International Journal of Reconfigurable Computing

Selected Papers from the Symposium on Integrated Circuits and Systems Design (SBCCI 2011)

Guest Massimo Conti, Elmar Melcher, Jürgen Becker, Alisson Brito, and Oliver Sander

Selected Papers from the Symposium on Integrated Circuits and Systems Design (SBCCI 2011)

International Journal of Reconfigurable Computing

Selected Papers from the Symposium on Integrated Circuits and Systems Design (SBCCI 2011)

Guest Editors: Massimo Conti, Elmar Melcher, Jürgen Becker, Alisson Brito, and Oliver Sander

Editorial Board

Contents

Selected Papers from the Symposium on Integrated Circuits and Systems Design (SBCCI 2011), Massimo Conti, Elmar Melcher, Jürgen Becker, Alisson Brito, and Oliver Sander
Volume 2013, Article ID 942021, 2 pages

Development of a SoC for Digital Television Set-Top Box: Architecture and System Integration Issues, André Borin Soares, Alexsandro Cristóvão Bonatto, and Altamiro Amadeu Susin
Volume 2013, Article ID 783501, 10 pages

Open SystemC Simulator with Support for Power Gating Design, George Sobral Silveira, Alisson V. Brito, Helder F. de A. Oliveira, and Elmar U. K. Melcher
Volume 2012, Article ID 793190, 8 pages

Efficient Execution of Networked MPSoC Models by Exploiting Multiple Platform Levels, Christoph Roth, Joachim Meyer, Michael Rückauer, Oliver Sander, and Jürgen Becker
Volume 2012, Article ID 729786, 13 pages

Algorithm and Hardware Design of a Fast Intra Frame Mode Decision Module for H.264/AVC Encoders, Daniel Palomino, Guilherme Corrêa, Cláudio Diniz, Sergio Bampi, Luciano Agostini, and Altamiro Susin
Volume 2012, Article ID 813023, 10 pages

Modeling and Implementation of a Power Estimation Methodology for SystemC, Matthias Kuehnle, Andre Wagner, Alisson V. Brito, and Juergen Becker
Volume 2012, Article ID 439727, 12 pages

An Optimization-Based Reconfigurable Design for a 6-Bit 11-MHz Parallel Pipeline ADC with Double-Sampling S&H, Wilmar Carvajal and Wilhelmus Van Noije
Volume 2012, Article ID 786205, 17 pages

HoneyComb: An Application-Driven Online Adaptive Reconfigurable Hardware Architecture, Alexander Thomas, Michael Rückauer, and Jürgen Becker
Volume 2012, Article ID 832531, 17 pages

QoSS Hierarchical NoC-Based Architecture for MPSoC Dynamic Protection, Johanna Sepulveda, Ricardo Pires, Guy Gogniat, Wang Jiang Chau, and Marius Strum
Volume 2012, Article ID 578363, 10 pages

A Memory Hierarchy Model Based on Data Reuse for Full-Search Motion Estimation on High-Definition Digital Videos, Alba Sandyra Bezerra Lopes, Ivan Saraiva Silva, and Luciano Volcan Agostini
Volume 2012, Article ID 473725, 10 pages

An FPGA-Based Omnidirectional Vision Sensor for Motion Detection on Mobile Robots, Jones Y. Mori, Janier Arias-Garcia, Camilo Sánchez-Ferreira, Daniel M. Muñoz, Carlos H. Llanos, and J. M. S. T. Motta
Volume 2012, Article ID 148190, 16 pages

Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2013, Article ID 942021, 2 pages
http://dx.doi.org/10.1155/2013/942021

Editorial

Selected Papers from the Symposium on Integrated Circuits and Systems Design (SBCCI 2011)

Massimo Conti,[1] Elmar Melcher,[2] Jürgen Becker,[3] Alisson Brito,[4] and Oliver Sander[3]

[1] *Dipartimento di Ingegneria dell'Informazione, Università Politecnica delle Marche, Via Brecce Bianche, Ancona, Italy*
[2] *Centro de Engenharia Elétrica e Informática, Universidade Federal de Campina Grande, Brazil*
[3] *Institute for Information Processing Technology (ITIV), Karlsruhe Institute of Technology (KIT), 76131 Karlsruhe, Germany*
[4] *Centro de Informatica, Universidade Federal da Paraiba, João Pessoa, Brazil*

Correspondence should be addressed to Massimo Conti; m.conti@univpm.it

Received 13 May 2013; Accepted 13 May 2013

SBCCI is an international forum dedicated to Integrated Circuits and Systems Design, Test and CAD, held annually in Brazil, copromoted by SBC, SBMicro, IEEE CAS, ACM SIGDA, and IFIP WG 10.5. The 24th edition of the Symposium on Integrated Circuits and Systems Design (SBCCI) was held in João Pessoa, PB, Brazil, from August 30 to September 2, 2011.

Track 2.1 of SBCCI 2011 was dedicated to Reconfigurable Computing. Some of the papers of this track have been selected for this special issue.

This special issue presents some of the latest developments in the area of design, specification, and modeling languages and applications of reconfigurable computing; reconfigurable architectures and novel applications of FPGAs; hardware-software codesign and coverification; emulation and prototyping techniques.

Ten articles are in this issue.

The main disadvantages of the reconfigurable approaches are still the costs in area and power consumption. In "*HoneyComb: an application-driven online adaptive reconfigurable hardware architecture*," A. Thomas et al. present a solution for application driven adaptation of a reconfigurable architecture at register transfer level to reduce the resource requirements and power consumption while keeping the flexibility and performance for a predefined set of applications. A prototype chip of this architecture designed in 90 nm standard cell technology manufactured by TSMC is presented.

Multiprocessor system-on-chip (MPSoC) security is becoming an important requirement. The challenge is to provide MPSoC security that makes possible a trustworthy system that meets the performance and security requirements of all the applications. The network-on-chip (NoC) can be used to efficiently incorporate security. In "*QoSS hierarchical NoC-based architecture for MPSoC dynamic protection*," J. Sepulveda et al. propose the implementation of Quality of Security Service to overcome present MPSoC vulnerabilities. The paper presents the implementation of a layered dynamic security NoC architecture to overcome actual MPSoC vulnerabilities.

Networked multiprocessor system-on-chips are used to implement novel embedded applications characterized by increasing requirements on processing performance as well as the demand for communication between several devices. Such systems require a detailed exploration on both, architectures and system design. In "*Efficient execution of networked MPSoC models by exploiting multiple platform levels*," C. Roth et al. present a methodology that embeds previous work into a simulation platform, which facilitates efficient execution of cross-domain simulation models on different abstraction levels.

In "*Open systemC simulator with support for power gating design*," G. S. Silveira et al. present an open source SystemC simulator with support to Power Gating design. This simulator is an alternative to assist the functional verification accomplishment of systems modeled in RTL. The possibility of controlling the retention and isolation of Power Gated Functional Block is presented turning the simulations more stable and accurate.

In "*Modeling and implementation of a power estimation methodology for systemC*," M. Kuehnle et al. describe a methodology to model power consumption of logic modules. A detailed mathematical model is presented and incorporated in a tool for translation of models written in VHDL to SystemC. The power analysis is based on a statistical model of the underlying HW structure and an analysis of input data.

In "*Development of a SoC for digital television set-top box*: *architecture and system integration issues*", A. B. Soares et al. present the development of a set-top box for Digital Television compliant to the SBTVD standard. It is a complex digital system integrating audio and video decoders and a CPU to run user interface and applications. Practical strategies used to solve integration problems are discussed. The SoC architecture is validated and is prototyped using a Xilinx Virtex-5 FPGA board.

In "*Algorithm and hardware design of a fast intra-frame mode decision module for H.264/AVC encoders,*" D. Palomino et al. present a fast intra mode decision algorithm and its hardware architecture design for an H.264/AVC video encoder. The proposed algorithm allows the complete elimination of the encoding loop present in Rate-Distortion-Optimization based mode decision, which is substituted by simple distortion calculations and comparisons, decreasing the complexity of the video encoder. The designed architecture of the fast intradecision algorithm was described in VHDL and synthesized targeting Stratix II FPGA and TSMC 0.18 μm standard cell library. The motion estimation is the most complex module in a video H.264/AVC encoder requiring a high processing throughput and high memory bandwidth, mainly when the focus is high definition videos. The throughput problem can be solved increasing the parallelism in the internal operations. The external memory bandwidth may be reduced using a memory hierarchy. In "*A memory hierarchy model based on data reuse for full search motion estimation on high-definition digital videos*", A. S. B. Lopes et al. present a memory hierarchy model for a full search motion estimation core. The proposed memory hierarchy expressively reduces the external memory bandwidth required for the motion estimation process and it provides a high data throughput. A case study for the proposed hierarchy was implemented and prototyped on a Virtex 4 FPGA.

In "*An FPGA-based omnidirectional vision sensor for motion detection on mobile robots*," J. Y. Mori et al. present a FPGA-based omnidirectional vision system for mobile robotic applications. The proposed architecture is suitable for robot localization, allowing to compute the distance between the robot and the surrounding objects. The overall architecture has been mapped onto a Cyclone II FPGA device, using a hardware/software codesign approach, which comprises a NIOS II embedded microprocessor and specific image processing blocks implemented in hardware.

In "*An optimization-based reconfigurable design for a 6-bit 11-MHz parallel pipeline ADC with double-sampling S&H,*" W. Carvajal and W. V. Noije present a 6 bit, 11 MS/s time-interleaved pipeline A/D converter design. The specification process, from block level to elementary circuits, is covered to draw a design methodology. Prelayout simulations of the complete ADC are presented to characterize the designed converter, which consumes 12 mW while sampling a 500 kHz input signal. The circuit will be sent to fabrication in a CMOS 0.35 μm AMS technology, and some postlayout results are shown.

Massimo Conti
Elmar Melcher
Jürgen Becker
Alisson Brito
Oliver Sander

Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2013, Article ID 783501, 10 pages
http://dx.doi.org/10.1155/2013/783501

Research Article

Development of a SoC for Digital Television Set-Top Box: Architecture and System Integration Issues

André Borin Soares, Alexsandro Cristóvão Bonatto, and Altamiro Amadeu Susin

PGMICRO, UFRGS, Avenida Bento Gonçalves 9500, 91501-970 Porto Alegre, RS, Brazil

Correspondence should be addressed to André Borin Soares; borin@inf.ufrgs.br

Received 21 January 2012; Revised 14 November 2012; Accepted 16 December 2012

Academic Editor: Oliver Sander

This work presents the integration of several IPs to generate a system-on-chip (SoC) for digital television set-top box compliant to the SBTVD standard. Embedded consumer electronics for multimedia applications like video processing systems require large storage capacity and high bandwidth memory. Also, those systems are built from heterogeneous processing units, designed to perform specific tasks in order to maximize the overall system efficiency. A single off-chip memory is generally shared between the processing units to reduce power and save costs. The external memory access is one bottleneck when decoding high-definition video sequences in real time. In this work, a four-level memory hierarchy was designed to manage the decoded video in macroblock granularity with low latency. The use of the memory hierarchy in the system design is challenging because it impacts the system integration process and IP reuse in a collaborative design team. Practical strategies used to solve integration problems are discussed in this text. The SoC architecture was validated and is being progressively prototyped using a Xilinx Virtex-5 FPGA board.

1. Introduction

Video processing systems require high performance processing elements and an efficient memory hierarchy design to reach real-time performance in the decoding of high-definition video sequences. Dedicated high performance modules are integrated into a single system which decodes the incoming bitstream and produces the output video images. In this process, reference frames are stored to be reused in the decoding process. A large size memory as an off-chip DRAM (Dynamic Random Access Memory) is mainly used and the memory accesses are directed to a single off-chip memory interface. The memory hierarchy design and computation complexity are bottlenecks to reach real-time high-definition video decoding [1].

This work presents an architectural design and FPGA (Field Programmable Gate Array) implementation of a SoC for H.264/AVC video decoding [2]. A SoC is a complex system, and the integration and validation of the design are the challenges of the development process. The architecture considers a four-level memory hierarchy [3] composed of local SRAM memories and by off-chip DRAM memories. Off-chip DRAM memories can guarantee the necessary storage capacity at low cost if compared to embedded SRAM. The memory controller is designed with a multichannel data interface because different processing modules need to share the same data port. The multichannel controller manages data access requests and optimizes the reference memory utilization, enabling data processing modules to interact efficiently in order to satisfy the performance requirements.

This paper is organized as follows. Section 2 discusses related works; Section 3 presents the set-top box hardware architecture, the H.264/AVC video decoder, the audio decoder, and the implemented memory hierarchy; integration issues and strategies employed are discussed in Section 4; FPGA implementation results are discussed in Section 5 and finally we conclude in Section 6.

2. Related Works

Related works can be grouped in three main categories: software decoding, hardware/software decoding, and hardware decoding. Pescador et al. [4] presented an H.264

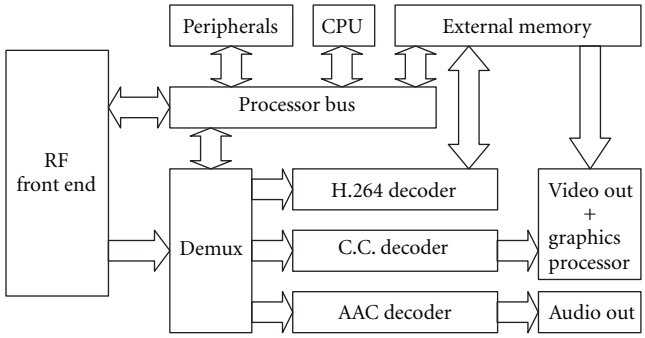


Figure 1: SBTVD compatible set-top box.

video decoder running on a VLIW DSP at 600 MHz with software highly optimized to use DSP native features. Results presented the minimum of 57% of CPU utilization for 480p sequences in the baseline profile. Yang et al. [1] developed a single chip decoder SoC for H.264 decoding using an RISC processor and hardware accelerators. The basic decoding flow and synchronization are controlled by software. The system operates at 100 MHz at the baseline profile, decoding CIF videos at 20 MHz. Lin et al. [5] presented an ASIC implementation of the H.264 video decoder with dedicated hardware modules and two memory controllers, decoding HD1080p videos at 120 MHz. It uses only 4.5 KB of local memory. Modern SoCs like [6] have all the algorithms developed in hardware to reach high performance with low power.

The video decoder presented on this work is completely implemented in hardware and presents itself as an intermediary step for the future production of a chip. Currently it is capable of decoding 720p videos at 50 MHz in real-time on an FPGA. Based on previous experiments [7], a silicon version is expected to get the benefit of low power consumption by operating at a lower frequency than software-based approaches and by the utilization of a design flow with low power techniques. Also, the inclusion of a dedicated CPU on the system will enable an effective implementation of a user interface and the execution of applications. Beyond set-top boxes, this solution could also be reused in other systems such as tablets or mobile phones. Nevertheless, a more detailed analysis concerning the overall power consumption must be performed for these applications. One of the main contributions of this work is to present the techniques used to reduce the complexity of the integration and test of the SoC design while keeping the RTL description used in the initial creation of the system components.

3. Set-Top Box Architecture

The SBTVD (Brazilian Digital Television System, in portuguese) follows the ISDB-t standard [8], which is based on the Japanese television system (ISDB). This standard comprises H.264 for video encoding and AAC for audio encoding. Video and audio decoding must be performed in hardware by specialized architectures in order to reach decoding efficiency. The Brazilian government funded the development of an H.264 decoder prototype and an SBTVD compatible SoC [9, 10]. An overview of the architecture of the SBTVD SoC developed in this work is shown on Figure 1. The design is being developed following Brazilian standards [8, 11] to enable compatibility with broadcast transmissions. The SoC is composed of an H.264 video decoder, an AAC audio decoder, a closed caption decoder, a Leon3 CPU [12], and a demultiplexer, receiving a compressed bitstream from an external RF front-end and demodulator. Additionally a graphics processor exhibits decoded images, superposes decoded subtitles and locally generated messages or images and a CPU is intended to run user interface and interactivity applications. Video decoder, main CPU, and graphics processor are connected to the external memory controller. The system bus is an AHB AMBA bus standard, implemented in VHDL language.

The LEON3 is a synthesizable VHDL model of a 32-bit processor compliant with the SPARC V8 architecture developed by Gaisler Research [12] and available for use in research projects under free license. SPARC is a CPU instruction set architecture (ISA), derived from a reduced instruction set computer (RISC) lineage.

3.1. H.264/AVC Video Decoder Architecture. H.264/AVC is the latest video coding standard of the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG). This state-of-the-art video coding standard outperforms previous standards by employing bipredictive motion estimation, spatial prediction, and adaptive entropy coding techniques. However, the cost of increased performance is an augmented computational complexity and the enlarged memory traffic when compared to previous standards. Dedicated modules must supply the high performance demand of the decoding processing. Memory access optimizations are also necessary to handle data of the reference frames while video is decoded and exhibited in real-time.

The H.264/AVC video coding standard defines a set of levels and profiles. Complexity and output bandwidth are related to the level, while profiles refer to different set of coding functions. There are three different profiles defined by the standard: baseline, main, and high. The main profile includes television broadcasting and video storage, supporting interlaced videos, intercoding with bipredictive slices, intercoding with weighted prediction, and entropy coding using context-adaptive arithmetic coding (CABAC) [2]. The hardware decoder in development in this work targets decoding video streams in the main profile at level 4.0. Figure 2 shows the video reconstruction path in an H.264 decoder, starting with the compressed video bitstream and ending with the YCbCr video that is stored in the RPB (reference picture buffer).

Parsing with entropy decoding extracts from the compressed video bitstream the syntax elements to interframe and intraframe video reconstruction and compressed residual data. The residual data is decoded using fixed or variable length binary codes in the entropy parsing module and is processed in the inverse transform and inverse quantization

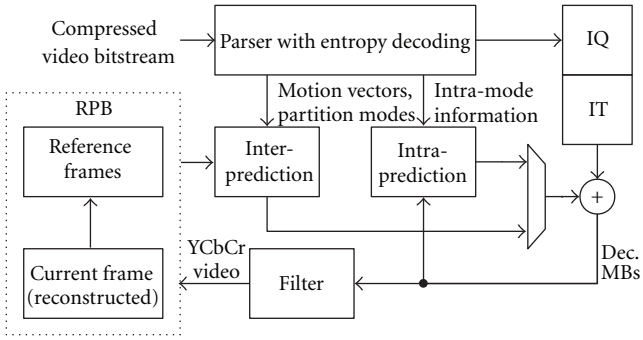


Figure 2: Block diagram of the H.264/AVC video decoder.

(IT and IQ) steps. Using information decoded from the bitstream, the decoder creates a prediction block using intra or inter prediction modes. Motion compensation (MC) is the hardware module which reconstructs an actual frame from reference frames. Intraprediction reconstructs each image block from its neighborhood.

The H.264 decoder consists of five main processing units, designed as IP (Intellectual Property) cores: the inter-prediction module, or MC (Motion Compensation), the intraprediction or Intra, the deblocking filter, the inverse transform (IT) and inverse quantization (IQ), and the entropy decoding module. The filter is the output of the decoder and generates the predicted macroblocks to the RPB by reducing block artifacts.

The RPB is the module designed to manage the reference frames used in the video decoding process. The main functions of the RPB are the frame data storage and control. Decoded frames are stored on the external memory, in which they can be located by address pointers. They are also indexed on reference lists to be used as references in the decoding process.

3.1.1. Storage Requirements for Video Applications. Video processing applications have high bandwidth requirements that increase with the video resolution, bit depths, and frame rates. For example, when decoding a SD (720 × 480) video stream of 30 frames per second, the video decoder output generates 10.4×10^6 pixels per second, while for HD (1280 × 720) video stream of 30 frames per second, the pixel output is 27.65×10^6 pixels per second.

The colorspace, pixel's depth, and picture subsampling are also determinant of the output bandwidth. There is an information compression if the pixel is represented in the YCbCr (luminance, chrominance blue and chrominance red) with 4 : 2 : 0 subsampling instead of use RGB (red, green, and blue) format. For example, a video picture in Full-HD resolution (1920 × 1080) represented in YCbCr 4 : 2 : 0 8-bit per pixel requires 3.1×10^6 bytes and in RGB 12-bit per pixel 9.4×10^6 bytes to be stored in memory.

The video decoding process in H.264 standard Main profile requires about 12.5×10^6 bytes for reference frame memory in YCbCr 4 : 2 : 0 8-bit format. This memory size is needed to store four video frames in Full-HD resolution.

3.1.2. Reference Picture Buffer Accesses Behavior in H.264/AVC Video Decoding. Image is processed in the form of regions of 16 × 16 pixels size called macroblocks of pixel (MBs) samples. Video processing is done by manipulating pixels in coarse-grain tasks that are mapped onto processing units. This is the video decoder granularity and each pixel sample in the macroblock is represented by its luminance (Y) and chrominance (Cb and Cr) components. The video decoder output is a 32-bit four pixels width line-of-pixels and a sequence of four line-of-pixels generates a 4 × 4 block of pixel samples. The macroblock thus consists of a sequence of 256 luminance samples followed by 64 Cb chrominance samples and 64 Cr chrominance samples, ordered in double-z.

There are three main modules accessing the RPB: the filter output (reference pictures), video display output, and motion compensation process. They have different access behaviors and bandwidth requirements.

The MC process is the most computationally demanding in the video decoder [13], and the one that generates more external data requests. As the video decoding process has an unpredictable behavior, the MC module can access the main memory at different data rates. Also, the region of pixels used to reconstruct the image can be different in each decoded macroblock. The MC cache uses an addressing scheme based on the (x, y) pixel block coordinates and the reference picture number.

The H.264 video decoder output generates decoded pixels in a sequence of macroblocks. It does not generate any addressing information, which must be controlled by the RPB.

Video display output module needs to fetch decoded frames in the right display order, which can be different from the decoding order. It scans an entire line of pixels in the exhibited image. As in the case of the decoder output, the video display does not generate addressing information. Pixels are requested by the video output and the RPB must perform address control.

We can estimate the memory interface access bandwidth for each processing unit by the decoded video resolution and macroblocks generation rates. The interpolation blocks in motion compensation process are all 4 × 4 size. It can be seen that MC process is the most bandwidth demanding in the H.264 decoder. The total memory bandwidth necessary to decode 1920 × 1080@30fps is 1.002 GB/s. More details can be found in [14, 15].

3.2. AAC Audio Decoder. The audio decoder was developed on a secondary prototyping board. A data link was designed to connect both boards. The MPEG-4 AAC decoder is capable of decoding a stereo bitstream with 48 kHz sampling rate in real-time operating with a clock frequency of 4 MHz. It includes a parser, a filter bank, and a spectrum decoder composed of noiseless decoding, an inverse quantizer and a rescaling module and decodes mono, stereo, and 5.1 audio

streams. The audio decoder architecture is described with more details in [14].

3.3. Multichannel Memory Controller. Currently the implementation of the quantity of memory necessary for the RPB as on-chip memory is not cost effective. On-chip SRAM is about 7 mm^2 per mega-byte in 65 nm technology [15]. Embedded DRAM has a higher density but requires additional mask layers for manufacturing, increasing system cost. Architectures of video processing systems require a single interface to off-chip DRAM memory in order to achieve the necessary storage capacity at low cost. The multichannel memory controller (MMC) designed in this work implements the RPB to the video decoder described in Section 4. An off-chip 64-bit wide DDR2 SDRAM memory running at 200 MHz is used as main system memory. Also, the multichannel memory controller implements a data/instruction interface for the Leon3 CPU through the AMBA bus standard interface.

3.3.1. DDR2 SDRAM. DDR2 SDRAM is a double-data rate synchronous DRAM interface designed to transfer data on the rising and falling edges of the bus clock signal. This memory allows higher bus speed than its previous memory standard, DDR SDRAM, and requires lower power by running the internal clock at one quarter the speed of the data bus. For example, considering that the internal memory clock is 100 MHz, the bus clock will be 200 MHz and the bus interface is capable to perform 400 megatransfers per second. Also, with data being transferred 64-bit at each bus clock edge, the maximum transfer rate is 3,200 mega-bytes per second.

Data accesses are linear and words are stored organized in banks and pages. Data can be transferred in bursts with 4 or 8 data words in each memory access. Data memory contents are accessed by page activation (ACT), using the row-address strobe command and a column-address strobe command. In the case of a read operation, data is available after the CAS Latency (CL) which can be 3 to 7 clock cycles plus an optional additive latency (AL).

DDR2 SDRAM latency can reduce significantly the system performance if data are not transferred in bursts. Also, frequent row changes or bank conflicts reduce the system performance because it is necessary to deactivate the current row or bank, and activate the next to be used. This is done by using a sequence of commands called precharge (PRE) and activate (ACT), taking about 10 cycles after the last data access operation.

By storing the images in the off-chip memory in the form of macroblocks of pixels organized as YCbCr 4 : 2 : 0 8-bit image format, it is possible to design the granularity of data transfers to the reference memory as being a macroblock. As presented before in [16], macroblock granularity provides better SDRAM efficiency and less power consumption. This granularity for data transfers is used in the H.264 decoder, video output, and graphics processor.

3.3.2. Multichannel Memory Controller (MMC) Architecture. Figure 3 shows the memory controller architecture. The off-chip memory interface consists of a physical IP (DDR2 PHY) from Xilinx [17]. The hardware modules are connected to the memory controller through channels which share the same command and address bus, being necessary to have an arbiter controlling the data requests. Each channel interface is defined as IF (interface) with an associated number, in the presented case IF0 to IF4. Each IF consists of data buffers and address generators. The address generators are necessary to index stored data in the external memory. In this system, only the CPU generates addressing information to manage data. Each IF contains data buffers in different sizes and data organization schemes, to optimize external memory data transfers.

An arbiter was designed to control read and write requests do the DDR2 PHY. Also, the MMC contains a data multiplexer and demultiplexer and a RPB control, necessary to manage the video frame buffer in the external memory. This design implements five interfaces to the external memory with corresponding command, address, data and control signals. Nevertheless, it is possible to extend the architecture to use more data channels. The multichannel interface uses a simple protocol where an acknowledge signal gives permission to the module after the received access request. The main advantages of this controller are the scalability to add more channels, the possibility to control the priorities of transfers, and the data conversion on the channel buffering. A more detailed analysis of the memory hierarchy was presented in [18].

Optimized Memory Hierarchy. When interfacing with DRAMs, it is necessary a time to setup a data transfer and this time cannot always be overlapped with another data transfer. Therefore, DRAM accesses need to have large burst sizes in order to use the DRAM interface efficiently. Whether all this data is actually needed depends on the spatial locality of the data accesses. The sizes of the DRAM bursts need to be aligned with the sizes of the cache blocks and the local buffers used to interface each processing unit. Figure 3 shows local and off-chip memory resources used by the multichannel memory controller, grouped within the levels of the memory hierarchy. It contains the following four levels.

Level 3: Off-chip DRAM memory: it is the biggest memory module with reduced cost if compared with the internal SRAM memory. This level has the larger latency.

Level 2: Memory controller buffer: it is the memory level used to store data for an off-chip memory transfers. Data is saved along with addresses and commands to be written in external memory. This memory level allows the controller (PHY) to manage off-chip memory information with autorefresh operations and different banks or row changing without disrupting the interfaces that are accessing data. The Read FIFO is only used with additive read latency (AL) option is enabled [17].

Level 1: Macroblock-level memory: in this level, data is stored in buffers until reach the minimum size of a macroblock before being transferred to external memory. This level is necessary to maximize the size of data transfer

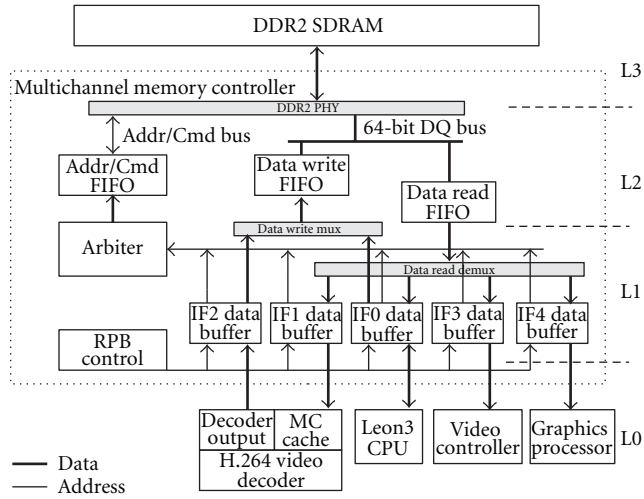


Figure 3: Multichannel memory controller block diagram. The memory resources are also grouped within the levels of the memory hierarchy (L0 to L3).

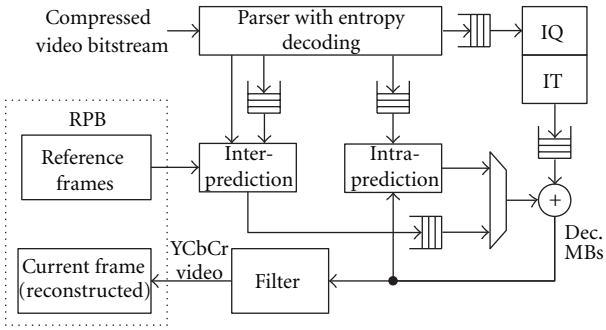


Figure 4: Buffering in H.264 video decoder.

to external memory, because the transfer takes the form of bursts of data in line memory address. It is also necessary to generate the addresses.

Level 0: Local SRAM processing buffers: it is the lowest level of memory, characterized by local memory in the processing units used to process information locally; it enables the execution of more local processing without external memory access. Figure 4 shows local buffers on the video decoder.

IF Channel Interfaces. Each channel interface consists of an address generator and a data buffer. The address generators are used to reorganize the macroblocks of pixels in reference pictures. The buffers can be parameterized for each access behavior requirement.

In the case of video display, the buffer has a size of 240 macroblocks. This is necessary to store two entire lines of macroblocks, considering full HD video resolution, while the other line is being exhibited. The decoder output contains a buffer with two macroblocks capacity. The MC requires a data buffer with storage capacity of three macroblocks. The MC architecture used in the video decoder is presented in [19]. It contains a local cache to store a tridimensional data structure which contains the requested reference pixels while the reconstruction process is executed. Cache size is $40 \times 16 \times 16$ 8-bit samples for Y and $20 \times 8 \times 16$ 8-bit samples for Cb and Cr.

The buffers are also used to align multiple 32-bit words (or a line-of-pixels LoP) to 64 or 128-bit data wide and to synchronize multiple clock domains. Larger buffer size increase memory bandwidth utilization due to the reduction of the utilization of memory commands.

The address generator receives information from the RPB control to store the reference frames, generating the addressing scheme to access data in the external memory. Each decoded frame is labeled as reference frame or not. This information is generated by the decoder during the bitstream parsing process.

Multichannel SDRAM Arbiter. The arbiter controls the access of each hardware module to the time-division multiplexed memory channel. It is associated a priority to each IF channel in the arbitration scheme, considering one classification associated to high or low latency and high or low bandwidth. Each processing unit accessing the off-chip memory is classified as latency sensitive (LS) or bandwidth sensitive (BS). LS processing units require immediate access to memory channel and use the channel for few memory transactions. In the other side, BS processing units uses the off-chip memory channel to for long memory transactions, transferring high data volumes, but without immediate access. In this implementation, the best approach to multiplex the memory channel between process units is the use of a priority-based arbitration scheme with preemption. More details are show in [18].

Reference Picture Buffer Control. Frame allocation in the RPB is performed by a dedicated buffer management module, which calculates the buffer size and the pointers to the buffers stored in the reference memory. This calculation is performed every time that the video resolution changes. Image

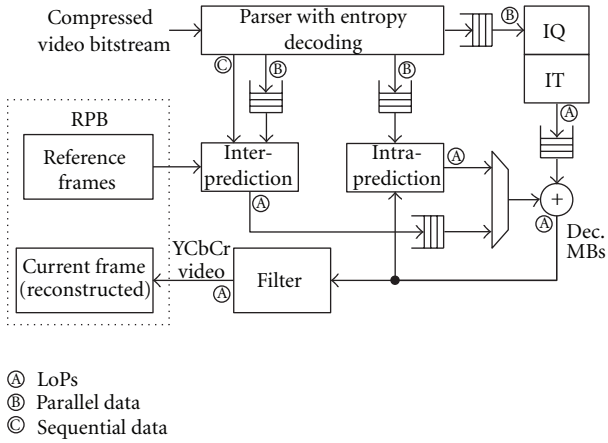


Figure 5: Data format on the H.264 decoder.

dimensions are stored in the bitstream, being recovered by the parser module.

Motion Compensation requires random memory access based on a Reference Index that points to specific buffers in the reference memory. Base pointers associated to the reference index are provided by the buffer control and fed the address generator of IF1 data buffer.

Reference list management must support a dynamic list with pointers to buffers. Two memories are used to implement this behavior. The first memory contains the pointers to the reference memory contents. The second memory contains the state of the list, including buffer state and an index to the first memory.

Operations performed by the RPB control module include buffer allocation for the video decoder output, identification of the buffer to be exhibited, and reallocation of an exhibited buffer. The graphics processor and Leon3 CPU also interface with the buffer management control when accessing data in the graphics buffer memory region. The CPU generates an on-screen display (OSD) and stores it in the graphics processor memory area located at the external memory. The OSD image is superimposed on a screen picture with the decoded video by the graphics processor. This commonly used feature was implemented in full image resolution, using the YCbCr 4 : 4 : 4 pixel format to achieve high quality graphics pictures.

Three submodules composes the buffer manager module. The first submodule receives requests from the decoder output and translate them into buffer insertion commands. It controls the start and end of a frame decoding operation. Buffer is allocated in the start of decoding operation and must be inserted in the reference list in the end of decoding procedure. The second submodule receives the requests from video output module and translate them into buffer removal commands. It also controls frame exhibition repetitions to adjust the video frame rate of the decoder to the video output module frame rate. This enables a 15 fps video to be exhibited into a 60 fps monitor, for instance. The third submodule centralizes the requests for buffer insertion and removal from the reference lists and the real state of the buffers in the DDR memory (allocated or not allocated).

This submodule provides different priorities for the reference list and buffer control operations when different clients request for buffer pointers. A request of a memory pointer to video exhibition has the highest priority, as the output timing is critical. Also, when the buffers are all occupied, a buffer must be released before it can be used in a request from the video decoder.

Buffer insertion and removal are not time-critical operations as they are executed one or two times in the time of a frame (33 ms). They are implemented as state-machines which scan the memories when a buffer must be inserted or removed from the reference list. Each operation can take up to 16 cycles due to the number of reference frames. The recovery of a buffer pointer based on a reference index is an operation that occurs one time to each 3 macroblocks. In this implementation, it takes one clock cycle.

4. Integration Issues

4.1. Integration Issues on the H.264 Video Decoder. The H.264 video decoder of this work is being developed as a collaborative design. The initial project specification established an architecture based on the algorithm structure described in [2]. The data transfers between IPs inside the video decoder were specified to follow the order in which data is produced by the decoding process, in three main formats: LoPs, parallel data, and sequential, as shown in Figure 5.

The decoder modules were developed simultaneously, following a reference software model [20]. They were validated with individual testbenches using bitstreams with the inputs and expected outputs, generated from a reference software. During integration phase, some issues required the redesign of some parts of the IPs of the video decoder. They can be grouped in two main categories: issues related to intermediate buffer insertion and issues related to data dependency. Although these issues may seem obvious at this stage of the project, they emerged during the system development due to the complexity of the design.

4.1.1. Issues Related to Intermediate Buffers. The decoder operates with a data flow regulated by the consumed frames in the output. The average rate in which the frames are consumed is determined by the exhibition rate. Frames are produced and consumed from the DDR in an irregular instantaneous rate, due to the channel multiplexing. This irregular rate propagates from the output to the input of the decoder by the full signal of the FIFOs. At the input side, the parser operation is irregular as it decodes an irregularly compressed input bitstream. If the parser is not fast enough, this produces an irregular operation in all the modules, as the FIFOs connected to the output of a given module pass by the full condition, or the FIFOs connected to the input of a module pass by the empty condition. This irregular rate can propagate from the input of the decoder to its output by the empty signal of the FIFOs. These two effects require that all modules of the decoder must be enabled or disabled in

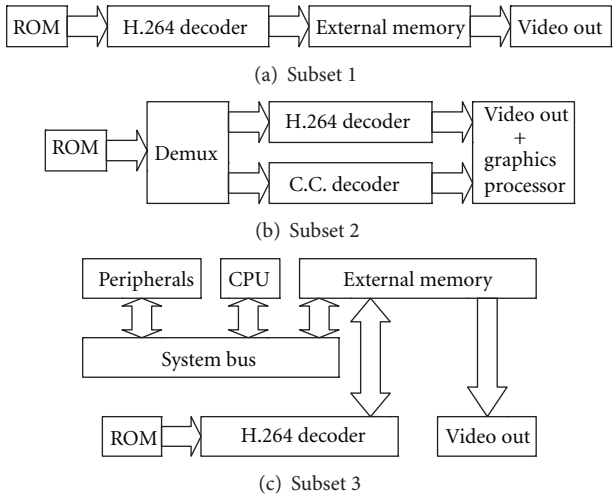


Figure 6: Subsets of SoC IPs used for incremental development.

some way by preceding or succeeding modules in the data flow chain.

IP integration required some interface adjusts to deal with these start/stop conditions, which can be grouped in two basic strategies. The first strategy consists of the insertion of an enabler signal in the IP. This strategy requires distribution of an enable signal within the IP and despite its conceptual simplicity, it requires considerable effort in debugging exception cases that can emerge due to incorrect synchronization of control and datapath operation with the enable signal. The second strategy consists of the inclusion of a buffer in the output of the module (on the level 0 of the hierarchy), disabling the module inputs when there is not enough space in the output buffer. This strategy assumes that the module process data in packets. When the output has enough space for one packet, the module processes one packet of data.

The first strategy enables smaller output buffers as it controls module operation with a finer temporal/data granularity than the second strategy. Nevertheless, it requires larger effort and development time. Also, it requires the access to the IP source code and detailed internal documentation. Access to an IP internals is not a severe issue when the IP is developed for a specific project, in a collaborative design, but it is a problem when applying this strategy in the integration of third party IPs. The second strategy requires a large buffer to store at least one data packet in the output. If this condition is not met, data loss can occur when the module maintains its operation and next module in the chain does not consume data from the buffer. This strategy does not require modifications in the interface and it is more suited to complex IPs. Both strategies were applied on the video decoder integration [21].

4.1.2. Issues Related to Data Dependency. The IPs composing the video decoder were designed with an initial functional specification of each module based on the operation of the corresponding part in the reference software. All local data dependencies were considered at design time. Nevertheless, due to the complexity of the algorithms, additional mechanisms were necessary to solve global unpredicted data dependencies and interactions between the internal decoder modules and I/O blocks on lower levels of the memory hierarchy. These data dependencies produced in the initial tests an incorrect behavior of the decoder due to the pipeline operation of some modules and to the macropipeline operation of the decoder.

Other issues are related to limitations of the platform used for hardware prototyping. As one example, Motion Compensation was designed with a cache considering an external memory interface with 320 bits wide data channel. This corresponds to 2.5 macroblock wide, and the cache must be fed by a specific arrangement of memory banks. The board used to prototype the decoder has a 64-bit wide data channel. The data conversion between MC and external memory is performed by the memory channel, designed to receive data in macroblocks and deliver it in parallel to MC cache, on the level 1 of the memory hierarchy. This strategy can be also applied to use standard off-the-shelf memory modules with a final SoC implementation in silicon. More information can be found in [21].

4.2. Integration Issues on the SoC

4.2.1. Issues Related to System Complexity. An SoC is generally a large and complex design. In the case of the SoC for the SBTVD, the presented H.264 video decoder is considered just one IP. Larger complexity implies in larger synthesis and validation time, increasing the development time.

Integration was planned to be performed as partial subsets of the final SoC. This strategy enabled to reduce developing time by working with smaller designs, thus reducing complexity, synthesis time and subsequently the verification effort. It also enabled distinct subteams to work in parallel, reducing development time.

The subsets of the SoC (shown on Figure 6) in the integration process can be enumerated as Subset 1, 2 and 3, and described as follows.

Subset 1. With the video decoder and a basic video output module with the multichannel memory controller, with test data fed by internal ROM; this subset enabled video decoder and decoded picture buffer testing and validation.

Subset 2. With the video decoder, graphics processor and demultiplexer, with test data fed by ROM; this subset enabled to test correct operation of the demultiplexer and graphics processor.

Subset 3. With the Leon3 CPU, the memory controller, and basic video controller; this subset enabled to develop and test applications on the CPU, the video frame buffer, and the development of a basic graphics library.

With these subsets, it was possible to start the software development before the end of the hardware integration process. All the subsets used the same interfaces between the corresponding modules, enabling to compose the final

(a) (b)

Figure 7: (a) Floorplanning and placed design of the H.264 video decoder of Subset 1; (b) on board demonstration of the prototype of Subset 3.

system as a composition of the subsets. The exception are the interfaces of the graphics processor and video controller used on the tests. The main difference is an input data channel for the closed caption data on the graphics processor, which displays only video if this channel is unused.

Large systems as this sometimes must to be prototyped in more than one development board. The interconnection between IPs on different boards must be designed to transfer data reliably avoiding problems related to clock differences between transmitter and receiver and skew between interface signals. The difference between clocks was treated with the use of a FIFO with different read and write clocks. As the data flows only from transmitter to receiver, the FIFO was located on the receiver and its potential consuming rate is larger than the producing rate of the transmitter. The skew between interface signals was treated by dimensioning the interconnections with the same length.

5. Results

5.1. Simulation Tests. The H.264 video decoder, using the described memory hierarchy and topology shown in Figure 3, was validated by simulation using video test sequences using Modelsim simulator. Simulation of the system shown in Figure 3 has limited application as the time necessary to simulate complex software (an operating system for instance) is very long.

5.2. On Board Tests. Board tests were initially performed on a XUPV2P board with a XC2VP30 Virtex II Pro device for individual tests with the video decoder and Leon3 CPU. As the size of the system increased it was migrated to a Xilinx ML509 board, with a XC5VLX110T Virtex-5 FPGA and 512 MB of external DDR2 SDRAM. The proposed architecture was synthesized using FPGA development tool ISE, from Xilinx. Synthesis results for the Virtex-5 are presented in Table 1.

On the ML509 board the proposed subsets were validated as follows.

(1) An Intra-only version of the decoder, with Multichannel Memory Controller configuration (Subset 1), decoding videos at 720p and 1080p.

(2) A baseline version of the decoder, with Multichannel Memory Controller configuration (Subset 1), decoding videos at 720p and 1080p. Currently the filter is also integrated, but it does not fit on the device with the decoder. This subset imposed a higher degree of effort to reach on board validation requiring the execution of manual floorplanning (shown on Figure 7(a)) before placement and routing.

(3) An Intra-only version of the decoder with demux and Closed Caption decoding (Subset 2).

(4) The SoC composed of the CPU Leon3, H.264 video decoder fed by a ROM and multichannel memory controller (Subset 3). The system runs with 50 MHz on the video decoder and CPU (shown on Figure 7(b)). The execution of an application

(5) Subset 1 using external (offboard) loopback between the ROM and the video decoder to test electrical interface between boards.

The interface with other boards, to transfer data for audio board and RF front-end, was developed and tested initially using a data source on the same FPGA which holds the video decoder and video output, passing data by a loopback connector, operating at 27 MHz. In a second step, the board with the video decoder was connected to a board with an RF demodulator and demux, with the link operating successfully. The audio decoder was not completely prepared for integration at the time of the realization of these experiments.

The SoC prototype with CPU (Subset 3) was tested with a small video being decoded in parallel with a small graphics

Table 1: Synthesis results for Xilinx XC5VLX110T FPGA.

	Slice Regs	Slice LUTs	BlockRAMs
PHY DDR2	2277	1749	3 (108 kb) L2
MMC	2714	2739	42 (1512 kb) L1
Parser	1346	3849	37 (6 kb) L0
MC	41411	23174	33 (381 kb) L0
Intra	2071	4164	3 (41 kb) L0
IqIt	5827	4792	3 (76 kb) L0
Filter	2254	2275	94 (5 kb) L0
Leon-3 CPU	4868	6618	29 (1044 kb) L0
Graphics processor	789	1331	13 (468 kb) L0

application created to test the video memory. This subset used a version of the H.264 decoder capable of decoding only frames with intraframe prediction. This version was chosen due to the limitation on the capacity of the XC5VLX110T device. When the migration to a platform with a larger FPGA is complete, the H.264 decoder will be replaced in the SoC prototype. The final goal is to develop an ASIC version of the SoC, after complete validation of the architecture on board.

The area overhead introduced in the output of MC module on level 0 is a buffer of two macroblocks, consuming 6.1 kb. The overhead in the channel of IF1 on level 1 to adapt the 128-bit port of PHY to 320-bit port of MC Cache is a buffer of 3 macroblocks, consuming 9.2 kb.

All individual modules were designed with an individual minimum performance which enables full HD decoding at maximum frequency of 100 MHz. The pipeline operation enables the system to decode one macroblock in 415 clock cycles for I frames and one macroblock in 384 clock cycles for P frames on the average. Latency is not a major concern in the decoder output as the number of cycles to decode one frame is up to 8160 times greater than one macroblock. Thus the system is capable of decoding 720p 30fps videos at 50 MHz and 1080p 30fps videos at a frequency near 100 MHz, depending on video content.

The measured performance showed that the memory elements on level 0 were correctly dimensioned. Attempts to reduce buffer sizing in certain elements caused the stop of the video decoding process due to data dependency.

In these tests, the DDR2 SDRAM memory controller runs at 200 MHz, the H.264 video decoder at 50 MHz and the output video controller runs at 110 MHz to generate 720p resolution or 148 MHz for 1080p. Tests in hardware with 1080p currently use the RPB capability to store a decoded picture and repeat it until the decoding process of the next one is finished.

6. Conclusions

In this work, we presented the development of a set-top box for Digital Television. It is a complex digital system integrating audio and video decoders and a CPU to run user interface and applications. A multiport memory interface with a memory hierarchy optimizes the off-chip memory usage.

The inclusion of this memory hierarchy has to be considered in the system design. It is known that buffering on lower levels of the hierarchy reduces potential bottlenecks in higher levels, where channel multiplexing is needed. Also, the sequential access characteristics of a video recovering task benefits from burst readings in DDR memory. Strategies were presented and analyzed to enable IP reuse and integration by buffering data and controlling processing on modules. The strategies proved to be effective, as the expected performance of the system was not reduced, and they introduced a small overhead in the architecture of the final system.

Complexity grows with the size of a system, and so does the time for design synthesis and validation. It has a strong impact on development time. In this work, the strategy of incremental integrations was crucial for reducing SoC complexity and verification effort, enabling to specify clear checkpoints in the development. With the execution of development and validation of independent parts of the system in parallel through independent on-board test of critical parts of the system, the strategy also enabled the development of the SoC in the available time for the project.

The shared memory in the higher levels enabled the integration of the video decoder with other elements, like an image processor to overlay an image from an access terminal over the decoded video or other systems through the insertion of more channels on the multichannel memory controller. The next step is the migration to a larger prototyping platform, with a device that can support the integration of the complete system.

Acknowledgments

This work is supported by CAPES and FINEP.

References

[1] K. Yang, C. Zhang, G. Du, J. Xie, and Z. Wang, "A hardware-software co-design for H.264/AVC decoder," in *Proceedings of the IEEE Asian Solid-State Circuits Conference (ASSCC '06)*, pp. 119–122, Beijing, China, November 2006.

[2] "ITU-T recommendation H. 264: advanced video coding for generic audiovisual services," Video Coding Experts Group, 2005.

[3] JEDEC, *JESD79-2F: DDR2 SDRAM Specification*, JEDEC Solid State Technology Association, Arlington, Va, USA, 2009.

[4] F. Pescador, G. Maturana, M. J. Garrido, E. Juarez, and C. Sanz, "An H.264 video decoder based on a DM6437 DSP," in *Proceedings of the International Conference on Consumer Electronics (ICCE '09)*, Las Vegas, Nev, USA, January 2009.

[5] C. C. Lin, J. W. Chen, H. C. Chang et al., "A 160K gates/4.5 KB SRAM H.264 video decoder for HDTV applications," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 1, pp. 170–181, 2007.

[6] ST SPEAr 1340 dual-core cortex A9 HMI embedded MPU, http://www.st.com/.

[7] A. C. Bonatto, A. B. Soares, A. Renner, A. A. Susin, L. M. Silva, and S. Bampi, "A 720p H.264/AVC decoder ASIC implementation for digital television set-top boxes," in *Proceedings of the 23rd Symposium on Integrated Circuits and Systems Design (SBCCI '10)*, pp. 168–173, September 2010.

[8] ABNT, *NBR15602 Digital Terrestrial Television—Video Coding, Audio Coding and Multiplexing*, ABNT, 2007.

[9] "Project Rede H. 264," http://www.lapsi.eletro.ufrgs.br/h264/wiki/tiki-index.php.

[10] "Project SoC-SBTVD," http://www.lapsi.eletro.ufrgs.br/soc-sbtvd/wiki/tiki-index.php/.

[11] ABNT, "NBR15604 digital terrestrial television—receivers," ABNT, 2007.

[12] "Gaisler research Leon3 processor," http://www.gaisler.com .

[13] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro, "H.264/AVC baseline profile decoder complexity analysis," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 704–716, 2003.

[14] A. Renner and A. Susin, "An MPEG-4 AAC decoder FPGA implementation for the Brazilian Digital Television," in *Proceedings of the 8th Southern Conference on Programmable Logic (SPL '12)*, pp. 1–6, Bento Gonçalves, Brasil, March 2012.

[15] P. Van Der Wolf and T. Henriksson, "Video processing requirements on SoC infrastructures," in *Proceedings of the Design, Automation and Test in Europe (DATE '08)*, pp. 1124–1125, ACM, Munich, Germany, March 2008.

[16] C. H. Li, C. H. Chang, W. H. Peng, W. Hwang, and T. Chiang, "Design of memory sub-system in H.264/AVC decoder," in *Proceedings of the Digest of Technical Papers International Conference on Consumer Electronics (ICCE '07)*, pp. 1–2, January 2007.

[17] Xilinx, http://www.xilinx.com/.

[18] A. C. Bonatto, A. B. Soares, and A. A. Susin, "Multichannel SDRAM controller design for H.264/AVC video decoder," in *Proceedings of the 7th Southern Conference on Programmable Logic (SPL '11)*, pp. 137–142, Córdoba, Argentina, April 2011.

[19] B. Zatt, A. Azevedo, L. Agostini, A. Susin, and S. Bampi, "Memory hierarchy targeting bi-predictive motion compensation for H.264/AVC decoder," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI '07)*, pp. 445–446, Porto Alegre, Brazil, March 2007.

[20] "H.S. Coordination," JM Software, http://iphome.hhi.de/suehring/tml.

[21] A. B. Soares, A. Bonatto, and A. Susin, "Integration issues on the development of an H.264/AVC video decoder SoC for SBTVD set top box," in *Proceedings of the 24th Symposium on Integrated Circuits and Systems Design (SBCCI '11)*, August-September 2011.

Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2012, Article ID 793190, 8 pages
doi:10.1155/2012/793190

Research Article

Open SystemC Simulator with Support for Power Gating Design

**George Sobral Silveira,[1] Alisson V. Brito,[2]
Helder F. de A. Oliveira,[3] and Elmar U. K. Melcher[3]**

[1] *Department of Electrical Engineering, Federal University of Campina Grande (UFCG), 58429-140 Campina Grande, PB, Brazil*
[2] *Department of Informatics, Federal University of Paraiba (UFPB), 58051-900 Joao Pessoa, PB, Brazil*
[3] *Department of System and Computer, Federal University of Campina Grande (UFCG), 58429-140 Campina Grande, PB, Brazil*

Correspondence should be addressed to Alisson V. Brito, alissonbrito@dce.ufpb.br

Received 17 June 2012; Revised 19 November 2012; Accepted 10 December 2012

Academic Editor: Oliver Sander

Power gating is one of the most efficient power consumption reduction techniques. However, when applied in several different parts of a complex design, functional verification becomes a challenge. Lately, the verification process of this technique has been executed in a Register-Transfer Level (RTL) abstraction, based on the Common Power Format (CPF) and the Unified Power Format (UPF). The purpose of this paper is to present an OSCI SystemC simulator with support to the power gating design. This simulator is an alternative to assist the functional verification accomplishment of systems modeled in RTL. The possibility of controlling the retention and isolation of power gated functional block (PGFB) is presented in this work, turning the simulations more stable and accurate. Two case studies are presented to demonstrate the new features of that simulator.

1. Introduction

Due to the new requirements that the consumer market has been imposing, the semiconductors industry suffered modifications in its manufacturing process. These evolutions introduced great challenges to the design with even more complex chips and high density of transistors in a tiny silicon area, leading to an inevitable increase of power and consequently heat dissipation in the chips [1]. Ahead of this fact, the industry and academic research centers are searching for new techniques to ease the power density problem and enable the development of low power ICs.

Techniques for reducing the power consumption can be applied during the development of an IC from the design and specification system to the layout stage [1]. Among the main techniques that can be highlighted are clock gating, multi-Vth, power gating, voltage islands, logic restructuring, and dynamic voltage and frequency scaling (DVFS). These techniques can be combined together and require additional features such as power management controllers, cell isolation power domains, and/or retention registers for logical values [2–4].

It is common to use Transaction-Level Modeling (TLM) and Register-Transfer Level (RTL) to accomplish the functional verification of complex system on chip (SoC) [1]. Functional verification is a processes used in order to demonstrate that the objectives of the design are preserved after its implementation [5]. In accordance with the state of the art, the power gating verification process has been executed at the Register-Transfer Level (RTL) [6–9], primarily, based on Common Power Format (CPF) and Unified Power Format (UPF) [10–15]. The purpose of this work is to demonstrate a SystemC simulator, open source, with support to functional verification of designs containing the principles of the power gating technique implemented in SystemC RTL. Power gating consists of powering down internal modules of the SoC. The analysis of verification methodologies for low power design is not the focus of this work, but an overview of the methodology used in the case studies will be demonstrated.

In the examples presented in this work, the VeriSC methodology was used [16]. It is based on SystemC language and features to propose a verification flow that does not begin by Design Under Verification (DUV) implementation.

Instead, the testbench and reference model are implemented before the DUV. The SystemC simulator is independent of VeriSC and of all other verification methodologies. In this simulator version, SystemC TLM is only used to implement the testbench and DUV is implemented using SystemC RTL.

The SystemC simulator was presented in [17]; in the occasion, some new features were added to the simulator allowing to switch any module on and off during simulation. In the present work, the possibility of controlling the retention and isolation of power gated functional block (PGFB) was added, turning the simulations to be more stable and more accurate in relation to the modeled hardware designs. This turned the modified SystemC simulator into an effective open-source tool to realize functional verification of low power designs.

The remainder of this work is organized as follows. Section 2 presents the current technologies for the functional verification of low power design; Section 3 shows a power gating overview; Section 4 shows the new functions added to the SystemC kernel; Section 5 shows two SystemC-LP simulator applications; Section 6 presents the results and analysis; and Section 7 presents the final considerations.

2. Technologies for the Functional Verification of Low Power Design

Several power saving techniques can be applied during the development of a SoC, from conception of the system's architectural design until prototyping. It is consensus that the techniques for low power SoC development, applied in initial phases of the conception flow, especially in the system design and architecture phase, possess a bigger impact factor in relation to techniques applied in the implementation phase [1].

The most efficient techniques for power savings are shutting down power or reducing the voltage level of some regions of the device, known as power domains [1]. In the first design generation, power-aware SoCs had few power domains, but recent designs have more than 20 domains, resulting in numerous power modes [18]. This leads to an exponential growth in the number of power-up and power-down transitions to be verified before silicon prototyping. Problems related to power can be extremely critical, resulting in the need to modify the SoC. Figure 1 shows a diagram expressing the impact ratio of the decisions, where 80% of the power savings are reached due to decisions taken before implementation [19].

Currently, five technologies are applied to the verification accomplishment: Power Definition Markup Language (PDLM) specification, power-aware simulation, verification of the powered structure, relationships power assertions, and formal analysis of the logic control of power. These technologies are essential components of an effective verification aiming at high reliability of low power design techniques application [18], as follows.

(i) PDML provides a way to specify the architecture of the powered design independently of the description in RTL. The PDML specification includes the powered connectivity, shutdown behavior control, and interaction between the different power domains.

(ii) Power-awareness simulation includes power management features, in which the RTL simulator reads and interprets the PDML, where the behavior in powerup and powerdown can be modeled.

(iii) The verification of the powered structures can be performed at the beginning of the architectural power modeling.

(iv) Some control functions and relationships with time specified in PDML can be automatically transformed into assertions using a standard format, such as the assertions in SystemVerilog or a proprietary specification language.

(v) Formal analysis can be used to find all faults related to the assertions, where in simulation a given set of tests does not exercise all important behaviors related to power.

Currently, low power techniques are commonly applied at RTL level due to available technologies. The advantage of RTL verification is that there exists in the market a vast verification tool chain possibility from RTL to final graphic database system (GDSII). Therefore, once a given design is verified at RTL level, subsequent design stages only need to incrementally cover the additional low-power issues that are specific to that particular design stage.

During implementation of RTL-to-GDSII flow, most projects of integrated circuits have successfully incorporated power management using techniques such as clock gating, power gating, multi-VDD. The implementation of these techniques are not fully automated and the evaluation of the tradeoff between them is not easy. Some of these power analysis and optimization are assisted by Electronic Design Automation (EDA) tools [20, 21].

3. Overview of Power Gating

The designers have several ways to manage power, some are easily implemented and others are complex with respect to the operation frequency or area. Power-gating strategy is based on adding mechanisms to turn off blocks within the SoCs that are not being used; the act of turning blocks off and on is performed in appropriate time to achieve power saving while minimizing performance impact [1].

When the event of turning off happens, power saving is not instantaneous due to internal capacitances and the nature of technology that is not ideal for power gating. The process of turning a block on requires some time that cannot be ignored by the system designer [1]. Figure 2 shows an example of the activity of a block with power gating implemented.

Differently of a block that is always active, a power-gate block is powered by a power-switching network that will supply VDD or VSS. The Complementary Metal-Oxide-Semiconductor (CMOS) switches are distributed within or

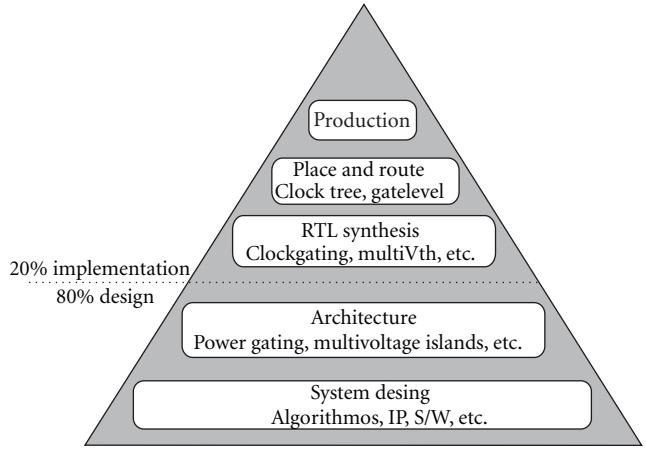


Figure 1: Impact factor of power savings [19].

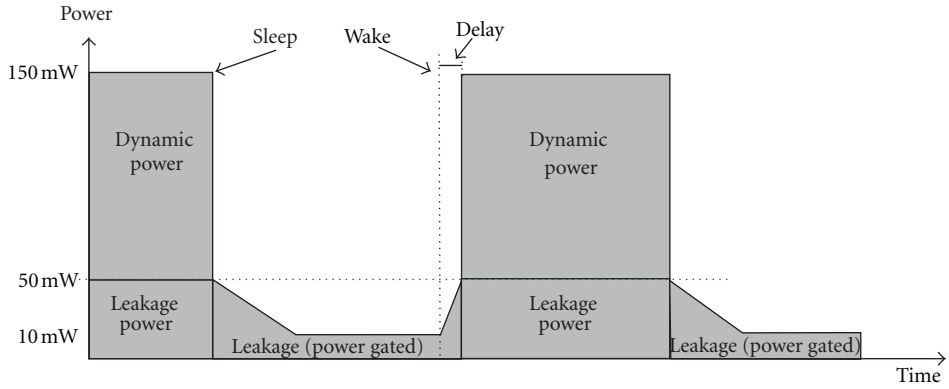


Figure 2: Profile with power gating, adapted [1].

around the block. The control of such switches is done by a power gating controller.

3.1. Signal Isolation. Each power domain represents a part of the chip's physical area, and even being independently switchable, these areas remain physically connected while they are turned on or off. Therefore, a logical isolation between the domains is necessary in order to avoid fluctuating signals and spread of undesired signals when the domain is off. The isolation is accomplished by logic gates which fix the input and/or output values when the power domain is turned off. In Figure 3, the isolation is represented by the block "isol."

3.2. State Retention. With the Power Gating Functional Block (PGFB) turned off, internal information about the state is lost. This may be inconvenience in certain applications. After the reactivation and restart of the PGFB activity, it should start from the initial state, reset, which can cause a significant consumption of energy and time. The retention can be accomplished in two ways: partial or total. There are several methods for saving and restoring the internal state of a PGFB, based on software, scan chains, and registers. Regardless the method, the goal is that the strategy adopted for retention of the state is fast and efficient, providing the PGFB with a method to quickly restart the full operation after activation [1, 22, 23].

4. SystemC-LP Simulator

The support to power gating design with SystemC was based on creating functions that assist the power gating technique in semantic description. These functions are based on an approach developed to simulate partial and dynamic reconfiguration [24, 25] that originated the first simulator prototype [17]. This is a bottom-up approach adding functions to activate and deactivate modules by the programmer, simulating the run-time reconfiguration.

Two new special functions were implemented to turn modules on and off during simulation named *sc_lp_turn_on* and *sc_lp_turn_off*, respectively. These functions were written modifying the SystemC kernel source code. Table 1 presents the functions signatures in *sc_simcontext.h* SystemC kernel file. The routine can be called by user code on regular simulations.

The simulator kernel was rewritten using a Hash Map that stores modules attributes and the turn-on delay, always present when a module is reactivated on chip. The module is

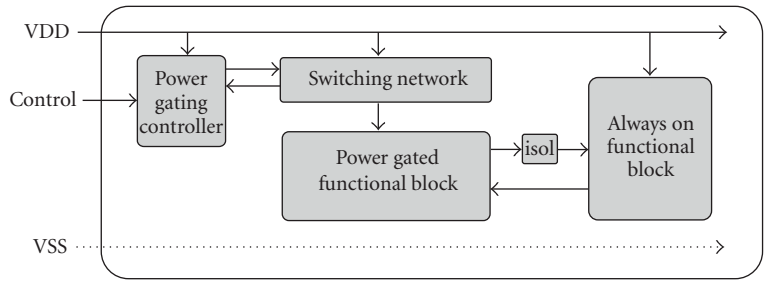


Figure 3: Block diagram of a SoC with power gating, adapted [1].

Table 1: Functions declarations.

void sc_lp_turn_on (const char module_name, sc_time wakedelay);*
void sc_lp_turn_off (const char module_name);*

Table 2: Functions signagtures of the PGLIB.

extern void pglib_int (const char name, void* var, int value);*
extern void pglib_int_set (const char name);*
extern void pglib_sc_lv (const char name, void* var, sc_lv<64> value);*
extern void pglib_sc_lv_set (const char name);*

described in SystemC RTL using SC_METHOD. Each Hash Map element represents a design module and is composed of a data structure containing two variables (bool and sc_time). The boolean variable is responsible for identifying whether the module is on or not, and the variable sc_time is responsible for storing the delay necessary to reactivate the module. The elements are accessed using a key, which is the name of the module.

As presented in Section 3, the retention can be partial or total. It is desirable for the signal isolation cell that its output may change to one of the following states: (i) the output keeps the value before deactivation; (ii) the output is equal to one of the logic states "0," "1," "*Z*" or "*X*." Thus, these features of signal retention and isolation were incorporated to the simulator.

Considering the signal retention, the default configuration stores the variable values, performing a total retention. However, for the output isolation the default is to keep the last value before deactivation. A library named Power Gating-Library (PGLIB) was developed to provide both, partial retention and output isolation, which must be added to the project source code to be used. These functions were not available in the first version of the simulator, shown [17]. The signatures of these functions are presented in Table 2.

In Table 2, two data types are used, Integer (*int*) and SystemC Logic Vector (*sc_lv*). For each data type, two functions were created. The first one is to declare the variables to be accessed (e.g., *pglib_int* and *pglib_sc_lv*) and the second one to set the values of those variables (e.g., *pglib_int_set* and *pglib_sc_lv_set*). The pointers to the variables and their values are stored in two Hash Maps, which use the name of the variables as keys to access their values.

The calculation of the power consumption is not the focus of this work and, therefore, was not implemented in the simulator.

5. Application of the SystemC-LP Simulator

To demonstrate the SystemC-Low Power (SC-LP) simulator, the functional verification of designs of [17, 26] is presented in this section.

5.1. Design 1. It consists of a circuit that serially receive words of 4 bits in the BCD format and convert to 7 segments format; after encoding, the data are available in the output [17]. This design has 450 code line and was implemented in RTL using SystemC version 2.2.0 with the features for power gating design. Figure 4 shows the design block diagram.

Two versions of the design were implemented in RTL using SystemC version 2.2.0, one with and another without the features for low power design.

(i) *Design Version 1 (DV1.1)*, without low power design: The implementation was done in a simple way. The Serial2Parallel and BCD2Segment7 converters were implemented, the output of Serial2Parallel converter is connected directly to the BCD2Segment7 converter. The operation of the circuit is as follows: while the Serial2Parallel converter parallelize the bits that are read, the BCD2Segment7 converter encodes the values provided by the Serial2Parallel converter; after reading a complete word of 4 bits by Serial2Parallel converter, this module provides the result in the output.

(ii) *Design Version 2 (DV1.2)*, with low power design: The implementation is similar to the preceding, differing in the addition of power gating controller (PGC) functions. The operation of this version differs from the previous in the following point: while the Serial2Parallel converter reads the word of 4 bits, the BCD2Segment7 module is turned off with its state retained. After reading the complete word BCD2Segment7 module is turned on, which from this point waits a time of 150 ns corresponding to the delay between wake up and activity; after this time the module executes its function and on the next step provides the value in the output. With the assistance of PGLIB, there is no retention values of the internal

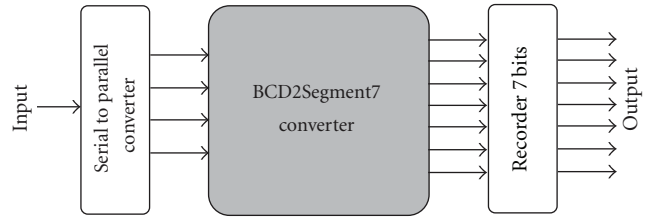


Figure 4: Design block diagram.

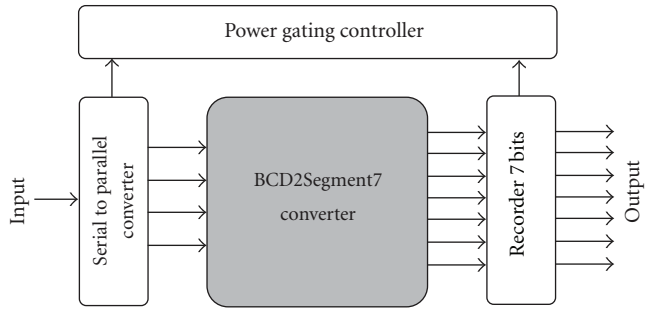


Figure 5: Design block diagram with PGC.

variables BCD2Segment7 module. When the module is disabled, all variables receive the value zero. The module output is isolated and its value is retained. Figure 5 shows the design block diagram with PGC.

5.2. Design 2. It consists of MPEG-4 decoder [26]; the IP core is a Simple Profile Level 0 movie decoder with approximately 21.000 line code and was implemented in a silicon chip, with 22.7 mm^2 at a 0.35 μm CMOS 4 ML technology with a 25 MHz working frequency. The block schematic of the MPEG-4 decoder is described in Figure 6.

The decoder circuit is divided into 10 modules, grouped into 4 functional groups, comprising bitstream demultiplexing, motion compensation, texture decoding, and image composition. The power gating technique was applied to the Inverse Discrete Cosine Transform (IDCT) submodule. This submodule was chosen because it presents inactivity time periods during its operation. The block schematic of the MPEG-4 decoder with PGC is described in Figure 7.

The PGC submodule has been added, described in SystemC RTL, with the function of calling the new SystemC-LP. The PGC receives information from Inverse Quantization (IQ) and Summing (SUM) submodules using wires. The PGC operates in the following way: it activates the IDCT whenever the IQ has information to be sent to the IDCT and the SUM is able to receive information from the IDCT. Otherwise, the later remains off. Using PGLIB, when the IDCT module is disabled only some configuration values of IDCT module are stored, the other internal variables receive the value zero. The module output is isolated and its value is retained.

6. Functional Verification

The VeriSC methodology supports projects with hierarchy concept; therefore, a project can be divided into parts to be implemented and verified [16]. The methodology consists of a verification flow, which is not initiated by the implementation of DUV. In this flow, the testbench implementation and the reference model precede the DUV. To allow the implementation of testbench before the DUV, the methodology has a mechanism to simulate the presence of DUV elements with their own testbench, without needing additional code generation. The testbench structure is described in Figure 8. The Reference Model, Source, and Check are described in ESL level, using SystemC TLM.

The source block is responsible for generating the stimuli. They are sent to DUV and Reference Model. The check block is responsible for checking whether the responses are the same. The TDriver converts transactions into signals, while the TMonitor converts signals into transactions. The reference model calculates all expected outputs of the system based on inputs that are used during functional verification. Ideally, separate engineers' teams must implement the DUV and the reference model. The goal is to decrease the probability of erroneous interpretations of the system specification propagate to DUV and verification environment.

BVE-Cover library was chosen to accomplish the functional verification with coverage of the design. Several simulations were performed with different versions of SystemC simulator and design.

(1) SC + DV1.1: At this stage the original SystemC version 2.2.0 and the first design implementation, were used.

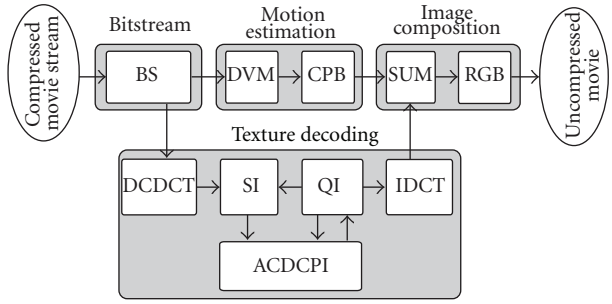


Figure 6: MPEG-4 decoder schematic.

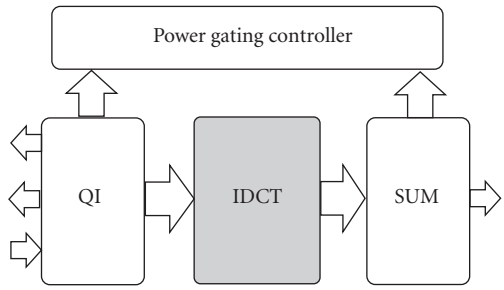


Figure 7: IDCT and PGC submodule connections.

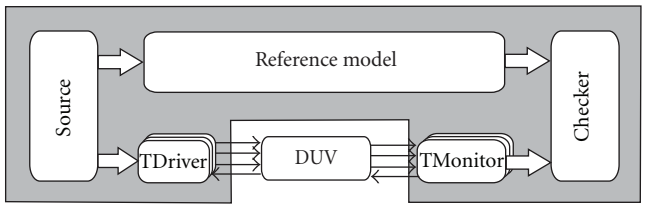


Figure 8: Testbench structure.

(2) SC-LP + DV1.1: At this stage the SystemC-LP and the first design implementation, without power gating design, were used.

(3) SC-LP + DV1.2: At this stage the SystemC-LP and the second design implementation containing the power gating design were used.

(4) SC + DV2.1: At this stage the original SystemC version 2.2.0 and MPEG-4 decoder were used.

(5) SC-LP + DV2.1: At this used the SystemC-LP and MPEG-4 decoder were used.

(6) SC-LP + DV2.2: At this stage the SystemC-LP and MPEG-4 decoder containing Power Gating technique were used.

7. Results and Analysis

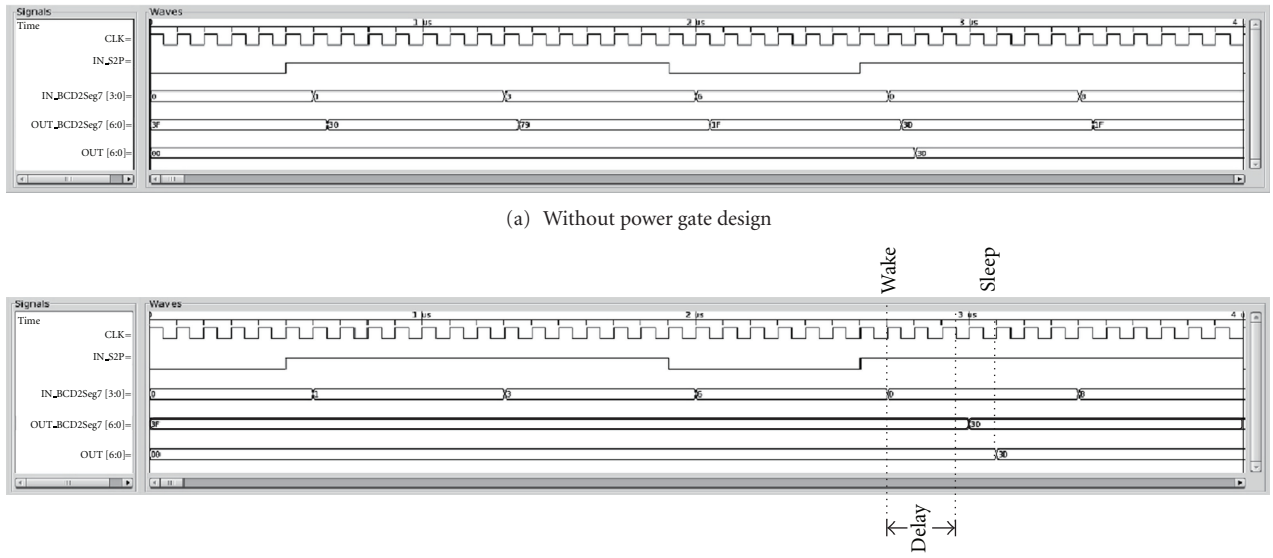
With the simulations results, it is possible to see the semantics of power gating design and that new features added to the simulator do not interfere with the other functions. The principles of power gating can be verified in Figure 9, which presents the waveform of simulations DV1.1. Figure 9(a) shows the changes of state of the BCD2Segment7 module while the Serial2Parallel module parallels the words.

In Figure 9(b), it is possible to see that while the parallels occur, the word of the BCD2Segment7 module maintains constant output state due to the isolation and retention that occurs prior to shut down. After the word reading by the BCD2Segment7 module plus the time delay, it turns back to activity and executes its functions to provide the values in the output.

Relevant information which can be extracted from the log is the 10 μs time that the submodule BCD2segment7 remains activated, which corresponds to 37% of the total time. For the IDCT submodule, it only remained activated for 41% of the total time. This value can be used as a parameter for estimating the power savings using the power gating technique.

About the simulator performance, satisfactory results were achieved. Table 3 shows values with the averages of the simulations times. It can be seen that the design simulations, using the SC-LP simulator presented a decrease of approximately 7,33% in simulation time, which shows that, even with the changes in the kernel simulator, this has performance similar with low and high complexity design.

The simulations of power gating design DV2.2 presented a decrease of approximately 6,12% in comparison with the original SystemC simulator using version 2.2.0. Comparing the simulation 5 with 6, it is possible to observe that the simulation 6 was faster than 5. It happens because in simulation 6 the IDCT module is not executed when it is deactivated. Based on that, it is possible to say that the power gating simulator can have an equivalent performance to the original SystemC simulator using RTL, depending on how the features are applied.



(a) Without power gate design

(b) With power gate design

Figure 9: Waveforms.

Table 3: Simulators performance.

Number	Simulator	Design	Time (s)
4	SC	DV2.1	466,403
5	SC-LP	DV2.1	503,315
6	SC-LP	DV2.2	496,851

8. Final Considerations

The research academic centers and industry have expended a great effort into finding alternatives to saving power in chips. This work shows an open-source simulator with technical support of power gating design. The SC-LP is an alternative tool for functional verification of the power gating design technique modeled in RTL. This presented a satisfactory performance in relation to SystemC simulator version 2.2.0 during the accomplishing power gating design simulations. In a preliminary analysis, although the main modifications in SystemC simulator version 2.3, the benefits presented in this work are still possible to be used, as the SystemC has support to block just models that use SC_THREAD, while our solution work mainly with SC_METHOD, which is the basis for RTL models.

Acknowledgment

This work is supported by "Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES)" by a postgraduate research scholarship.

References

[1] M. Keating, D. Flynn, R. Aitken, A. Gibbons, and K. Shi, *Low Power Methodology Manual: For System-on-Chip Design*, Springer, New York, NY, USA, 2007.

[2] S.-H. Chen and J. Y. Lin, "Experiences of low power design implementation and verification," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC '08)*, pp. 742–747, Seoul, South Korea, March 2008.

[3] S. Jadcherla, J. Bergeron, Y. Inoue, and D. Flynn, *Verification Methodology Manual for Low Power*, Synopsys, 2009.

[4] S. Henzler, *Power Management of Digital Circuits in Deep Sub-Micron CMOS Technologies*, Springer, 2007.

[5] J. Bergeron, *Writing Testbenches Using System Verilog*, Springer, Boston, Mass, USA, 2006.

[6] C. Eisner, A. Nahir, and K. Yorav, "Functional verification of power gated designs by compositional reasoning," *Formal Methods in System Design*, vol. 35, no. 1, pp. 40–55, 2009.

[7] V. Viswanath, S. Vasudevan, and J. A. Abraham, "Dedicated rewriting: automatic verification of low power transformations in RTL," in *Proceedings of the 22nd International Conference on VLSI Design (VLSI '09)*, pp. 77–82, New Delhi, India, January 2009.

[8] J. Kim, S. Kim, and K. H. Baek, "A low power SOC architecture for the V2.0+EDR bluetooth using a unified verification platform," *IEICE Transactions on Information and Systems*, vol. E93-D, no. 9, pp. 2500–2508, 2010.

[9] M. A. Pasha, S. Derrien, and O. Sentieys, "System level synthesis for ultra low-power wireless sensor nodes," in *Proceedings of the 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools (DSD '10)*, pp. 493–500, Lille, France, September 2010.

[10] T.-W. Hsieh, P. C. Hsiao, C. Y. Liao et al., "Energy-effective design & implementation of an embedded VLIW DSP," in *Proceedings of the International SoC Design Conference (ISOCC '08)*, pp. I252–I255, Busan, South Korea, November 2008.

[11] N. Joseph, S. Sabarinath, and K. Sankarapandiammal, "FPGA based implementation of high performance architectural level low power 32-bit RISC core," in *Proceedings of the International Conference on Advances in Recent Technologies in Communication and Computing (ARTCom '09)*, pp. 53–57, Kottayam, India, October 2009.

[12] S.-H. Chen and J. Y. Lin, "Implementation and verification practices of DVFS and power gating," in *Proceedings of the International Symposium on VLSI Design, Automation and Test (VLSI-DAT '09)*, pp. 19–22, Hsinchu, Taiwan, April 2009.

[13] G. Gammie, A. Wang, H. Mair et al., "Smart reflex power and performance management technologies for 90 nm, 65 nm, and 45 nm mobile application processors," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 144–159, 2010.

[14] M. Kuwahara, "Design and verification methods of Toshiba's wireless LAN baseband SoC," in *Proceedings of the 15th Asia and South Pacific Design Automation Conference (ASP-DAC '10)*, pp. 457–463, Taipei, Taiwan, January 2010.

[15] C. Trummer, C. M. Kirchsteiger, C. Steger, R. Weiß, M. Pistauer, and D. Dalton, "Automated simulation-based verification of power requirements for systems-on-chips," in *Proceedings of the 13th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS '10)*, pp. 8–11, Vienna, Austria, April 2010.

[16] K. R. G. da Silva, E. U. K. Melcher, I. Maia, and H. D. N. Cunha, "A methodology aimed at better integration of functional verification and RTL design," *Design Automation for Embedded Systems*, vol. 10, no. 4, pp. 285–298, 2005.

[17] G. S. Silveira, A. V. Brito, and E. U. K. Melcher, "Functional verification of power gate design in systemc RTL," in *Proceedings of the 22nd Symposium on Integrated Circuits and Systems Design (SBCCI '09)*, Natal, Brazil, September 2009.

[18] T. Anderson, "Power Mode Technologies Verify Today's SoCs," EETimes, 2008, http://www.eetimes.com/showArticle.jhtml?articleID=206900466.

[19] A. Mathur, *Sequential Transformations for Low Power and CPF*, 2008.

[20] Mentor Graphic's, "Mentor Graphics Low Power Solutions," Mentor Graphic's, 2012, http://www.mentor.com/solutions/low-power/functional-verification.

[21] Cadence, "Cadence—Low-Power Solution," Cadence, 2012, http://www.cadence.com/solutions/lp/pages/default.aspx.

[22] D.-C. Juan, Y. T. Chen, M. C. Lee, and S. C. Chang, "An efficient wake-up strategy considering spurious glitches phenomenon for power gating designs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 2, pp. 246–255, 2010.

[23] M.-C. Lee, Y. T. Chen, Y. T. Cheng, and S. C. Chang, "An efficient wakeup scheduling considering resource constraint for sensor-based power gating designs," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD '09)*, pp. 457–460, San Jose, Calif, USA, November 2009.

[24] A. V. Brito, M. Kühnle, M. Hübner, J. Becker, and E. U. K. Melcher, "Modelling and simulation of dynamic and partially reconfigurable systems using SystemC," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI: Emerging VLSI Technologies and Architectures (ISVLSI '07)*, pp. 35–40, Porto Alegre, Brazil, March 2007.

[25] A. V. De Brito, E. U. K. Melcher, and W. Rosas, "An open-source tool for simulation of partially reconfigurable systems using SystemC," in *Proceedings of the IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures (ISVLSI '06)*, pp. 434–435, Karlsruhe, Germany, March 2006.

[26] A. Rocha, P. Lira, Y. Ju, E. Barros, and E. Melcher, "Silicon validated ip cores designed by the brazil-ip network," in *Proceedings of the IP Based SoC Design Conference and Exhibition (IP/SOC '06)*, pp. 1–5, 2006.

Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2012, Article ID 729786, 13 pages
doi:10.1155/2012/729786

Research Article

Efficient Execution of Networked MPSoC Models by Exploiting Multiple Platform Levels

Christoph Roth, Joachim Meyer, Michael Rückauer, Oliver Sander, and Jürgen Becker

Karlsruhe Institute of Technology (KIT), Institute for Information Processing Technology (ITIV), 76131 Karlsruhe, Germany

Correspondence should be addressed to Christoph Roth, christoph.roth@kit.edu

Received 21 February 2012; Revised 3 June 2012; Accepted 21 July 2012

Academic Editor: Massimo Conti

Novel embedded applications are characterized by increasing requirements on processing performance as well as the demand for communication between several or many devices. Networked Multiprocessor System-on-Chips (MPSoCs) are a possible solution to cope with this increasing complexity. Such systems require a detailed exploration on both architectures and system design. An approach that allows investigating interdependencies between system and network domain is the cooperative execution of system design tools with a network simulator. Within previous work, synchronization mechanisms have been developed for parallel system simulation and system/network co-simulation using the high level architecture (HLA). Within this contribution, a methodology is presented that extends previous work with further building blocks towards a construction kit for system/network co-simulation. The methodology facilitates flexible assembly of components and adaptation to the specific needs of use cases in terms of performance and accuracy. Underlying concepts and made extensions are discussed in detail. Benefits are substantiated by means of various benchmarks.

1. Introduction

Today, two major trends can be observed in the embedded domain. (1) Multiprocessor System-on-Chips (MPSoCs) become more and more popular for embedded systems since functionality is evermore integrated directly into a single device. Besides that, this trend towards multicore is driven by the need to optimize performance per watt which results in an increase of parallelism instead of the clock frequency. (2) Applications of embedded systems evermore rely on communication between several or many devices. When considering, for example, the vision of the so-called Cyber-Physical Systems (CPS), this trend will continue or rather become much more intense in future [1]. CPS are characterized by their high adaptability and the capability of very tight coupled interaction among each other as well as the environment. In many imaginable scenarios, such networked embedded devices must meet hard requirements like high performance and low power consumption due to their autonomy in the field of application.

Due to the high degree of parallelism provided by networked MPSoCs system design, verification and validation become increasingly complex. Usually, system development starts on higher abstraction levels. The system is then iteratively refined until reaching the register transfer level (RTL) implementation. During each refinement step, often simulation makes a significant contribution to verify and validate the current system architecture. With increasing model fidelity such simulations can become an extremely time-consuming task.

We believe, simulation of multiple networked devices on multiple abstraction levels is essential for a comprehensive evaluation of networked MPSoCs since (1) the dynamic impact of network communication on alternative system configurations and vice versa needs to be considered, (2) a network-wide optimization of single nodes regarding size, power consumption, performance, and so forth demands for detailed simulations of several nodes in parallel, (3) a step by step refinement process including reuse of intellectual property (IP) cores directly induces simulation at various

abstraction levels, and (4) verification and validation of the system with respect to possible network and traffic conditions is important through all steps of system design, even in late phases, when the system model approaches the final implementation. In order to meet these requirements, new simulation tools and methodologies are necessary which support the process of system design by flexibly combining advantages of both system and network domain and by having special regard on simulation performance at the same time.

In this work, a methodology is presented which facilitates efficient execution of system/network co-simulation on different abstraction levels. Main part of the methodology is a construction-kit-like simulation platform that is divided into several platform levels. This approach allows flexibly interconnecting system domain tools (e.g., OSCI SystemC kernel or Modelsim [2]) with network domain tools (e.g., OMNET++ [3] or ns-3 [4]) and exploiting different hardware components like workstations and FPGA-based rapid prototyping systems for efficient co-simulation and co-emulation. In previous work, parallel simulation of a single MPSoC [5] as well as synchronization algorithms for system/network co-simulation [6] has been considered for applicability with a high-level-architecture- (HLA-) based simulation backbone. In contrast, key contributions of this work are as follows.

(i) Introduction of the co-simulation methodology.

(ii) Embedding of a realistic MPSoC model called HeMPS [7] into the methodology by extending it with wireless networking capabilities through a virtual network interface.

(iii) Extension of the platform of [6] towards a multilevel construction kit by equipping it with a co-emulation interface for fast execution of detailed RTL-based system models.

(iv) Benchmarking and functional verification of the simulation platform by means of the newly integrated HeMPS system using different abstraction levels.

The remainder of this paper is organized as follows. Section 2 summarizes fundamentals and presents related works in the fields of parallel discrete event simulation and network/system modeling and design as well as combined approaches. The proposed simulation methodology is presented in Section 3. Extensions on targeted simulation models as well as the implementation of the simulation platform are described in Sections 4 and 5. In Section 6 performance of different platform configurations is evaluated followed by a demonstration of applicability on a realistic scenario in Section 7. Finally, in Section 8 the work is concluded and an outlook is given to possible further work.

2. Fundamentals and Related Work

2.1. Parallel and Distributed Discrete Event Simulation. Parallel and distributed discrete event simulation (PDES and DDES) is a research topic for more than three decades now. It denotes a technology that enables execution of discrete event simulations (DES) on parallel and/or distributed platforms. Major objectives are the following [8]:

(i) reduction of execution time,

(ii) enabling larger simulations by incorporating distributed resources,

(iii) reuse in terms of integration of possibly spatially distributed simulators of different types.

An integral part of PDES research deals with the so-called *synchronization problem* [9] since the chosen synchronization mechanism in combination with the underlying hw/sw platform fundamentally influences performance. Basically, a PDES is made up of several *logical processes* (LP) that maintain separate event queues. Within each LP time progress is controlled by a scheduler that always selects the smallest time stamp event for execution. Whenever causal dependence between local events in the queue and events generated by other LPs cannot be excluded, the LPs must also consider these remote events during event scheduling. Various algorithms have been proposed for this issue in the literature. They can be divided into *conservative* and *optimistic* approaches. In short, conservative synchronization algorithms avoid violating the causality relationships between LPs by always guaranteeing event delivery and execution in the correct time order. The approaches in [10–12] are examples of conservative algorithms. Also the method we proposed in [6] for system/network co-simulation is a conservative approach. In contrast, optimistic approaches allow violating the causality relationships but provide for mechanisms to restore already past points in time (e.g., [13] or [14]).

Simulator interoperability is another major research topic that directly derives from the afore-mentioned "reuse" objective. Up to now, standards like HLA [15] have mainly addressed the syntatic interoperability between different simulation systems. One major reason why PDES/DDES is still not really established for common use is the lack of adequate tools that automate the process of configuring and setting up a distributed simulation for efficient execution. In this context, Strassburger et al. mention in [16] as one of the main research challenges and compelling problems in the field of distributed simulation systems the need of a technical approach for *true plug-and-play capability* between simulation packages of different domains.

2.2. Network Modeling and Design. For modeling and design of computer networks, there exists a wide range of so-called network simulation tools. Well-known examples are OMNET++ [3], ns-2 [17], ns-3 [4], or JiST [18]. These tools generally facilitate specification and simulation of the behavior of complete protocol stacks referring to the ISO/OSI reference model. A network model typically consists of several or many devices being interconnected by virtual wired or wireless links. Each device implements the functional description of the targeted network protocol including protocol state machines, packet structure, and data manipulations. Beyond that, possible environmental influences

on communication behaviour like mobility or obstacles can also be defined. During simulation execution, device models interact with each other through virtual network channels. Typical observable parameters are network traffic, transmission latencies, packet loss, or link failures. The so-called *network emulation* introduced by Fall [19] tries to increase realism and precision of generated network traffic by integrating physical systems with a network simulation. Since the approach of Fall demands for real-time execution, recent work [20] introduced an approach for synchronized network emulation using virtual machines.

Generally, the main drawback of network simulation tools is their ineptitude for detailed system modeling, since they do not reproduce behavior and interaction of underlying hardware and software components accurately.

2.3. Simulation Platforms for Embedded Systems. Today, simulation plays a major role during both design space exploration (DSE) of embedded systems as well as software development for such systems. OVP [21] and GEM-5 [22] are two well-known simulation platforms in the embedded domain. OVP is optimized for very fast execution and targets the provision of virtual prototypes for software development. Due to the lack of enough accuracy, models provided by OVP are typically not suitable for a DSE. GEM-5 is a simulator for computer architecture research with which microarchitectural characteristics like pipelines, caches, or interconnects can be modeled and simulated, making it suitable for a DSE.

In a typical DSE process, a basic architecture platform consisting of hardware and software is assumed, that is, parameterizable to a certain extend. Starting from a given target application, the architecture platform is iteratively refined towards the final implementation. In the domain of MPSoCs, recent approaches employ multiaccuracy simulation models that allow choosing between faster or more accurate simulation. In this context, Moreno et al. [23] propose an MPSoC design flow that integrates NoC models of different accuracy. Indrusiak et al. [24] propose, a layered approach that allows obtaining accurate figures for communication latency from an abstract application model. In [25], a model-based methodology and supporting toolset is presented that lets designers estimate application-specific network on chip power dissipation at early stages of the design flow. Authors employ a unified model and define different application-mapping platform layers. Last but not least, an example of a simulation platform that can accelerate DSE by exploiting parallelism provided by SMP workstations is described in [26].

The presented frameworks and methodologies for DSE all target evaluation and analysis of embedded system architectures. However, they focus on optimization of single systems without considering interaction with an outer network.

2.4. Network Centric Development of Embedded Systems. Pioneering work in the area of system/network co-simulation has been done by Fummi et al., for example, in [27, 28], reporting about a co-simulation environment for SystemC and ns-2. In [28], the authors show by means of several application examples that using system/network co-simulation reduces the modeling effort of networked devices and allows easy design verification and validation. In [27], they focus on timing accurate integration and synchronization and describe necessary kernel extensions for a tightly simulator coupling by linking the kernels together. Recent work from the same authors [29] introduces a TLM-based methodology for system/network design-space exploration of networked embedded systems which extends the traditional transaction level modeling [30] refinement process with a new dimension to represent network configuration alternatives. They derive a general criterion to map functionalities to system and network models and apply the proposed methodology to the design of a Voice-over-IP client which is partially refined down to RTL level. An example for a co-simulation framework based on a loosely coupling of SystemC and OMNET++ is given in [31] which is motivated by the evaluation of automotive IT architectures and the fact that in the automotive industries predominantly so-called "rest bus" simulations are state of the art which rely on real working hardware, and therefore, can only take place in late design phases. Finally, the approaches in [32, 33] examples for co-simulation platforms in the domain of wireless sensor networks. The former focusses on the evaluation of hardware/network interactions in WSNs in general, whereas the approaches in [33] focusses on investigating the application of reconfigurable hardware in the WSN context. Thereby, the authors explicitly model on cycle accurate level in order to reproduce hardware behaviour in detail.

Except this work, there exists no research paper targeting a system/network co-simulation environment that can be specialized towards different use cases and allows integration of simulation tools in a construction kit-like manner. Furthermore, none of the related work supports co-simulation, co-emulation, and parallel/distributed execution at the same time.

3. Simulation Methodology

The proposed simulation methodology describes the way of proceeding for a network centric evaluation of embedded MPSoCs using various abstraction levels. It is illustrated in Figure 1.

The methodology is composed of three steps.

(1) *Scenario Definition.* A *scenario* is defined for which a system analysis by means of simulation should be performed. The scenario definition is given by a specification of the communication protocol including the target application and the environmental context (e.g., node position, distance, mobility, etc.) in which the application will be executed. The scenario definition is done in the network domain using a respective network simulation tool, since such tools inherently provide support for defining a scenario in a formal way using a specification file.

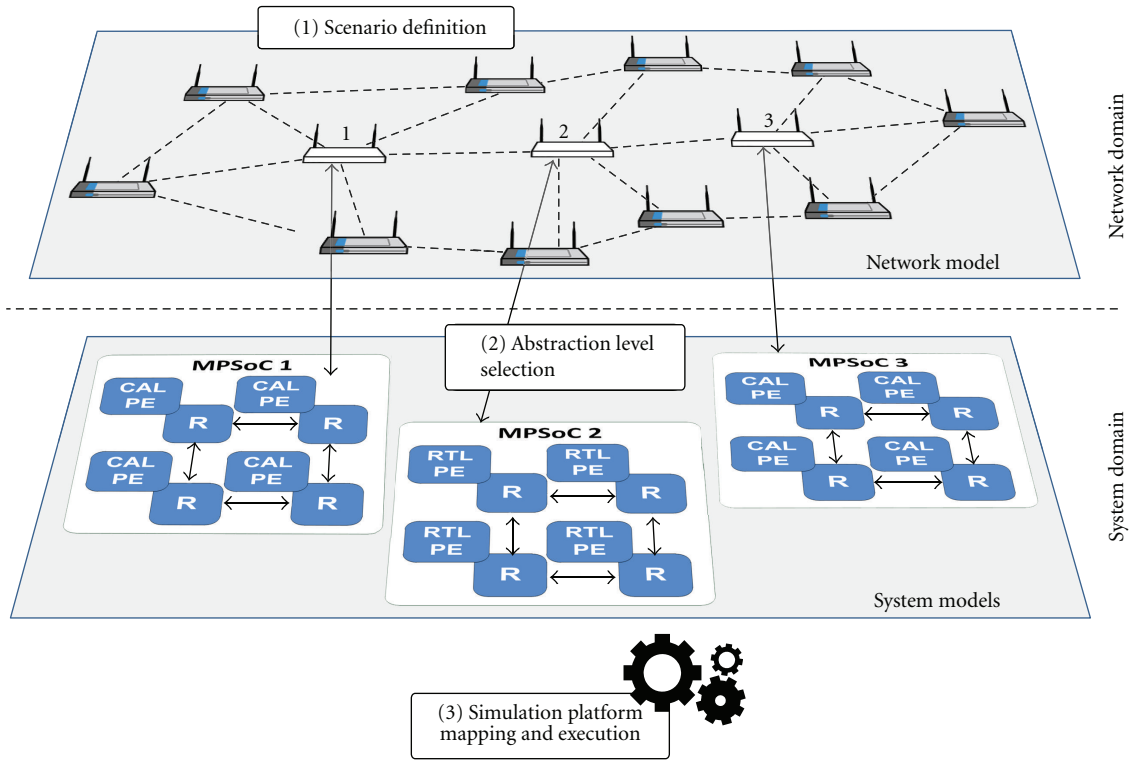


Figure 1: Simulation methodology.

(2) *Abstraction Level Selection.* A *cross-domain simulation model* (CSM) is generated that links the network domain to the system domain. Therefore, it is defined, which of the nodes of the scenario definition remain an abstract artifact in the network domain model and which of them are modeled in more detail. In the latter case, an abstract artifact in the network domain needs to be linked with an associated detailed artifact in the system domain (*system domain model*). In Figure 1, only cycle accurate and register transfer level are illustrated. Other abstraction levels or even mixed abstraction levels within a single node are also imaginable. In [6], the foundation for dividing a node into network and system domain artifacts is described. From the communication protocol perspective nodes can be either *local nodes* or *remote nodes*. Local nodes implement the complete protocol stack of the network protocol that is simulated within the network domain. Remote nodes only implement lower protocol layers within the network domain. Upper protocol layers are implemented in the system domain. The position of the cut between protocol layers determines the data that needs to be exchanged between network and system domain artifacts of a remote node. The decision whether a node is extended to a remote node depends on the targeted design or verification *use case* (e.g., network-wide optimization of several node architectures or verification of a single node). Basically a nodes internal hardware/software interactions need to be reproduced in detail if there is the necessity of gaining information about hw/sw-system characteristics, for example, in terms of latency and throughput.

(3) *Simulation Platform Mapping and Execution.* The CSM is mapped and executed on an appropriate instance of the *Simulation Platform* (SP) (see Figure 2). The SP is made up of two *platform levels* (not to be confused with *abstraction levels* of step two) called *simulator interoperability level* (SIL) and *execution platform Level* (EPL). Each platform level is assigned a distinct task. The SIL provides the *simulation infrastructure* for the CSM in terms of interconnected and correctly synchronized domain specific simulators. The EPL is the *execution infrastructure* in terms of different workstations, rapid prototyping systems, and local area network (LAN). Different manifestations of the platform levels can be combined in a construction kit-like manner. A concrete manifestation again depends on the targeted *use case*, characteristics of the CSM like number of system domain models and execution performance requirements.

4. Target Cross-Domain Simulation Models

The targeted cross-domain simulation models consist of instances of the HeMPS MPSoC [7] in the system domain and a 802.11 wireless network model in the network domain.

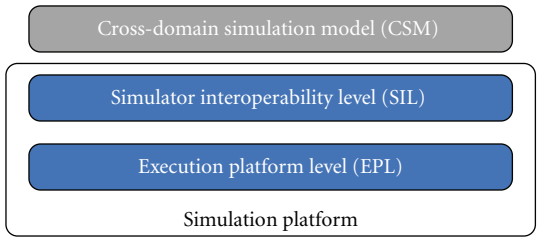


Figure 2: Simulation platform.

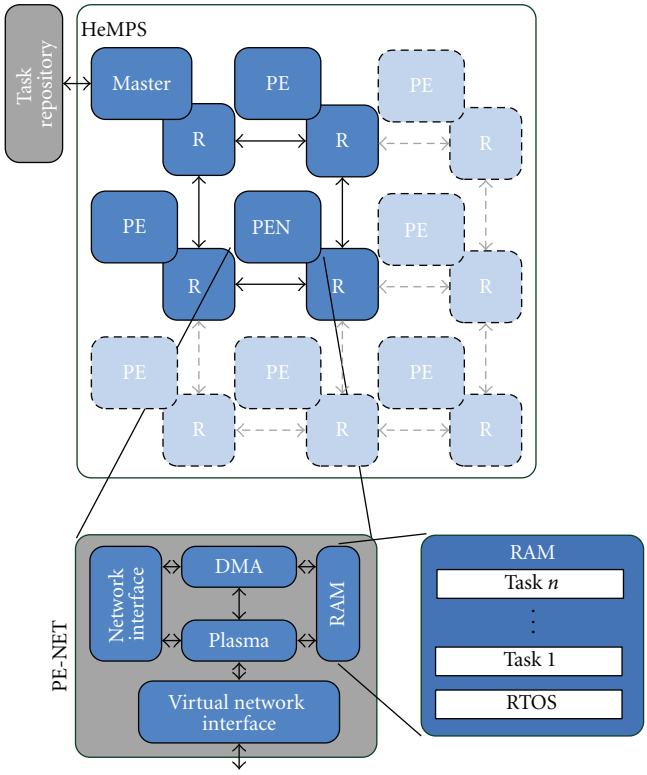


Figure 3: HeMPS with virtual network interface extension.

In the following, HeMPS itself as well as newly made interface extensions for integration of the model into a CSM are described. Besides that, details of interface extensions of the network model that have not been treated in previous work [6] are also shortly explained.

4.1. System Domain Model. HeMPS is a homogeneous MPSoC that consists of a configurable number of processing elements (PE) being interconnected by an NoC with mesh structure. It is completely available in SystemC and can be executed with the OSCI system kernel or modelsim. In contrast to the models used in previous work [6], HeMPS is much more realistic. Figure 3 gives an overview of the HeMPS architecture.

Main hardware components of HeMPS are the HERMES NoC [34] and the MIPS processor plasma [35]. The plasma cores execute a multithreaded real-time operating system (RTOS). Applications running on HeMPS are modeled using

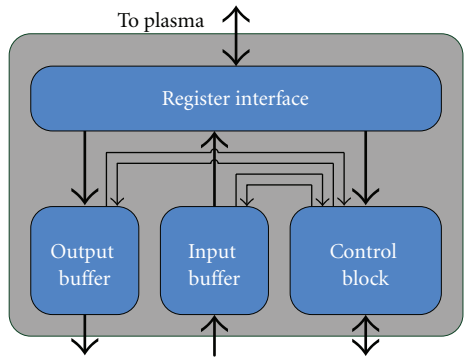


Figure 4: Virtual network interface.

task graphs. They are stored in an external memory named *task repository* and are allocated by a *master* to the remaining PEs (also called *slaves*). HeMPS has been chosen since it also fulfills the following requirements:

(i) scalability of the NoC size,

(ii) availability of RTL models and faster models on cycle accurate level (CAL) of the processing elements,

(iii) easy configurability (size, task mapping, routing strategy, etc.) and model generation,

(iv) HeMPS is open source.

The model has been extended with a virtual network interface (VNI) that can be connected to any of the PEs within the whole HeMPS array. The VNI is accessed from the SIL via a model adaptor (see Section 5.1) and forms the basis for data exchange with a network domain artifact on MAC layer. In Figure 4, the inner structure of the VNI is depicted. By means of a register interface, the peripheral is integrated into the address space of the plasma core. To exchange data with the SIL, input and output buffers are present. Both can hold 1024 bytes of data. Access to the buffers is directed by the control block which gathers the buffer states for generation of respective status signals and accepts read/write commands from the remaining model and the SIL.

4.2. Network Domain Model. Typically, nodes in network simulators are represented by hierarchically nested modules that communicate by passing messages to each other. Module nesting allows users to reflect the logical structure of network protocols. The modular approach of network simulation tools is exploited for instantiation of an invisible dummy module called VNI_Master which can be accessed by the model adaptor of the SIL. It is the counterpart to the VNI in the system domain. The VNI_Master acts as a kind of gateway and forwards network packets from the SIL to network domain artifacts (remote nodes). In turn, it also forwards network packets from network domain artifacts to the SIL. The approach of using a dummy module can be applied in any available network simulator since basic concepts are equivalent. However, due to its good extensibility, OMNET++ together with the MiXiM framework has been used.

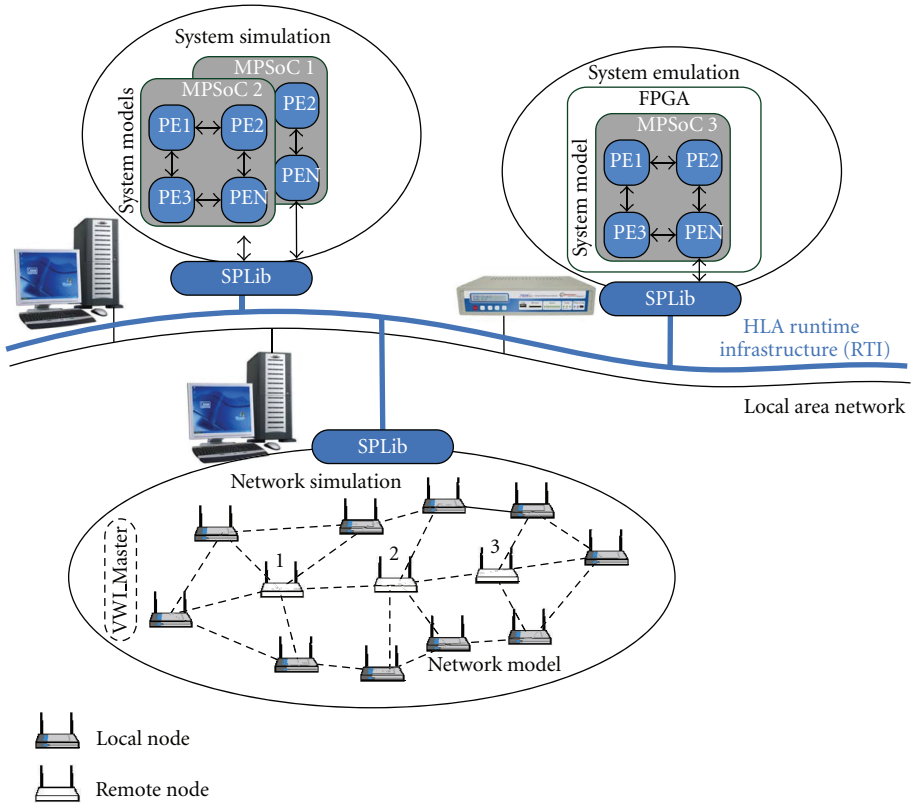


Figure 5: Simulation model and platform instance.

5. Simulation Platform

The construction kit-like design of the simulation platform facilitates

(i) assembly of different CSM, SIL, and EPL components,

(ii) reuse of domain specific simulation tools and models,

(iii) adaptation to specific use cases.

Figure 5 provides an overall view of an instantiation of the simulation platform that is feasible with the currently available components on the EPL and the SIL. Regarding the EPL, execution of either network or system simulators is done on Linux workstations. Emulation of system models is done on FPGA-based rapid prototyping systems called *Hardware-Prototyping and Emulation System—Industrial Reference Platform* (IRP) [36]. Since co-emulation has not yet been considered in previous work [6], SIL and EPL are described with a special regard on emulation and the difference to simulation. For the sake of completeness, available synchronization mechanisms on the SIL, as they were already described in [6], are shortly depicted.

5.1. Simulation Interoperability Level. The implementation of the SIL is given by the *Simulation Platform Library* (SPLib). The SPLib encapsulates an HLA simulation backbone called CERTI [37]. The SPLib complements the HLA with domain specific interfaces that facilitate controlled data exchange between domain specific models. The link that is formed by the SPLib between HLA, simulators, emulators, and models is basically fixed but parameterizable within certain degrees of freedom, that is,

(i) type of network simulator,

(ii) number and type of system simulators,

(iii) number of system emulators,

(iv) number of models per system simulator or emulator,

(v) synchronization method used for controlling data flow between models.

The library structure is illustrated in Figure 6.
It consists of the following:

(i) different types of adaptors for model connection,

(ii) different types of controllers implementing different synchronization methods,

(iii) a database for configuring the library regarding different use cases,

(iv) ambassadors that access the HLA services.

A model adaptor interacts via function calls with the connected model. However, while the OMNET++ and SystemC model adaptors directly access the VNI_Master respectively, VNI (see Section 4), the functions of the emulator

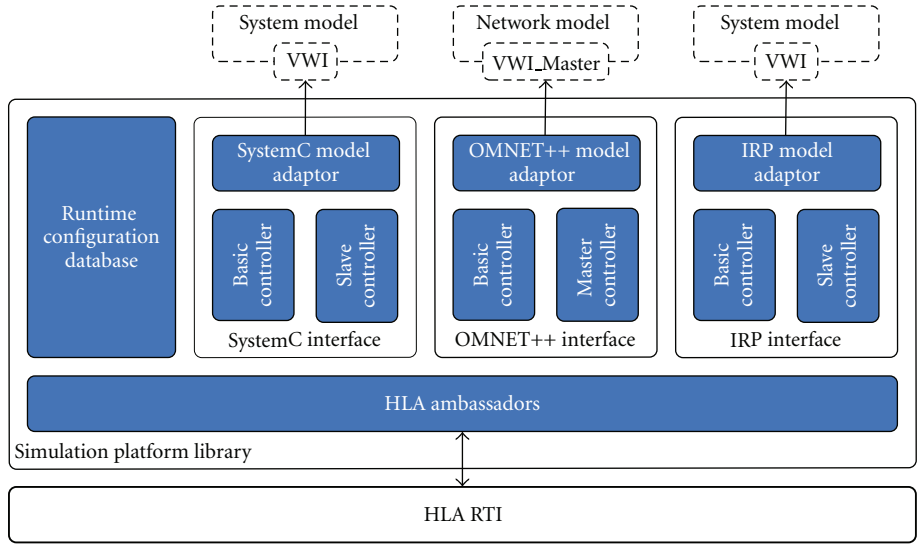


Figure 6: Simulation platform library.

adaptor need to be able to access the emulation hardware. As will be described in more detail in Section 5.2, the IRP emulation system consists of an Atom processor that connects to a FPGA via PCIe. The SPLib runs on the Atom while the model runs on the FPGA. Because of that, the adaptor functions have been equipped with routines that allow reading and writing from/to the PCIe bus by means of a PCIe driver and thus accessing the model indirectly.

Different synchronization methods are implemented by means of different controller instances. In case of SystemC or OMNET++ simulation, a controller implementation performs time advancement of the respective kernel by directly interacting with its scheduler. In contrast, in case of emulation, time advancement of the model that is running on the FPGA is controlled by accessing a special cycle register via PCIe that allows model execution for a configurable amount of cycles. Currently, basic *symmetric* as well as master/slave based *asymmetric* controllers are available for simulation and emulation. For the sake of completeness, their implementation is shortly summarized in the following.

5.1.1. Symmetric Conservative Synchronization. Symmetric conservative synchronization is already inherently supported by the HLA. CERTI provides two conservative algorithms for this purpose, the *Null Message Algorithm* (NMA) [10] and the *Null Prime Message Algorithm* (NPMA) [11], an improved version of the NMA. Regardless of whether the NMA or the NPMA is configured in CERTI, the HLA API remains the same. As has been shown in our previous work [6], when relying on system models of a very fine time granularity of events, symmetric synchronization can become a strong bottleneck in co-simulation since it incorporates all events that are generated in the system and the network domain. Most of the generated synchronization cycles are superfluous since no causal dependencies between simulators are generated.

5.1.2. Asymmetric Conservative Synchronization. The asymmetric synchronization method was proposed in [6]. It takes advantage of communication delays between simulated devices as guarantee for model independence. Time advancement is exclusively controlled by the network simulator which acts as master. It specifies how far system simulators (slaves) are allowed to run ahead. Hence, the amount of synchronization no longer depends on the fine time granularity of cycle accurate system models, but on network characteristics like message rate and transmission delays being of much higher scale. The only disadvantage of the asymmetric approach is that "node internal" events are possibly delayed. Internal events are events that are transmitted from one domain to another and that effect an immediate event in the opposite direction. For instance, a parameter request from an upper protocol layer simulated in the system domain to lower protocol layers simulated in the network domain is such a internal event.

5.2. Execution Platform Level. The execution platform level (EPL) is the lower level of the simulation platform. In the current implementation, two types of components are available, workstations with Linux OS as well as the newly supported IRP-based emulation systems which can be used for direct FPGA-based execution of RTL system models without the need of a simulator within the SIL. Basically, the choice between system simulation and emulation depends on both, use case and performance requirements. Exploiting IRPs for emulation is much more performant compared to workstation-based simulation (see Section 6); however, emulation is only possible if the system model is present as synthesizable RTL description and its resource requirements do not exceed available resources on the FPGA. Besides that, it is basically limited to use cases in the context of functional verification and application development or in which realistic traffic needs to be generated efficiently

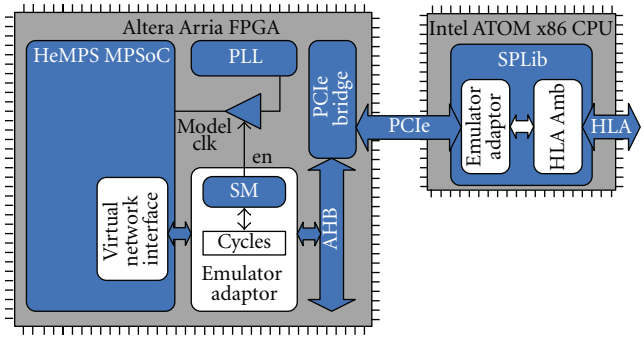


Figure 7: Emulator setup.

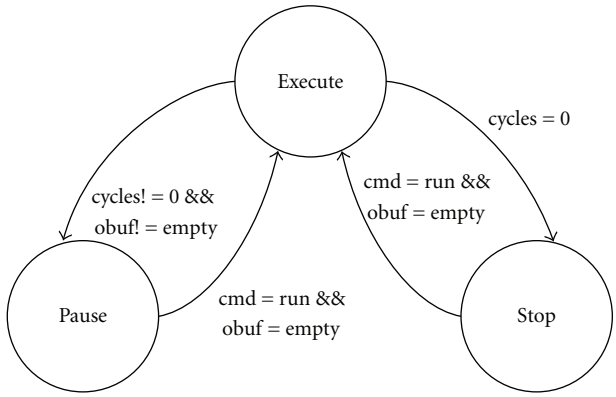


Figure 8: Hardware adaptor state machine.

(keyword *synchronized network emulation*). If the design use case demands, for example, for determination of the power consumption, then simulation on higher abstraction levels including a power estimation model is probably the best choice in order to do this.

The structure of the IRP together with the extensions towards emulation is illustrated in Figure 7. The IRP consists of a 1.6 GHz Intel Atom processor and an Altera Arria FPGA. Both are interconnected via a PCIe interface. The Atom runs the SPLib on top of a Gentoo Linux distribution and serves as mediator between FPGA, and HLA. On the FPGA the synthesized system model as well as some additional hardware components are executed. The model adaptor of the SPLib is complemented with a hardware counterpart that is connected to PCIe via AHB bus and accesses the model via the VNI.

In order to enable synchronized model execution and data exchange with the HLA, time advancement must be performed in a well-controlled manner which means regular start and automatic suspension of execution after a configurable number of clock cycles. For that reason, the hardware part of the model adaptor is equipped with a cycle register that allows specifying the number of clock cycles the model is allowed to run ahead. Activation and deactivation of the model clock is subject to the state machine illustrated in Figure 8.

During the *execute* state, the PLL clock is interconnected to the model and the cycle register is counted down with the processing frequency f_p until either reaching zero or a message is available in the output buffer. In the first case, the state machine switches to *stop* and waits for a new clock cycle value as well as a *run* command. In the second case, the output buffer must first be read out before the state machine can switch back to *execute*. The processing frequency f_p must not be confused with the model frequency f_m. The model frequency f_m expresses the "virtual" clock frequency of the system in relation to simulation time instead of the wall-clock time. When neglecting the communication and synchronization overhead between FPGA and Atom/HLA, the overall wallclock time $T(\Delta t)$ needed for executing a RTL model for a simulation time delta of Δt can be expressed as a relation of f_m and f_p times Δt

$$T(\Delta t) = \frac{f_m}{f_p} \times \Delta t. \quad (1)$$

The maximum value of f_p depends on factors like FPGA type, synthesis tool, and size of the model. Its value typically ranges in the order of several tens/hundreds of MHz.

Table 1: Variable benchmark parameters.

Parameter	Location
HeMPS abstraction level	CSM
Clock frequency	CSM
NoC size	CSM
Remote node fraction	CSM/SIL
Synchronization method	SIL
Execution platform	EPL/SIL
Workstation cores	EPL

6. Performance Evaluation

In the following, performance characteristics of the platform are evaluated. While focus in previous work [6] was on the comparison of different synchronization methods, the following evaluations target a more comprehensive consideration and evaluation of performance benefits of the overall simulation platform when using different

(i) abstraction levels on the CSM,

(ii) configurations on the SIL and EPL.

Hence, advantages of the construction kit-like design are emphasized. Starting point is a wireless communication scenario. From this, three different benchmarks are derived. Each benchmark covers one or several realistic use cases. Adjustable parameters that are applied in the benchmarks are summarized in Table 1.

6.1. Scenario Specification. Within the considered scenario, 64 nodes communicate via a 11 Mbps 802.11b wireless channel. They are arranged in a mesh structure. Neighbouring nodes have a distance of 20 m to each other. Each node transmits broadcast packets with a frequency of 100 Hz. Pure local nodes implement the broadcast application by means of a simple traffic generator that is triggered every millisecond.

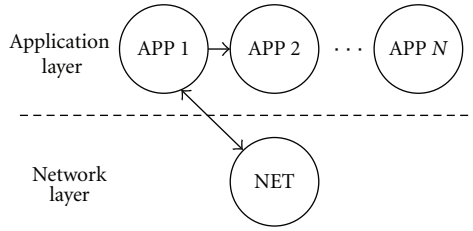


Figure 9: HeMPS software structure.

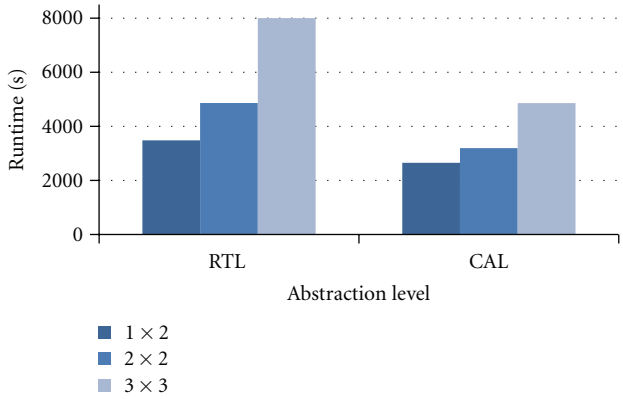


Figure 10: Results abstraction benchmark.

6.2. Additional Requirements for the CSM. Since the cut between network and system domain artifacts is done on MAC layer, the HeMPS systems need to implement application and network layer. Therefore, the MPSoCs run a network layer task that is regularly triggered by an application task pipeline in order to transmit the broadcast packets (see Figure 9). The application tasks in the pipeline simply generate workload in an endless loop.

Measurements have been performed with the OSCI SystemC kernel V2.2.0 and OMNET++ 4.1. Beside that, the CERTI null prime message algorithm (NPMA) is taken as basis for all measurements. The EPL was composed of a linux workstation (2.0 GHz QuadCore CPU, 8 GB RAM), an IRP system and a 100 MBit/s local area network.

6.3. Abstraction Benchmark. In this benchmark, a single remote node is co-simulated with 63 local nodes. This co-simulation use case is motivated by the need to analyse hw/sw characteristics of a single node (e.g., number of processing elements, task distribution, etc.) with respect to certain network traffic with the remaining nodes. RTL processing elements are needed if, for example, a detailed exploration of different types of MPSoC processing elements should be done. CAL processing elements can be used if only processing performance of the whole MPSoC with regard to network traffic shall be evaluated in detail. Symmetric synchronization is used for maintaining full accuracy also for node internal events. Sequential execution is necessary in the case that only a single workstation core is available. Parameters are summarized in Table 2.

Table 2: Abstraction benchmark configuration.

Parameter	Range
HeMPS abstraction level	RTL, CAL, NET
Clock frequency	100 MHz
NoC size	1×2, 2×2, 3×3
Remote node fraction	1/64
Synchronization method	Symmetric
Execution platform	1 workstation
Workstation cores	1

Results are illustrated in Figure 10. The simulation has been executed for 20 ms. As can be seen, in both cases, RTL and CAL, the runtime of co-simulation increases with the model size. However, when raising the abstraction level and using CAL instead of RTL, speedups of 1.3 (for the 1×2 system) to 1.65 (for the 3×3 system) are gained. The value of the speedup increases with the model size, since with increasing model size the computational overhead prevails the communicational overhead, introduced by the symmetric synchronization method. As a comparison, when simulating all 64 nodes as local nodes (pure network simulation, the most abstract case), the simulation only lasts three seconds instead of 2600 to 8000 seconds that were measured in case of co-simulation.

6.4. Parallelization Benchmark. In this benchmark, several remote nodes are co-simulated with several local nodes. This co-simulation use case is motivated by the need to analyse performance of a distributed application that is mapped onto different processing elements of distributed MPSoCs. The focus thereby would be on the mutual impact of selected MPSoC systems. CAL is used, since performance of each MPSoC as a whole is of interest. Parallel execution is possible in the case that several workstation cores are available. Parameters are summarized in Table 3.

Measured speedups are illustrated in Figure 11. In case of two, three, and four workstation cores, the network model was always executed separately on a single workstation core. As can be seen, except the 1×2 case, the speedup generally increases with increasing the degree of parallelization from two to three or four workstation cores. In case of a model size of 1×2 and four workstation cores, the computational overhead is obviously too low in order to provide an additional gain compared to three workstation cores. Beside that, the speedup provided by two workstation cores compared to a single one is marginal and partially even smaller than one. This is due to the fact that the network simulation always has been executed on a separate core and the bulk of computational overhead is generated by the system simulation. Thus, the network simulation is idle most of the time. However, for example, in case of a 1×2 model and three workstation cores (one running the network model exclusively), a larger speedup than two is measured. This in turn can be traced back to cache effects within the memory hierarchy of the used processors. When using a single workstation core for execution, the amount

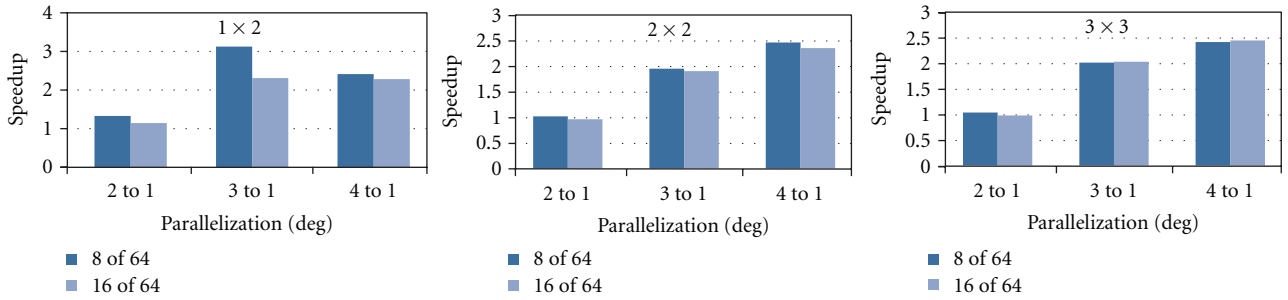


Figure 11: Results parallelization benchmark.

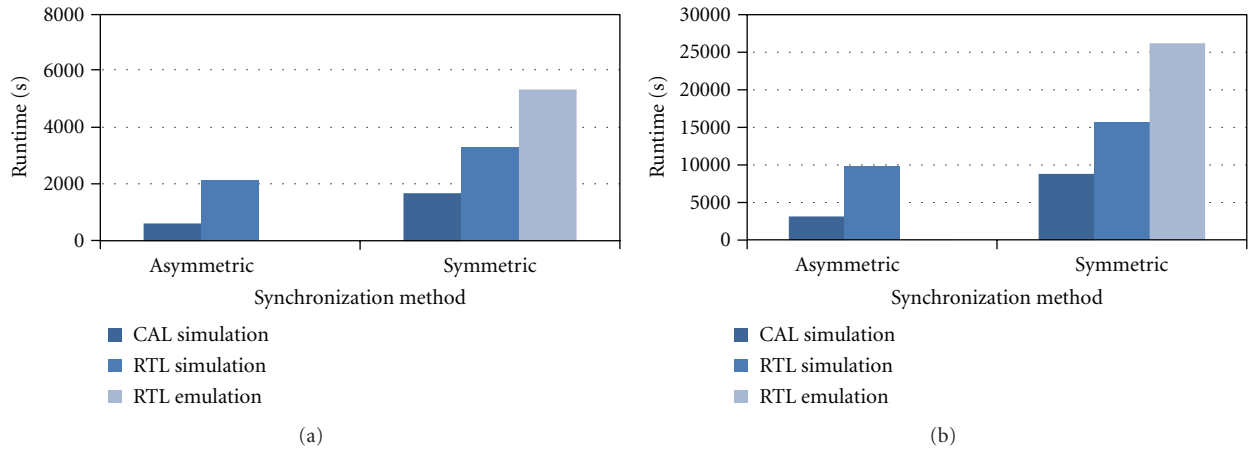


Figure 12: Results emulation benchmark (100 MHz (a) and 500 MHz (b)).

Table 3: Parallelization benchmark configuration.

Parameter	Range
HeMPS abstraction level	CAL
Clock frequency	100 MHz
NoC size	1×2, 2×2, 3×3
Remote node fraction	8, 16 of 64
Synchronization method	Symmetric
Execution platform	1 Workstation
Workstation cores	1–4

Table 4: Emulation benchmark configuration.

Parameter	Range
HeMPS abstraction level	RTL
Clock frequency	100 MHz, 500 MHz
NoC size	2×2
Remote node fraction	1/64
Synchronization method	Symmetric, asymmetric
Execution platform	1 workstation, 1 IRP
Workstation cores	1

of data that needs to be processed obviously outvalues the amount of fast cache memory. If increasing the number of workstation cores, more fast cache memory is available for holding data. This reduces the number of slow RAM accesses and provides the performance gain.

6.5. Emulation Benchmark. In this benchmark, a single remote node is co-emulated with 63 local nodes. This co-emulation use case is motivated by the need to speedup execution of detailed RTL models and to evaluate a hardware prototype of a single node with respect to its behaviour under realistic network traffic conditions. Asymmetric synchronization can be used without a loss of accuracy if no node internal events occur. Parameters are again summarized in Table 4. For the purpose of comparison, the same measurements are also done using co-simulation.

Execution times for 20 ms of simulation time are shown in Figure 12. The following characteristic can be observed in case of simulation. Raising the simulated clock frequency from 100 MHz to 500 MHz goes along with an increase of the execution time by a factor of about five. This characteristic is observed for both RTL and CAL simulation. Also the synchronization method has no significant influence on this characteristic. Both RTL- and CAL-based processing elements obviously provide enough computational overhead to hide the synchronizational overhead.

In case of emulation, the following can be observed. When using asymmetric synchronization co-emulation

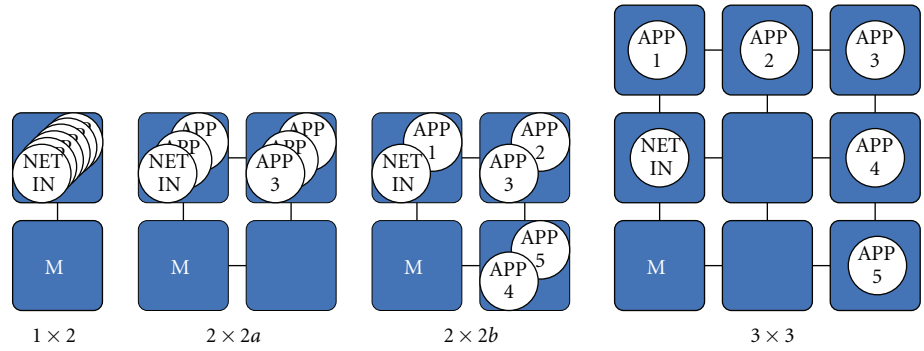


Figure 13: MPSoC configurations.

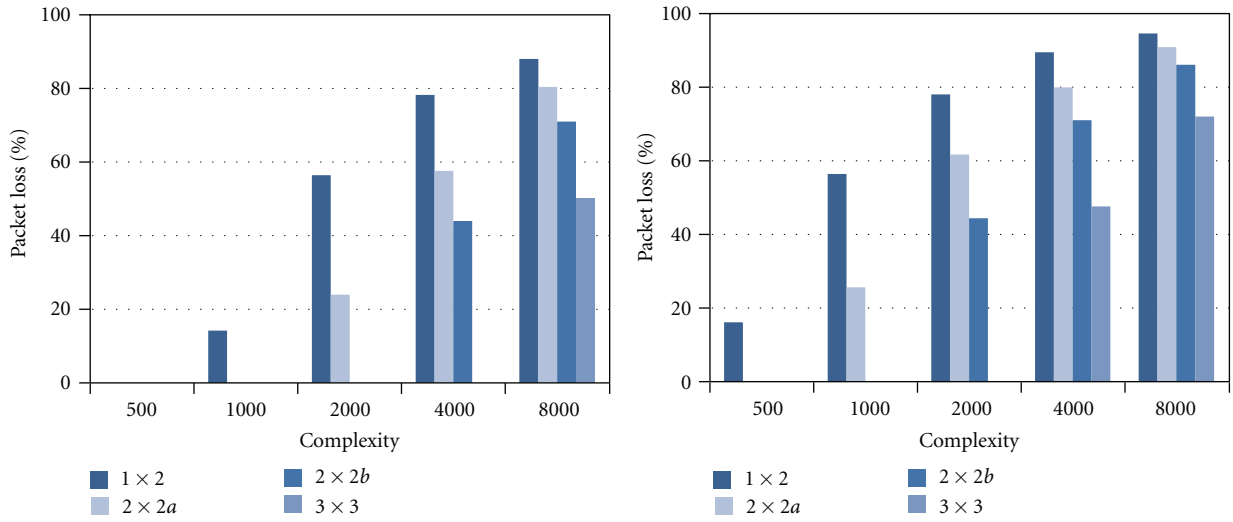


Figure 14: Dummy application: 500 Pkt/s (a) and 1000 Pkt/s (b).

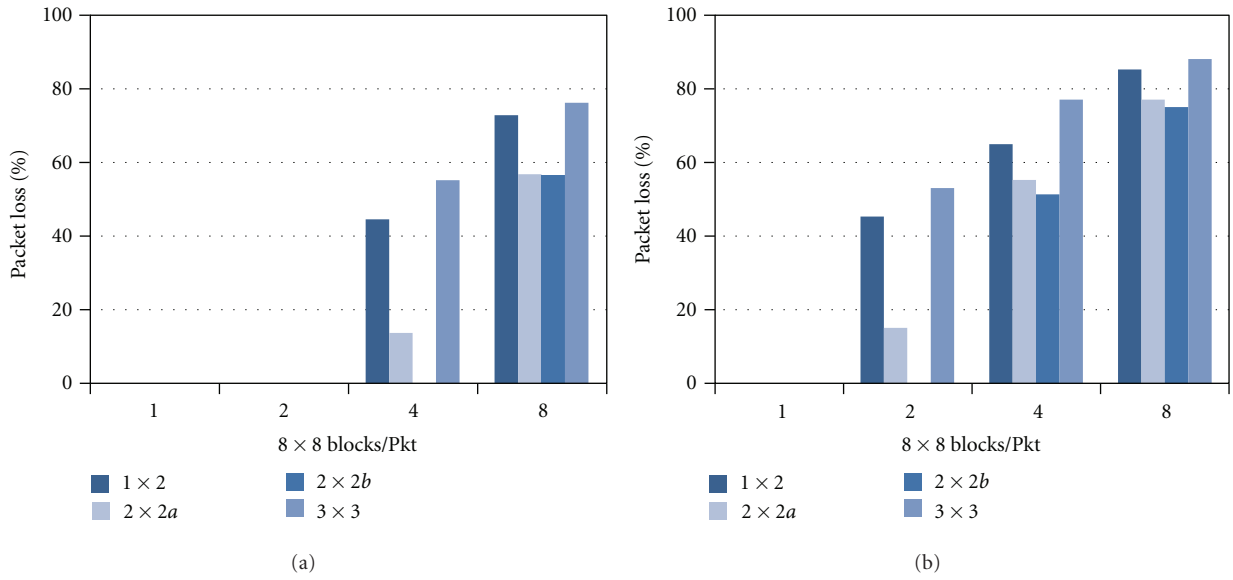


Figure 15: MPEG decoder: 500 Pkt/s (a) and 1000 Pkt/s (b).

strongly outperforms co-simulation for both simulation clock frequencies. Reasons are the independence of asymmetric synchronization of the time granularity [6] and the reduced computational overhead through FPGA execution (after logic synthesis the 2×2 RTL model runs with a processing clock frequency f_p of 25 MHz on the FPGA). In contrast, symmetric co-emulation generates even worse results than any co-simulation benchmark. This can be traced back to the additional delay of the LAN that connects the IRP system to the workstation.

7. Functional Simulator Verification

In order to verify that the simulation platform is working correct and appropriate for the intended objectives, a further example scenario has been set up. By the help of this, the effect of different network packet rates and different MPSoC configurations on the packet loss within the virtual network interface is exemplarily investigated.

7.1. Scenario Specification. A number of nodes are arranged in a circle having a diameter of 20 m. In the middle of the circle is an additional local node that stimulates the others via a 11 Mbps 802.11b channel at different packet rates.

7.2. Additional Requirements for the CSM. Each node around the circle is a remote node. Used MPSoC configurations are characterized by a certain number of available processing elements in combination with a certain kind of task mapping which results in varying throughput constraints. Measurements are performed by means of two types of applications, a five task dummy application, and a realistic five task MPEG decoder, both based on the pipelined software structure of Figure 9. Applied configurations and task mappings are illustrated in Figure 13.

Beside the packet loss of 1% that is induced by the wireless channel model, each MPSoC configurations induces an additional packet loss within the virtual network interface. This packet loss is measured.

7.3. Dummy Application. In case of the dummy application, each received network packet is assigned a certain amount of workload w in terms of integer multiply operations. Each application task processes 1/5 of the overall workload w by executing $1/5 \times w$ integer multiply operations. Benchmarking results are illustrated in Figure 14.

As can be seen, the hw/sw configuration strongly influences throughput and therewith the packet loss. In case of 500 pkt/s, the 1×2 system is able to cope with a packet complexity of 500 integer multiply operations, whereas the 3×3 system can process complexities of 4000 operations without packet loss. Doubling the packet rate to 1000 pkt/s expectably results in a left shift of the diagram.

7.4. MPEG Decoder. The same analysis has been performed by means of a realistic MPEG decoder consisting of an initiator task (INIT) that divides network packets into 8×8 MPEG blocks, a variable length decoder task (VLC), an inverse quantization task (IQUANT), an inverse discrete cosine transformation task (IDCT), and an output task (OUT) for data printing. Each received network packet contains a configurable number of 8×8 blocks of an encoded MPEG stream which is forwarded into the pipeline. As measurement results in Figure 15 reveal, variant $2 \times 2b$ is the most appropriate since it can cope with 4 blocks/pkt at 500 pkt/s and 2 blocks/pkt at 1000 pkt/s. The additionally emerging communication overhead of the 3×3 system is obviously too large to further speedup MPEG processing. This results in an increase of the packet loss compared to the $2 \times 2b$ system of 21%, respectively, 13% for 500 pkt/s, respectively, 1000 pkt/s.

8. Conclusion

We presented a methodology that enables efficient execution of cross-domain simulation models on different abstraction levels. Basis of the methodology is a construction kit-like simulation platform that allows assembly of different components and eases adaptation to different design and verification use cases. The usefulness of the methodology and its underlying simulation platform have been demonstrated by performance benchmarks that included consideration of different abstraction levels, parallelization and emulation. Beside that, the platform has been functionally verified by a realistic use case. We are sure, the methodology can support productivity and efficiency during development of networked embedded systems like future Cyber-Physical Systems.

Further research will focus on automatization of the simulation platform mapping step that is part of the methodology. Therefore, the development of an online profiling tool is planned which allows discovering bottlenecks during simulation execution. Results shall serve as basis for the development of a formal model of the platform that allows estimating performance. In this context, system/network co-simulation shall also be combined with distributed simulation of single systems. In addition, further abstraction levels are planned to be integrated into the HeMPS MPSoC model.

References

[1] P. Tabuada, *Cyber-Physical Systems: Position Paper*, 2006.

[2] "Modelsim," http://www.model.com/.

[3] A. Varga, "The OMNeT++ discrete event simulation system," in *Proceedings of the European Simulation Multiconference (ESM '01)*, June 2001.

[4] "ns-3," http://www.nsnam.org/.

[5] C. Roth, G. Almeida, O. Sander et al., "Modular framework for multilevel multi-device MPSoC simulation," in *Proceedings of the 25th IEEE International Parallel and Distributed Processing Symbposium*, 2011.

[6] C. Roth, O. Sander, and J. Becker, "Flexible and efficient cosimulation of networked embedded devices," in *Proceedings of the 24th Symposium on Integrated Circuits and Systems Design (SBCCI '11)*, pp. 61–66, ACM, New York, NY, USA, 2011.

[7] E. A. Carara, R. P. de Oliveira, N. L. V. Calazans, and F. G. Moraes, "HeMPS—a framework for NoC-based MPSoC generation," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '09)*, pp. 1345–1348, May 2009.

[8] R. M. Fujimoto, "Parallel simulation: distributed simulation systems," in *Proceedings of the 35th Conference on Winter Simulation: Driving Innovation (WSC '03)*, Winter Simulation Conference, 2003.

[9] R. M. Fujimoto, *Parallel and Distribution Simulation Systems*, John Wiley & Sons, New York, NY, USA, 1999.

[10] M. Chandy and J. Misra, "Distributed simulation: a case study in design and verification of distributed programs," *IEEE Transactions on Software Engineering SE-51979*, no. 5, pp. 440–452.

[11] J. Chaudron, E. Noulard, and P. Siron, "Design and modeling techniques for real-time RTI time management," in *Proceedings of the Spring Simulation Interoperability Workshop*, 2011.

[12] B. D. Lubachevsky, "Efficient distributed event-driven simulations of multiple-loop networks," *Communications of the ACM*, vol. 32, no. 1, pp. 111–131, 1989.

[13] D. Jefferson, B. Beckman, F. Wieland, L. Blume, and M. Diloreto, "Time warp operating system," in *Proceedings of the eleventh ACM Symposium on Operating systems principles (SOSP '87)*, vol. 21, no. 5, pp. 77–93, ACM, 1987.

[14] F. Mattern, "Efficient algorithms for distributed snapshots and global virtual time approximation," *Journal of Parallel and Distributed Computing*, vol. 18, no. 4, pp. 423–434, 1993.

[15] "IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)," IEEE Std 1516.x-2010, 2010.

[16] S. Strassburger, T. Schulze, and R. Fujimoto, "Future trends in distributed simulation and distributed virtual environments: results of a peer study," in *Proceedings of the Winter Simulation Conference (WSC '08)*, pp. 777–785, December 2008.

[17] "ns-2," http://nsnam.isi.edu/nsnam/.

[18] R. Barr, Z. J. Haas, and R. van Renesse, "JiST: an efficient approach to simulation using virtual machines: research articles," *Software-Practice & Experience*, vol. 35, no. 6, Article ID 106016, pp. 539–576, 2005.

[19] "Network emulation in the vint/ns simulator," in *Proceedings of the 4th IEEE Symposium on Computers and Communications*, p. 244, IEEE Computer Society, 1999, http://dl.acm.org/citation.cfm?id=876890.880459.

[20] E. Weingärtner, F. Schmidt, T. Heer, and K. Wehrle, "Synchronized network emulation: matching prototypes with complex simulations," *ACM SIGMETRICS Performance Evaluation Review*, vol. 36, no. 2, pp. 58–63, 2008.

[21] "Open Virtual Platform," http://www.ovpworld.org/.

[22] "The gem5 simulator system," http://www.m5sim.org.

[23] E. I. Moreno, K. M. Popovici, N. L. V. Calazans, and A. A. Jerraya, "Integrating abstract NoC models within MPSoC design," in *Proceedings of the 19th IEEE/IFIP International Symposium on Rapid System Prototyping (RSP '08)*, pp. 65–71, June 2008.

[24] L. S. Indrusiak, L. C. Ost, F. G. Moraes et al., "Evaluating the impact of communication latency on applications running over on-chip multiprocessing platforms: a layered approach," in *Proceedings of the 8th IEEE International Conference on Industrial Informatics (INDIN '10)*, pp. 148–153, July 2010.

[25] L. Ost, L. S. Indrusiak, G. M. Guindani, F. G. Moraes, and S. Määttä, "Exploring NoC-based MPSoC design space with power estimation models," *IEEE Design and Test of Computers*, vol. 28, no. 2, pp. 16–28, 2011.

[26] I. M. Pessoa, A. Mello, A. Greiner, and F. Pêcheux, "Parallel TLM simulation of MPSoC on SMP workstations: influence of communication locality," in *Proceedings of the International Conference on Microelectronics (ICM '10)*, pp. 359–362, December 2010.

[27] F. Fummi, P. Gallo, S. Martini, G. Perbellini, M. Poncino, and F. Ricciato, "A timing-accurate modeling and simulation environment for networked embedded systems," in *Proceedings of the 40th Design Automation Conference*, pp. 42–47, June 2003.

[28] N. Drago, F. Fummi, and M. Poncino, "Modeling network embedded systems with ns-2 and systemc," in *Proceedings of the 1st IEEE on Circuits and Systems for Communications (ICCSC '02)*, pp. 240–2245, 2002.

[29] N. Bombieri, F. Fummi, and D. Quaglia, "System/network design-space exploration based on TLM for networked embedded systems," *ACM Transactions on Embedded Computing Systems*, vol. 9, no. 4, pp. 1–37, 2010.

[30] F. Ghenassia, *Transaction-Level Modeling with Systemc: Tlm Concepts and Applications for Embedded Systems*, Springer, Secaucus, NJ, USA, 2006.

[31] B. Müller-Rathgeber and H. Rauchfuss, "A cosimulation framework for a distributed system of systems," in *Proceedings of the IEEE 68th Vehicular Technology Conference (VTC-Fall '08)*, pp. 1–5, September 2008.

[32] X. Deng, Y. Yang, and S. Hong, "A flexible platform for hardware-aware network experiments and a case study on wireless network coding," in *Proceedings of the 29th conference on Information communications*, March 2010.

[33] J. Zhang, Y. Tang, S. Hirve, S. Iyer, P. Schaumont, and Y. Yang, "A software-hardware emulator for sensor networks," in *Proceedings of the 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON '11)*, pp. 440–448, June 2011.

[34] F. Moraes, N. Calazans, A. Mello, L. Möller, and L. Ost, "HERMES: an infrastructure for low area overhead packet-switching networks on chip," *Integration, the VLSI Journal*, vol. 38, no. 1, pp. 69–93, 2004.

[35] "OpenCores," http://www.opencores.org/.

[36] G. E. Research, *Industrial Reference Platform*, Whitepaper, 2009.

[37] E. Noulard, J.-Y. Rousselot, and P. Siron, "CERTI, an Open Source RTI, why and how," in *Proceedings of the Spring Simulation Interoperability Workshop*, 2009.

Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2012, Article ID 813023, 10 pages
doi:10.1155/2012/813023

Research Article

Algorithm and Hardware Design of a Fast Intra Frame Mode Decision Module for H.264/AVC Encoders

Daniel Palomino,[1] Guilherme Corrêa,[1] Cláudio Diniz,[1] Sergio Bampi,[1] Luciano Agostini,[2] and Altamiro Susin[1]

[1] *Microelectronics Group, INF, Federal University of Rio Grande do Sul, Avenida Bento Gonçalves 9500, P.O. Box 15064, 91501-970 Porto Alegre, RS, Brazil*
[2] *Group of Architectures and Integrated Circuits, CDTEC, Federal University of Pelotas, Campus Universitário s/n, P.O. Box 354, 96001-970 Pelotas, RS, Brazil*

Correspondence should be addressed to Daniel Palomino, dmvpalomino@inf.ufrgs.br

Received 19 January 2012; Revised 29 May 2012; Accepted 21 July 2012

Academic Editor: Massimo Conti

In the rate-distortion optimization (RDO), the process of choosing the best prediction mode is performed through exhaustive executions of the whole encoding process, increasing significantly the encoder computational complexity. Considering H.264/AVC intra frame prediction, there are several modes to encode a macroblock (MB). This work proposes an algorithm and the hardware design for a fast intra frame mode decision module for H.264/AVC encoders. The application of the proposed algorithm reduces in more than 10 times the number of encoding iterations for choosing the best intramode when compared with RDO-based decision. The architecture was synthesized to FPGA and achieved an operation frequency of 98 MHz processing more than 300 HD1080p frames per second. With this approach, we achieved one order-of-magnitude performance improvement compared with RDO-based approaches, which is very important not only from the performance but also from the energy consumption perspective for battery-operated devices. In order to compare the architecture with previously published works, we also synthesized it to standard cells. Compared with the best previous results reported, the implemented architecture achieves a complexity reduction of five times, a processing capability increase of 14 times, and a reduction in the number of clock cycles per MB of 11 times.

1. Introduction

H.264/AVC is the state-of-art video compression standard proposed by ITU-T and ISO-IEC [1]. It has shown significantly better coding performance than existing video coding standards, for example, about 50% bit-rate reduction compared with MPEG-2, and, for this reason, it has been adopted by most of electronic devices that encode digital video, such as camcorders and mobile phones. A huge amount of coding options have been included in the H.264/AVC encoder to achieve a better coding efficiency for different video sequences. Since all video codecs, as well as H.264/AVC, include lossy compression schemes, the video encoder must know how much the video quality is degraded for a given target quality, that is, there is a direct relation between the amount of bits of the coded video and its visual quality.

Figure 1 shows how the video quality behaves for a target bit-rate of a coded video. The video quality is measured in peak signal-to-noise ratio (PSNR), and the bit-rate is measured in bits per second (bps). The point **a** in Figure 1 shows a very low bit-rate video with very degraded quality, which is not desirable for most applications. The point **c** in Figure 1 shows a higher quality video but with very large bit-rate, which is not useful for practical applications with restricted bandwidth. The point **b** in Figure 1 shows a video with balanced relation between quality and bit-rate.

Those various coding tools, included mainly on the prediction modules (intra frame prediction and interframes prediction) of an H.264/AVC encoder, must be evaluated

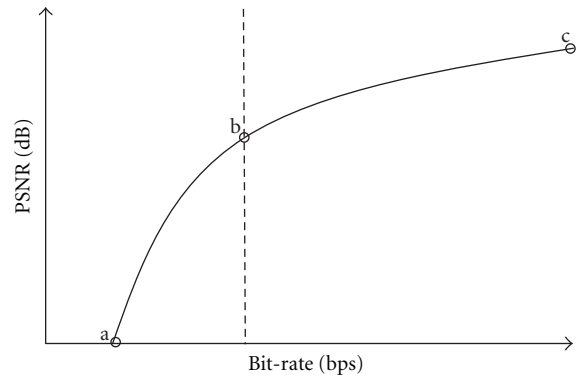


Figure 1: Rate-distortion behavior of an encoded video.

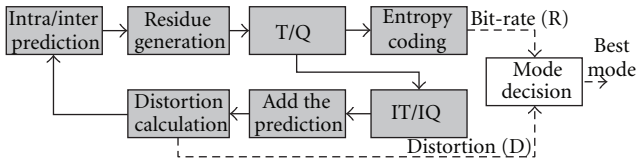


Figure 2: Diagram of RDO-based decision.

in terms of their rate-distortion results to be selected, and then the residual data between the coding possibility that generates the best predicted block and the original block is encoded to form the final video stream. The rate-distortion optimization (RDO) [2] is a well-known technique for maximizing coding quality and minimizing the amount of coded bits. It is usually employed in video encoder implementations to achieve the best coding result. However, the RDO technique demands a high computational complexity in H.264/AVC encoders, since it exhaustively examines all intra- and interprediction modes, performing a complete encoding process for each mode. Up to 90% of total encoding time may be spent in the mode decision stage [3].

Figure 2 shows the diagram of the RDO-based mode decision to better explain its complexity. Grey blocks are performed once for each prediction mode, while the mode decision block (in white) receives all bit-rates and distortions (dashed lines) of the candidate modes and the best mode selected is the one that presents a better tradeoff among bit-rate (R) and distortion (D).

In the intra frame prediction for the luminance layer, there are two possible block size partitions to encode one macroblock (MB): (1) I16MB, with four possible prediction modes applied to the whole MB (16×16 samples) and (2) I4MB, with nine possible prediction modes applied to the sixteen blocks of 4×4 samples which compose the MB. For the chrominance layer there are also four possible modes to predict each 8×8 block (Cr and Cb) in a MB. Considering an HD1080p video sequence (1920×1080 pixels), 138.720 iterations of prediction, forward transform and quantization, inverse transform and quantization and entropy coding (called in this work as *encoding loop*) (Figure 2) are needed to encode each intra frame using the RDO technique. This

M	A	B	C	D	E	F	G	H
I	a	b	c	d				
J	e	f	g	h				
K	i	j	k	l				
L	m	n	o	p				

Figure 3: Neighbor samples used in the prediction of a 4×4 block.

way, it is clear that the RDO-based decision is hard to be used when high resolution and real-time applications are considered. Besides that, an encoder architecture that uses the RDO-based decision will spend a lot of clock cycles and energy to perform the encoding loop for all those candidate modes that will not be included in the bit stream at the end.

Due to this complexity, some works as [4–6] have proposed fast intra frame mode decision algorithms and hardware designs to decrease the encoding time for one MB. However, all these works focus only on reducing the number of modes that will be evaluated by the RDO-based decision. The work [4] proposes an intradecision based on the dominant edge strength. The authors use a filtering technique to perform the edge detection. This way, the number of modes is reduced from nine to four in a 4×4 luma block. For 16×16 luma or 8×8 chroma block, only the detected mode and the DC mode are selected to be evaluated by the RDO technique. In the work [5], the authors proposed a modified low complexity mode decision algorithm based on a cost function composed by distortion and an estimated rate. The work [6] proposes a modified three-step algorithm [7] to perform the intradecision. As well as the work in [4], the main goal is to decrease the number of I4MB candidate modes (from nine to seven) to be evaluated by the RDO technique.

The main drawback of the related works [4–7] is that the proposed techniques can only reduce the number of candidate modes to be evaluated by the RDO process. This is a good technique but the gains in computational complexity are limited, since the RDO process is still executed for some modes. In this work, our approach is very different. There are other relevant works [8–12] in the literature proposing fast mode decision algorithms to improve the intra prediction performance. These works present interesting techniques to reduce the computational complexity of the intra prediction process. However, none of these works takes the hardware implementation issues into account and all of them still use the RDO-based decision.

We propose a fast algorithm and its hardware architecture in order to completely eliminate the RDO-based decision of the encoding process, thoroughly decreasing the time needed to encode one MB. The algorithm uses a threshold value to choose the partition type (I16MB or I4MB), and for each partition type the encoding mode is selected with a simple distortion metric, for example, sum of absolute differences

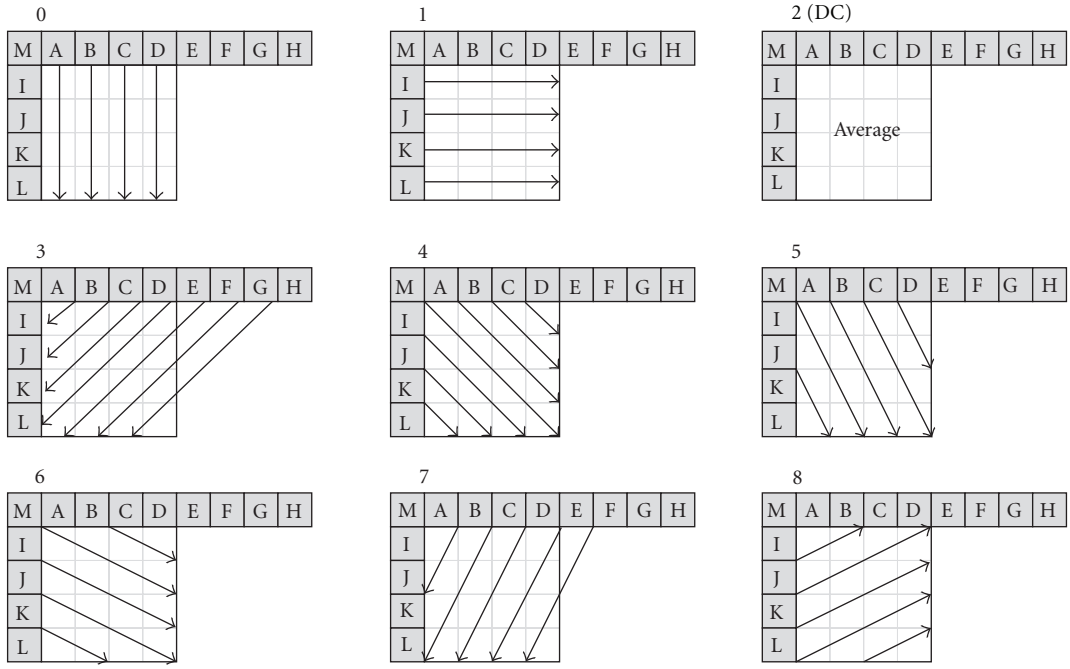


Figure 4: Nine possible prediction modes for I4MB macroblocks.

(SAD). The threshold value is based on the difference of distortions (DD) and is determined by offline simulations with several video sequences which represent a variety of illumination and texture patterns.

The paper is organized as follows. Section 2 presents the intra frame prediction and all possible modes to perform the prediction. Section 3 shows our fast intradecision algorithm. Section 4 shows the designed architecture of the intradecision and compares it with state of the art. Finally, Section 5 concludes this work.

2. Intra Frame Prediction

The intra frame prediction module exploits the spatial redundancy inside a video frame. It is performed by an interpolation of the neighbor pixels previously coded and decoded. In the H.264/AVC standard, the intra prediction is applied either for luminance or chrominance samples. Considering the luminance samples, there are two possible partition sizes: (1) 4×4 (I4MB) and (2) 16×16 (I16MB). When the I4MB partition is chosen, the 16×16 luminance block is divided into 16 4×4 blocks and, then, they are all individually predicted. In the I16MB partition, the entire 16×16 luminance block is predicted. The chrominance samples are always predicted using the same size (8×8), when the 4 : 2 : 0 subsampling is used.

2.1. 4×4 *Partition Size (I4MB).* For the I4MB partition size, there are nine different ways to generate the prediction of one 4×4 block. Figure 3 shows the neighbor samples that can be used to perform the prediction over a 4×4 block. Samples A to M were previously encoded and reconstructed. Samples a to p represent the current block that is being predicted.

The interpolation of the samples A to M is performed according to an angle direction that defines which and how the samples will be interpolated to generate the predicted block. Figure 4 shows the nine prediction modes for I4MB macroblocks.

In the modes 0 and 1, only a copy is performed (following the presented directions) to generate the predicted block. In the mode 2, an average over the neighbor samples is performed to generate all pixels of the predicted block. In the modes 3, 4, 5, 6, 7, and 8, an interpolation following the directions is performed to generate the predicted block [13].

2.2. 16×16 *Partition Size (I16MB).* When the partition size used is 16×16 (I16MB), there are four different ways to perform the prediction over a block. There are 32 neighbor samples (H and V in Figure 5) to be used in the prediction modes. Figure 5 shows the four possible modes to the I16MB partition.

In modes 0 (VERTICAL, in Figure 5) and 1 (HORIZONTAL, in Figure 5), the samples H and V are copied in the vertical and horizontal directions, respectively, for the whole 16×16 block. In mode 2 (DC, in Figure 5), the samples in the predicted block are generated using the average value between all neighbor samples (H and V).

The mode 3 (PLANE, in Figure 5) is the most complex, since its calculation uses a linear function where three parameters are used for each sample in the predicted block [13]. The four chrominance modes are the same as these used for I16MB partitions, but for smaller blocks (8×8).

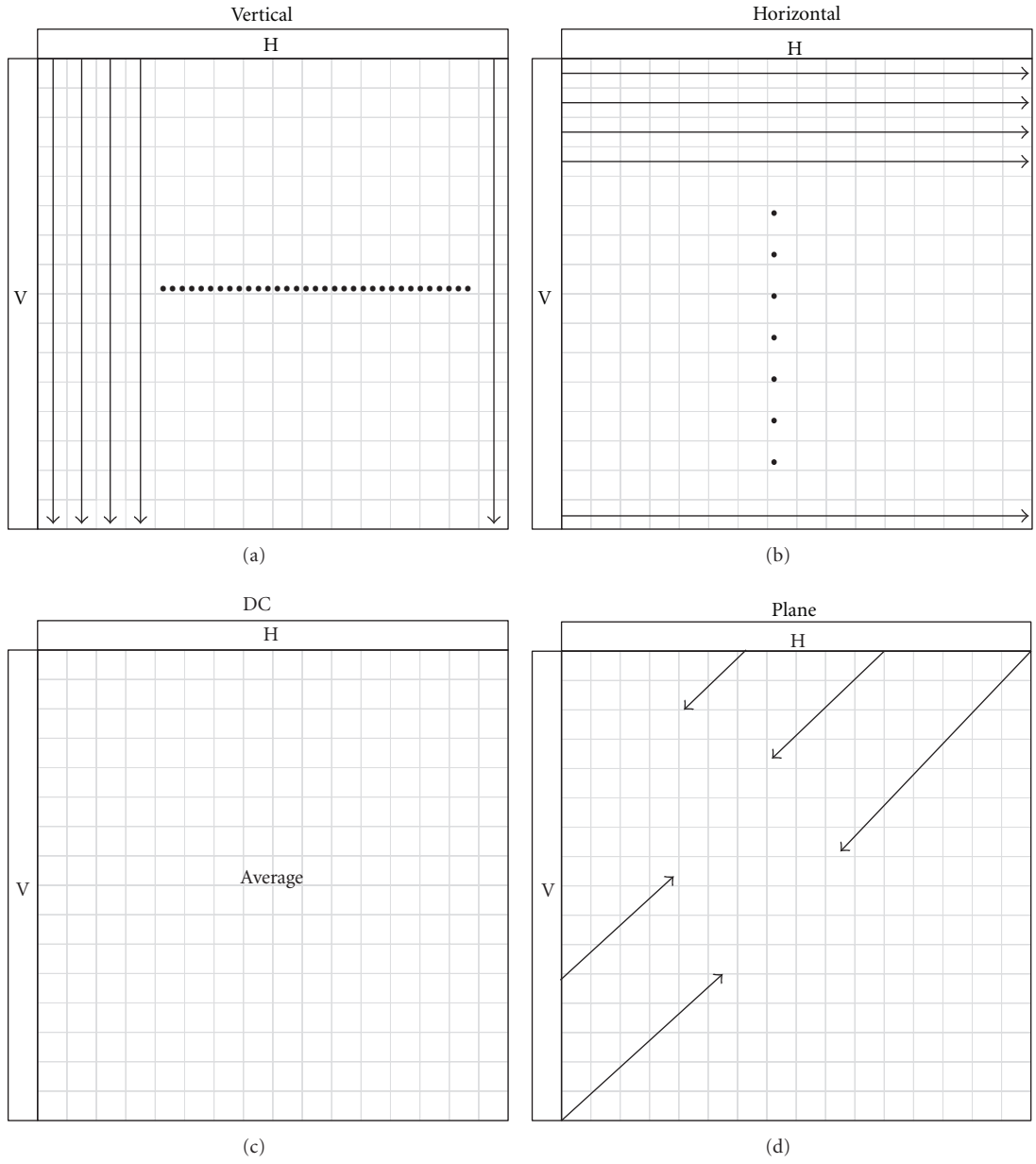


Figure 5: Four possible modes for I16MB macroblocks.

3. Fast Intra Frame Mode Decision Algorithm

Our fast intramode decision algorithm is based only on distortion calculation. The decision is performed in a hierarchical way in two steps: (1) decision among equal partition sizes and (2) decision among different partition sizes. In the first step, distortion calculation for different partition types (I4MB and I16MB) is calculated independently. In the second step, we applied a metric that we called difference of distortion (DD) and a heuristic to choose which partition shall be coded to obtain the best RD result. The next subsections will better explain our approach.

3.1. Decision among Equal Partition Sizes. The first decision step is based on distortion calculation to choose the best I16MB partition considering the four possible modes and the best I4MB partition considering the nine possible modes. Several simulations considering three different distortion metrics were performed: sum of absolute differences (SAD), sum of squared differences (SSD), and sum of absolute transformed differences (SATD). The results (for bit-rate and video quality) obtained using these three metrics were compared among each other, and a comparison was performed considering computational complexity measured by the number of sums (see Table 1).

Table 1: Comparison among SAD, SSD, and SATD.

SSD versus SAD			SATD versus SAD		
PSNR (dB)	Bit-rate (%)	Sums (%)	PSNR (dB)	Bit-rate (%)	Sums (%)
+0.008	−0.63	+361.29	+0.079	−1.13	+348.39

SATD and SSD metrics show better RD results (bit-rate and video quality). However, the computational complexity of these two metrics is extremely much larger than SAD (about 361% larger when the SSD metric is considered). As the main goal of this work is to design a faster intramode decision, we decided to use SAD as the distortion metric. In addition to that, the hardware architecture design for SAD calculators is simpler than one for SATD (which includes a 4×4 Hadamard transform) and SSD (which includes a multiplier and a square root). So in this step, a SAD calculation is performed for all modes considering each 4×4 block and the partial results are accumulated to obtain the SAD of the entire MB in I4MB modes. A SAD calculation is also performed for the four I16MB modes and four chrominance modes.

3.2. Decision among Different Partition Sizes. The second step of the proposed intradecision is to choose which partition size (I4MB or I16MB) will be used to encode the luminance channel of MB. This decision is made using the information generated by the first step: the distortion of the best I4MB partition and the distortion of the best I16MB partition measured with SAD. A simple comparison between these two values would cause, in most cases, the choice for I4MB partitions, because of their finer prediction granularity and more coding modes. However, analyzing the behavior of the difference between these two distortion values when RDO-based decision is applied, it is possible to make a good choice on which partition shall be used for each MB.

Simulations were performed with various video sequences using JM H.264 reference software [14] set in full-RDO-based decision and intra-only MB modes (to choose only between I4MB or I16MB modes). It was possible to notice that in most cases when I16MB partition had been chosen the distortion values of the best I4MB partition and the best I16MB partition were very close. It means that most of the 16 4×4 modes were the same, so it is more beneficial to choose a 16×16 partition (I16MB mode), since it will generate less modes information, that is, less bits in the bit stream to signalize mode information. On the other hand, when I4MB partitions were chosen, the distortion values of the best I4MB and the best I16MB partition were very different. It means that most of the 16 4×4 modes were different and even choosing only one 16×16 mode it will generate a lot of residual information.

Then, we created a metric called difference of distortions (DD) and several simulations were performed to classify in which situation each intra prediction mode is selected by the RDO technique. Equation (1) shows the difference of distortion (DD) calculation where the SAD_{I4MB} is the sum of all residual generated by the 16 best 4×4 modes and the SAD_{I16MB} is the residual generated by the best 16×16 mode:

$$DD = SAD_{I4MB} - SAD_{I16MB}. \quad (1)$$

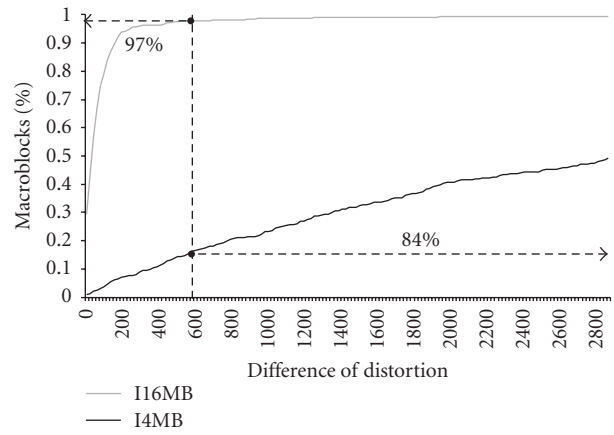


Figure 6: Percentage of modes chosen by the RDO technique and difference of distortion (DD) for the MBs.

Figure 6 shows a graph where the percentage of chosen modes (I4MB and I16MB) for each MB selected by the RDO technique is compared with the difference of distortion generated by the best I4MB and I16MB partitions (first decision step). With this metric, the partition (I4MB or I16MB) is chosen and then the rate-distortion calculation is performed.

With the difference of distortion set to 600, for example, it is possible to see that in most cases when the I16MB partition is chosen (97%) the difference of distortion is very small (lesser than 600), while when the I4MB partition is chosen the difference of distortion is very large (84% are bigger than 600). This way, it is possible to use this information in comparison with a threshold value to choose the partition size for intra frame MBs.

The threshold value that presents the best results in terms of PSNR and bit-rate was obtained as follows: (1) all videos used in the simulations were first encoded using the RDO technique for the decision among different block sizes. During that, the distortion values and the chosen partition were saved. Then, the differences of distortion were compared with the chosen partition to define the threshold value. Simulations were performed with a threshold ranging from 0 to 1000, adjusting it to bit-rate and video quality. Threshold value set to 600 generated the best results considering the bit-rate and video quality relation for the video sequences evaluated.

The results obtained by using the proposed intradecision algorithm are presented in Table 2. The first columns present the results using RDO-based decision. The central columns present the results obtained by the proposed decision. Finally, the last columns show a comparison between the two approaches in terms of bit-rate increase and image quality

Table 2: Results of the proposed intramode decision and comparison with RDO-based decision.

Video	RDO		Proposed decision		RDO versus proposed decision (threshold 600)	
	PSNR (dB)	Bit-rate	PSNR (dB)	Bit-rate	PSNR loss	+Bit-rate (%)
Station 2	39.643	34022 k	39.338	35729 k	0.305	5.02
Sunflower	42.766	33030 k	42.413	35391 k	0.353	7.15
Tractor	39.400	58208 k	39.035	60939 k	0.365	4.69
Traffic	39.643	50884 k	39.308	53070 k	0.335	4.30
Manincar	42.609	8818 k	42.518	9018 k	0.091	2.27
Pedestrian area	40.878	24949 k	40.699	26875 k	0.179	7.72
Riberbed	38.652	56735 k	38.280	58962 k	0.372	3.93
Rolling tomatoes	40.475	12170 k	40.387	12537 k	0.088	3.02
Rushhour	41.694	20013 k	41.484	21306 k	0.210	6.46
Average	40.640	33203 k	40.385	34870 k	0.255	5.02

Table 3: Comparison with related works.

Works	PSNR loss (dB)	Bit-rate increase (%)	Number of iterations reduction (times)
Wang et al. [4]	0.021	2.92	2.6
Kao et al. [5]	0.010	—	—
Lin et al. [6]	0.110	1.03	1.1
This work	0.255	5.02	13

(PSNR) difference. These results were obtained through simulations performed with HD1080p video sequences [15] which represent a variety of illumination and texture patterns.

The application of the proposed heuristic resulted in an average increase of 5.02% in the bit-rate and an average decrease of 0.255 dB in the image quality (PSNR). The increase of bit-rate and the decrease of image quality are very small and are justifiable by the enormous computational complexity reduction achieved in the decision process. As presented in Figure 2, the RDO-based encoding process is finished only after the execution of all possible intra frame prediction modes by the encoding loop. The decision proposed in this work is performed after the generation of the predicted blocks by the intra prediction followed by the SAD-based distortion calculation and then the difference of distortion operation. This way, the encoding loop presented in Figure 2 is completely eliminated resulting in enormous gain in terms of computational complexity reduction of the intra frame decision process. When RDO-based decision is performed, four I16MB modes and nine I4MB intra frame modes must be evaluated, totalizing 13 encoding iterations per MB. Considering the proposed decision method, the encoding process is performed only once for each MB. Table 3 presents a comparison with related works in terms of bit-rate, image quality (PSNR), and reduction in RDO calculations.

While other works have shown a reduction of coding iterations from 1.1 to 2.6 times in comparison with RDO-based decision, the proposed decision allows a reduction of 13 times (one order-of-magnitude). The cost of this gain resides, however, in the bit-rate increase of 5.02% and image quality loss 0.255 dB which do not compromise coding efficiency when the enormous gain in terms of computational complexity reduction is considered. Moreover, reducing the number of encoding loop iterations, it is possible to reduce the number of clock cycles and energy consumption needed by the hardware architecture to perform the whole prediction of one MB.

4. Designed Architecture and Comparisons

In order to further improve the performance of our fast intra frame decision scheme, a hardware architecture was designed. The fast intramode decision architecture is shown in Figure 7, which consists of 17 SAD calculators (nine for I4MB modes, four for I16MB modes, and four for chroma modes), three comparators, and the DD mode decision module. The distortion calculation is performed by the SAD calculators using as input the predicted block and the original block. Considering the distortion calculation for I4MB partitions, the SAD value of each mode is compared and then the 16 lowest SADs of each 4×4 block are accumulated to generate the total distortion for the I4MB partition. For I16MB partitions, the SAD values among the four modes are compared and then the lowest one is chosen as the best I16MB partition. As chrominance samples are predicted considering only one partition size (8×8), the decision among these samples is easier. The SAD values for each mode are compared, and the lowest one is chosen as the best.

Figure 8 shows the RTL diagram of each SAD calculator. The SAD calculator consumes eight samples (two lines of a 4×4 block) per cycle. It was designed with two pipeline stages,

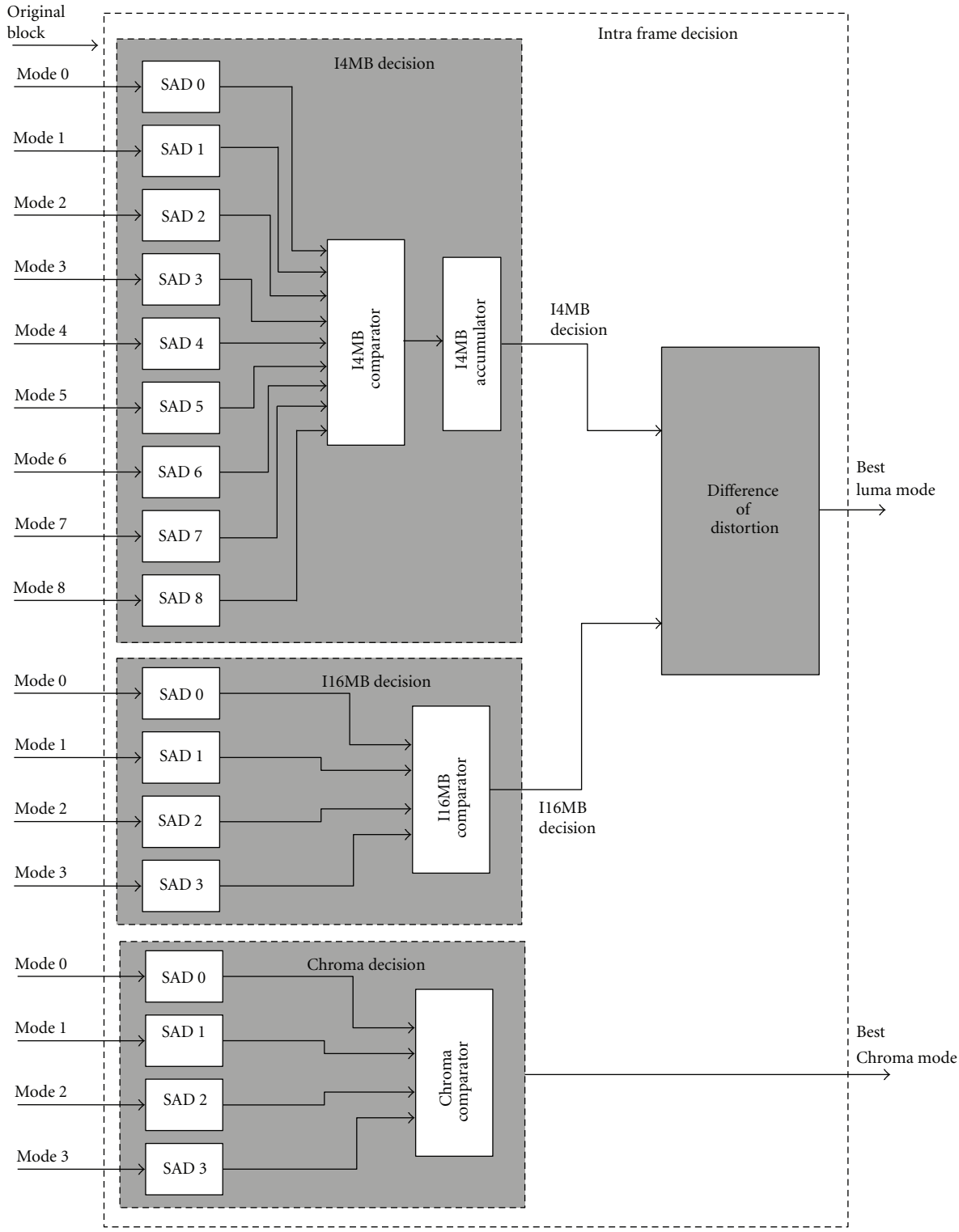


Figure 7: Fast mode decision architecture.

that is, it takes two clock cycles to deliver the first result. There is a little difference between the SAD calculator used in the I4MB distortion decision and the SAD calculator used in the I16MB distortion decision. Considering I4MB partition, the accumulated value is used by the comparator each time that one 4×4 block was processed, since the prediction is performed over 4×4 blocks. After that, the lowest SAD of the nine 4×4 blocks is chosen as the best, and then it is

Table 4: Synthesis results and comparison with related works.

Works	Technology	Hardware resources	Number of cycles	Max. frequency	Number of HD1080p frames/s	Frequency to process HD1080p@30 fps
Wang et al. [4]	UMC 0.18 μm	10,302 gates	416	66 MHz	31	62.21 MHz
Kao et al. [5]	TSMC 0.13 μm	11,229 gates	672	75 MHz	Not supported	Not supported
Lin et al. [6]	TSMC 0.13 μm	*94,700 gates	560	140 MHz	30	140 MHz
This work	FPGA	3,267 ALUTs 2,312 DLRs	36	98.43 MHz	335	8.26 MHz
	TSMC 0.18 μm	28,518 gates	36	129.1 MHz	439	8.26 MHz

* Hardware resources considering the complete intra-coder.

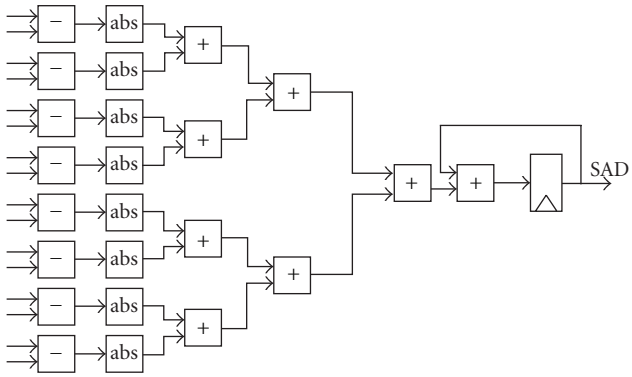


Figure 8: RTL diagram of one SAD calculator.

accumulated again until all the 16 4 × 4 blocks are read. On the other hand, when the best I16MB and the best 8 × 8 chroma partitions are considered, only one accumulator is needed, since the prediction is generated over the whole block. Finally, the comparator chooses the best mode.

Figure 9 shows the time diagram of the designed architecture. As the SAD calculators were designed with two pipeline stages and with eight samples of parallelism, the architecture takes 3 clock cycles to deliver the first valid SAD value of a 4 × 4 block. When the pipeline is filled, there is a valid SAD value every two clock cycles. This way, the architecture takes 34 clock cycles to evaluate all the nine I4MB modes and one more clock cycle to accumulate the last best SAD. Considering the I16MB partition decision, the SAD values are accumulated in the SAD calculator itself. This way, the architecture takes 33 clock cycles to accumulate the SAD for the four modes and one more clock cycle to compare them. The chroma decision is similar to the I16MB decision; however, the block size is of 8 × 8 samples. Then, it only takes 10 clock cycles to perform the whole chroma decision. After the calculation of the best I4MB distortion and the best I16MB distortion, the difference of distortion is evaluated in one clock cycle to decide which partition size will be used.

The architecture was described in VHDL and synthesized targeting the EP2S130F1508C3 Stratix II FPGA [16]. Even though we have performed an extensive research in the technical literature, no works using FPGA as device target in intra prediction mode decision were found. This way, the architecture was also synthesized for TSMC 0.18 μm standard cells [17] to allow for a fairer comparison. Table 4 presents the synthesis results for both technologies and compares them with previous works. The results are presented in terms of hardware resources usage, number of clock cycles needed to process one MB, maximum operation frequency achieved, and throughput measured in HD1080p frames per second.

When synthesized for FPGA, the architecture used 3.267 ALUTs (look-up tables) and 2.312 DLRs (dedicated logic register), totalizing 4% of the resources in the device. The FPGA synthesis achieved 98.43 MHz as maximum operation frequency, being able to process up to 335 HD1080p frames per second. When synthesized to TSMC 0.18 μm, the total gate count was 28.518. The maximum operation frequency achieved was 129.1 MHz. This way, the architecture is able to process 439 HD080p frames per second.

Compared with previous works [4–6], the designed architecture consumes the lowest number of cycles to process an intra frame: more than 11X reduction compared with [4], 18X reduction compared with [5]. It also results in the highest throughput among the related works: more than 11X and 14X increase in number of HD1080p frames encoded per second for FPGA and TSMC 0.18 μm versions, respectively, compared with [4]. All these results were obtained considering our architecture operating at maximum frequency. An interesting result is that with higher throughput we can reduce the operating frequency down to 8.26 MHz and still process HD1080p videos at 30 fps, which is the target frame rate normalized by H.264/AVC standard for real-time systems [1]. With this really low frequency, we can achieve very low power using our architecture, which is an excellent alternative for battery-powered devices. However, if the whole encoder design is considered, this low frequency

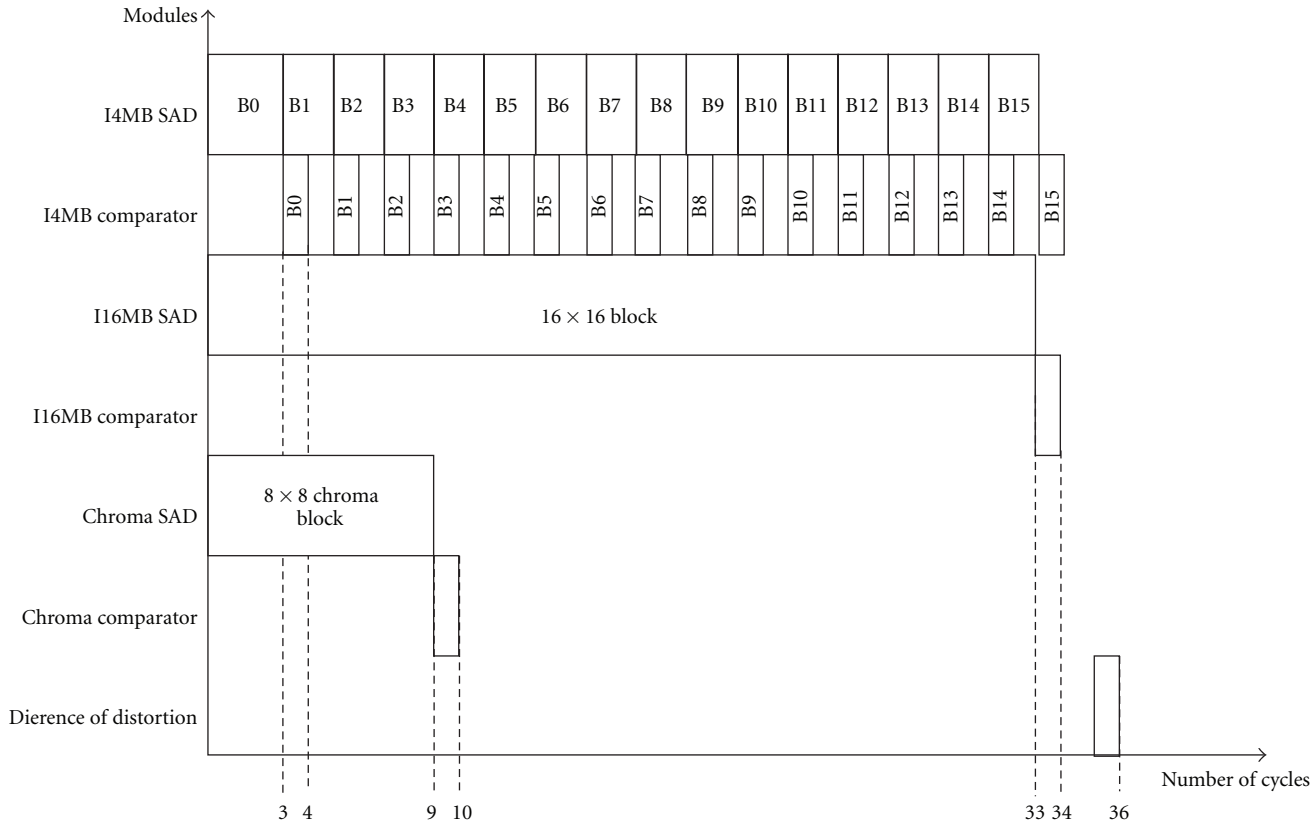


Figure 9: Schedule of the designed architecture.

could not allow HD1080p processing in real time if the other encoder modules do not achieve such performance.

5. Conclusions

This work has presented a fast intramode decision algorithm and its hardware architecture design for an H.264/AVC video encoder. The proposed algorithm allows the complete elimination of the encoding loop present in RDO-based mode decision, which is substituted by simple distortion calculations and comparisons, thoroughly decreasing the complexity of the video encoder. The number of encoding iterations was reduced in 13 times when compared with RDO-based decision, at the cost of relatively small bit-rate increase (5.02% in average) and image quality loss (0.255 dB in average).

When compared to other works, the proposed algorithm achieves a complexity reduction more than five times higher, while the bit-rate increase and the image quality loss are slightly higher and still similar to the compared works. The designed architecture of the fast intradecision algorithm was described in VHDL and synthesized targeting two technologies: (1) Stratix II FPGA and (2) TSMC 0.18 μm standard cell library. The synthesis results have shown that the architecture is able to process up to 439 HD1080p frames per second.

References

[1] ITU-T Recommendation H.264/AVC (05/03): advanced video coding for generic audiovisual services, 2003.

[2] G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for: video compression," *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 74–90, 1998.

[3] S. B. Solak and F. Labeau, "Complexity scalable video encoding for power-aware applications," in *Proceedings of the International Green Computing Conference*, pp. 443–449, Chicago, Ill, USA, August 2010.

[4] J.-C. Wang, J.-F. Wang, J.-F. Yang, and J.-T. Chen, "A fast mode decision algorithm and its vlsi design for H.264/AVC intra-prediction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 10, pp. 1414–1422, 2007.

[5] Y.-C. Kao, H.-C. Kuo, Y.-T. Lin et al., "A high-performance VLSI architecture for intra prediction and mode decision in H.264/AVC video encoding," in *Proceedings of IEEE Asia Pacific Conference on Circuits and Systems (APCCAS '06)*, pp. 562–565, Singapore, December 2006.

[6] Y.-K. Lin, C.-W. Ku, D.-W. Li, and T.-S. Chang, "A 140-MHz 94 K gates HD1080p 30-frames/s intra-only profile H.264 encoder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 3, pp. 432–436, 2009.

[7] C. C. Cheng and T. S. Chang, "Fast three step intra prediction algorithm for 4x4 blocks in H.264/AVC," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 2, pp. 769–772, May 2005.

[8] M. Parlak, Y. Adibelli, and I. Hamzaoglu, "A novel computational complexity and power reduction technique for H.264 intra prediction," *IEEE Transactions on Consumer Electronics*, vol. 54, no. 4, pp. 2006–2014, 2008.

[9] Y. H. Huang, T. S. Ou, and H. H. Chen, "Fast decision of block size, prediction mode, and intra block for H.264 intra prediction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 8, pp. 1122–1132, 2010.

[10] J. H. Min, S. Lee, I. K. Kim, W. J. Han, J. Lainema, and K. Ugur, "Unification of the directional intra prediction methods in TMuC," in *Proceedings of the 2nd Meeting of Joint Collaborative Team on Video Coding*, JCTVC-B100, Geneva, Switzerland, July 2010.

[11] J. H. Min, S. Lee, I. K. Kim, W. J. Han, J. Lainema, and K. Ugur, "TE4: Results for Simplification of Unified Intra Prediction," JCTVC-C042, Guangzhou, China , 2010.

[12] L. Zhao, L. Zhang, S. Ma, and D. Zhao, "Fast mode decision algorithm for intra prediction in HEVC," in *Proceedings of IEEE Visual Communications and Image Processing*, pp. 1–4, Tainan, Taiwan, November 2011.

[13] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, 2003.

[14] JM H.264 reference software version 16.2, http://iphome.hhi.de/suehring/tml/.

[15] "Video Library and Tools," http://nsl.cs.sfu.ca/wiki/index.php/Video_Library_and_Tools#1080p.

[16] Altera Corporation, "Altera: The Programmable Solutions Company," http://www.altera.com.

[17] Artisan Components, "TSMC 0.18μm 1.8-Volt SAGE-XTM Standard Cell Library Databook," 2001.

Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2012, Article ID 439727, 12 pages
doi:10.1155/2012/439727

Research Article

Modeling and Implementation of a Power Estimation Methodology for SystemC

Matthias Kuehnle,[1] Andre Wagner,[1] Alisson V. Brito,[2] and Juergen Becker[1]

[1] *Institute for Information Processing Technology, KIT, 7602 Karlsruhe, Germany*
[2] *Department of Informatics, Federal University of Paraiba (UFPB), 58051-900 João Pessoa, PB, Brazil*

Correspondence should be addressed to Alisson V. Brito, alissonbrito@dce.ufpb.br

Received 19 March 2012; Revised 12 May 2012; Accepted 18 June 2012

Academic Editor: Massimo Conti

This work describes a methodology to model power consumption of logic modules. A detailed mathematical model is presented and incorporated in a tool for translation of models written in VHDL to SystemC. The functionality for implicit power monitoring and estimation is inserted at module translation. The translation further implements an approach to wrap RTL to TLM interfaces so that the translated module can be connected to a system-level simulator. The power analysis is based on a statistical model of the underlying HW structure and an analysis of input data. The flexibility of the C++ syntax is exploited, to integrate the power evaluation technique. The accuracy and speed-up of the approach are illustrated and compared to a conventional power analysis flow using PPR simulation, based on Xilinx technology.

1. Introduction

The need for more abstract system on chip development techniques is evident due to rising system complexity. Consequently, accurate system evaluation in less time will increase the productivity. According to the Semiconductor roadmap, especially the consideration of energy consumption is becoming more important and is also a limiting factor for many applications [1, 2]. Modeling strategies are driven by system and software engineers on the one hand and hardware engineers on the other hand. The first group develops RTL models written in hardware description languages (HDLs) since they are the basis for synthesis tools. The second group uses transaction level models (TLMS), most commonly written in SystemC [3] since these models enable fast system simulation. SystemC is a library based on the object-oriented programming language C++. A TLM specification extends SystemC to separate communication from computation. TLM improves modeling and simulation speed. The simulation speed depends on the level of abstraction [4]. Also, modeling at different abstraction levels is possible. This increases the flexibility of SystemC.

A remaining problem is the trade-off between accuracy and simulation speed and with that, the link and synchronization between the two layers. Translation tools are solving this problem to some extent. They inherit some limitations in the translation of syntax constructs that do not have direct counterparts. The presented tool extends this feature list. The main goal of this work, however, is the integration of a power analysis methodology into the translation process. The power estimation methodology estimates switching activities in DSP units such as adders or multipliers according to actual input data. The approach is based on a statistical methodology. It implements the measurement functionality implicitly into the SystemC model by defining overloaded operators, in the sense of object-oriented programming. These can be differently characterized based on technology parameters. The operators can be automatically integrated at system translation. DSP units are considered, since a power analysis of such systems shows that the major part of the dynamic power dissipation is consumed in the data-processing part of the architecture. In addition, the power dissipation is highly data dependent. Therefore a fast but reasonably accurate estimation of the dynamic

power dissipation of such data-driven kernels is of high interest and the analysis using representative input data is essential. From the hardware point of view, the additional flexibility inherent in reconfigurable architecture and easy-to-use implementation flows make reconfigurable computing, especially FPGA architectures, attractive for power-aware computing solutions. Since CPUs are not sufficiently energy efficient (power consumption up to 200 W) and, on the other hand, ASICS are unaffordable for a low market volume, reconfigurable computing is considered an alternative especially for data-driven applications. Their benefit has been shown, for example, in [5, 6].

The described situation in the system development landscape motivates the development of a strategy for abstract, hence faster data-dependent power analysis for LUT-(lookup-table-) based systems in this work. The results were presented in [7] and are compared here to a standard power estimation flow. Beside it a detailed mathematical modeling strategy for power consumption estimation is presented. To ease and accelerate the development cycle, the power estimation method is embedded in a translation tool, so that the process of power estimation is transparent for the user and can be validated.

The remainder of this work is structured as follows. Section 2 summarizes related work. Section 6 embeds the strategy in an overall tool flow. Section 3 discusses the strategy to evaluate power dissipation for LUT-based hardware based on a statistical model to estimate toggle rates. This is related to a hardware mapping analysis of representing macroblocks (adders and multipliers) for data processing. Section 7 describes the implementation of the technology into the VHDL to SystemC converter. Section discusses the results. Section 9 concludes and gives an outline on future extensions.

2. Related Works

Code translation tools can help guaranteeing the consistency of translated and original models across languages and speeding up development time, since tool-supported translation takes seconds instead of hours to days for manual translation. In the system development cycle, top-down approaches [8–10] are used in HLS (high-level synthesis) tools; bottom-up approaches [11, 12] are used for IP reuse and module abstraction to achieve faster simulation models. In many hardware engineering problems, optimization is necessary on RTL Level. However, readability and efficiency of the translated code are two major problems of the HLS code translators. Because of that a bottom-up strategy is followed in this work. The authors in [13] further illustrates the effects of IP reuse on design time, hence motivates the bottom up approach.

The presented methodology differs from the existing solutions since it targets, beside correct code translation, the automatic integration of further functionality: bus interface wrapping and a power estimation methodology. The power estimation methodology estimates switching activities in DSP units such as adders and multipliers according to actual input data. The approach is based on probability theory. With that it differs from approaches that need to run a conventional time-consuming power estimation flow (e.g., XPA from Xilinx) or other estimators that are based on synthesis results without considering input data (e.g., XPE from Xilinx).

The approach presented in [14] is also implemented for SystemC. It enables the usage of different power models. In comparison with this work, it uses a strategy based on logging the execution of special modules and signals extended from regular SystemC ones, instead of a probability model, as presented here. In [15] a SystemC class library is proposed to the calculate the energy consumption of hardware described with SystemC TLM, and the power model was based on experimental results performed in laboratory, while approach of the presented work is based on the translation of modules from VHDL to SystemC RTL and on probability models. Further works concern the extension of the approach also to TLM.

The work [16] presents an architectural level framework for power analysis, based on parameterized power models of common structures of microprocessors, but does not consider any probability model, as presented in this work. In [17] a methodology is presented for simulation and verification of low-power systems using SystemC. It is based on disabling modules during execution to simulate the power switch-off used by the technique of power gating. The technique for disabling SystemC modules at simulation time is detailed in [18]. At the same time, these related works are not specific to reconfigurable architectures, in contrast to our work, which considers specific characteristics of FPGAs.

3. Power Evaluation Strategy

This project uses an activity-based power estimation and calculation based on the switching frequency of the inputs, outputs, and the internal signals of the individual sub-modules. The proposed power estimation and calculation is tailored for signal processing units where the major part of the power consumption is produced by multiply-accumulate instructions, which is computed in hardware by ripple carry adders and field multipliers based on ripple carry adders. To guarantee that the synthesized hardware for computing MAC is realized on the given adder and multiplier structures, a hardware model on register transfer level of the processing unit is needed. Both implementation and calculation are optimized for area consumption on FPGAs with little slice count. The accuracy of the estimation is based on two areas.

First there is the realization on hardware which is done by the synthesis tools for a specific LUT-based FPGA. Later in this section synthesis results for given FPGAs are examined for getting the fan-out parameters which are needed for a exact calculation of the power dissipation of each element. Further general rules for optimal synthesis of ripple carry adders and field multipliers are formulated.

Secondly, input data is analyzed for predicting the switching frequency on the expected synthesized hardware structure. This means that the probability distribution of the

input data has to be computed. The probability distribution of the output data is derived from that. The input data should be taken from the original implementation. This means, for example, in case of an audio decoder, a corresponding audio stream should be used. For better comprehension an complete workflow for power estimation is shown.

The first step to be done is the analysis of the input data. For example, the input data is uniformly distributed on the complete accepted range with the boundaries 0 and u, where u is a natural number. This assumption is suitable if the distribution of the input data is unknown. By the way this concept was first formulated by Gauss and is known as the Gaussian indifference principle.

Next the binary coding of the numbers between 0 and u is analyzed. The value of the MSB is distributed as followed:

(i) in the range between 0 and $u/2$, the value of the MSB is 0,

(ii) in the range between $u/2$ and u, the value of the MSB is 1.

With the condition of the uniform distribution, all numbers in range are selected with the same frequency. This means, for the probability that the MSB bit is set to 1,

$$p(\text{MSB}) = \frac{u/2}{u} = \frac{1}{2}. \tag{1}$$

The analysis of the next less significant bit is made the same way. The value of the (MSB-1) bit is the follows:

(i) in the range between 0 and $u/4$, value of (MSB-1) is 0,

(ii) in the range between $u/4$ and $u/2$, value of (MSB-1) is 1,

(iii) in the range between $u/2$ and $3u/4$, value of (MSB-1) is 0,

(iv) in the range between $3u/4$ and u, value of (MSB-1) is 1,

The calculation of the probability that (MSB-1) bit is set to 1:

$$p(\text{MSB} - 1) = \frac{2 * (u/4)}{u} = \frac{1}{2}. \tag{2}$$

Based on the calculations for the MSB and the (MSB-1) bit, a prediction can be made.

For every step you make from MSB towards LSB, the number of intervals in which the bit is set to 1 is doubled but the length of the intervals is halved. In consequence this means that the accumulated length of the intervals in which the bit is set to 1 is equal for all bits. Because of this fact the probability that a bit on the input is set to 1 is calculated for all bits:

$$p = \frac{1}{2}. \tag{3}$$

After the computation of the probabilities on the input vector, the probabilities for the output vectors of the ripple carry adder and the field multiplier can be calculated. First the probability of the occurrence of carry bits of the ripple carry adder is determined.

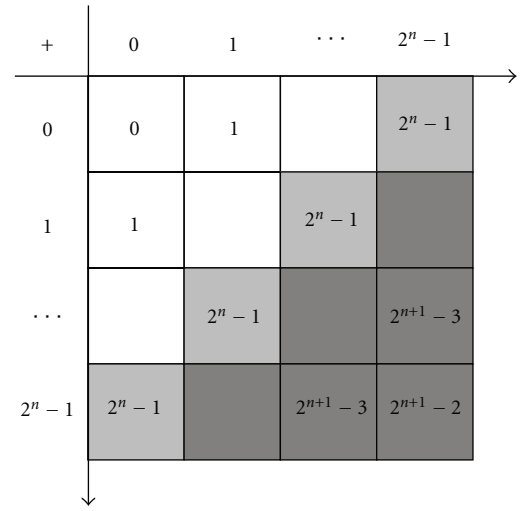


Figure 1: Carry generation in the adder.

For the possibility that a carry-on location n is generated, all input vectors of the adder between 0 and $2^n - 1$ are relevant. Figure 1 shows the addition of two summands in this range.

Each cell in this figure corresponds to the sum of the row and column number. To calculate the the probability that the carry is set, it is suitable to count the number of cell in which a carry is generated and divide this count by the total number of cells. This figure has 2^{2n} entries of which $2^{2n-1} - 2^{n-1}$ generate the carry. The division leads to

$$p_{c,n} = \frac{2^{2n-1} - 2^{n-1}}{2^{2n}} = \frac{1}{2} - 2^{n-1}. \tag{4}$$

Additionally the following statement can be made: because the generation nth carry is only possible if the upper bound of the accepted input values is 2^n, all the probabilities for all carry bits on locations higher than $ld(u)$ can be set to zero.

After calculating the probability of the carry bits, the probability of the sum bits of the adder can follow. The boolean equation of the sum on location n is

$$\begin{aligned} s_n = {} & \left(i1_n \wedge in2_n \vee \overline{i1_n} \wedge \overline{i2_n}\right) \wedge c_n \\ & \vee \left(\overline{i1_n} \wedge in2_n \vee i1_n \wedge \overline{i2_n}\right) \wedge \overline{c_n}. \end{aligned} \tag{5}$$

Transferring this equation to the probability domain leads to

$$\begin{aligned} p_{s,n} = {} & \left(p_{i1,n} * p_{i2,n} + \overline{p_{i1,n}} * \overline{p_{i2,n}}\right) * p_{c,n} \\ & + \left(\overline{p_{i1,n}} * p_{i2,n} + p_{i1,n} * \overline{p_{i2,n}}\right) * \overline{p_{c,n}}. \end{aligned} \tag{6}$$

Setting $p_{i1,n} = 0.5$, $p_{i2,n} = 0.5$ (uniform distribution) and $p_{c,n}$ equals

$$p_{b,n} = 0.5. \tag{7}$$

Like the statement for carry bits with location near the MSB, a likewise statement can be made for the sum bits: because the generation nth sum bit is only possible if the upper bound of the accepted input values is 2^n, all the probabilities for all carry bits on locations higher than $ld(u)$ can be set to zero. Similar to these calculations, likewise calculations for other probability are possible. The following paragraph describes the approach for calculating the probability of sum and carry bits with calculated nonequal probabilities for each bit of the adder input. The given input probabilities are named $p_{i1,n}$ and $p_{i2,n}$. In this project a 32-bit adder was analyzed so the valid indices are in the range from 0 to 31. With this new assumption, the proceeding is as follows.

First the boolean equation of the carry bit is annotated:

$$\overline{cn} = \left(i1_n \wedge \overline{i2_n} \vee \overline{i1_n} \wedge i2_n\right) \wedge c_{n-1} \vee \overline{i1_n} \vee \overline{i2_n}. \quad (8)$$

Based on this equation, a transformation to probability domain is made:

$$\overline{p_{c,n}} = \left(p_{i1,n} * \overline{p_{i2,n}} + \overline{p_{i1,n}} * p_{i2,n}\right) * p_{c,n-1} + \overline{p_{i1,n}} * \overline{p_{i2,n}}. \quad (9)$$

With the extra knowledge that the carry bit in the least significant adder is never set (this means $p_{c,n}$ is zero), all the carry probabilities can be calculated in a recursive manner. With the formula for calculating the $p_{s,n}$ probabilities the examination of the ripple carry adder with nonequal distributed input vectors is completed.

Based on these observations a derivation for the calculation of the signal probabilities of the field multiplier is possible.

The idea is to connect the calculations for the ripple carry adder with individual input probabilities. According to Figure 2 , the input probability p_{i1n} of one adder is the same value like the $ps(n+1)$ probability of the previous level. With the extra assumption of uniform distributed input vectors of the adder and the knowledge that a AND-gate is equal to a multiplication of its input probabilities, the $p_{i2,n}$ probability equals the multiplication of the probabilities of the inputs of the multiplier; that is, $0.5 * 0.5 = 0.25$. If the distribution is restricted to the range 0 to u, all $p_{i2,n}$ with a index and level higher than $ld(u)$ have to be set to zero.

Also these calculations can be performed on other input probability distributions. After the calculation of the probabilities of occurrence, the transition to the switch frequency is made. The normalized statistical switching frequency is given by

$$f_{\text{norm,stat}} = p(1-p). \quad (10)$$

That definition of the frequency is equal to the probability of the occurrence of a positive transition of the signal. By means of this frequency definition, the dynamic power dissipation of CMOS circuits is described as follows:

$$P_{\text{dyn,stat}} = C * U^2 * f_{\text{norm,stat}} * f_{\text{base}} * \text{fanout}, \quad (11)$$

where f_{base} defines the bit rate on the observed signal. The parameter C for the input capacity and the supply voltage

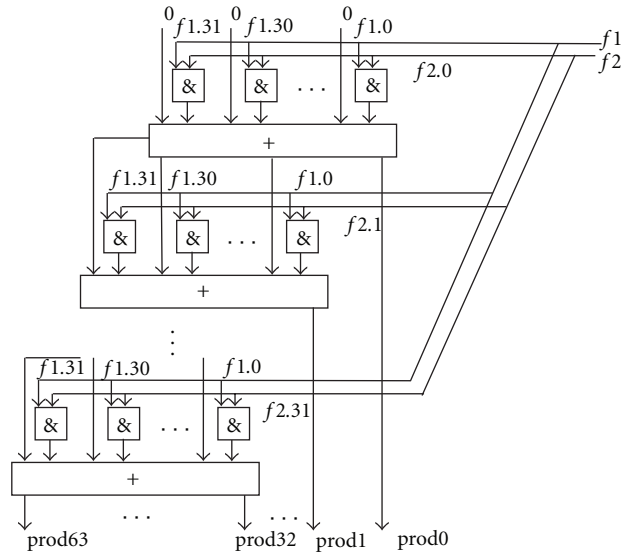


Figure 2: Structure of a field multiplier.

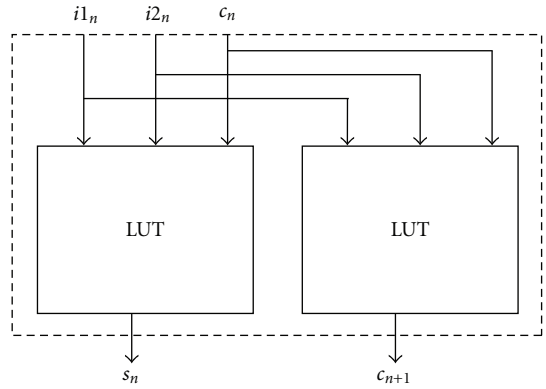


Figure 3: Optimized elementary cell for three-to-four-input LUT.

U can be extracted from the data sheet while the fanout parameter which describes the number of inputs to be driven is dependent on the circuit synthesis on the FPGA. For getting the fanout parameters of the individual signals, the synthesis result of Xilinx ISE is analyzed. The following pictures show the elementary cells of the adder for different FPGAs.

Figure 3 shows the synthesis result of the basic cell of a ripple carry adder for a LUT-FPGA with three to four inputs and one output, Figure 4 shows the synthesis result of the basic cell of a ripple carry adder for a LUT-FPGA with five to six inputs and one output, Figure 5 shows the synthesis result of the basic cell of a field multiplier for a LUT-FPGA with four to five inputs, and Figure 6 shows the synthesis result for a LUT-FPGA with six to seven inputs.

Out of the synthesis result, it is obvious that the maximal size of a elementary cell is limited by the LUT fanin which calculates the carry for the next elementary cell. The elementary cell synthesized for the adder equals a radix-floor $((n-1)/2)$ adder which is shown in Figure 7.

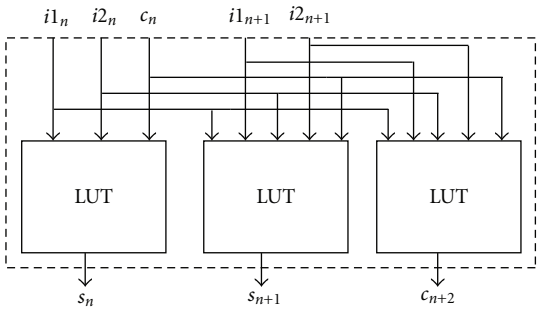


Figure 4: Optimized elementary cell for five-to-six-input LUT.

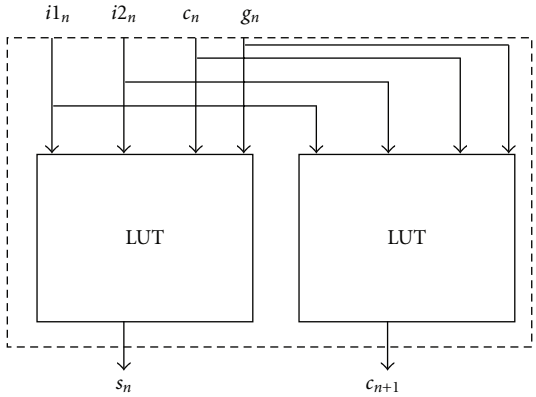


Figure 5: Optimized elementary cell for four-to-five-input LUT.

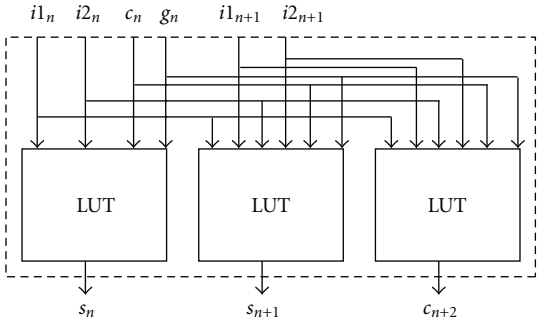


Figure 6: Optimized elementary cell for six-to-seven-input LUT.

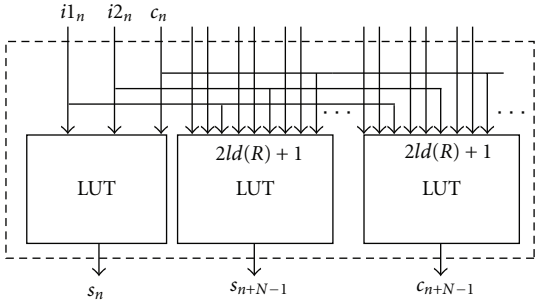


Figure 7: The high radix adder mapped to LUTs.

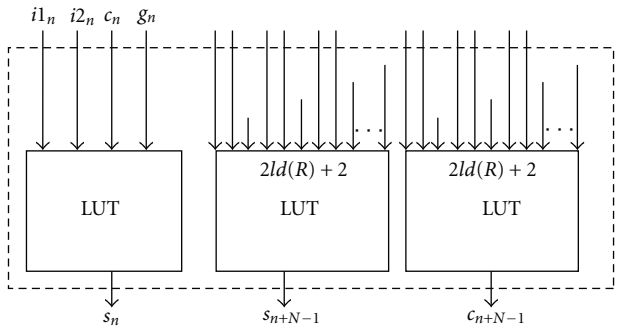


Figure 8: The high-radix adder with gate mapped to LUTs.

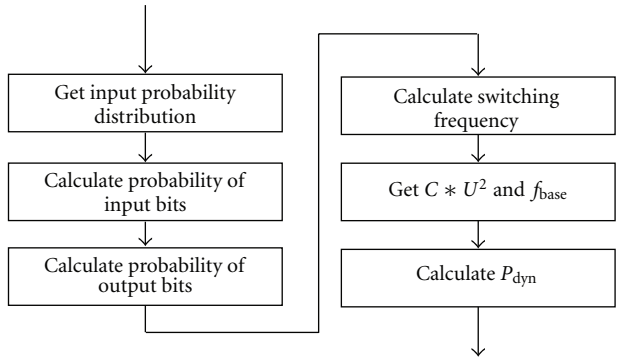


Figure 9: Setup of a new power estimation.

Similar to the high-radix adder elementary cell, the elementary cell of the multiplier consists of a high-radix adder and set of AND-Gates which are controlling the optional addition of summand no. 2 (shown in Figure 8).

With help of this circuit diagrams, it is possible to get the fan-out parameters for the power dissipation formula. The overall workflow can be seen in Figure 9 and is described as follows.

4. Improving Probability Analysis by Carry-Computation

Another possibility for the quality of the probabilities of signal occurrence is the replacement of signal by transfer of a specifically calculated value. It raises the question of which carry bit achieved through the replacement of an exact signal calculation has the best effect. The first consideration is the simplified assumption that all bits are equally distributed, then, for the probability of occurrence for each carry bit of the formula:

$$p^N(c,n) = \frac{1}{2} - \frac{1}{2^{n+1}}. \tag{12}$$

In case that the n_1th bit is calculated exactly for the signal, the maximum difference applies to

$$\begin{aligned}
\mathrm{Diff}_{c,n_1} &= \sum_{n=0}^{N-n_1} \left| p_{c,n}^{N} - /p_{c,n}^{N} \right|, \\
\mathrm{Diff}_{c,n_1} &= \sum_{n=0}^{N-n_1} 1 - 2 * p_{c,n}^{N}, \\
\mathrm{Diff}_{c,n_1} &= \sum_{n=0}^{N-n_1} \frac{1}{2}^{n}, \\
\mathrm{Diff}_{c,n_1} &= \frac{1}{2} - \frac{1}{2}^{N-n_1}.
\end{aligned} \tag{13}$$

This maximum deviation occurs with the probability of $p_{cn}^{N}a$. Hence, for the probable deviation,

$$\begin{aligned}
p_{\mathrm{Diff}_{c,n_1}} &= \mathrm{Diff}_{c,n_1} * p_{c,n_1}^{N}, \\
p_{\mathrm{Diff}_{c,n_1}} &= \left(\frac{1}{2} - \frac{1}{2^{n+1}}\right) * \left(\frac{1}{2} - \frac{1}{2}^{N-n_1}\right).
\end{aligned} \tag{14}$$

To get the maximum deviation probability, the first derivative of $p_{c\mathrm{diff},n_1}$ should be used:

$$\begin{aligned}
p'_{\mathrm{Diff}_{c,n_1}} &= \ln(2) * \frac{1}{2}^{n_1} \left[\frac{1}{2} - \frac{1}{2}^{N-n_1}\right] - \ln(2) \\
&\quad * \frac{1}{2}^{N-n_1} \left[\frac{1}{2} - \frac{1}{2}^{n_1+1}\right] \neq 0.
\end{aligned} \tag{15}$$

It can be seen from the equation that it must apply for a zero:

$$\begin{aligned}
N - n_1 &\stackrel{!}{=} n_1 + 1, \\
n_1 &= \frac{N-1}{2}.
\end{aligned} \tag{16}$$

With an additional graphic, it can be seen that it is found at the point where only the maximum of the function is located. Now it is possible to find the point in the equation that determines the maximum:

$$\max\left(p_{\mathrm{Diff}_{c,(N-1)/2}}; \mathrm{Diff}_{c,(N-1)/2} * p_{c,(N-1)/2}^{N}\right) \leq 0.25. \tag{17}$$

At this point it becomes clear why the exact calculation of the carry is not worthwhile: the probability of occurrence of a input signal bit is already, by assumption, 0.5 and can be determined without calculation; the calculation of a carry must be made in each run of the adder and helps only to avoid errors in average of 0.25. The distribution of the most probable estimation error for different length adders is presented in Figure 10. The calculation of a carry bit, however, is costly. The n_1th carry is calculated as follows:

$$\begin{aligned}
\mathrm{Carry}_{n_1} &= ([\mathrm{Summand\ 1} \bmod (1 \ll i)] \\
&\quad + [\mathrm{Summand\ 2} \bmod (1 \ll i)]) \div (1 \ll i),
\end{aligned} \tag{18}$$

the total of

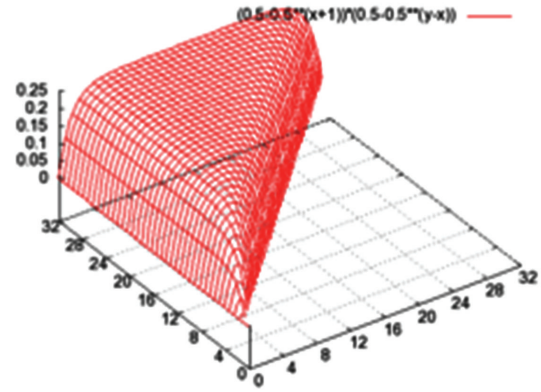


Figure 10: Distribution of the most probable estimation error for n-bit adder.

(i) 3 identical left-shift operations

(ii) 1 division

(iii) 2 modulo (division with remainder).

5. Interface: Power Loss Estimation

In order to generate as little code overhead, a basic C data type was created for unsigned integers `unsigned int` for the equivalent class `sc_int_power`, which is equipped with the same operator overloads like the original data type. These include all the arithmetic operators that are required for signal processing (addition, subtraction, multiplication, division), the assignment operator, and the operator for streaming output with `cout`. In addition, a cast operator for `unsigned int` casting in a safe situation in which only the numeric value of the `sc_int_power` object must be such as that with an array indexing. The real power estimation takes place only in the multiplication and addition operators, because these functions are the main focus of this work. The functions occurring in this energy consumption model are stored by the class static member `power`.

To take advantage of the presented class `sc_int_power` in a SystemC project, it should be compared to a standard implementation requiring the following additional code fragments:

(1) include the header file `power.h`;

(2) call the static function `set_calc` to indicate the accuracy of calculations;

(3) call the static function `set_mu` set parameters for the estimation;

(4) call the static function `set_lut` to adjust the power estimation of the target FPGA.

Anything else due to the replication of the functionality of `unsigned int` in `sc_int_power` must be ignored. It is also possible to define a simple design already in the VHDL source code with pragmas, such as the type `integer` in the types `sc_int_power` by the modified V2SC during translation. Instead of the function calls in an equivalent

SystemC project boundary bonds are defined by the above pragmas as follows:

(1) `set powerestimation_lut number;`

(2) `set powerestimation_mu number;`

(3) `set powerestimation_calc 0 or CALC_ALL;`

(4) `powerestimation on:` all of the following integers in `sc_int_power` translated;

(5) `powerestimation off`: all of the following integers in `int` translated.

These pragmas are recognized when translated from VHDL to SystemC and automatically converted into the corresponding SystemC functions. By this procedure, additional user settings are generated into the source code.

5.1. The Class prob_bit. The class `prob_bit` implements the functions to estimate the amount of time for the inputs, outputs, and internal signals of a 32-bit adder to be high and the estimation of the switching frequency. To fill the individual estimated, six arrays are implemented:

(1) `input` contains the estimation of bit = "1" for input bits;

(2) `input_rising_edges` contain the estimation for the switching frequency of the input;

(3) `output` contain the estimation of bit = "1" for the sum bit;

(4) `output_rising_edges` contain the estimation for the switching frequency of the sum bits;

(5) `signal` contains the estimation of bit = "1" for the carry bit;

(6) `signal_rising_edges` contains the estimation for the switching frequency of the carry bit.

To store a new estimate for bit = "1" in each field, the functions `add_input_prob`, `add_output_prob` and `add_signal_prob` provided that again over the functions `get_input_prob`, `get_output_prob`, and `get_signal_prob` can be retrieved. To simplify these getter and setter methods, the class has the static methods `in1`, `in2`, `carry` and `out`, which return the corresponding array index in the fields. For example, an estimate probability of bit = "1" for the 3rd bit of the adder `adder1` is set to 0.5 with the following call:

```
adder1.add_input_prob(prob_bit::in2(3),0.5).
```

The estimated switching frequencies on the power dissipation of the adder can have the class of functions `input_weight_function`, `signal_weight_function`, and `output_weight_function`, corresponding fan-ins for the individual inputs, outputs, and signal. By multiplying the switching frequencies associated with the switching frequency, we have the following relationship:

$$P_v = C_{\mathrm{LUT}} * V^2 * (f * \mathrm{fanin}). \tag{19}$$

Thus, all weighted switching frequencies must be multiplied only by the same factor in order to infer the loss of performance. The actual value for each weighting function depends on the specific type of LUT-optimized design.

The weighting factors provided with the switching frequencies can be accessed through the functions `get_input_power`, `get_output_power`, and `get_signal_power`. The parameters of these functions can also have the static function `in1`, `in2` be `carry` and out to select the desired bits.

These functions only refer to individual bits of each type. A new group of functions was defined to access the accumulated probabilities of occurrence of bit = "1" and the weighted switching frequencies. The functions are named following the same pattern as the getter and setter methods:

(1) `get_weighted_input_sum` accumulates all the probabilities of occurrence of the inputs;

(2) `get_weighted_input _power_sum` accumulates all the switching frequency of the inputs;

(3) `get_weighted_signal_sum` accumulates all the probabilities of occurrence of the signals;

(4) `get_weighted_signal_power_sum` accumulates all the switching frequency of the signals;

(5) `get_weighted_output_sum` accumulates all the probabilities of occurrence of the outputs;

(6) `get_weighted_output_power_sum` accumulates all the switching frequency of the outputs.

It is also possible to accumulate over the functions `get_weighted_all_sum` and `get_weighted_all_power_sum` all weighted probabilities of occurrence or all the weighted switching frequencies. For further support of statistics on different `prob_bit` objects, two other functions were created for each port and signal to return the probability of occurrence and the switching frequency, as follows

(1) `print_input_all_prob` returns the probability of occurrence of all inputs;

(2) `print_input_all_power` outputs the switching frequency of all inputs;

(3) `print_signal_all_prob` returns the probability of occurrence of all signals;

(4) `print_signal_all_power` outputs the switching frequency of all signals;

(5) `print_output_all_prob` returns the probability of occurrence of all outputs;

(6) `print_output_all_power` returns the frequency switching of all outputs.

The class uses a custom constructor with three parameters for the transfer of the number of input and output signals, which also allows the formation of any combinational logic result.

5.2. The Class exact_bit. The class `exact_bit` implements functions for the accurate determination of probability of occurrence and switching frequency of 32-bit adders and is analogue of the class `prob_bit` which estimates these parameters. In order to not duplicate code and to keep

the naming of the methods most consistently, the class exact_bit inherits from the parent class prob_bit and can largely use the methods of the parent class. The difference from the parent class makes itself felt in the determination of the exact switching frequency. During the estimation in the class prob_bit, an estimation of the switching frequency can be made directly from occurrence probability, this is not an exact determination because positive edges can be made only in the context of the last state of the adder. For this reason, the class attribute last_state holds a pointer to an associated exact_bit object that stores the last state of the input and output signals. With this additional information, it is now the exact determination of the number of positive edges occurred and therefore the exact determination of the switching frequency possible.

The determination of the accumulated amount of time in which one remains on high bit (equivalent to the probability of occurrence) is also possible without the context of the last condition; therefore the implementation of the class prob_bit is used. All other functions such as statistical functions to accumulate or print functions can also be used without modification, as there are the fields' input, input_rising_edges for the stored values to access from the direct calculation.

5.3. Addition Class. The class addition provides the methods for estimation and exact determination of the probabilities of occurrence and switching frequencies of individual bits in a prepared 32-bit ripple-carry adder, which are used in the class sc_int_power. Addition has the functions power_add_approx2, power_add_approx2_complex, and power_add_exact2, where the first two estimate the switching frequency and the last ones accurately determine the switching frequency of the individual inputs, broadcast, and outputs.

The two estimators power_add_approx2 and power_add_approx2_complex differ in the specification of the probability of occurrence of the input bits. For the case when the adders are used as a single unit, it can be assumed as a multiplier, according to the principle of indifference when all input vectors are equally distributed within an interval. The function power_add_approx2 can be used to get the average of the input vectors and a statistic value that the prob_bit object returns. There is the case when the adder is part of a multiplier, the perception of the equal distribution is no longer made, and the function power_add_approx2_complex is used. This function takes an input statistic in the form of a prob_bit counter object and returns the output statistics again in a prob_bit object.

The function for accurate calculation of the switching frequency power_add_exact2 was implemented, two terms for the associated switching frequencies, whilst the output statistics are in an exact_bit object are used as parameters. The exact determination of the required associated switching frequency is associated to exact_bit object with the latest state of the adder stored by the class in the private attribute last_state.

5.4. Multiplication Class. The class Multiplication provides methods for accurate determination and estimation of the probability of occurrence and the switching frequency in a 32-bit multiplier, which are required in the parent class sc_int_power. The class has the functions power_multiply_approx2 for the estimated and the power_multiply_exact2 for accurately determined switching frequency. The parameters of both functions are analogous to the methods of the class Addition. The estimator gives the mean value for the factors, while the function is used to the exact determination of the two factors to be multiplied. Because of the considered carry of the multiplier, the class Addition takes into account the wiring structure and take intermediate AND gates completely.

5.5. Random Generator for gcc. First, to test the estimation of the classes Addition and Multiplication, the built-in random generator of C was used with the function rand from stdlib library. In the first test the estimations had similar variations (about 10%) to the exact values calculated as in the subsequent tests with the Mersenne-Twister random number generation. Due to the fact that the C standard requires no algorithm for random number generation, the author does not know which algorithm is used in the current implementation by gcc.

5.6. Random Generator LFSR. In a second test the classes Addition and Multiplication were used for a linear feedback shift register. In the present implementation, the first 32 CRC polynomials were stored; that is, it can pseudo-randomize sequences to generate numbers in intervals up to 2^{32}. Tests, however, showed that the generated pseudo-random numbers are distributed very unevenly, most obviously in direct comparison with the C-random number generator or the Mersenne-Twister. Due this statistical property, this random number generator was considered unsuitable and was not included in further tests.

5.7. Mersenne-Twister Random Number Generator. The GNU suite delivers a Mersenne-Twister random number generator. This type of random generator works on an extremely long interval of $2^{19937-1}$ and its most important feature is the uniform distribution of all output bits. Due to these excellent statistical properties, the Mersenne-Twister random number generator has been selected as a reference for testing the power loss estimation. In a comparison to integrated random in C, the Mersenne-Twister cuts marginally better due to the fact that in the random number generator rand of C, the least significant bits are not equally distributed.

5.8. Random NDIST Based on gcc. To test the implementation of the power estimation with normally distributed random variables, the C-function NDIST was used, which approximates to function rand, but with uniformly distributed random numbers. With the aid of the central limit theorem, which states that the mean of a sufficiently large number of independent random variables, each with finite mean and variance, will be approximately normally

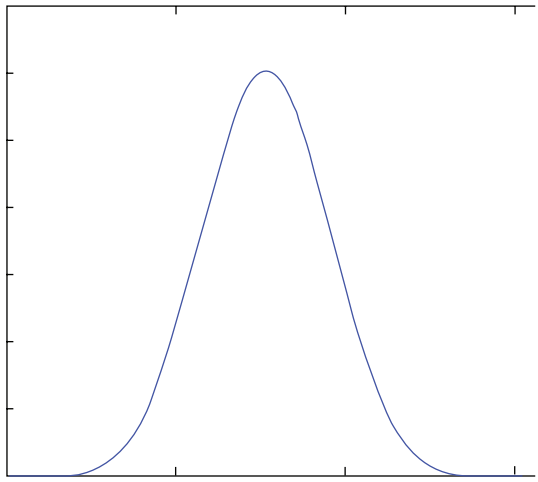


Figure 11: Normal distribution.

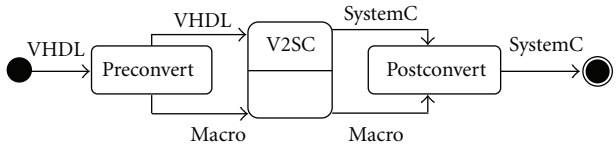


Figure 12: Combination of wrapper and V2SC.

distributed, six different random numbers are added and displayed as a normally distributed random number. The actual number of six random numbers to be accumulated was determined experimentally. The chart of Figure 11 shows the theoretical distribution of the generated numbers in this six-accumulated random variable.

5.9. Generating a Large Random Number Set. The problematics when comparing different implementations of the estimation with the classes `Addition` and `Multiplication` in comparison with the VHDL implementation are to regenerate the random numbers used in each run. For this reason a common set of uniform distribution of $2 * 1000$ random numbers 0–1024 was generated and from this set all the relevant statistical characteristics of was calculated the probability of occurrence of individual bits. But the numbers from 0 to 1, which are the basis for determining the switching frequency, change. In addition to this, statistical characteristics were raised to what extension they can differ from the requirement to assess the validity of the estimation result.

6. Module Translation and Integration

An integrated system simulation can be accomplished in homogeneous or heterogeneous environments. The adoption of homogeneous system simulations has several advantages: (i) it achieves faster simulation since synchronization tasks between different simulation kernels can be omitted; (ii) the usage of SystemC instead of for example, VHDL and the possibility to simulate models on higher abstraction levels further increment the simulation speed and additionally; (iii) since SystemC models are inevitable for system simulations and RTL models are necessary for synthesis flows, a module translation helps in keeping synchronism among the different module implementations and reduces error sources. On the other hand, this strategy requires a module translator. The VHDL to SystemC translator (V2SC) that was used in this work as basis has been developed by [19].

Table 1: V2SC features.

Feature	Included
If	Yes
While	Yes
For	Yes
Function	Yes
Procedure	Yes
Packet	Yes
Record	Partly
Array	Partly

Table 2: V2SC + wrapper extras.

Feature	Included
Alias	Yes
Cast functions	Yes
Generate	Yes
Port map	Yes
Records	Yes
Arrays	Yes
Configuration	No
File IO	No

It was designed for SystemC 1.0 and basically supports constructs that can be directly translated from VHDL to SystemC. In this work, the translator has been extended to be compatible with the actual SystemC version 2.2 and to translate synthesisable VHDL constructs. Table 1 presents the features provided by V2SC and Table 2 shows a selection of extensions included as a contribution of this work. Further convertible VHDL syntax elements are listed in [20]. The converter has been verified against a set of VHDL modules, among them a IMDCT (for calculating the inverse modified discrete cosine transform) module including an AHB master and a APB slave interface with a complexity of >5000 lines of code, a FFT module (for calculating the fast Fourier transform) with about 250 lines of code, and a GCD module (for calculating the greatest common divisor) with about 100 lines of code.

The extensions are implemented with the compiler building tools flex and bison in the same way as the original converter. Furthermore the macroprocessor m4 and the text manipulation tool sed were used. The whole extension is build as a wrapper around V2SC (Figure 12) and is divided in two parts. A preconvert filter simplifies syntax elements by converting them to V2SC compatible constructs, or the filter masks constructs such that they can pass the V2SC

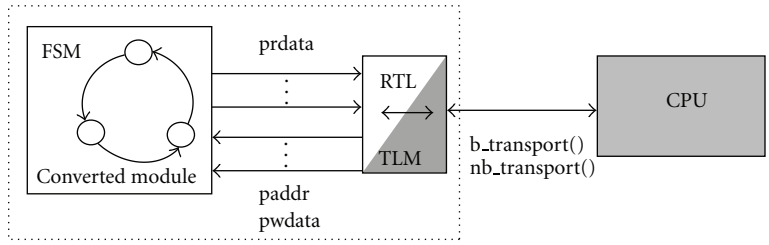


Figure 13: Interface conversion for generating a heterogeneous accuracy simulation model.

Table 3: Simulation speed comparison.

SystemC	SystemC with `int`	SystemC with `sc_int_power` (approx.)	SystemC with `sc_int_power` (exact)
Addition	1	0.5	0.03
Multiplication	1	0.5	0.001

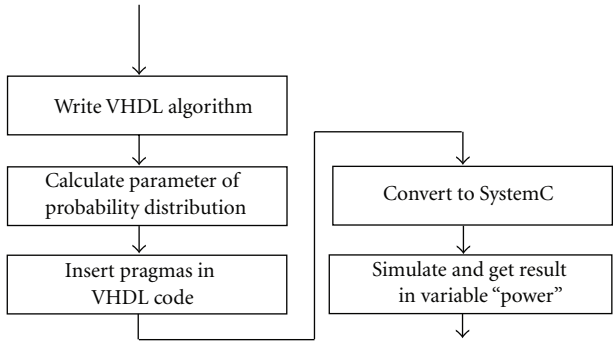


Figure 14: Power estimation workflow.

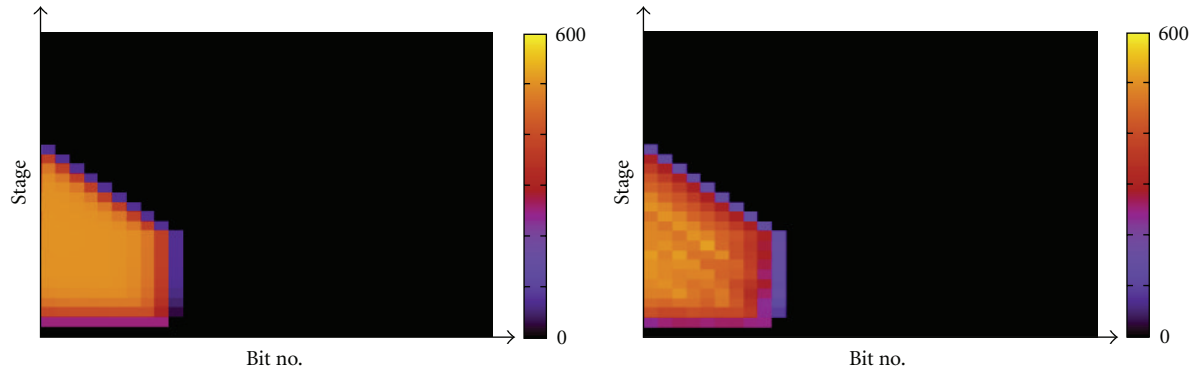
without modification, if the V2CS cannot handle them. The masked blocks are converted by the postconverter to SystemC constructs.

The converted IPs own a pin accurate RTL interface whereas the system simulator provides TLM interfaces. To decrease the design time, a protocol conversion has been integrated in the translation tool. The modified V2SC enables connecting VHDL RTL IPs to a SystemC TLM system by implementing a library that contains design patterns for pin accurate to TLM functional units according to the interface specifications. Appropriate modules are exemplary implemented for AMBA, AHB, and APB. The resulting translated modules can be directly connected to a AMBA-TLM 2.0 base system (see Figure 13).

7. Implementation of Power Estimation Features

The extension contains beside the compatibility pack, as described in Section 6, a tool for the automated integration of the previously discussed power estimation and calculation. This section explains a strategy, with which the conventional flow may be omitted if a certain decrease of accuracy can be accepted. Generally, the extension of functional parts with monitoring capabilities is an efficient way for automated system analysis. Since accurate power analysis flows are one the most time-consuming steps in the development process, their integration in a SystemC environment, with the techniques discussed in Section 3, are expected to speed up the evaluation process. Conventional accurate power analysis tools need mainly two kinds of input information. On the one hand a model of the placed and routed design allows to calculate capacitive loads of each net by evaluating fan-out numbers and the characteristics of the primitives (LUTs for FPGA technology) and with that the energy consumption per net per activity. On the other hand dynamic activities are collected during post-place and route simulation in value Change Dumps (VCD). Especially the accurate simulation with enabled signal tracing is a very time-consuming step.

The power evaluation extension of the modified V2SC can be enabled by the flag `-powerestimation` of preconvert. After setting this flag, preconvert converts the VHDL-type integer in the new-user-defined type `sc_int_power` and not to the standard SystemC-type `int`. `sc_int_power` is a new type which is constructed in this project to do all power estimation and calculation. It acts a replacement for `int` and its behaviour is similar to it. To get this functionality, all common arithmetic operators (+,−,∗,/) are overloaded while the addition and the multiplication operator definitions contain the power estimation and calculating algorithms. Additionally all arithmetic operators are overloaded for mixed-type operation with `int`. Another feature is the overloaded cast operator of `int`, which is especially for indexing arrays. With a view to the use of `sc_int_power` as a complete SystemC data type, the streaming operator ≪, the test of equality ==, the test of inequality !=, the assignment operator =, and the function `sc_trace` are overloaded. Thus in this manner defined type can be used as a template class in `sc_signal<>`, for example. To get a high prediction quality of the power estimation, the algorithms have to be parameterised. This is done by the static class methods `set_mu` (defines the average value of the input vectors) and the `set_lut` (for optimizing the design to architecture). The static class method `set_calc` defines a switch between estimation (value is zero) and calculation (value is equal `CALC_ALL` constant). The result of the calculation respectively estimation can be retrieved



(a) Approximation (b) Exact calculation

Figure 15: Toggle distribution estimation.

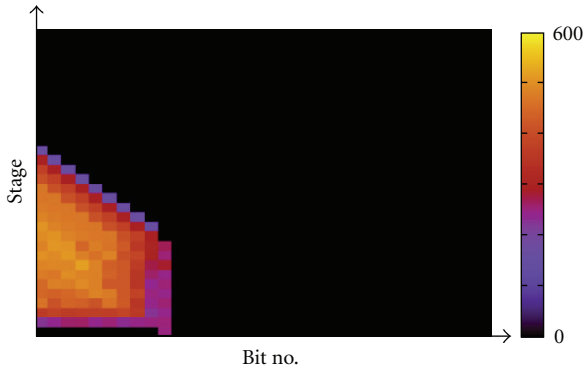


Figure 16: Reference from XPA tool.

from the static variable `power`. For improvement of the manageability of the power analyser tool the parametrisation of the algorithms is also possible in VHDL code by special comments called pragmas. These pragmas consist of the prefix "powerestimation" followed by the parametrisation function (i.e., `set_calc`, `set_mu` or `set_lut`) and the value to set. The following example shows the proceeding.

The original VHDL code:

```
(1) signal a,b,c: integer;
(2) --powerestimation set_lut 5
(3) --powerestimation set_calc 0
(4) --powerestimation set_mu 10.0
(5)
(6) c<=a+b.
```

Is translated in SystemC code:

```
(1) #include "power.h"
(2) ..
(3) sc_int_power a,b,c;
(4) c=a+b;
(5) //the powerestimation result
(6) //is stored in sc_int_power::power
(7) return 0;
```

The whole workflow is like that presented following that Figure 14.

8. Results

8.1. Accuracy of Toggle Estimation. The accuracy of the power estimation algorithms in `sc_int_power` to the also implemented exact calculation is about 13% for a average of more than 100 single additions or multiplications. Figure 15 shows the estimated toggle counts on the left and the exact calculated toggle count on the right of the partial sums (summands no. 1 of each stage) for a data set of 1000 two factors namely summands which are equally distributed in the range between 0 and 1024.

In comparison to the Xilinx XPower tool (see Figure 16), an almost similar accuracy was reached. On the following picture is the count of toggles for the partial sum bits visualized.

8.2. Speed Improvement. The integration of the proposed power estimation slows the calculation in contrast to the calculations on the standard type `int` about the factor two, but the also implemented exact power calculation slows the additions and multiplications about the factor 32, namely, 1024 (see Table 3). But the estimation is only slightly worse than the exact determination of the power dissipation; however the estimation is about the factor 16, namely, 512 faster. In sum this means that the proposed estimation is a good tradeoff between accuracy and simulation speed.

9. Conclusion and Future Work

This work proposes a methodology for switching activity estimation, taking into account the underlying hardware structure. The methodology has been exercised for MAC units. Different FPGAs have been used to show the portability of the method to other technologies. The loss of accuracy of 13% in the case of the MAC unit compared to post-place

and route simulation results comes along with a simulation speed up of a factor up to 1024. The transparent implementation of that methodology into a VHDL to SystemC converter further accelerates and eases the development process. The general approach of the methodology can be also applied to other regular computation structures. With the implementation of further computational units (e.g., dividers or other adder architectures) and the support for other FPGA architectures, an analysis of complex data paths and a faster evaluation of design alternatives are envisioned.

References

[1] N. Dhanwada, I. C. Lin, and V. Narayanan, "A power estimation methodology for SystemC transaction level models," in *Proceedings of the 3rd IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and Systems Synthesis (CODES+ISSS '05)*, pp. 142–147, September 2005.

[2] N. Dhanwada, R. A. Bergamaschi, W. W. Dungan et al., "Transaction-level modeling for architectural and power analysis of PowerPC and CoreConnect-based systems," *Design Automation for Embedded Systems*, vol. 10, no. 2-3, pp. 105–125, 2005.

[3] M. Lang, "System C for Embedded System Design," Seminar, 2006, http://dl.acm.org/citation.cfm?id=339657.

[4] S. Boukhechem and E. B. Bourennane, "TLM platform based on systemC for STARSoC design space exploration," in *Proceedings of the NASA/ESA Conference on Adaptive Hardware and Systems (AHS '08)*, pp. 354–361, June 2008.

[5] S. Hauck and A. Dehon, *Reconfigurable Computing the Theory and Practice of FPA-Based Computing*, Elsevier, 2008.

[6] N. Voros, A. Rosti, and M. Hübner, *XDynamic System Reconfiguration in Heterogeneous Platforms*, Elsevier, 2009.

[7] M. Kuehnle, A. Wagner, and J. Becker, "A statistical power estimation methodology embedded in a SystemC code translator," in *Proceedings of the 24th Symposium on Integrated Circuits and Systems Design (SBCCI '11)*, pp. 79–84, IEEE Computer Society, 2011.

[8] Forte, "Cynthesizer," 2011, http://www.forteds.com/products/cynthesizer.asp.

[9] M. Graphics, "Catapultc," 2011, http://www.mentor.com/esl/catapult/overview.

[10] Cadence, "C-to-silicon compiler," 2011, http://www.cadence.com/products/ sd/silicon compiler/pages/default.aspx.

[11] N. Bombieri, "Hif suite 2.0: hdl translating and manipulation tools," 2009, http://hifsuite.edalab.it/.

[12] University of Cincinnati, "Savant," 2011, http://www.clifton-labs.com/savant/.

[13] M. e. a. Bocchi, "A system level IP integration methodology for fast SOC design," in *Proceedings International Symposium on System-on-Chip*, pp. 127–130, 2003.

[14] G. B. Vece and M. Conti, "Power estimation in embedded systems within a SystemC-based design context: the PKtool environment," in *Proceedings of the 7th Workshop on Intelligent Solutions in Embedded Systems (WISES '09)*, pp. 179–184, June 2009.

[15] M. Giammarini, M. Conti, and S. Orcioni, "System-level energy estimation with Powersim," in *Proceedings of the 18th IEEE International Conference on Electronics, Circuits and Systems (ICECS '11)*, pp. 723–726, December 2011.

[16] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," *ACM SIGARCH Computer Architecture News*, vol. 28, no. 2, pp. 83–94, 2000, http://portal.acm.org/citation.cfm?id=339657.

[17] G. S. Silveira, A. V. Brito, and E. U. K. Melcher, "Functional verification of power gate design in systemc RTL," in *Proceedings of the 22nd Symposium on Integrated Circuits and Systems Design (SBCCI '09)*, I. S. Silva, R. P. Ribas, and C. Plett, Eds., ACM, September 2009.

[18] A. V. Brito, M. Kühnle, M. Hübner, J. Becker, and E. U. K. Melcher, "Modelling and simulation of dynamic and partially reconfigurable systems using System C," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI '07)*, pp. 35–40, 2007.

[19] U. Tuebingen, 2001, http://www-ti.informatik.uni-tuebingen.de/systemc/.

[20] A. Wagner, *Diplomarbeit Randbedingungen der HW-Modellierung auf RTL-und Systemebene*, 2011.

Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2012, Article ID 786205, 17 pages
doi:10.1155/2012/786205

Research Article

An Optimization-Based Reconfigurable Design for a 6-Bit 11-MHz Parallel Pipeline ADC with Double-Sampling S&H

Wilmar Carvajal and Wilhelmus Van Noije

Laboratory of Integrable Systems (LSI), Polytechnic School, University of São Paulo, 05403-900 São Paulo, SP, Brazil

Correspondence should be addressed to Wilmar Carvajal, wilco@lsi.usp.br

Received 17 February 2012; Accepted 22 May 2012

Academic Editor: Alisson Brito

This paper presents a 6 bit, 11 MS/s time-interleaved pipeline A/D converter design. The specification process, from block level to elementary circuits, is gradually covered to draw a design methodology. Both power consumption and mismatch between the parallel chain elements are intended to be reduced by using some techniques such as double and bottom-plate sampling, fully differential circuits, RSD digital correction, and geometric programming (GP) optimization of the elementary analog circuits (OTAs and comparators) design. Prelayout simulations of the complete ADC are presented to characterize the designed converter, which consumes 12 mW while sampling a 500 kHz input signal. Moreover, the block inside the ADC with the most stringent requirements in power, speed, and precision was sent to fabrication in a CMOS 0.35 μm AMS technology, and some postlayout results are shown.

1. Introduction

The ADC design for a multistandard receiver system has different ways to be developed seeing that both the involved standards and the selected architecture face their own drawbacks and implementation issues. A multistandard receiver is not only a combination of isolated systems operating under each of the standards, but a system capable of working in an efficient way under those dynamic conditions. To do that, some desired capabilities are reconfigurable computing and the possibility of sharing and reusing as many blocks as possible between the operation modes.

The time-interleaved pipeline architecture is frequently used to satisfy the previous requirements in high speed, moderate resolution applications [1–3]. Its main advantage is the flexibility, hence different number of time-interleaved branches and pipeline stages can be enabled/disabled to configure variable resolution and sampling frequency, thus leading to a reconfigurable system. Figure 1 shows a 2-channel, 4-stage version of the architecture, which could provide 12 bits @ 2.75 MS/s and 6 bits @ 11 MS/s for a GSM/Bluetooth receiver. There are, however, some drawbacks related to the parallelism of time-interleaved pipeline ADCs, such as channel offset, gain and timing mismatch. A front-end sample and hold (S&H) circuit is the most straightforward way to avoid timing skew between channels, as shown in Figure 1 [3]. After this S&H block operating at the full-sample rate of the converter, input signals are not anymore continuous. Thus, exact sampling moments of the first pipeline stages over these new ideally constant input signals are no longer critical. Additionally, if double sampling techniques are used, changes between sampling and hold phases are identical for both branches, reducing timing mismatch [4]. Channel offset and gain mismatch are also diminished by reusing amplifiers, making capacitor mismatch the most important error source [5].

This work presents a single-standard version of the time-interleaved pipeline ADC, which meets Bluetooth specifications while minimizing power consumption and mismatch issues inherent to the architecture. This simpler design allows going deeper into the architecture details as well as preparing it toward a multistandard implementation. In addition, optimization techniques are also applied and explored looking for an even lower power consumption in the most elementary circuits of the A/D converter [6]. Finally, as an extended version of [7], this work upgrades the

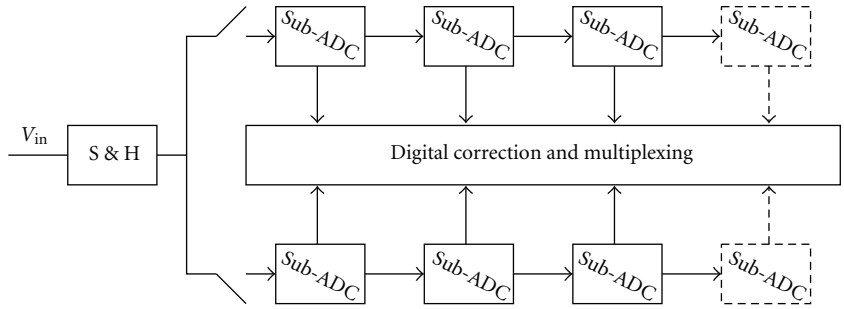


Figure 1: Time-interleaved pipeline ADC with S&H.

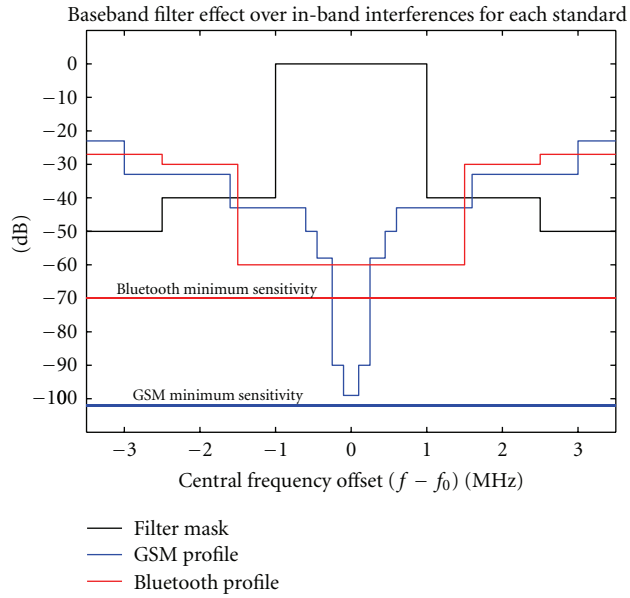


Figure 2: Blocker profiles for 2 wireless standards.

supporting material for GP and the design strategy. Also, new intermediary results are included to show how the complete ADC and the S&H work in pre- and postlayout simulations, respectively.

The architecture of the ADC is described in Sections 2 and 3 presents some of the blocks in the topology. Next, design process goes down to a lower hierarchy level, to describe the OTA and comparator designs using GP in Section 4. Simulation results are presented in Section 5 and finally, conclusions are drawn in Section 6.

Table 1: Some specifications for Bluetooth standard.

Parameter	Value
Frequency band	2400–2483.5 MHz
Constant amplitude modulation	*GFSK*
Number of channels	79
Channel separation	1 MHz
Maximum bandwidth signal	500 kHz
Sensitivity/BER	-70 dBm/10^{-3}
Channel time slot length	625 μs
Packet preamble length	4 μs

2. System and Architecture Level

The first step in the ADC design is to know the selected wireless standard. Table 1 shows the main Bluetooth specifications affecting the converter. Albeit general for the entire receiver chain, these specifications can be used in system level simulations and analysis to determine the ADC requirements, as can be seen from Table 2. Resolution specification is taken from SNR-BER graphs, along with profiles of channel adjacent interferences (see Figure 2) and design margins. Sampling frequency is derived from system level simulations taking into account settling times and preamble length. Furthermore, linearity requirements (DNL and INL) are defined to guarantee a monotonic converter [3].

As seen from Figure 1, there are 2 variables to play with in a time-interleaved pipeline architecture: the number of parallel channels and the resolution of pipeline stages. First, the inverse relation between number of channels and

Table 2: Some design specifications for the ADC.

Parameter		Value	
Power supply		3.3 V	
Maximum input bandwidth		500 kHz	
Sampling rate		11 MHz	
Resolution		6 bits	
Full-scale input voltage (V_{FS})		2 V_{pp}	
INL	DNL	<0.5 LSB	<1 LSB

Table 3: Specifications for pipeline ADC stages.

Spec	Input S&H	Stage 1 4 bits	Stage 2 3 bits
Sub-ADC error e_{ADC} (bits)	16	6	3
Offset voltage v_{offset} (mV)		31.25	125
Gain error e_G (%)	1.6	12.5	
DAC error e_{DAC} (bits)		6	
Noise level (dBc)	−34	−34	−25
Clock jitter		226 ps	

current consumption of the ADC is mainly due to double-sampling techniques and amplifier reuse. Nevertheless, mismatch issues get worse as the number of branches increase. Furthermore, the less bits every stage resolves, the more stages are needed to provide the required total resolution. This leads to a larger power consumption, yet also to looser specifications for the comparators into the sub-ADC. On the other hand, each additional bit duplicates the number of comparators and divides by two the allowed offset in the sub-ADCs of the pipeline stages. Nonetheless, this also requires less power consuming MDACs (Multiplying DAC) [5].

After this discussion as well as some parametric analysis, an architecture with 2 channels and 2 pipeline stages is proposed in Figure 3. Each parallel pipeline chain operates at 5.5 MS/s to get a total sampling rate of 11 MS/s. In spite of the first 4 bit complete stage, which includes sub-ADC and MDAC blocks, the second 3 bits stage has only the sub-ADC block. Moreover, from the 7 output bits, only 6 are effective and 1 (from the first stage) is used for Redundant Sign Digit (RSD) correction. The previous task is done with additional digital circuitry, which also combines and multiplexes the bits towards a single 6-bit digital output word.

The errors within the pipeline stage may appear in four different points, as shown in Figure 4. The sub-ADC nonidealities produce e_{ADC}, while circuit limitations of the S&H, DAC, and residue amplifier add $e_{S\&H}$, e_{DAC} and e_G components, respectively. Nonetheless, the previous three components are jointly generated by one single block: the MDAC. By doing so, error sources are identified in Figure 4, as well as the circuit performance parameters to eliminate those undesired characteristics. Some of these parameters are stage resolution, amplifier gain precision, offset voltages and noise levels, among others [3]. Therefore, taking into account linearity specifications from Table 2 and error scaling throughout the pipeline stage gains backwards (1), restrictions in similar forms to (2) can be used to determine the different stage specifications presented in Table 3.

$$e_T^2 = e_1^2 + \left(\frac{e_2}{G_1}\right)^2 + \left(\frac{e_3}{G_1 \cdot G_2}\right)^2 + \cdots + \left(\frac{e_m}{\prod_{k=1}^{m-1} G_k}\right)^2, \quad (1)$$

$$G_k \cdot e_{ADC} < \text{INL} \Big|_{\text{INL}=(1/2)\cdot \text{LSB}_{k+1}=(1/2)\cdot(V_{FS}/2^{n_k}).}^{\text{for stage } k} \quad (2)$$

3. Block Level

Going deeper into the architecture, the block details and functions are revealed. Accordingly, Figure 5 shows the details of the implemented converter. Not only are the analog blocks presented, which consist of S&H, sub-ADCs, MDACs, buffers, and bias circuits; but the digital circuitry is also unfolded in clock generation, synchronization, combination, multiplexers, and correction circuits. All of these blocks were full-custom designed and will be introduced in the next lines as well as some of their design considerations.

The S&H circuit is normally implemented with switched capacitors (SCs) architectures including an amplifier as their central component. Because of the pure capacitive load, the central amplifier employs single-stage topologies (OTAs), which are considered the fastest and most power-efficient ones [3, 5]. Though the OTA improves S&H performance, its non-idealities, including finite gain and bandwith, margin phase, slew rate, offset and noise, limit the circuit specifications. During operation, it is required only in the hold phase, thus remaining idle when sample action takes place. The double sampling technique takes advantage of this idle time for using the amplifier. Despite providing samples at double speed, power consumption remains almost unchanged since the referred power is dominated by the amplifier, which normally uses class A architectures that dissipate power even when idle [5].

The S&H schematic is shown in Figure 6, besides the signals controlling its operation. This S&H employs bottom-plate sampling with switches $S7N(P) - S8N(P)$ and phases $\phi_{1,2e}$ to reduce the component of charge-injection depending on the input signal. By using the fully differential architecture, the other constant components of error are also diminished. In addition, the shared switches $S9N(P)$ with phase ϕ at the full sample rate f_s eliminate the parallelism at the sampling instant, thus making the S&H timing-skew insensitive.

To specify the S&H block, a minimum sampling capacitance is first determined from noise (3), mismatch, and parasitic components requirement. Since input signals are stored into only one pair of capacitors at each sampling moment, and that OTA also adds thermal noise from its active devices ($\overline{v_{n,\text{amp}}}^2$), total output (and input, because it is a unity gain circuit) referred S&H noise is given by (3), where γ is a channel length dependent noise-excess factor. If assumed that OTA design guarantees a minimum input noise

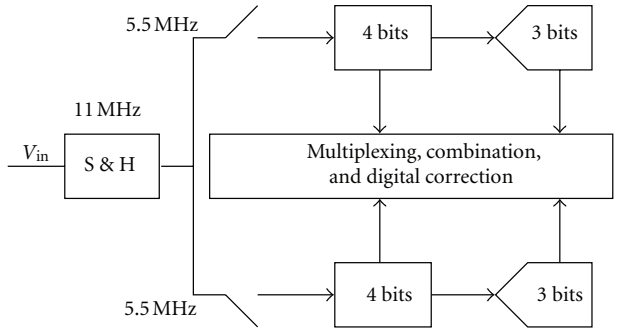


Figure 3: 2-ch x 2-st ADC for Bluetooth standard.

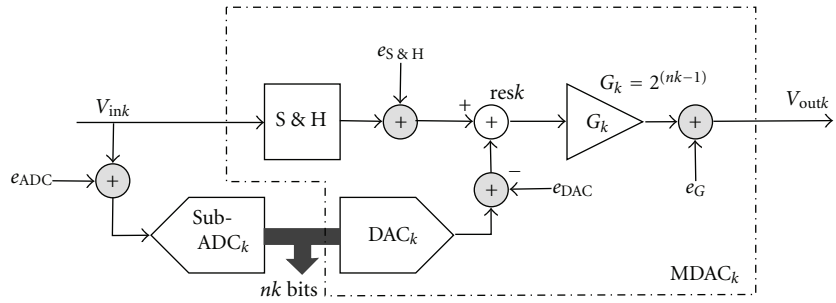


Figure 4: Error model for pipeline stage.

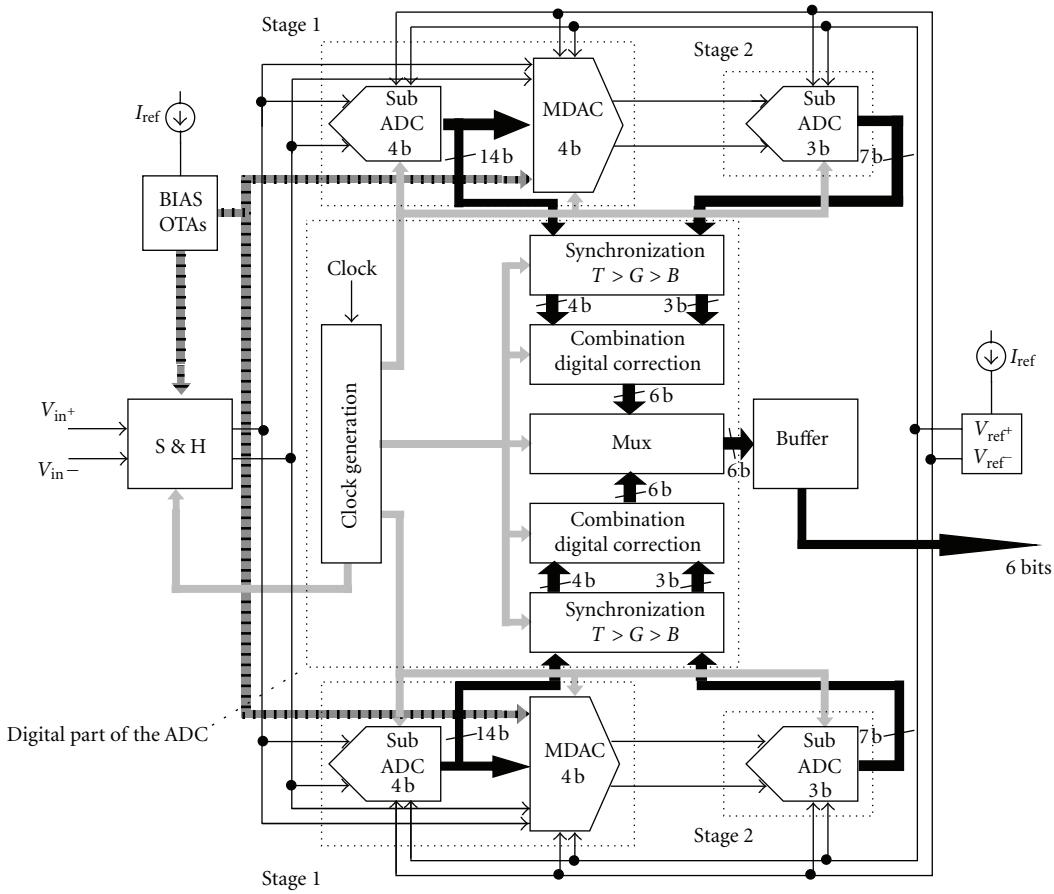


Figure 5: Detailed block schematic of the ADC.

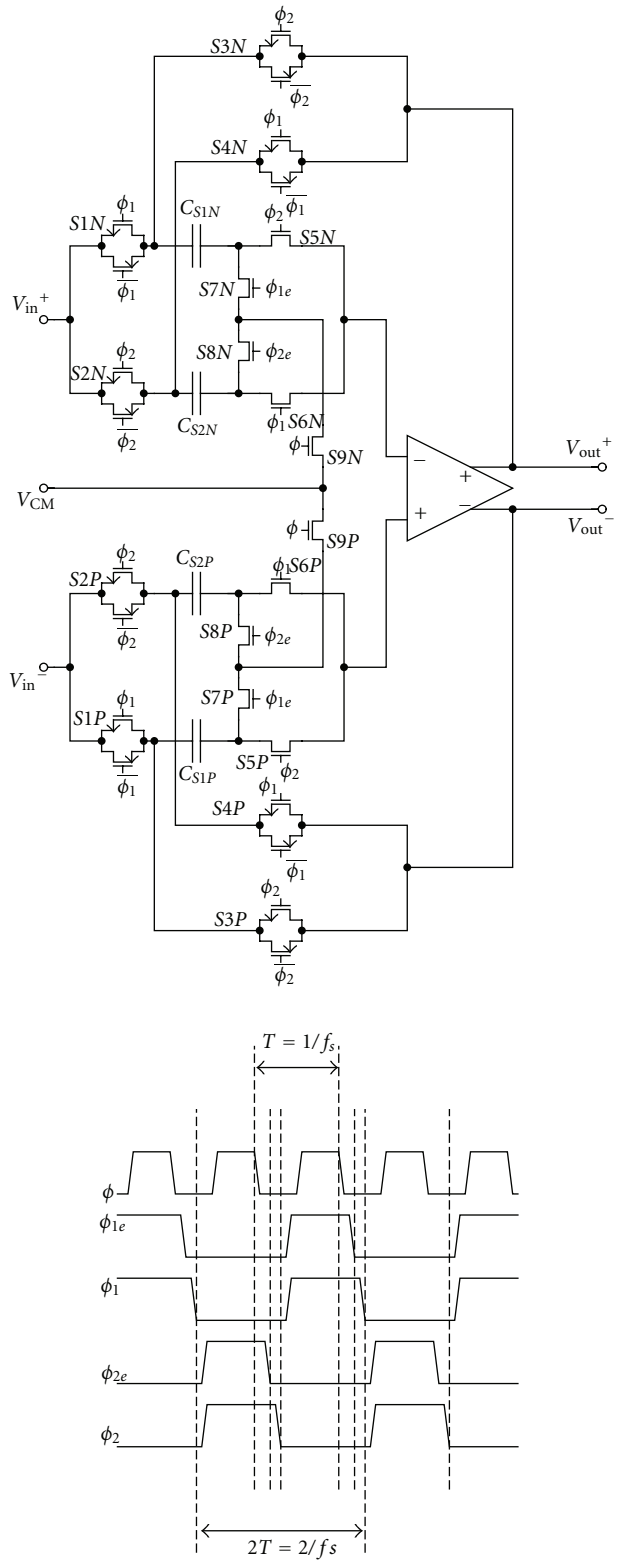


Figure 6: SC-S&H block architecture.

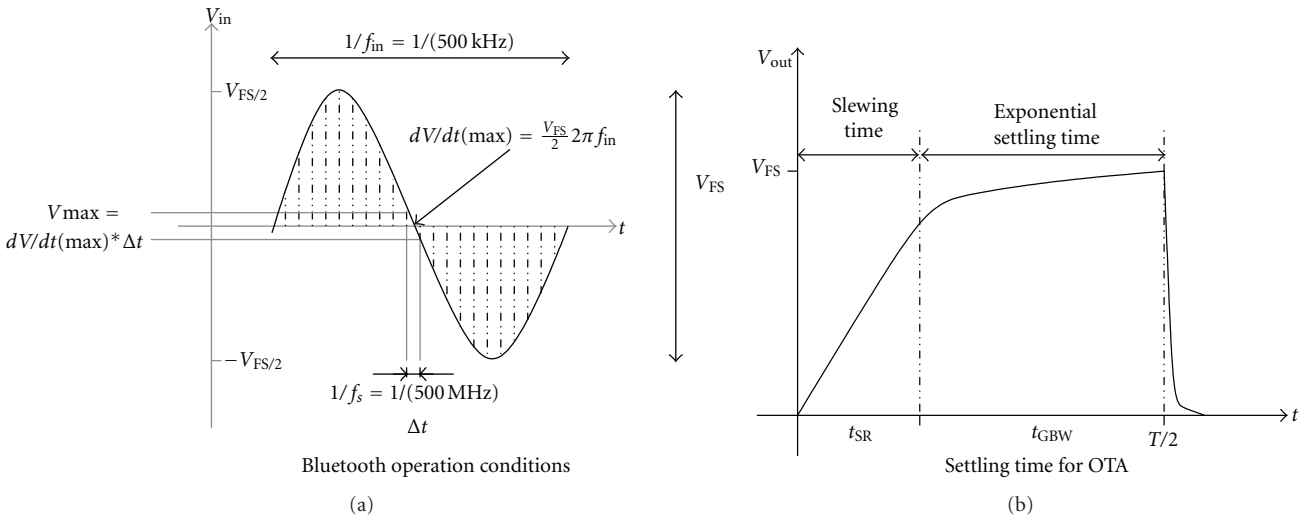


FIGURE 7: Timing details.

level at S&H operation frequency, its component in (3) may be considered negligible. As result, sampling capacitors are the dominant thermal noise source in the S&H.

$$\overline{v_{n,\text{out}}^2} = 2\gamma \frac{kT}{C_S} + \overline{v_{n,\text{amp}}^2}. \tag{3}$$

Once the MOS switches are sized in order to avoid degrading the frequency response of the amplifier or limiting the finite-bandwidth input signal [5], OTA specifications can be described. The combination of the S&H e_G specification in Table 3 and expression (4) gives the DC gain (A_o) requirement for the OTA. The transfer function in right side of (4) is obtained by applying the charge conservation principle in circuit of Figure 6 and assuming that no charge leakage occurs between phases ϕ_1 and ϕ_2, giving (5) which can be approximated as in (6) if C_{ip} is considered negligible:

$$e_G = G_{\text{S\&H-ideal}} - G_{\text{S\&H-real}} = 1 - \frac{1}{1 + (1/A_o)} = \frac{1}{1 + A_o}, \tag{4}$$

$$Q_S = Q_H \Longrightarrow V_{\text{out}} = \frac{C_S \cdot V_{\text{in}}}{C_S \cdot (1 + (1/A_o)) + C_{ip} \cdot (1/A_o)}, \tag{5}$$

$$V_{\text{out}} = \frac{V_{\text{in}}}{1 + (1/A_o)}. \tag{6}$$

When single pole model is used for OTA during hold phase, settling time for a step input voltage is determined by the unity gain-bandwidth product GBW. This assumption, however, is valid only if OTA is designed in such a way that its frequency response is close to the single pole response. Consequently, there is one dominant low-frequency pole, while the others poles and zeros lie at much higher frequencies [5]. As result, (6) can be re-written as in (7), where $pA_0 = 2\pi\text{GBW}$ with p beeing the dominant pole. By applying inverse Laplace transformation, expression (8) is obtained, where t_{GBW} is the exponential settling time for V_{out}:

$$\begin{aligned} V_{\text{out}} &= \frac{V_{\text{in}}}{1 + ((1 + (s/p))/A_o)} \\ &\simeq V_{\text{in}} \cdot \frac{1}{1 + (s/pA_o)} \\ &= V_{\text{in}} \cdot \frac{1}{(1 + (s/(2\pi \cdot \text{GBW})))}, \end{aligned} \tag{7}$$

$$V_{\text{out}} = V_{\text{in}} e^{-2\pi \cdot \text{GBW} \cdot t_{\text{GBW}}}. \tag{8}$$

Right side of Figure 7 shows t_{GBW} and t_{SR} times. Because of double sampling, S&H output has to settle before the half clock period to guarantee the sub-ADC and MDAC in the first pipeline stage has enough time to sample it. In addition, it is a good practice to reserve 1/3 of the settling time for the slewing and the rest for the GBW limited part [5]. By using the latter ideas, to achieve the DNL and INL requirement after settling, (9) needs to be satisfied if a single pole system is guaranteed by means of the OTA frequency response, that is, the unity gain-bandwidth (GBW) specification

$$\begin{aligned} & V_{\text{in-max}} e^{-2\pi \cdot \text{GBW} \cdot t_{\text{GBW}}} < \frac{1}{2}\text{LSB} \\ & \Longrightarrow \frac{V_{\text{FS}}}{2} e^{-2\pi \cdot \text{GBW} \cdot (2/3)(1/2)(1/f_s)} < \frac{1}{2} \frac{V_{\text{FS}}}{2^N} \\ & \Longrightarrow \text{GBW} > \frac{3}{2\pi} \ln 2 \cdot N f_s, \end{aligned} \tag{9}$$

$$\text{SR} = \frac{V_{\text{step-max}}}{t_{\text{SR}}} = \frac{V_{\text{step-max}}}{(1/3)(1/2)(1/f_s)} = 6 \cdot f_s \cdot V_{\text{step-max}}. \tag{10}$$

Besides the unity gain-bandwidth, the finite OTA output current to charge and discharge the capacitive load limits

settling time as well. This prevents OTA outputs to follow large voltage steps faster than its slew rate. By the way, (10) can be used to find the SR requirement taking into account that $V_{\text{step-max}}$ is the maximum step size at the OTA output and t_{SR} is shown in the right side of Figure 7. Unlike the very common assumption $V_{\text{step-max}} = V_{\text{FS}}$ in the literature [3], this work goes deeper into this specification because of the strong and direct SR effect on power consumption of the OTA. The previous assumption is valid when S&H circuit applies a reset between consecutive samples. Nevertheless, S&H of Figure 6 behaves as a track and hold topology, the output of which tracks its input during the sampling mode. In addition, the acquisition phase has been suppressed to use double sampling technique, and the S&H operates in a very similar way to a zero-order hold. For the latter kind of system, consecutive samples can be very close in amplitude depending on f_s and f_{in} ratios, as shown in the left side of Figure 7. Thus, it is evident that there is no possibility that OTA output has to follow voltage steps as large as V_{FS}. A more realistic value of $V_{\text{step-max}}$ can be obtained from the maximum derivative of a sinusoidal input signal $V_{\text{in}} = (V_{\text{FS}}/2)\text{sen}(2\pi f_{\text{in}} t)$ times the time slot between consecutive samples, as illustrated in Figure 7 [7].

The quantization process in each pipeline stage is executed by low-resolution sub-ADCs. In order to maximize the available settling time for S&H/MDAC outputs, so that delays are reduced as well as signal dependent conversion errors, the flash architecture is chosen for these blocks. The topology consists of a comparator bank followed by registers that synchronize the output bits with the system, using a carefully selected V_{LATCH} signal. Furthermore, thermometer output bits have to be converted into binary codes. Gray codification is also applied as an intermediate step to diminish spike errors from thermometer transitions. Owing to RSD correction, 4-bit sub-ADC in the first stage has 14 comparators (Figure 8), while second stage uses 7 comparators to produce 3 bits. The two main comparator specifications, offset and speed, are derived from e_{ADC} in Table 3 and timing (sampling frequency) in Figure 7, respectively.

The MDAC is the other main block within the pipeline stages, except for the last one in the chain. Its function is bringing the sub-ADC output back to analog domain, subtracting it from the previously sampled stage input, and then, amplifying the final residue, which will be used as input for the next stage (Figure 5). The MDAC is also based on an SC architecture and has a capacitor bank rather than a single sampling capacitor (Figure 9), so that all the S&H design considerations and analysis are valid, too. Using again the noise and the other stage specifications from Table 3, the different components of the MDAC architecture can be specified [5, 8].

The above-mentioned necessity of clock phases generation, bit combination and codification now brings digital circuitry into focus. The circuit in Figure 10 generates the different clock pases required by the SC circuits of Figures 6 and 9 and the control signals for the comparators and registers in Figure 8 [5]. It is a standard divide-by-2 architecture using a D-flipflop [9] and 2 cross-coupled NAND gates along with delay chains to control the phase duty cycles.

Figure 11 shows an arrangement of registers, inverters, and NAND, OR, XOR gates that codes into thermometer representation, synchronizes with an extra stages and finally applies RSD to digitally correct the output bits coming from the sub-ADC in the pipeline stages. A glitch-free intermediary Gray coding stage is included as well. RSD combination was developed with a Carry-Lookahead Adder and static logic gates were used all over the digital block.

4. GP on Circuit Level

GP is applied here to minimize power consumption and optimize performance under specific requirements, for both the OTA and comparator design presented as follows.

4.1. Geometric Programming. GP is a special kind of mathematical optimization problem in which the objective function and restrictions belong to a set of functions with a particular form, thus satisfying some specific conditions. A geometric program is itself a complex nonlinear optimization problem, however, it can be turned into a convex problem through variable changes and transformations of the related functions. Then, it can be solved by very efficient algorithms available from a number of companies and research groups working on the matter.

Despite the above benefits, GP is very restrictive about its formulation. Just monomial and posynomial expressions can be part of a geometric program. These function types are shown, respectively in (11) and (12) where $c_k > 0$, a_i is any real number and $x_1, \ldots, x_n$ are n real and positive variables. It is really important to identify which operations do not modify the original monomial and posynomial structures because GP is very restrictive in this issue

$$g(x) = c x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n}, \tag{11}$$

$$f(x) = \sum_{k=1}^{K} c_k x_1^{a_{1k}} x_2^{a_{2k}} \cdots x_n^{a_{nk}}. \tag{12}$$

A geometric program in standard form is an optimization problem with the format:

$$\begin{aligned} &\text{minimize} && f_0(x) \\ &\text{subject to} && f_i(x) \le 1, \quad i = 1, \ldots, m, \\ &&& g_i(x) = 1, \quad i = 1, \ldots, p, \end{aligned} \tag{13}$$

where f_0 is called the objective function, f_i are inequality restrictions and g_i are equality restrictions. In a geometric program in standard form as described in (13), functions f_{i} are posynomials, g_i are monomials, and x_i are the optimization variables, with the implicit constraint that the variables be positive ($x_i > 0$). In standard form GP, the objective must be posynomial (and it must be minimized); the equality constraints can only have the form of monomial equal to one, and the inequality constraints can only have the form of posynomial less than or equal to one [10, 11]. This

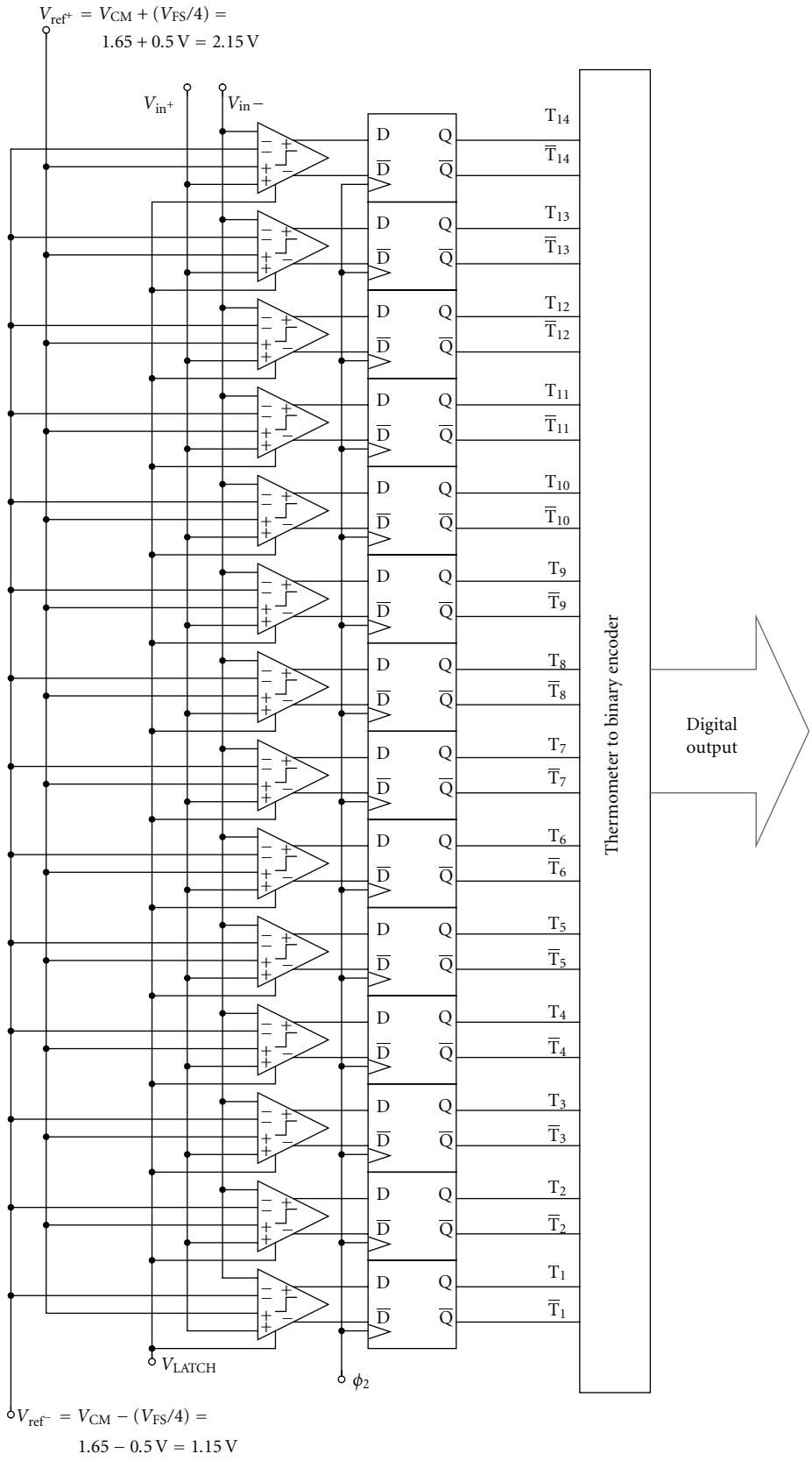


Figure 8: 4 bit sub-ADC applying RSD.

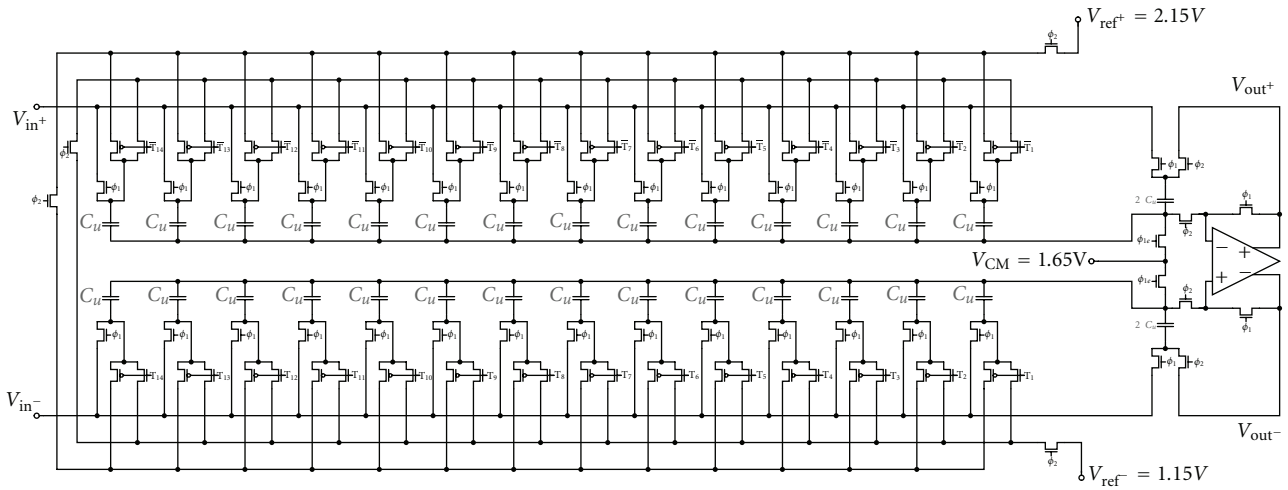


Figure 9: Multiplying DAC architecture.

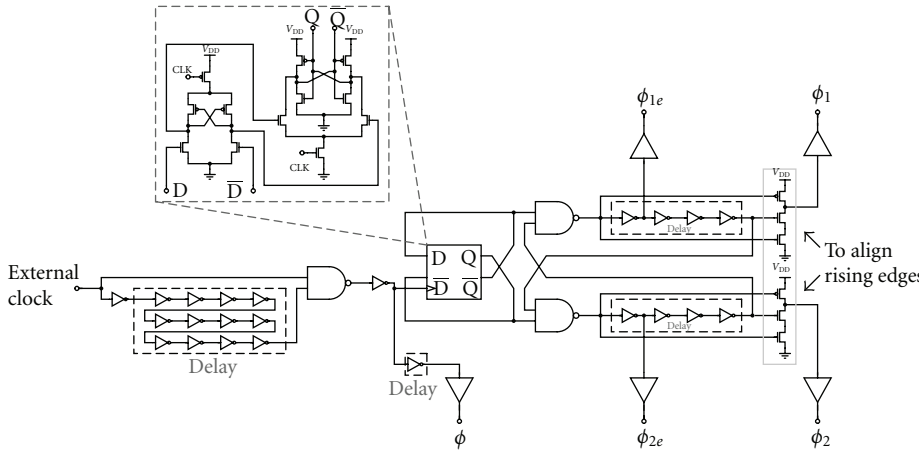


Figure 10: Clock phase generator.

GP solution is based on very efficient algorithms, specially designed for convex optimization.

By developing design via GP, a basic strategy (it requires some modifications for GP formulation incompatibilities) to obtain optimal circuits can be detailed as follows:

(1) circuit mathematical formulation in GP standard form,

(2) required transistor parameters identification and modeling,

(3) optimization file construction taking models and design specifications as inputs,

(4) results verification using a circuit simulator,

(5) new modeling regions identification after GP solution and return to point 2.

4.2. OTA and Comparator Design. In this work, fully-differential folded cascode topology was chosen for the OTA circuit, as shown in Figure 12. This architecture is preferred instead of the telescopic one because of its wider input and output dynamic ranges. These voltage swings are very important for this application because some signals may have maximum amplitudes as large as V_{FS}, mainly in the S&H. A SC-CMFB circuit controlling the output common-mode $V_{CM,OUT}$ is shown in Figure 12, too [7].

Before applying GP, OTA design space is delimited owing to offset requirements in Table 3. Accordingly, a random offset theoretical estimation is made through parametric variations of the involved transistor sizes shown as.

$$v_{off} = \Delta V_{th1} + \frac{g_{m3}}{g_{m1}}\Delta V_{th3} + \frac{g_{m9}}{g_{m1}}\Delta V_{th9} + \frac{V_{GS1} - V_{th1}}{2}\Delta K, \tag{14}$$

where ΔV_{th} and ΔK stand for threshold voltage and gain mismatch parameters, respectively, while g_m is the transconductance and V_{GS} the gate-source voltage. A prototype designed via GP and fabricated in a 0.35 μm technology is presented in [12] along with some experimental results.

Point 1 in GP methodology requires the main performance parameters to be expressed as restrictions for

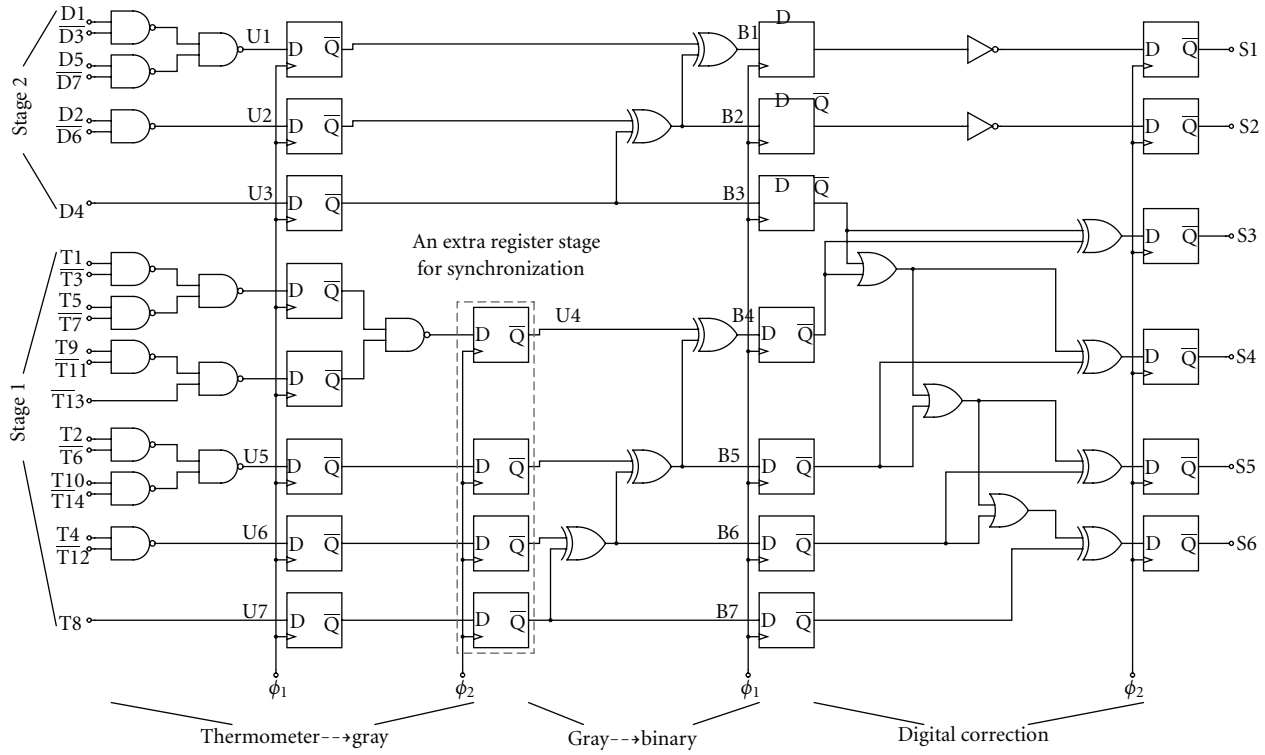


Figure 11: Combination and digital correction.

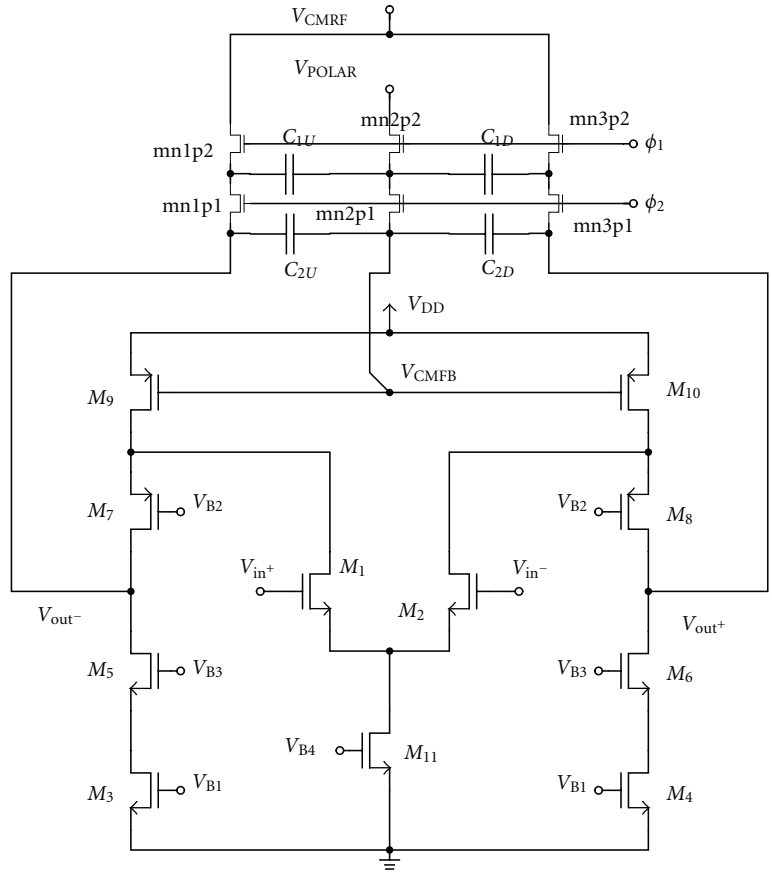


Figure 12: Folded cascode OTA architecture.

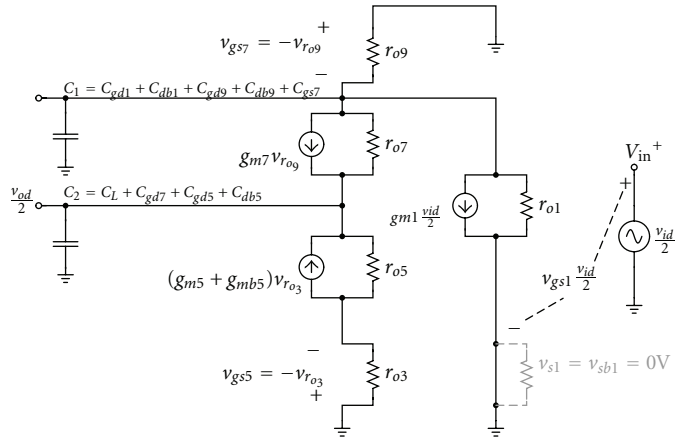


Figure 13: OTA half differential small signal circuit.

minimum values of DC gain in (15), unity gain bandwidth in (16), SR in (17), and an approximation of phase margin in (18), where ρ_2, $C_{L,\text{tot}}$ and $C_{d(M9),\text{tot}}$ are the nondominant pole and the total capacitances including parasitics at the output node and drain terminal of $M9(10)$ in the schematic of Figure 12, respectively. All these expressions were formulated from the equivalent circuit shown in Figure 13, where the body effect of transistors $M1(2)$ was ignored since its small signal source voltage $v_s \simeq 0$ (ideally balanced differential pair). A similar simplification was applied to transistors $M7(8)$ because they are PMOS that can have their own isolated N-wells in a P-substrate CMOS process. In addition, biasing and devices geometry conditions are also formulated into the GP, the objective of which is minimizing OTA power consumption established in (19)

$$\frac{g_{m1}}{(g_{m5}+g_{mb5})^{-1}g_{ds3}g_{ds5} + g_{m7}^{-1}g_{ds7}(g_{ds1}+g_{ds9})} \geq |A_{Vd\min}|, \tag{15}$$

$$\frac{g_{m1}}{C_{L,\text{tot}}} = \frac{g_{m1}}{C_L + C_{gd7} + C_{gd5} + C_{db5}} \geq \omega_{0\min}, \tag{16}$$

$$\frac{I_{M11}}{C_{L,\text{tot}}} = \frac{I_{M11}}{C_L + C_{gd7} + C_{gd5} + C_{db5}} \geq \text{SR}_{\min}, \tag{17}$$

$$\pi - \frac{\pi}{2} - \frac{\omega_0}{\rho_2} = \pi - \frac{\pi}{2} - \frac{g_{m1}}{C_{L,\text{tot}}}\frac{C_{d(M9),\text{tot}}}{g_{m7}} \geq \text{PM}_{\min}. \tag{18}$$

$$\text{Power} = 2 \cdot V_{\text{DD}} \cdot I_{M9}. \tag{19}$$

A low area, low power dynamic architecture (Figure 14) can be used for the comparators since the RSD correction loosens the offset restrictions in the sub-ADC [5]. Inner threshold generation prevents the use of the typical resistance ladder in Figure 8. Even though it suffers from offset, this can be tolerated in low-resolution applications, as 3 and 4-bit flash sub-ADCs. Threshold voltage is set by current division in the crossed differential pairs. Assuming $W_3 = W_4 = f \cdot W_1 = f \cdot W_2$, $I_{D5} = d \cdot I_{D6}$, $V_{\text{in}} = e \cdot V_{\text{ref}}$ in Figure 14, and using large signal models for transistor currents, expression (20) is obtained, with $K = \mu \cdot C_{ox}$ as the transistor gain factor.

GP is initially applied to optimize power consumption and delays in the basic comparator, which needs an external threshold voltage. Afterwards, (20) is used to establish the threshold voltages needed in the sub-ADC by modifying the transistor size and current ratios:

$$2de^2 I_6 \frac{W_1}{L} - Ke^4 V_{\text{ref}}^2 \left(\frac{W_1}{L}\right)^2 = 2I_6 \frac{W_3}{L} - KV_{\text{ref}}^2 \left(\frac{W_3}{L}\right)^2. \tag{20}$$

5. Results

Even though the complete ADC of Figure 5 was designed, only the S&H block reached the silicon fabrication phase so far. The complete converter needs a longer time to get to its finished layout and be sent to chip integration due to the size and complexity of its architecture. Therefore, this section exhibits some prelayout simulations for the Time-Interleaved Pipeline ADC from the circuit level up to the top system, and some postlayout results for the Sample and Hold block together with other auxiliary circuits to be introduced later.

First, from the circuit level, comparator yield was simulated to evaluate whether the offset of the dynamic topology would affect the required resolution. In Figure 15, each point is the result of 100 Montecarlo runs, indicating the percentage of right decisions owing to a known input of the comparator. The closer the input signal is to the threshold, the lower the yield is. Yield rises over 95% when the difference between when the difference between input and threshold voltages exceeds $\pm v_{\text{offset}}$. In a different manner, this can be understood as the probability of the comparator having an offset voltage lower than v_{offset} [13]. The offset voltage was simulated for the OTA too, as shown in the left side of Figure 16 together with an input dynamic range result, whose specified value was greater than 1 V and it requires the amplifier to be simulated in a unity feedback loop. Offset voltage is given by $v_{\text{offset}} \leq \mu \pm 3 \cdot \sigma$, where μ and σ are the mean and the standard deviation in Figure 16, respectively. Consequently, $v_{\text{offset}} < 7$ mV for the S&H OTA simulated in this case, fulfilling the requirement of Table 3.

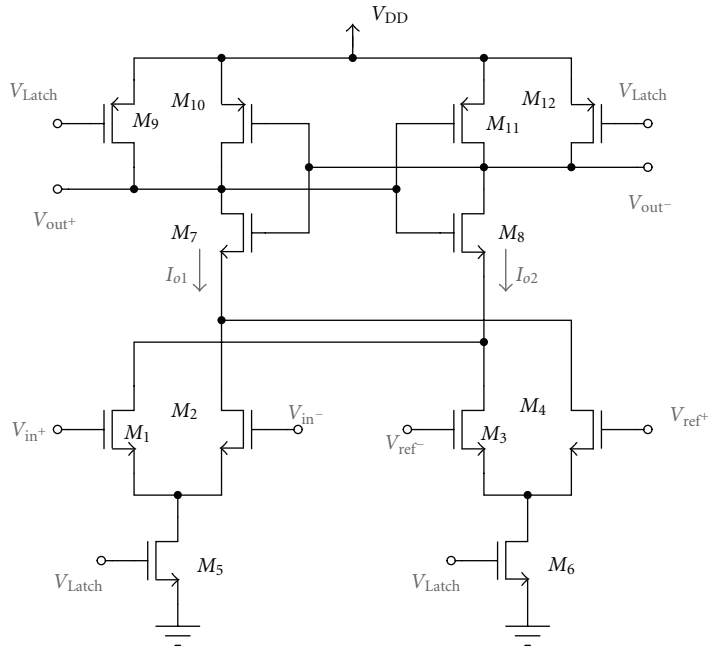


Figure 14: Comparator in the flash subADC.

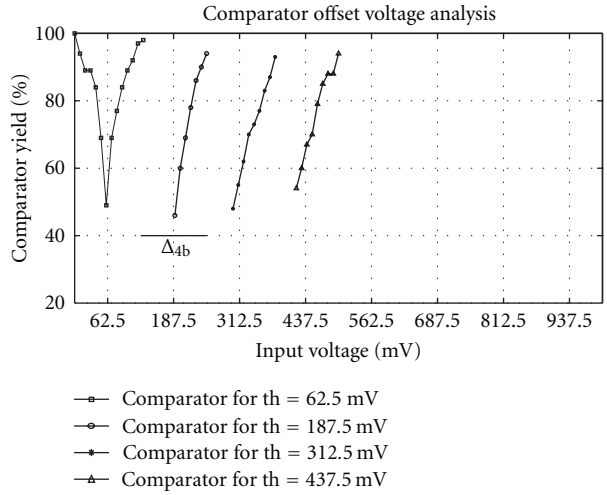


Figure 15: Comparator threshold simulations.

From the block level, the S&H circuit was simulated while sampling a slow ramp signal in Figure 17. Discrete output against analog input are shown, as well as nonideality details and block current consumption. Observed glitches come from nonlinear variation of CMOS switch resistances and reduced output resistance in OTA cascode transistors (Figures 12 and 13), since they start working close to their triode region when V_{in} approaches V_{FS}. This is why output and input DR (see Figure 16) are important specifications for the OTA design. CMOS bootstrapped switches can be used so as to further reduce those glitches.

Finally, simulations results from the complete ADC are shown in Figures 18 and 19, while sampling a maximum frequency (500 kHz) input tone using maximum sampling rate (11 MS/s). Time-interleaved pipeline operation can be observed at digital output word in Figure 19 as result of multiplexing the 6-bit outputs from 2 parallel channels, whereas its FFT transform in Figure 18 allows making an estimation of the ADC frequency characteristics. Linearity can be quantified from curves in Figure 20. Better DNL and INL measures require Montecarlo simulations, yet that would take so much longer.

Table 4 shows some results from the prelayout simulations, which characterize the designed ADC along with the other specifications presented in Table 2. It is difficult to compare these prelayout values with other works because

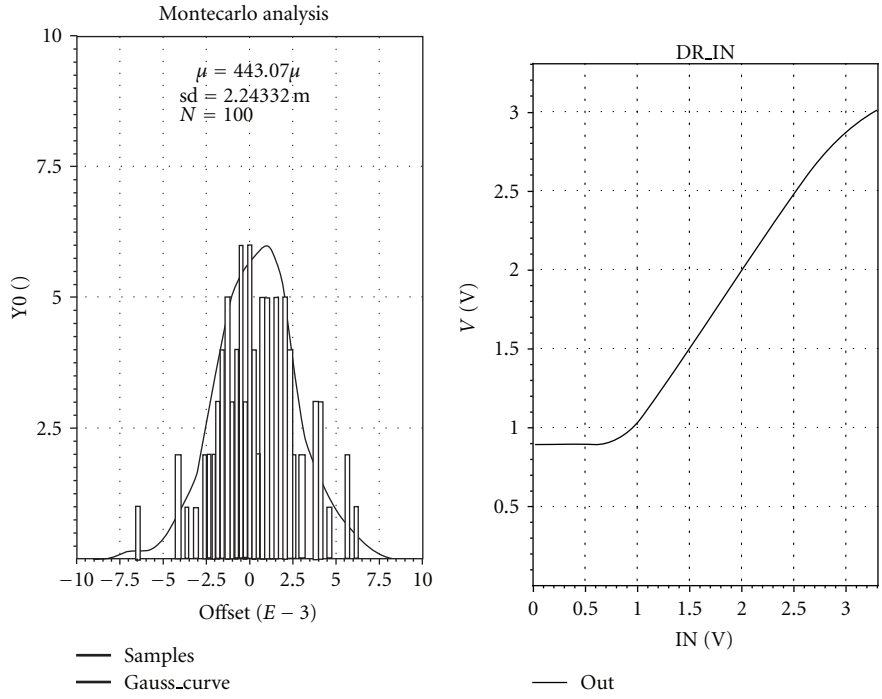


Figure 16: OTA offset and input DR simulations.

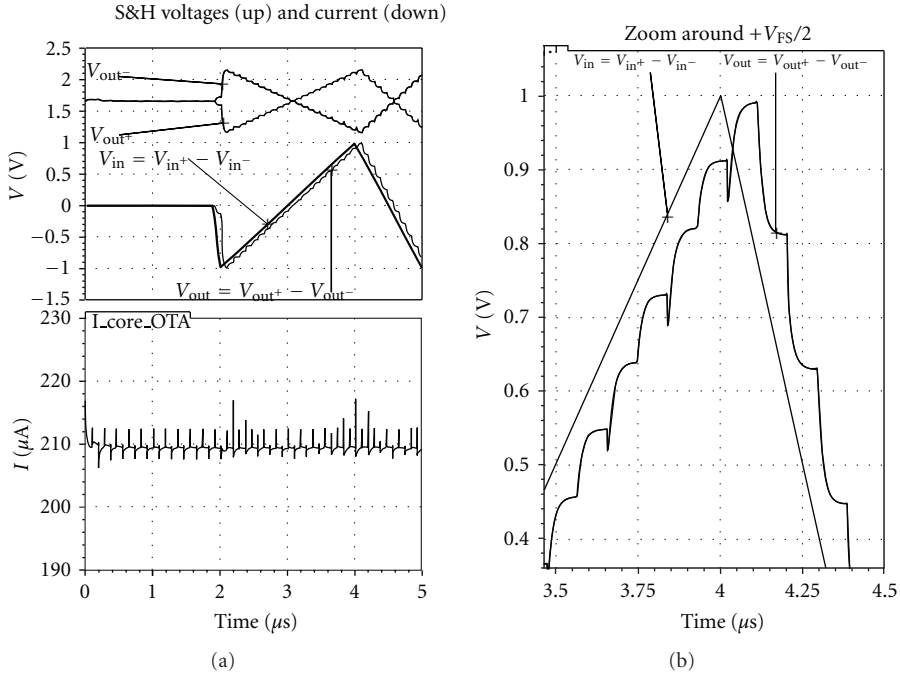


Figure 17: S&H operation.

of 2 reasons. The first of them is that these numbers are not silicon measurements, and the second is that it is really hard to find another converter with similar resolution-sampling frequency-technology-architecture characteristics in the literature, mainly due to the particularity of this application. In a future work, when the complete ADC is sent to fabrication, it will be worth the comparison.

As previously stated, the S&H block was sent to fabrication. Indeed, the complete chip including testing circuits looks like the schematic in Figure 21, and Figure 22 shows

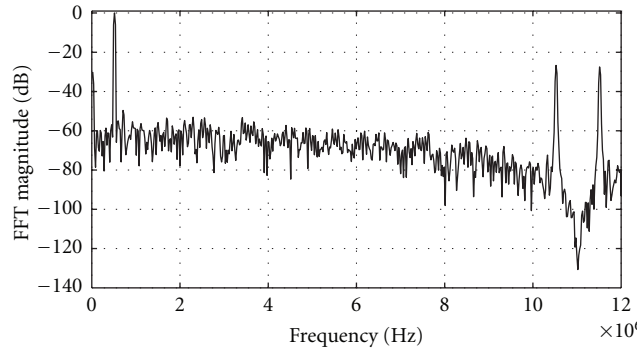


Figure 18: Output signal frequency spectrum.

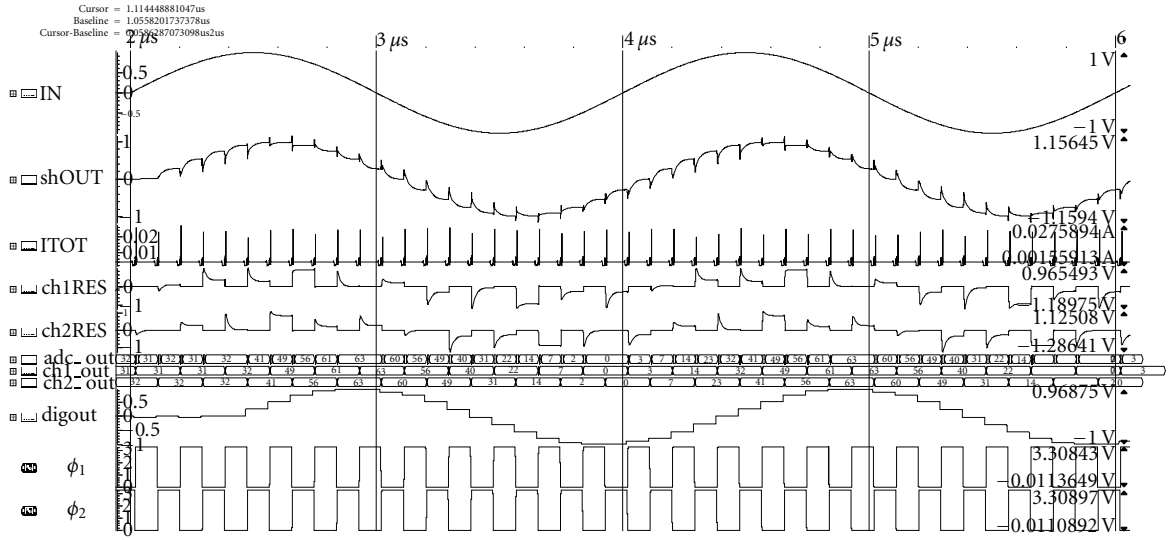


Figure 19: Complete ADC simulation.

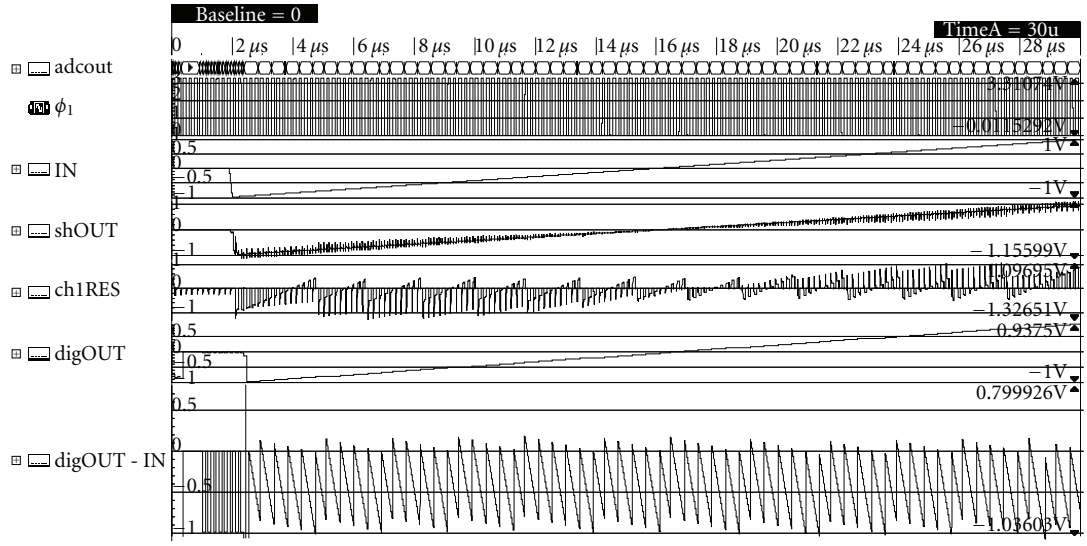


Figure 20: ADC linearity simulation.

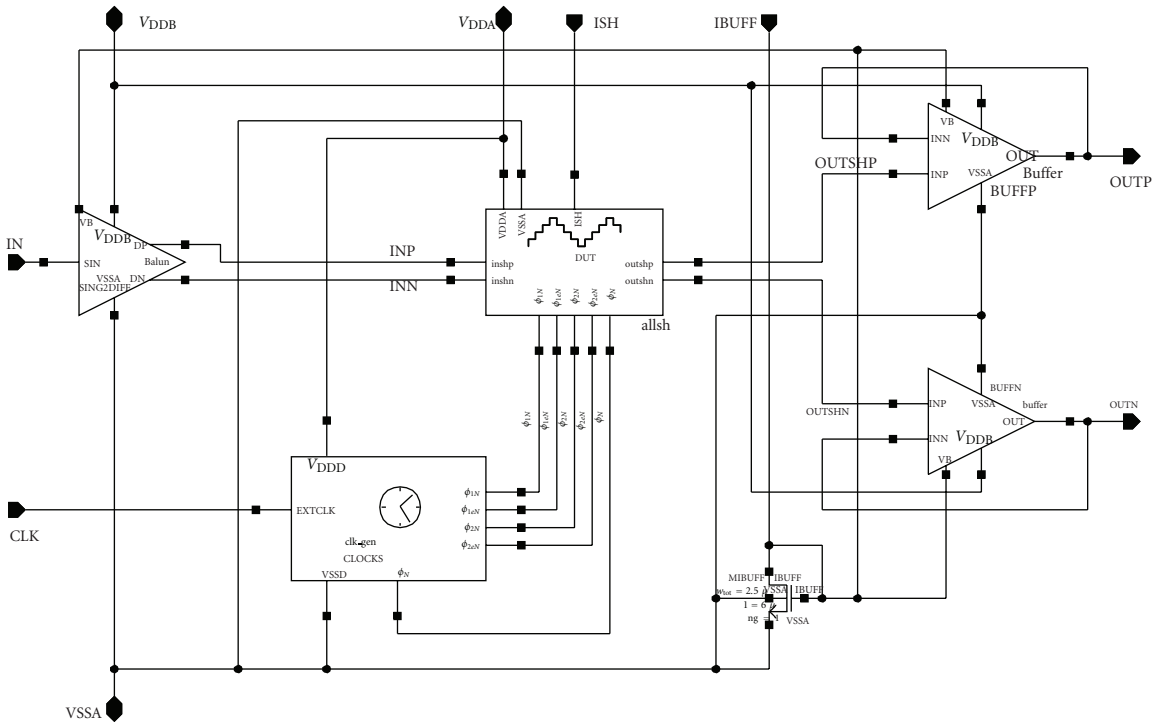


Figure 21: Chip to be fabricated.

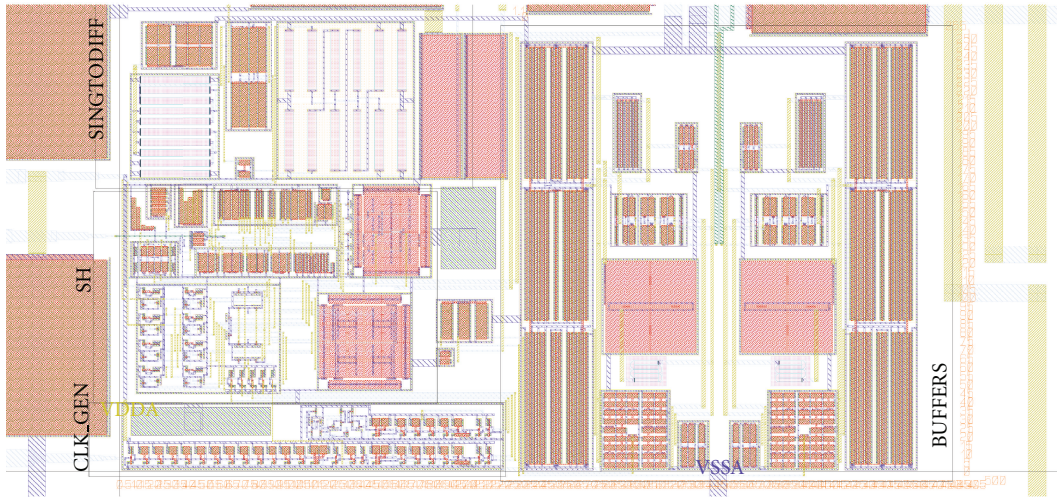


Figure 22: Core layout.

Table 4: Prelayout specifications for the ADC.

Parameter		Value	
Current consumption (rms)		3.64 mA	
Latency		5 clock cycles	
SNR in the channel	up to $f_s/2$	59.4	46.5 dB

the entire layout of the core without pads. As seen in those figures, along with the S&H block, four additional circuits were implemented. First, a clock generator was included to provide all the phases the sample and hold requires to function properly. Second, a single-ended to differential converter allows the chip input to be single-ended, making the test bench simpler. The S&H common mode voltage is also set by this input converter. Finally, two output buffers make the S&H capable of driving the measurement probes.

Layout techniques were applied all over the chip of Figure 22 in order to improve matching characteristics of the circuits. All passive component were drawn in common centroid arrangements, and when necessary, transistors used these structures too. The OTA inside the S&H, whose layout

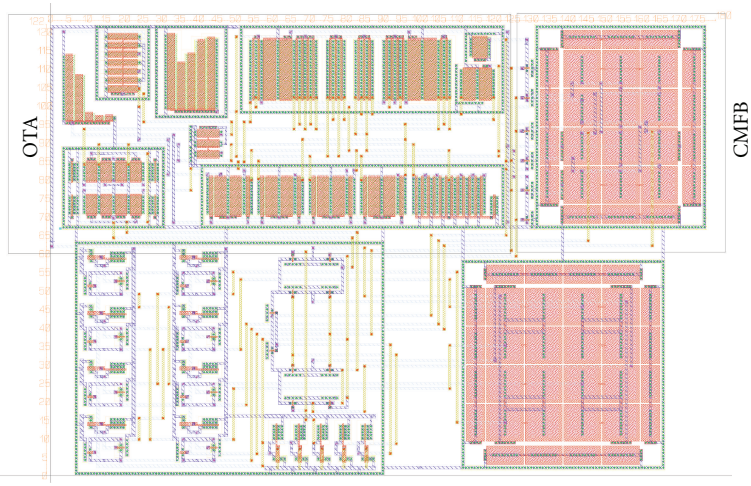


Figure 23: S&H layout.

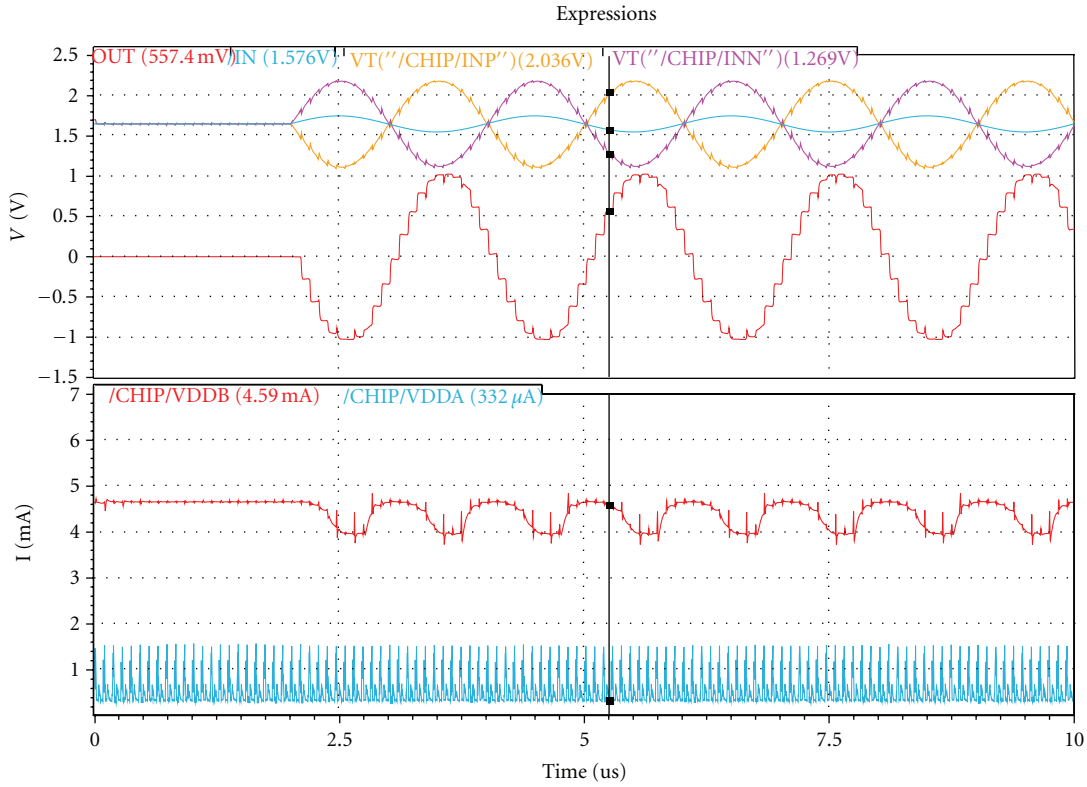


Figure 24: S&H postlayout simulations.

is zoomed in Figure 23, was placed in a stacked architecture to improve matching with its bias circuit as well.

Given that the chip has not returned for testing, some postlayout results are presented instead. Thus, Figure 24 shows the sample and hold operation over a 500 kHz single-ended input signal (blue at the top). The input converter amplifies and splits this wave in two differential versions (yellow and purple at the top), which are processed by the S&H and passed through the output buffers to produce the amplitude and time discrete signal in red. This simulation was executed with the entire chip, including pads and ESD structures, after parasitic extraction of the layout. The bottom panel of Figure 24 shows the separated current consumption of the S&H (blue) and testing circuits (red).

This prototype will operate under the same sampling frequency and bandwidth conditions that it would face as part of the ADC. The devices under test (S&H + clock generator) drive 370 μA from a 3.3 V power supply while the testing circuits (single-ended to differential converter and output buffers) draw around 4 μA from an auxiliary 3.3 V

power supply. The final layout size including pads is 800 μm $\times$ 1400 μm, yet the core is only 500 μm $\times$ 255 μm.

6. Conclusions

By carefully deriving key circuit specifications, it is possible to reduce their strong impact on total system power dissipation. Following this idea, a low power time-interleaved pipeline ADC for Bluetooth standard was designed. A survey on the specification process from standard to the elementary circuits was made to justify the 6-bit, 11 MS/s, 2 time-interleaved channel, 2 pipeline stage topology selection. This is the first step toward a reconfigurable system implementation. Indeed, the skill and architecture knowledge gained from this work will turn, easier and faster, the multistandard application into a power-efficient solution. It is difficult to compare this work results due to its very specific characteristics, which do not follow the actual trends of rising either the sampling rate or the resolution, but meet the Bluetooth standard requirements, instead.

The main support for this paper contribution is the application of GP, seeing that it provides a better knowledge and experience of circuit behavior. When designs are developed using GP, it is easier to detect relations and trends between circuit requirements and design variables, allowing identifying possible optimization focuses for global system performance, as in the complete ADC presented here.

As the main contribution of this work over its original version in [7], a prototype 11 MHz S&H was designed in a 0.35 μm 3.3 V CMOS process to verify its central OTA design via GP. Some testing blocks are also in the chip, yet not intended to influence the S&H performance itself. Over a stacked and common centroid-structure chip layout, future measurements aim to demonstrate the optimized power consumption while operating under the highest speed requirements a block would face into this time-interleaved pipeline ADC.

References

[1] J. Oliveira, J. Goes, M. Figueiredo, E. Santin, J. Fernandes, and J. Ferreira, "An 8-bit 120-MS/s interleaved CMOS pipeline ADC based on MOS parametric amplification," *IEEE Transactions on Circuits and Systems II*, vol. 57, no. 2, pp. 105–109, 2010.

[2] N. Petrellis, M. Birbas, J. Kikidis, and A. Birbas, "Asynchronous ADC with configurable resolution and binary tree structure," in *Proceedings of the 4th International Symposium on Communications, Control, and Signal Processing (ISCCSP '10)*, pp. 1–4, March 2010.

[3] B. Xia, A. Valdes-Garcia, and E. Sánchez-Sinencio, "A 10-bit 44-MS/s 20-mW configurable time-interleaved pipeline ADC for a dual-mode 802.11b/bluetooth receiver," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 3, pp. 530–539, 2006.

[4] B. Xia, *Analog-to-digital interface design for wireless receivers [Ph.D. thesis]*, Texas and A&M University, 2004.

[5] M. Waltari, *Circuit techniques for low-voltage and high speed A/D converters [Ph.D. thesis]*, Helsinki University of Technology, 2002.

[6] J. Kim, S. Limotyrakis, and C. K. K. Yang, "Multilevel power optimization of pipelined A/D converters," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 5, pp. 832–845, 2011.

[7] W. Carvajal and W. M. V. Noije, "Time-interleaved pipeline ADC design: a reconfigurable approach supported by optimization," in *Proceedings of the 24th Symposium on Integrated Circuits and System Design (SBCCI '11)*, pp. 17–22, ACM, João Pessoa, Brazil, September 2011.

[8] L. Sumanen, *Pipeline analog-to-digital converters for wide-band wireless communications [Ph.D. thesis]*, Helsinki University of Technology, 2002.

[9] J. Yuan and C. Svensson, "New single-clock CMOS latches and flipflops with improved speed and power savings," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 1, pp. 62–69, 1997.

[10] M. Del Mar Hershenson, "Design of pipeline analog-to-digital converters via geometric programming," in *Proceedings of IEEE/ACM International Conference on Computer Aided Design (ICCAD '02)*, pp. 317–324, November 2002.

[11] M. Del Mar Hershenson, *CMOS analog circuit design via geometric programing, [Ph.D. thesis]*, Stanford University, 1999.

[12] W. Carvajal, E. Roa, and W. M. V. Noije, "A 23 MHz GBW 460 μW folded cascode OTA for a sample and hold circuit using double sampling technique," in *Proceedings of the Conference on Integrated Circuits and Systems (DCIS '08)*, Grenoble, France, Novembre 2008.

[13] B. Wicht, T. Nirschl, and D. Schmitt-Landsiedel, "Yield and speed optimization of a latch-type voltage sense amplifier," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 7, pp. 1148–1158, 2004.

Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2012, Article ID 832531, 17 pages
doi:10.1155/2012/832531

Research Article

HoneyComb: An Application-Driven Online Adaptive Reconfigurable Hardware Architecture

Alexander Thomas, Michael Rückauer, and Jürgen Becker

Institut für Technik der Informationsverarbeitung, Karlsruher Institut für Technologie (KIT), Engesserstraße 5, 76131 Karlsruhe, Germany

Correspondence should be addressed to Michael Rückauer, michael.rueckauer@kit.edu

Received 21 February 2012; Accepted 24 May 2012

Academic Editor: Elmar Melcher

Since the introduction of the first reconfigurable devices in 1985 the field of reconfigurable computing developed a broad variety of architectures from fine-grained to coarse-grained types. However, the main disadvantages of the reconfigurable approaches, the costs in area, and power consumption, are still present. This contribution presents a solution for application-driven adaptation of our reconfigurable architecture at register transfer level (RTL) to reduce the resource requirements and power consumption while keeping the flexibility and performance for a predefined set of applications. Furthermore, implemented runtime adaptive features like online routing and configuration sequencing will be presented and discussed. A presentation of the prototype chip of this architecture designed in 90 nm standard cell technology manufactured by TSMC will conclude this contribution.

1. Introduction

Reconfigurable architectures aim to reach the performance and energy-efficiency of application-specific integrated circuits while the flexibility is increased, therefore closing the gap between ASICs and general-purpose processors.

For data-oriented applications an increase in performance compared to general-purpose processors can be reached by mapping operations to a possibly large set of functional units, which are working in parallel. In contrast to ASICs, their actual function and the interconnection between the units are not determined during design and manufacturing but may be changed at runtime to support a wider range of applications.

For example, in a mesh-based architecture, a flexible communication network connects the functional units (FUs) on demand. Since the FUs are communicating directly by exchanging the intermediate results through the communication network, memory accesses for temporary data storage are avoided and memory bandwidth usage is reduced to a minimum. The overall data throughput is at maximum and very close to the ideal performance that can be reached by ASIC implementations.

However, this approach is not without limitations. The increased flexibility comes at the cost of additional hardware. The flexible communication network for FUs requires a lot of multiplexers, communication lines, configuration registers, and additional logic to control the configuration mechanisms. Depending on the type of the reconfigurable approach (coarse-grained or fine-grained) the overhead of the configuration registers and control logic can be considerable. An example for this fact is given by field-programmable gate arrays (FPGAs) [1, 2], which require a lot of configuration data (in the area of several MBs) for specifying the device function. However, FPGAs with their fine-grained approach mark the worst case for this problem. Coarse-grained architectures [3–5] reduce the amount of configuration data to a fraction of the FPGA requirements. This is achieved by vector-based routing and simplified FUs that support arithmetic operations instead of LUT-based Boolean logic. However, the programming models of such architectures are still limited either to native languages or

subsets of known paradigms like C/C++, which prevents the commercial success additionally.

When different application domains, which may have different computational and communication requirements, should be supported, it is not reasonable to integrate FUs with an arbitrary set of operations into a configurable architecture. Therefore, an architecture template, which allows to adapt the set of operations supported by each FU as well as the communication network connecting the FUs at design time, aids in implementing hardware that is even more energy efficient while maintaining flexibility were it is beneficial.

This contribution presents the HoneyComb architecture, a coarse-grained reconfigurable hardware architecture (CGRA) template. It is designed to compute stream-based as well as control-based applications. Therefore, in addition to coarse-grained components, the architecture includes fine-grained components, which are used to compute Boolean operations and control program execution.

To support design-time adaption of the architecture to application requirements in addition to the dynamic reconfiguration, compiler-supported application-tailored hardware reduction/RTL adaption techniques have been developed and implemented.

The following section describes related work. Section three describes the HoneyComb (HC) architecture, its structural characteristics and functions shortly. Section four is devoted to the parameterizable RTL-model and the application-tailored reduction methodology. In section five the RTL-dependent programming model is described. The details of the final prototype and Printed Circuit Board (PCB) design are presented in section six. In section seven we show how the application kernels are mapped onto the architecture. Results and conclusion sections close this contribution.

2. Related Work

In the past decades a number of architectures were proposed in the field of reconfigurable systems that aim to efficiently solve computationally intensive problems while being flexible enough to support a wide range of applications. All these architectures have their advantages and disadvantages. In the following we give an overview of existing architectures.

The Pleiades project [6] proposes an architecture template for ultralow-power high-performance reconfigurable computing. It includes a general purpose processor coupled with a heterogeneous array of autonomous application-specific satellite processors. The resulting assembly is strongly catered to the target application, resulting in limited support for other applications. A well-known example for the use of the Pleiades template is the Maia Chip, which is catered to the requirements of speech coding.

The ACM architecture [7] from QuickSilver Technology is a low-power architecture designed for use in mobile devices. The architecture is derived from an analysis of target applications, resulting in a "fractal" architecture. Heterogeneous nodes are hierarchically connected in a tree fashion with each nonleaf node laid out like the previous layer. Nodes consist of either fine-grained or coarse-grained functional units. The hierarchical assembly facilitates fast reconfiguration of functional units. However, the scalability of the tree-like communication network seems to be a limiting feature in bigger configurations.

The concept of PACT XPP-technologies [8] assumes that applications consist of both regular and irregular parts. Therefore it features a regular array of processing units for dataflow-oriented applications as well as a set of supplementary processors (function folding units, FNC) for control-flow-intensive algorithms. Both components are optimized for 16-bit applications and tightly coupled to support high-speed data transfers. Nevertheless, two completely different components in one architecture require a manual partitioning step. Despite the fact that partial reconfiguration is supported, applications used at the same time on this architecture have to be planned in advance.

Another interesting approach is the DRP architecture [9] from Renesas Electronics (formerly NEC). It consists of a homogeneous multicontext array of processing elements (PE), with each PE having an 8-bit ALU and a register file. In addition there is a context memory, which can choose a new configuration in each cycle. The selection of context is done centrally by a sequencer. The sequencer is a finite state machine that changes states depending on the inner state of the array or depending on external control signals. Memory modules located at the array boundaries provide high-bandwidth data storage. Here, partial reconfiguration seems to be very difficult to realize. Furthermore, data transport to and from the array seems to need additional logic.

The Montium architecture [10] implements a processor that works similar to a VLIW processor, but differs considerably in programming. Instead of instructions, Montium processes sequences of preloaded configurations. This is done by five integrated 16-bit ALUs that are able to execute multiple instructions in a single cycle. Ten local memories with 512 entries each ensure that the ALUs are used to capacity. Multiple Montium processors can be integrated in a System-on-Chip as needed. The maximum parallelism given by the five ALUs seems to be also the limiting factor of this architecture.

The PipeRench Architecture [11] is based on a several times implemented pipeline structure. The individual stages are separated by registers and an interconnect network, in a way that data can be interchanged between pipelines. In addition there is a global network that facilitates data transport contrary to the pipeline flow. The actual configuration of the architecture is determined by parameters and can be adapted to specific applications. The number of concurrent pipeline implementations seems to have a huge effect on the resulting complexity of the interconnect networks and the resulting timing.

PADDI [12] is a multiple-instruction multiple-data (MIMD) architecture. Here, several simple processors that process VLIW-like instructions are connected through a switch structure, which allows conflict-free communication between processors. PADDI is a quite simple architecture.

However the partitioning on the available processors to reach full utilization seems to be a not neglectable task and reminds one of current problems with programming of multicore processors.

The RaPiD architecture [13] is a linear array of functional units, which is configured like a linear pipeline. It is well suited for irregular applications. However, it has weaknesses when processing block-oriented algorithms.

MorphoSys [14] combines all components needed for execution control and data processing in a single design. Execution control is done by a TinyRISC processor, while data processing is performed by an array of processing elements. Array nodes are connected through a multilevel interconnect network. Local memories that hold configuration and data deliver all information needed for program execution and at the same time decouple the array from the host interface.

MATRIX [15] is an architecture similar to MorphoSys, but does not feature an integrated processor for array control. Therefore array control is not as comfortable as with MorphoSys. Data processing is performed with 8-bit precision. MorphoSys and MATRIX both are not able to support concurrent application executions if not planned in advance. This shortcoming is common to the most of the presented architectures, except the ACM.

REMARC [16] is an array based on simple 16-bit nanoprocessors that communicate through local connections. The array is controlled by a global control unit. It carries out transport of data and configurations, but does only provide low bandwidth to the host system. The architecture is designed for multimedia applications. However, it does not have an integrated multiplier, which is a considerable weakness.

Besides the ACM and Montium architectures, most architectures are not designed to run in a multitasking environment. If more than one application is supposed to be running on the same hardware, it is required to plan such a scenario in advance or it is simply not possible to share the resources. For this functionality the target architecture requires additional logic to manage the resource sharing. In case of the HoneyComb architecture, this problem is solved with the adaptive online routing. There, resources for a communication stream are reserved at runtime.

3. HoneyComb Architecture

The HC architecture is an adaptable dynamically reconfigurable cell array with a hexagonal cell layout. The underlying RTL-model is highly parameterizable. Except for the basic structure of the architecture the specification of every component within the array can be enabled, disabled, or modified. A detailed description of the architecture is given in [17, 18]. This section gives only a short overview required for understanding the presented concepts.

The HC array is based on structurally similar cells, which consist of a routing unit and a functional module (see Figure 1). The routing units of all cells are connected to their neighbors and compose the communication network, which is meant to establish point-to-point connections (streams) between functional modules. Supported data types are 32-bit coarse-grained words and multigrained vectors of 1 to N bits.

The routing of streams is performed during runtime and is fully realized in hardware. Therefore routing instructions have to be defined and implanted into the source routing unit. Once a routing instruction is received, the routing unit starts the routing process by propagating the routing instruction to the next cell along the path to the destination cell. The implemented routing algorithm is depth-first search in combination with a backtracking algorithm. Each routing unit requires 3 cycles for the routing process and can process one routing request at a time. Once the destination is reached and the stream is established, data can be sent through this point-to-point channel. Each data word is buffered at the input of each cell, which defines the communication latency by the cycle count equivalent to the count of passed cell edges. The adaptable routing techniques are applied to coarse-grained as well as multigrained data-connections. Each transfer is fully synchronized by a handshake protocol that assures data consistency. Application and configuration data share the same communication network, which increases the reconfiguration performance by using multiple reconfiguration streams at once. It is only limited by the number of cells and the external interface bandwidth.

The functional module specifies the type of the HoneyComb (HC) cells and can be defined as I/O module (IOHC), memory module (MEMHC) or datapath module (DPHC). Cells carrying an I/O module include a specialized microcontroller for data transfers in or out of the array. Therefore the IOHC contains an interface to the system bus (AMBA, WISHBONE, or a proprietary interface). IOHCs initiate all processes within the array by transferring routing instructions into associated RUs, establishing data and configuration streams to destination cells, configuring these cells and controlling data transfers to and from these cells. Streams between cells can be routed by tunneling the necessary routing instructions to the source and starting the desired routing process. All transfers can be done through DMA without the interference of a system controller. Therefore an optimized μ Controller has been integrated into the IOHC, which includes parallel working address generators for fast data transfers.

The MEMHCs provide storage space for the applications within the array. Therefore multiple memory modules can be included in each MEMHC and store coarse-grained and multigrained data simultaneously. Logical merging of memory modules can be done to offer bigger storage space for applications. Each memory configuration can be used as RAM as wells as a FIFO or a LIFO.

DPHCs realize the main arithmetic or logical data manipulation units within the array. Therefore their functional modules include ALUs, LUTs, coarse-grained and fine-grained registers, data type converters, and data branchers, which are required to split synchronized data streams. The combination of ALUs and LUTs in one module allows the evaluation of ALU operation flags (carry, sign, overflow, etc.) at once and influence the next operations. Therefore each ALU includes a context memory, which selects

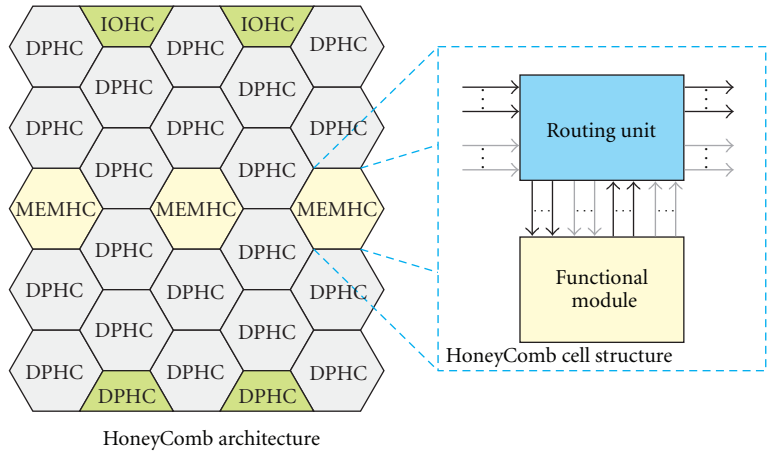


Figure 1: Hexagonal HoneyComb (HC) array with unified cell structure and three cell types—DPHC, MEMHC, and IOHC.

the opcodes, operands, output registers and generated flags. It can be addressed by LUTs or directly by output flags and change the output on every cycle. LUTs and fine-grained registers can implement finite state machines (FSMs) for even more complex functions. By sending fine-grained or coarse-grained output data to IOHCs, the removal of the current configuration can be triggered and a new one started. To increase the bandwidth of the I/O operations IOHCs are provided with a separate clock. Usually this clock is higher compared to the array clock. Array configurations consume the incoming data much faster than a system bus can deliver especially if multiple data streams are required. In this case the ratio of both clocks can be adjusted by the system.

Since every cell is capable to be configured independently partial reconfiguration is possible. Runtime routing even allows overlapping of configurations without being considered at compile time if enough resources are available. Clock gating of idle parts of the architecture is implemented and performed in two levels. The first level controls the routing units of each cell independently. Thereby, a routing unit is activated if the neighbor cell is establishing a path to this cell. Functional modules of MEMHCs and DPHCs are clock-gated if no configurations are programmed. Incoming configurations activate the cells and keep them active until the configurations are deleted. If the user detects a nonfunctional routing unit the affected cell can be deactivated. In this case the HC array routes the routing requests without using the deactivated cell.

The idle HC architecture is started by providing one of the available IOHCs with a memory address where program execution should start. This program includes steps to initiate the configurations, transfer the data to and from the array, and clean up the array by deleting the configurations. Several IOHCs can be used at the same time if the addressed cells within the array are disjunctive. The scheduling and checking of configurations is supposed to be performed by a runtime system, which is not finished yet. Control operations of the array are done by the HC Controller, which offers a separate interface for the system's hosts and a set of registers for checking cell states and writing control values.

The architecture can be programmed using the HC assembly language or the HC language, which are both introduced in the latter sections. The HoneyComb assembly language is a low level language for structural programming while HoneyComb language is a higher level approach.

4. Parameterizable RTL-Model

This section describes the methodology we have used to design the HC architecture and additional tools we have developed to support the configuration and verification process.

4.1. Design Methodology

The HC architecture is a highly parametrizable architecture. The complete model has been developed based on VHDL and its generic capabilities. Almost every possible way of specifying generic structures in VHDL has been used, including:

(i) generics definitions;
(ii) constant definitions;
(iii) conditional and looping generate structures;
(iv) function library for parameter evaluation;
(v) package definitions for configuration management.

The combination of these techniques is a very powerful way to describe parametrizable designs. We have defined two main sets of configuration parameters. The first set is defined within the global constant packages and includes constant definitions considering global parameters like

(i) array size;
(ii) CG/MG data width;
(iii) routing instructions formats;
(iv) IOHCs instruction formats;
(v) clock gating enable/disable.

The second part describes local definitions regarding

(i) cell port count;

(ii) DPHC configurations;

(iii) MEMHC configurations;

(iv) IOHC configurations.

Depending on the given parameter sets, specific features can be activated or module instances removed.

To ease the debugging process careful signal definitions have been implemented. With a few exceptions there are no unused signals available. This way every structural problem causes undefined signal states and can be identified very fast. Additionally, all modules have been designed with intention of exhaustive reuse ability. So, if fixed in a specific module this change is automatically applied to all instances of this module in the hierarchy. The more modules of the same type are instantiated the bigger is the impact.

Following these simple rules we designed the whole architecture in VHDL in about two years. The structural correctness and the functional verification took additional two weeks. After this time the architecture was able to perform first tests and simple applications for verification purposes. It took additional 4 years for the development of the programming languages, debugging tools, and demonstration application (see later sections). The final step was the IC layout in 90 nm for the prototype.

4.2. Configuration Manager. The amount of RTL configuration parameters is enormous. The biggest part has been spent for specifying the functional modules within the DPHCs, which includes over 60.000 parameters. Parameters can specify the operation sets, the count or available modes (single context/multicontext) of each single ALU, LUT configurations, the number and type of registers, the interconnection between these modules, and so on. Manual control of this amount of parameters is simply not possible and requires additional tool support. Therefore, the configuration manager has been developed.

The main purpose of this tool is the management of configurations. However, the functionality is going a step further and includes the generation and merging of configurations as well. The generation of configurations can be done manually or by analyzing previously compiled applications (see Figure 2).

Based on predefined templates or ideal array representations the analysis of applications is performed by the HCL compiler (see Section 4). The result is a functional description of the application in assembly language. One or several of those descriptions can be used to extract the necessary specifications for the target RTL description of the array, so-called super RTL configuration. Since configuration code for DPHCs and MEMHCs is structural, it is quite simple to extract the necessary information. Figure 3 shows how several applications impact the functional units. This kind of merging is performed on all levels of functional units and results in a hardware structure that is able to support every considered application.

The configuration manager has been developed in VBA for Microsoft Excel. The table management of this application is very well suited for configuration management and allows manual configuration creation. With additional VBA code necessary consistency checks have been implemented to support manual work. The super configuration Generator is part of this application as well as the Assembler application, which is described in Section 4.

5. Programming Model and Tools

The structural composition of the HC architecture predefines the programming model. This model is composed of three layers according to Figure 4 and is executed by specific architecture parts. Transport layer is performed by IOHC and is meant for conditional or unconditional data transfers. The communication layer controls the routing network and influences the placement of configurations. The configuration layer describes the functional modules of MEMHCs and DPHCs.

The programming model impacts the definitions of the HC Language as well as the definition of the HC assembly language; both are described in the following subsections.

5.1. HoneyComb Assembler (HCA). The HoneyComb Assembler application is integrated into the configuration manager. Thus, all configuration parameters of the array are available to the assembler and are considered during assembly. The HCA language definition consists of three parts, one for each layer of the programming model. By specifying the target cell coordinates and the layer type the user tells the assembler which kind of code to generate. The code specification for the transport and communication layers is globally the same for the given array. In case of the configuration layer the resulting binary code characteristics can differ from cell to cell if the RTL configurations vary, that is, if the HC array is heterogeneous. Code adapting to RTL configuration helps to reduce the resulting binary code size and required hardware structures but makes it incompatible to other cells.

Since the multicontext capability of HCA can be disabled at RTL it is required to perform code transformations during the assembly process. The following example demonstrates the necessity for transformations:

```
ALUOP   0, ADD
```

This instruction programs the given ALU to perform the ADD operation. In case of the ALU with multicontext capability this instruction has to be translated to the following piece of code:

```
ALULCFG   0, C0=[ADD]    #context 0
ALULCFG   0, C1=[ADD]    #context 1
...
```

Here, every active context will perform the ADD operation what in fact results in the same behavior as the ALU without multicontext capabilities. The assembly application transforms this kind of transformations automatically. Therefore,

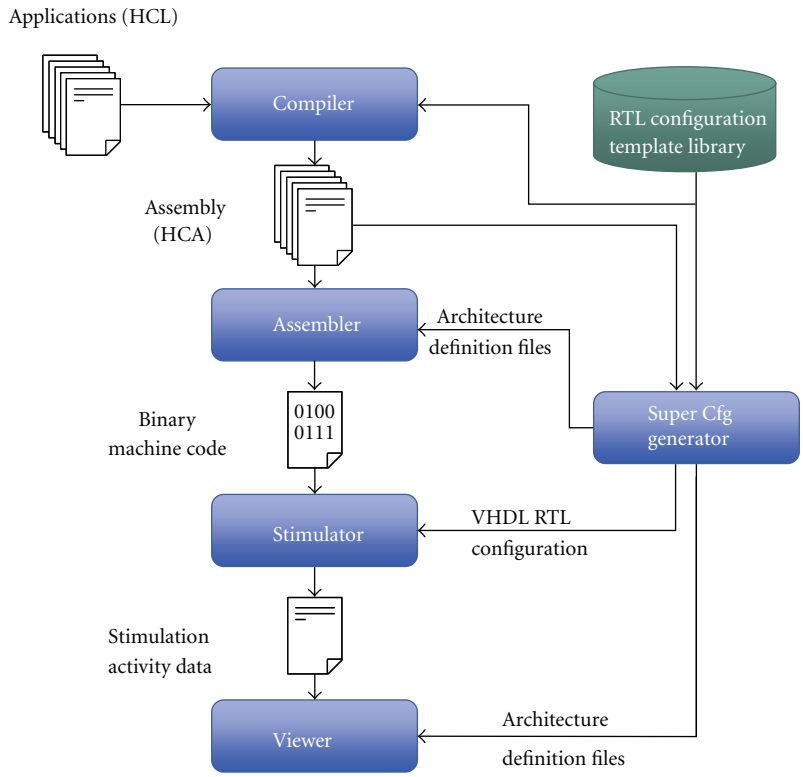


Figure 2: HoneyComb application-tailored design flow for generation of RTL configurations, development, and debugging of applications.

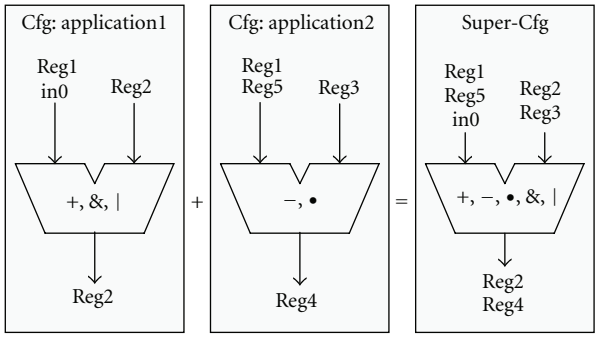


Figure 3: Example for ALU configuration generation derived from two applications and merged to the final operation and register sets.

the programmer does not have to consider every detail of the architecture and would still be able to compile applications. This kind of transformations performs the assembly tool in the extended mode which has to be separately activated. If incompatibilities are detected, the user will be notified and can adapt his code manually.

5.2. HoneyComb Language (HCL). The HoneyComb assembly language describes the configuration of a HoneyComb array in both a low level and structural fashion, making it hard to describe the desired behavior. Therefore we introduced the HoneyComb language (HCL), a high-level language that makes it much simpler to describe the behavior of the three cell types. In addition it is possible to specify sub-configurations consisting of several cells that together perform a more complex function than is possible with a single cell.

A structural programming language is used to describe the behavior of an IO cell. It features instructions for configuration management, off-chip communication, and streaming data transfers between the IO cell and the array cells. Using subconfiguration descriptions, configurations consisting of several cells can be established and removed with a single statement.

The behavior of a data path cell is described as a sequential process executed once per clock cycle. It supports control flow statements and assignment statements that assign the result of a complex expression to a register or output port.

5.3. HCL Compiler. A program written in the HoneyComb language is transformed into HoneyComb assembly language with the HCL Compiler software.

In the first step of compilation the HCL description is read in and transformed into an abstract syntax tree (AST) representation. The further proceeding depends on the type of cell description.

The AST of an IO cell description is transformed into an intermediate representation where all source language statements are replaced by sequences of one or more assembly instructions. Control flow statements are transformed into equivalent structures utilizing only conditional jumps.

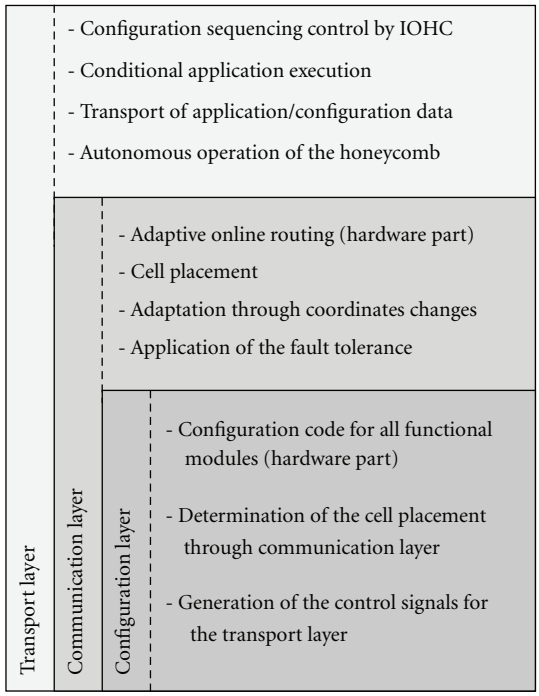


Figure 4: Programming model for the HoneyComb architecture based on the structural hierarchy.

This yields a control flow graph consisting of instruction base blocks that are connected by edges where a control flow transfer from one block to another may occur. There is no dedicated instruction selection and scheduling except for complex expressions, for which instructions are ordered to minimize the number of registers needed to hold intermediate values. Register allocation is done with the graph coloring approach described in [19]. In assembly code, basic blocks connected by unconditional control flow edges are emitted consecutively if possible to reduce the number of control flow transfers.

When the compiler processes a sequential description of a data path cell's behavior it first puts all noncontrol flow statements into a list. For each statement in the list, its actual execution condition is calculated by looking at the conditional statements surrounding it and the execution conditions of statements writing to the same variables. Consider the following example:

$$\text{IF } a \text{ THEN IF } b \text{ THEN } x \leq y + z; \text{ END IF; END IF;} \quad (1)$$

$$\text{IF } c \text{ THEN } x \leq z - y; \text{ END IF;} \quad (2)$$

The execution condition of the first statement is (a and b and not c) as it is only executed when the conditions of the both surrounding if statements are true (a and b) and the execution condition of the second statement (c) is false (equivalent to not c = true). Because of the synchronous nature of the HoneyComb architecture we also need to record the condition under which the value of a variable is consumed. Again consider the above example: the values of a and c are always consumed, while the value of b is only consumed if a is true. The values of y and z are only consumed if either statement one or two is executed. This information is collected for each statement in the list. With this information the statements in the list can be executed in any order assuming that any writes will be visible in the next cycle.

The statements in the list are now mapped to the ALUs and LUTs of the cell. Variables used in the statements are mapped to registers and IO ports of the cell depending on their type. In order to minimize resource usage, operations with mutual exclusive execution conditions and matching evaluation conditions for common variables are mapped to the same function unit.

This is done using a clique-partitioning algorithm [20] that is employed twice, once for the mapping to the ALUs and once for the mapping to the LUTs. A graph is created whose vertices represent the statements to be mapped to the functional units. Two vertices are connected by an edge if they can be mapped to the same functional unit. The edges are weighted by the cost reduction that is achieved by merging the two adjacent vertices. In each iteration the edge with the highest weight is selected and its adjacent nodes are merged into one. Nodes that were adjacent to both merged nodes are connected to the resulting node. This merge process is repeated until no edges are left or all remaining edges have a cost increase associated with them. Each resulting node is mapped to one ALU or LUT respectively.

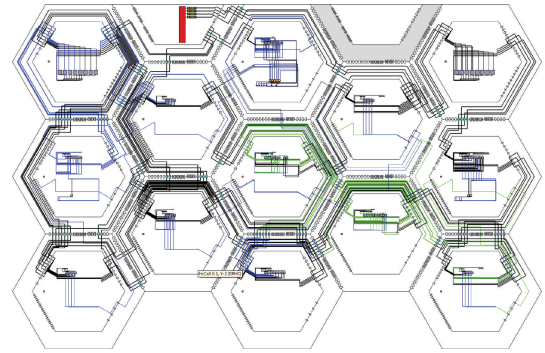


Figure 5: Execution of the AES256 algorithm executing on the HoneyComb architecture visualized by the HCViewer.

5.4. HoneyComb Viewer (HCViewer). The last step in the software development for the HC architecture is the debugging using the HCViewer tool (see Figure 5). This tool requires a current HC configuration generated by the Super Configuration Manager and the simulation activity data, which can be generated by the HC VHDL model during execution of an application. Once loaded, the HCViewer visualizes the processes within the array and reports all user relevant data and values.

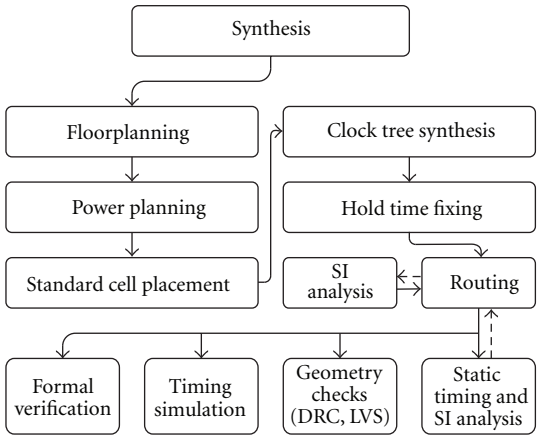


Figure 6: Implementation and Sign-Off Flow.

6. Prototype IC

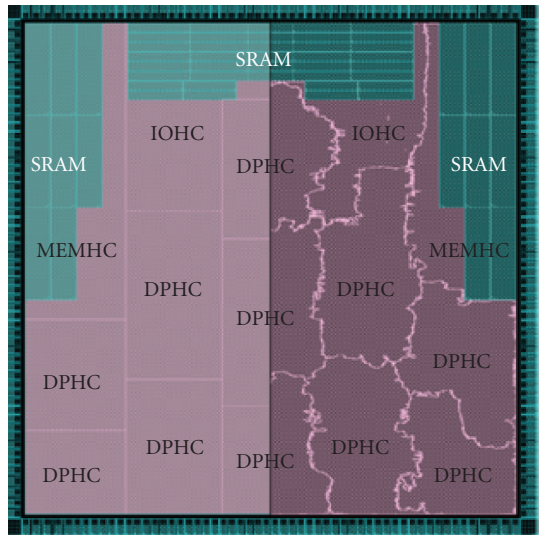
6.1. Initial Technology Decisions. We had to make the choice between different IC manufacturers providing a 90 nm process. As area was our main concern, we chose TSMC, which had the process with the best area efficiency. We selected a low-power library, which trades performance for a higher transistor density. Therefore the test chip was expected to achieve a lower clock frequency than a production chip would.

Initial synthesis of the array's RTL model was done with the design compiler software from Synopsys. The resulting netlist was imported into the SoC Encounter software from Cadence, where the entire layout work took place. Figure 6 gives an overview over the implementation and verification flow, which is described in detail in the next two sections.

6.2. Layout. From the total die area of 16 mm^2, a border of 183 μm had to be reserved around the standard cell area for the seal ring, the IO and bond pad area and the core power ring. This left an area of 13.2 mm^2 for the actual design. We chose a flat design methodology with a single consecutive workflow. This saved the overhead of having to implement and characterize the submodules separately.

Figure 7(a) shows the layout of the cells and the placement of the SRAM modules on the actual chip. Unlike the hexagonal cells in the logical layout, the physical cells have a rectangular shape. To connect each cell to its six neighbors despite the rectangular shape, the arrangement of the cells is staggered, so that an inner cell still adjoins its six neighbors.

Figure 8 shows the chip's power network. It consists of a power ring around the chip's core area and a regular power grid that spans the entire core area. With the chosen technology, nine layers of metal are available for routing. The two topmost layers are approximately three times thicker than the others, giving them a lower resistance and a higher current tolerance compared to the others. Therefore, they were chosen for carrying the power grid. The geometry of the power grid was chosen based on a suggestion found



(a) (b)

Figure 7: Cell layout as defined by floorplanning (a) and after standard cell placement, respectively (b).

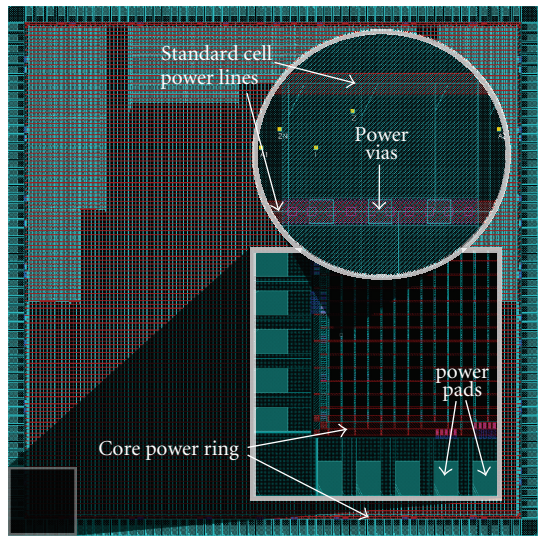


Figure 8: Power network.

in the application note for the employed standard cell library. An estimation of the chip's power consumption was made with the VoltageStorm power analyzer. That value was doubled to obtain a comfortable safety margin and the grid was planned to meet these requirements.

We added as many power pads as allowed by the spacing rules of the IO pads: 32 pads carrying power and ground for the IO domain and 58 pads carrying power and ground for the internal power supply, which is well above the requirements.

Afterwards the standard cells were placed. Initially the placer tended to create crowded areas with local placement densities of nearly 100%, which made it impossible to add buffers during hold time fixing or to obtain a valid routing

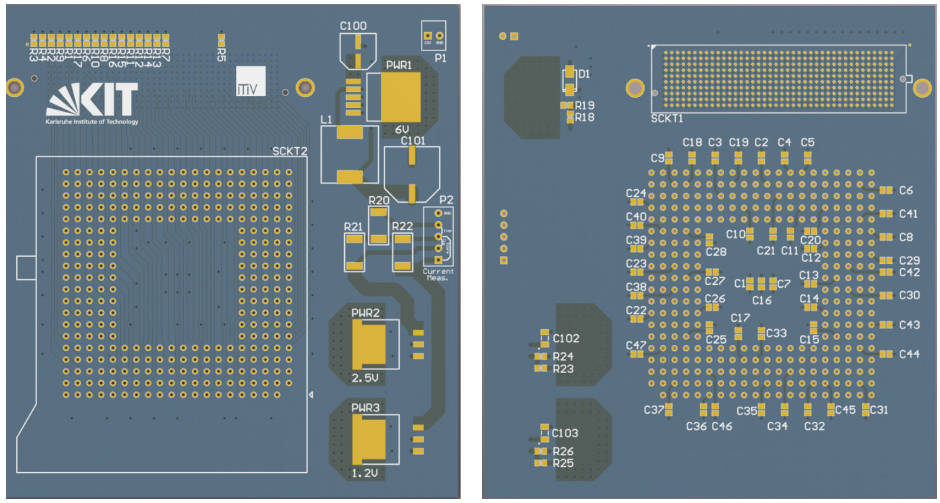


Figure 9: 3D rendering of final HoneyComb board layout.

of signal nets later on. Therefore the placement flow was altered. After an initial placement multiple iterations of timing optimization and incremental placement were performed. Thereby the allowable local placement density was raised from an initial 70% up to 80%. This approach created a more uniform standard cell distribution and leads to better timing results. Figure 7(b) shows the final cell shapes after placement. They are irregular and blend into each other, which is beneficial for timing.

Clock tree synthesis was done with the automatic synthesis mode, where the software created the clock tree geometry automatically from few parameters, which are the maximum insertion delay and the maximum skew between two flip flops' clock signals. With the default wire geometry the current through the clock tree exceeded the current limit. Therefore a custom rule was used that tripled the wire width between the inner clock tree buffers. After clock tree synthesis, another timing optimization step was performed that analyzed and repaired any remaining hold time violations by inserting buffers into affected signal paths.

Up until this point all work was done on a partially routed design likely to contain unrouted nets and shorts between different nets. A timing-driven routing algorithm legalized this routing, while trying to minimize the timing impact caused by the wiring.

Finally a signal integrity analysis was performed on the fully routed and timing-clean design. Hereby, pairs of nets that could influence each other through capacitive coupling were identified. If this effect could have caused a net to carry an invalid value, it was rerouted to reduce the coupling. This analysis-and-repair step was repeated until the design was free of signal integrity errors.

6.3. Chip Sign Off. Timing and signal integrity were checked with PrimeTime SI, using a 10% derating of clock and signal path runtimes to account for timing uncertainties introduced by manufacturing tolerances and different voltage levels across the chip. First, the software detected a few hundred transition time and signal integrity violations. A script extracted the violations from the timing reports and created another script that repaired the errors within encounter. The transition time violations were fixed by upsizing drivers; the signal-integrity violations were fixed by rerouting the affected nets. It actually took a lot of iterations and more than a week to fix all of these violations.

In addition, design rule checks (DRCs) and layout versus schematic (LVS) checks were performed with the Calibre software. As we did not have access to the actual layout data of the memories and standard cells, they had to be treated as black boxes during LVS. Therefore only the connectivity of the cells could be checked.

To verify the functional equality of the synthesis netlist and the netlist representing the final layout, the netlists were compared with the formality software. The software found three discrepancies between the synthesis netlist and the netlist representing the final layout. Despite this, we verified that the postlayout circuit performs the same function as the original one.

Before the layout was sent to manufacturing we performed a successful simulation run with timing data from Primetime SI.

6.4. Printed Circuit Board Design. The HoneyComb prototype IC has a proprietary parallel interface that is optimized to maximize throughput of streaming data transfers. It has two 32-bit data interfaces, of which one is dedicated to data input while the other is dedicated to data output. At 125 MHz this results in a combined bandwidth of 1000 MB/s, 500 MB/s in each direction.

The prototype IC is supposed to be connected to an FPGA, which serves as an interface between the HoneyComb IC and the controlling host system as well as an external memory controller that connects the IC to external DRAM memory. This allows for easy evaluation of different interfacing options as well as slave (accelerator) and standalone mode.

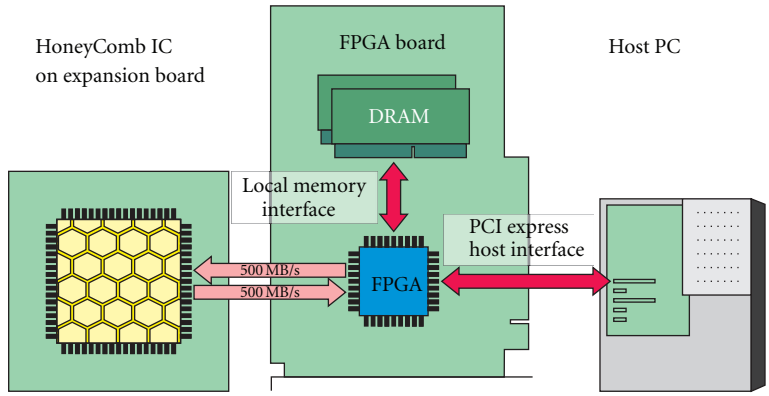


Figure 10: HoneyComb IC in coprocessor configuration with PCI-express host interface.

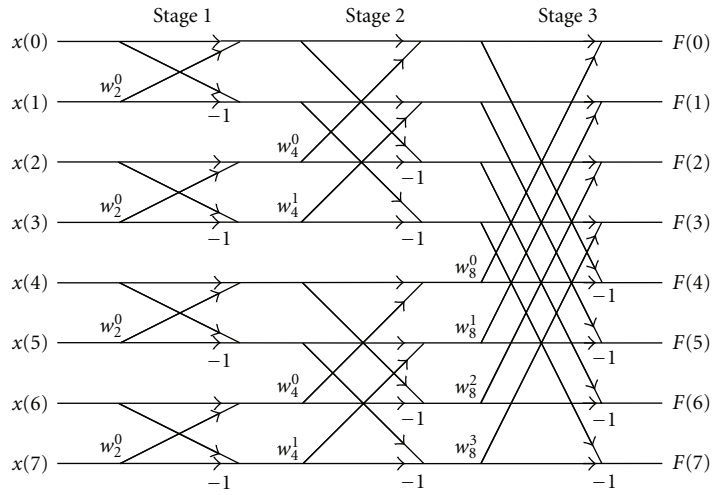


Figure 11: Radix-2 butterfly structure for 8-point FFT implementation.

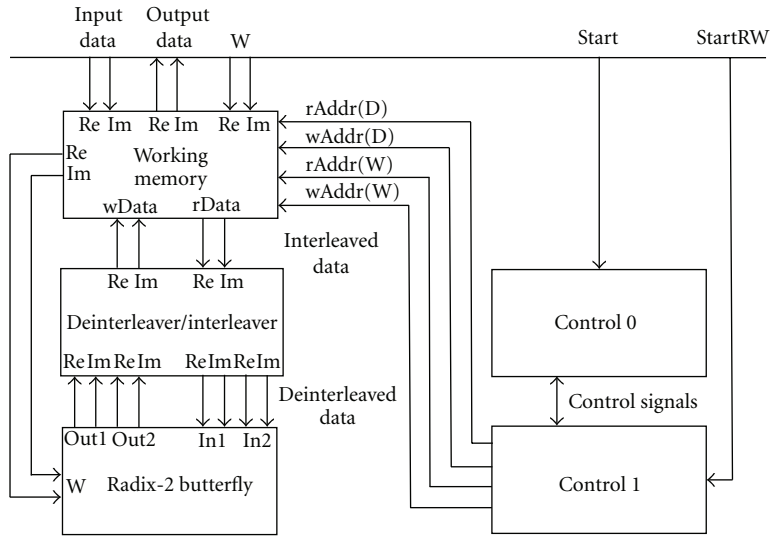


Figure 12: HoneyComb implementation of the FFT algorithm based on radix-2 butterfly approach.

The test board for the HoneyComb prototype is designed as an FPGA Mezzanine Card (FMC) as defined by the VITA 57 standard [21], which is an expansion card standard supported by recent Xilinx FPGA evaluation boards. The HoneyComb IC has 134 signal connections to the FPGA, therefore the board features a 400 pin high pin count (HPC) connectors, which supports up to 160 single-ended I/Os. It can be used with board featuring an FMC HPC connector like the Virtex-6 ML605, Kintex-7 KC705, or Virtex-7 VC707 evaluation kits. A 3D render image of the final layout of the HoneyComb board, which is currently in production, is shown in Figure 9.

In any of these configurations, the HoneyComb IC can be used as an application accelerator within a Standard-PC, to which it is connected via the PCI-Express slot of the base board. In this configuration, which is shown in Figure 10, the HoneyComb IC has access to the memory located on the FPGA board as well as to the PC's main memory. Configuration and computation are initiated by the host-PC.

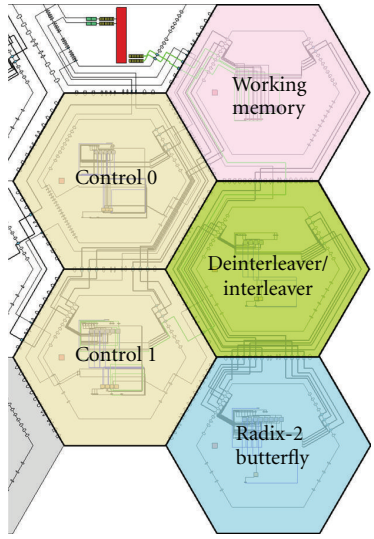


Figure 13: Mapping of the radix-2 FFT algorithm on the HoneyComb architecture.

7. Mapping Applications onto the HoneyComb Architecture

For demonstration purposes we developed a set of applications, which includes a 1024-point FFT [22], Wavelet algorithm according to JPEG2000 specification [23], iMDCT [24], and Advanced Encryption Standard (AES256) [25] implementation. These applications have been mapped on the HC array and the calculated data sets have been compared with reference implementations in C/C++ with a perfect match. Since both implementations are working with 32 bit precision these results were possible and expected. As described in the layout section, we applied the application-tailored reduction technique using these applications to reduce the resource requirements of the prototype in order to fit the design into an area of 16 mm^2.

The following procedure was used to generate the executable program code and to adapt the HoneyComb model to application requirements. We implement the initial high level program in HCL based on the C/C++ reference implementation. After compiling the HCL description we received the HCA description, which is a low level assembly program for the HoneyComb array. After additional optimizations we used the assembly code to determine the target HoneyComb model by adding additional resources to the architecture according to FFT requirements. Finally, the HC-assembler has been used to generate the final binary program code which can be executed by the modified HoneyComb model.

The following subsections will give an overview how the target applications have been mapped onto our architecture.

7.1. 1024-Point FFT Algorithm. The fast Fourier transform [22] is an efficient algorithm (Cooley-Tukey [26]) for discrete Fourier transform (DFT). It computes frequency components for a given sequence of values which can be a number of consecutive samples for a given signal. Besides the digital signal processing this algorithm is important for a wide variety of applications.

While direct DFT calculation results in complexity of $O(N^2)$ the FFT reduces it to O(NlogN) and allows calculation of datasets with thousands of point in a relative short time. The usual hardware implementation is based on butterfly structures. Each butterfly has a specified number of inputs and outputs which is specified as radix-parameter. In case of two inputs we are talking about radix-2 implementation. Figure 11 shows the basic structure for a radix-2 butter fly structure with eight points.

For the mapping of this algorithm we chose the radix-2 implementation. Since the resulting array composition has not enough resources to implement the complete radix-2 FFT directly we decided to map this algorithm partly in the area and partly into the time domain. This approach is supported by the HoneyComb array by using the multicontext capability. Therefore the working data has to be stored in memory modules (MEMHCs) and retrieved according to the butterfly structure. To generate the required address sequences two separate HoneyComb cells are dedicated to this task and are marked as control cells (see Figure 12). One separate MEMHC has been assigned for holding the coefficients W_i and current working data. Depending on incoming addresses the memory cell retrieves addressed data and coefficients. Since current MEMHCs only include single port memories retrieved data is read one by one on each cycle. An additional deinterleaver cell is used to parallelize sequential data values and forward each pair of values (In1, In2) to the butterfly cell. Thus, the butterfly cell is able to calculate one output pair (Out1, Out2) every two cycles and forward the results through the interleaver to the memory cell.

Two external signals control the execution of the FFT algorithm: StartRW and Start. StartRW controls whether the working data and coefficients are read from or written

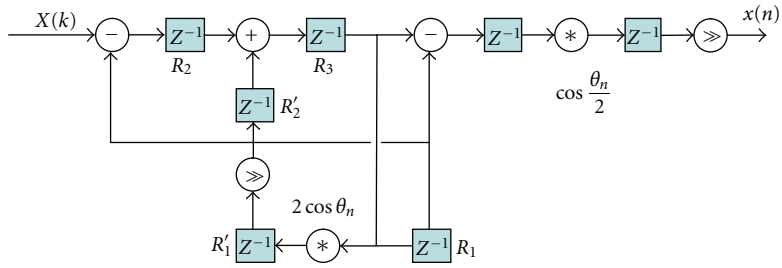


Figure 14: iMDCT structure fitting hardware implementation.

to the working memory. It triggers a linear address generator which generates an address stream on predefined output port of the control 1 cell. Whether working data or coefficients are transfers depends on the connected memory interface. These connections are established by dynamic reconfiguration when memory has to be filled or calculated data is to be retrieved. Once the memory is loaded the calculation can be started with the Start signal.

The butterfly uses a fixed point representation for the calculation which can be adapted to application requirements by resetting the fixed point position. The resulting algorithm is not limited to 1024-point FFT. Depending on the stored coefficients W_i it is possible to calculate data sets of different sizes. The limiting factor is only defined by the maximum size of the memory modules in the MEMHC. The ASIC implementation includes eight 1024 × 32bit memory modules. So, the maximum size is limited to 2048-point FFTs by this version.

Figure 13 shows one possible FFT configuration on the HoneyComb array as described in the sections above. The whole configuration requires 4 DPHCs and 1 MEMHC.

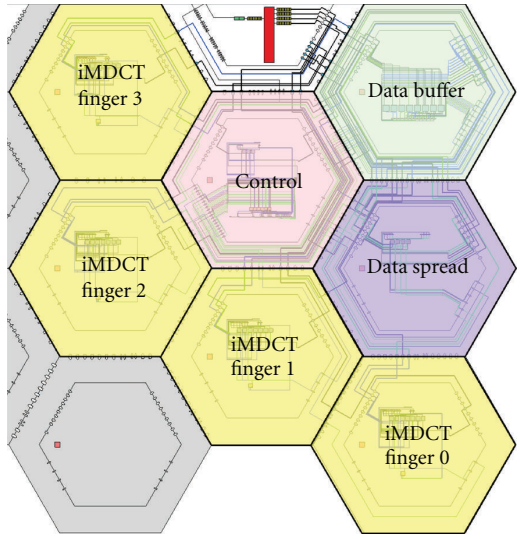


Figure 15: Mapping of the iMDCT algorithm onto the HoneyComb architecture.

7.2. 1024-Point iMDCT Algorithm. The inverse modified discrete cosine transform is an algorithm quite similar to the prior mentioned DFT. This modified version is primarily used for audio compression standards like MP3, AAC, and OggVorbis. The direct calculation of as iMDCT has the same complexity as the DFT which is $O(N^2)$. Like in case of FFT similar butterfly-based approaches exist to reduce the complexity. There are two particularities of this algorithm worth mentioning: for once the fact that it works with real numbers only and for second that a given number N of spectral values results in 2*N samples. The first point halves the memory requirements compared to FFTs since only real numbers have to be stored. In case of 1024-point iMDCT we get 2048 sample values, which is specified according to OggVorbis Audio compression.

Since one butterfly approach has already been implemented on the HoneyComb array we decided to use a more direct approach. Therefore we used the transformations introduced by Nokolajevic and Fettweiss [24] resulting in a structure which can be implemented directly in hardware. Figure 14 shows a slightly modified version after adding a few additional registers.

Even though the algorithm does not pursue the same approach as the FFT, the implementation resulted in a similar configuration. One MEMHC is used to store working data as well as cosine coefficients. But instead of using the MEMHC in RAM mode it is used in FIFO mode. Though, once a value has been read it has to be put back if it is supposed to be available again. This function is performed by a separate cell (data spread), which distributes the values from memories and forwards a copy to the iMDCT finger cells. Those cells implement the complete iMDCT structure as shown in Figure 15.

To compute a single output the iMDCT structure requires to receive 1024 spectral and cosine ($2\cos\theta_n$) values. On the final iteration the multiplication with the cosine value ($\cos(\theta/2)$) finalizes the calculation. Since the algorithm is working with fixed point numbers shift operations are required to correct the results. The resulting values are forwarded through the IOHC cell directly out of the array.

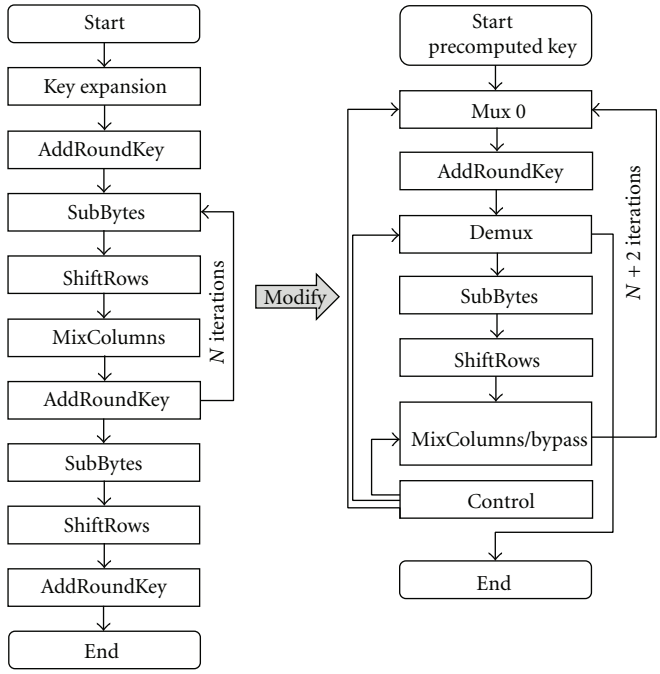


Figure 16: Original AES operation scheme and the modification of the implementation for the HC-Array.

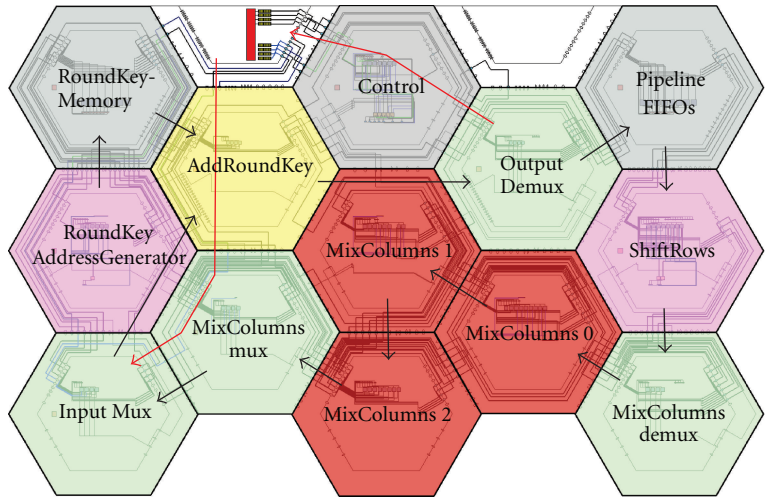


Figure 17: Resulting mapping if the AES algorithm on the HoneyComb architecture.

This approach is quite slow compared to butterfly solution. However, it is a good example to demonstrate multi-context capabilities of the DPHCs.

7.3. The Advanced Encryption Standard. The Advanced Encryption Standard (AES) [25] is a symmetric-key encryption algorithm standardized by NIST [27] in 2000. The symmetric character of the algorithm allows encryption and decryption of electronic data with the same cypher key. The algorithm is fast in software as well as in hardware. However, due to its nature the latter is always more efficient.

AES is using a block code of 4×4 bytes blocks. A set of transformation operations are defined which are executed in repetitions in several rounds to perform the encryption of the input text. Additionally, reverse rounds are defined to reverse the encryption with the same encryption key.

Four high level steps are defined for the encryption rounds: AddRoundKey, SubBytes, ShiftRows and MixColumns. AddRoundKey combines current block elements with the cypher key by applying the xor operation. SubBytes substitutes the bytes within the blocks due to a given lookup table. ShiftRows reorders the rows of the given block, while the MixColumns function reorders the columns of the block.

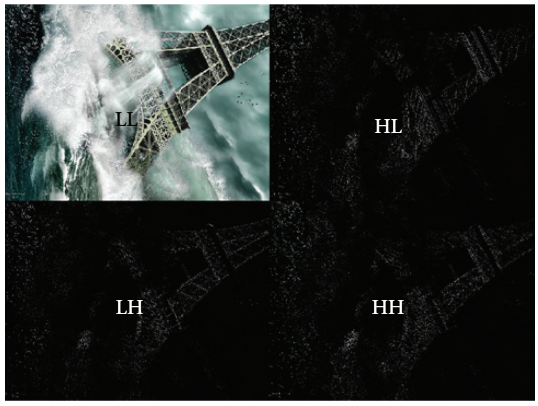


Figure 18: 2D wavelet transformed image.

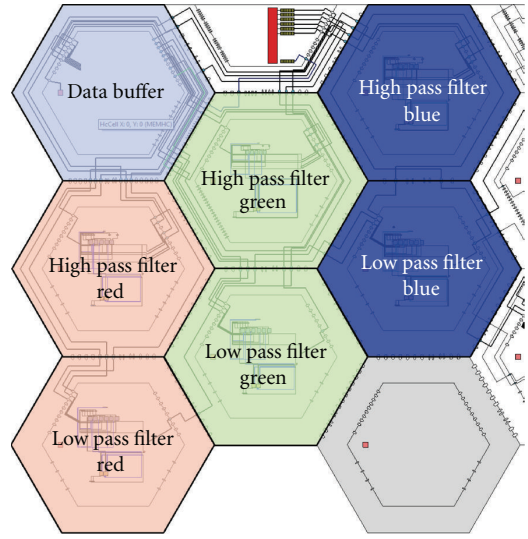


Figure 19: 3-time wavelet algorithm implementation on the HoneyComb architecture.

AES key sizes are specified with 128, 192, and 256 bits. Before starting the actual data encryption the plain text key has to be expanded. Here, the key is divided in 128-bit blocks (RoundKeys), whereas the blocks are partly filled with the key data and partly are calculated recursively. After the expansion the RoundKey can be applied with the AddRoundKey operation on the data during the encryption. Since the key expansion is not limiting the AES performance it will be done in software and the RoundKey will be transferred into the HoneyComb array for the actual encryption process.

The AES algorithm specifies a very specific order for the execution of those operations. However, to reduce the resource requirements, the block diagram has been slightly modified. Due to hardware restrictions, it is not possible to call any function at will, so each path has to be defined in advanced, what is shown with each path on the right hand side of Figure 16. It was important to avoid multiple implementations of the same function to save as many resources as possible. Therefore, the whole algorithm is working both ways: in parallel mapped in the hardware area and sequential by iterating the data by reusing the same resources over and over again. Depending on the key size the number of iterations changes, so all key sizes are supported by the HoneyComb implementation.

Figure 17 shows the results mapping of the AES algorithm on the HoneyComb architecture. The darts in this figure represent data paths between operational cells and have always four concurrent connections to transport one column of a block at once. Thus, every four cycle one block is transferred from one cell to another. The MEMHC with the pipeline functionality is used for once to increase the pipeline depth of one round and secondly holds the lookup table data for the SubBytes operation. The increasing of the pipeline depth increases the efficiency of the mapping enormously. Since this algorithm is working iteratively, it is required to empty the pipeline completely before starting the calculation for the new data set. This refill period leads to idle time on the array and decreases the resulting performance. By using deeper pipelines it is possible to reduce the idle time relatively to calculation time.

The result is high performance and efficiency. The maximum pipeline depth given by the FIFOs has 1024 entries, so including the calculation cells with additional register stages over 256 blocks can be calculated in one pass. The final mapping of the AES algorithm requires all available cells on our ASIC: two MEMHCs and eleven DPHCs.

7.4. Wavelet Application. The Wavelet algorithm [23] works on whole images to filter the higher and lower frequencies of the color dispersion. It can be applied horizontally and vertically to achieve 2Dtransformation. The JPEG2000 standard uses this algorithm to separate frequencies for better compression results. Quite similar to the JPEG approach loss of high frequency shares cannot be noted by the eye, so loss of this kind of information does not degrade the reextracted image noticeably.

Simply spoken, the algorithm compares three neighboring pixels and calculates the high and low frequency shares. When this is done horizontally, the results are two half-sized images, whereas the left image contains the low pass results and the right image contains the high pass results. When the algorithm is applied vertically this results in an image with the same size but consisting of four separate quarters (see Figure 18). Each quarter is the result of horizontal and vertical low and high pass filtering, what is marked with the letter L/H. The first letter indicates horizontal filtering, the second the vertical filtering.

The low and high pass filters require each one cell on the HoneyComb architecture. So, to implement a complete wavelet filter we need two DPHC cells to handle one data stream, which can be a color component, like red, green or blue. Because of tight timing constraints between these two cells a FIFO has been used to improve the performance.

To be able to compute all three color components of an RGB image and to improve the resulting performance we implemented three times the complete wavelet filter,

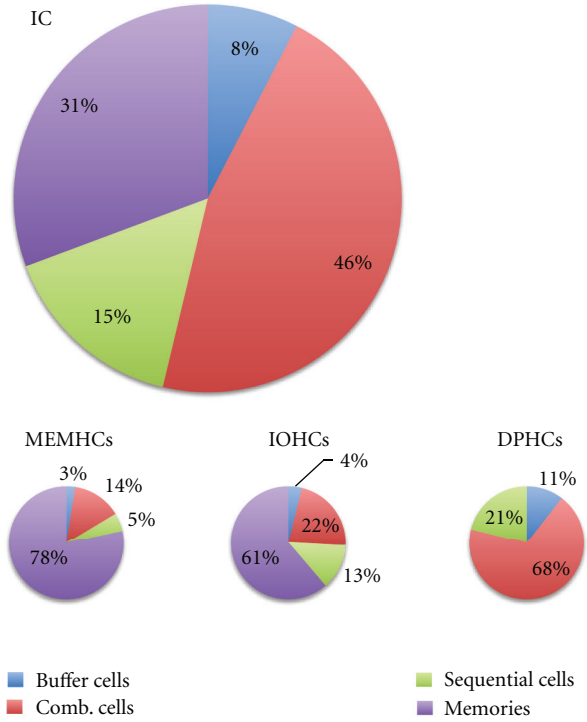


Figure 20: Distribution of the standard cell types within the HoneyComb array prototype.

what resulted in a configuration with six DPHCs and one MEMHC (see Figure 19).

8. Results Overview

The applications described in the last section have been taken for performance evaluation. These applications have been mapped on the HC array and the calculated data sets have been compared with reference implementations in C/C++. As described in the layout section, these applications have been used for the reduction of the prototype to reduce the resource requirements and to fit the design into an area of 16 mm^2.

The application results, which were obtained by cycle-accurate simulation of the architecture's VHDL model and normalized to 100 MHz, can be found in Tables 1 and 2. The configuration time specifies the amount of time it takes to configure the HC array for the application execution. These values are quite small and allow the architecture to switch between applications several thousand times per second. Therefore, configuration sequencing and resources reuse becomes practicable with this approach.

The AES implementation delivers the most impressive results. There, the maximum performance without reconfiguration interruptions is in the range of up to 26 MB/s which is very high considering the clock speed of only 100 MHz. Figure 5 shows the running application represented by the HCViewer debugging tool.

The maximum power consumption for the AES256 application, which is the maximum for this application set, is about 150 mW. This value, which was obtained with PrimeTime, includes the dynamic as well as the static power consumptions of the core cells and is right until now an estimated value by the synthesis and layout tools. The exact value will be evaluated once the prototype is finished.

Figure 20 shows the breakdown of the standard cell types for each HoneyComb cell type of the ASIC prototype. It is noticeable that the MEMHCs and IOHCs are mostly composed of integrated SRAM blocks. In case of the MEMHC these memory blocks are the main part of the configurable cell functions. In case of the IOHC the memory blocks are part of the FIFOs for clock domain crossing. The DPHC cells are mainly dominated by the combinational logic required for arithmetic units, multiplexers, and decoders. However, 8% of the design is composed of buffer cells, which is a quite good value considering the fact that we did not optimize design to minimize the use of buffers and did the layout analysis quite conservative.

Figure 21 reflects the area distribution of the HoneyComb architecture. Since the prototype includes 11 DPHCs 57% of the area is allocated by those cells. However, the breakdown of the functional unit (FU) and routing unit (RU) of the DPHCs shows optimization potential for the future. Right now, we used simple multiplex structures to design the architecture; especially the RUs are using multiplex structures extensively. By substituting the multiplexers with more efficient crossbar structures, according to our experiments, we expect to save up to 50% of the area.

9. Conclusion

This contribution presented an application-tailored methodology for a reconfigurable architecture, the HoneyComb

Table 1: Performance results for selected applications at 100 Mhz.

Application	DPHCs	MEMHCs	Config. time	Performance
AES256	11	2	6,85 μs	25,6 MB/s
iMDCT 1xfinger	3	1	24,06 μs	47,6 blocks/s
iMDCT 7xfingers	11	2	25,60 μs	333,46 blocks/s
FFT1024	4	1	7,65 μs	10850 blocks/s
Wavelet	6	1	3,15 μs	0,6 cycles/pixel

Table 2: Synthesis and power evaluation results at 100 MHz.

Application	DPHC area (μm^2)	MEMHC area (μm^2)	IOHC area (μm^2)	Leakage power (mW)	Dynamic power (mW)
AES256	362636	1226638	623680	5.59	146.73
iMDCT 1xfinger	461697	1290972	812529	7.11	66.23
FFT1024	472477	948950	787658	7.26	75.02
Wavelet	250042	1246197	728551	4.2	87.84
ASIC	652285	1299802	868313	10.76	

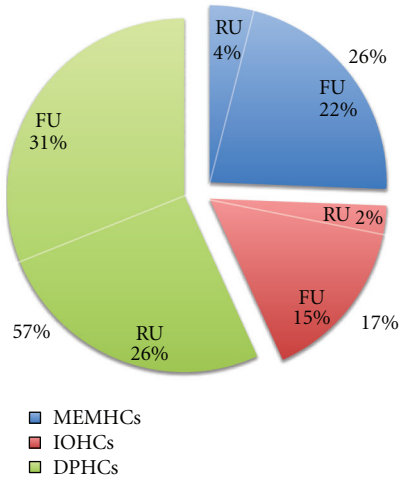


Figure 21: Area distribution across cells within the HoneyComb array prototype.

architecture. Since a fully flexible approach is simply too expensive and usually not desired it is possible to reduce the architecture according to a predefined set of applications. This approach saves silicon area and therefore money. In case of our prototype we reduced the initially required area to less than 50%. The prototype of the HoneyComb architecture has been produced and is already delivered. The layout of the PCB is complete, and the PCB is currently in production. We are expecting to have a running system within the next few months.

The performance results are also promising. Compared to Intel Core 2 Quad processor with 2666 MHz, which reaches about 200 MB/s executing the AES256 application resulting in an overall performance per core of about 50 MB/s, our results are excellent.

Still a lot of work has to be done regarding the area efficiency and programming interface. Currently the used simple multiplexing structures within the array can be replaced by more efficient cross-connect structures. Also, support for at least C is required and would grant access to a wide range of applications.

Acknowledgments

The authors acknowledge support by Deutsche Forschungsgemeinschaft and Open Access Publishing Fund of Karlsruhe Institute of Technology.

References

[1] Xilinx Inc., http://www.xilinx.com/.

[2] Altera Corp, http://www.altera.com/.

[3] J. Becker, T. Pionteck, and M. Glesner, "DReAM: a dynamically reconfigurable architecture for future mobile communication applications," in *Proceedings of the 10th International Conference on Field Programmable Logic and Applications*, Villach, Austria, 2000.

[4] R. Kress, *A fast reconfigurable ALU for Xputers [Ph.D. dissertation]*, Kaiserslautern University, 1996.

[5] T. Oppold, T. Schweizer, J. F. Oliveira, S. Eisenhardt, and W. Rosenstiel, "CRC—concepts and evaluation of processor-like reconfigurable archtitectures," *IT-Information Technology*, vol. 49, no. 3, p. 147, 2007.

[6] A. Abnous, H. Zhang, M. Wan, G. Varghese, V. Prabhu, and J. Rabaey, "The Pleiades Architecture," in *The Application of Programmable DSPs in Mobile Communications*, John Wiley & Sons, Chichester, UK, 2002.

[7] P. Master, "The next big leap in reconfigurable systems," in *IEEE International Conference on Field-Programmable Technology (FPT '02)*, pp. 17–22, December 2002.

[8] E. Schüler and M. Weinhardt, "XPP-III: the XPP-III reconfigurable processor core," *Lecture Notes in Electrical Engineering*, vol. 40, pp. 63–76, 2009.

[9] N. Suzuki, S. Kurotaki, M. Suzuki et al., "Implementing and evaluating stream applications on the dynamically reconfigurable processor," in *Proceedings of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM '04)*, pp. 328–329, April 2004.

[10] P. M. Heysters, G. J. M. Smit, and E. Molenkamp, "Energy-efficiency of the MONTIUM reconfigurable tile processor," in *Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA '04)*, pp. 38–44, Las Vegas, Nev, USA, June 2004.

[11] S. C. Goldstein, H. Schmit, M. Budiu, S. Cadambi, M. Matt, and R. R. Taylor, "PipeRench: a reconfigurable architecture and compiler," *Computer*, vol. 33, no. 4, pp. 70–77, 2000.

[12] D. C. Chen and J. M. Rabaey, "A reconfigurable multiprocessor IC for rapid prototyping of algorithmic-specific high-speed DSP data paths," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 12, pp. 1895–1904, 1992.

[13] C. Ebeling, C. Fisher, G. Xing, M. Shen, and H. Liu, "Implementing an OFDM receiver on the RaPiD reconfigurable architecture," *IEEE Transactions on Computers*, vol. 53, no. 11, pp. 1436–1448, 2004.

[14] G. Lu, H. Singh, M.-H. Lee et al., "The MorphoSys dynamically reconfigurable system-on-chip," in *Proceedings of the 1st NASA/DoD Workshop on Evolvable Hardware*, pp. 152–160, 1999.

[15] E. Mirsky and A. DeHon, "MATRIX: a reconfigurable computing architecture with configurable instruction distribution and deployable resources," in *Proceedings of the IEEE Symposium on FPGAs for Custom Computing Machines*, pp. 157–166, April 1996.

[16] T. Miyamori and U. Olukotun, "A quantitative analysis of reconfigurable coprocessors for multimedia applications," in *Proceedings of IEEE Symposium on FPGAs for Custom Computing Machines*, pp. 2–11, April 1998.

[17] A. Thomas and J. Becker, "New adaptive multi-grained hardware architecture for processing of dynamic function patterns (Neue adaptive multi-granulare Hardwarearchitektur)," *IT-Information Technology*, vol. 49, no. 3, p. 165, 2007.

[18] A. Thomas and J. Becker, "Multi-grained reconfigurable hardware architecture with online-adaptive routing techniques," in *Proceedings of the IFIP International Conference on Very Large Scale Integration (IFIP VLSI-SOC '05)*, Perth, Western Australia, October 2005.

[19] P. Briggs, *Register Allocation via Graph Coloring*, Rice University, Dissertation, 1992.

[20] C. J. Tseng and D. P. Siewiorek, "Automated synthesis of data paths in digital systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 5, no. 3, pp. 379–395, 1986.

[21] VMEbus International Trade Association (VITA)—FMC Marketing Alliance, http://www.vita.com/fmc.html.

[22] E. O. Brigham and R. E. Morrow, "The fast Fourier transform," *IEEE Spectrum*, vol. 4, no. 12, pp. 63–70, 1967.

[23] St. Mallat, *Phane: A Wavelet Tour of Signal Processing*, Academic Press, 2009.

[24] V. Nikolajevic and G. Fettweis, "New recursive algorithms for the unified forward and inverse MDCT/MDST," *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, vol. 34, no. 3, pp. 203–208, 2003.

[25] J. Daemen and V. Rijmen, "AES proposal: Rijndael," 1999.

[26] W. J. Cooley and W. J. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics of Computation*, vol. 19, pp. 297–301, 1965.

[27] National Institute of Standards and Technology, http://www.nist.gov/index.html.

Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2012, Article ID 578363, 10 pages
doi:10.1155/2012/578363

Research Article

QoSS Hierarchical NoC-Based Architecture for MPSoC Dynamic Protection

Johanna Sepulveda,[1] Ricardo Pires,[2] Guy Gogniat,[3] Wang Jiang Chau,[1] and Marius Strum[1]

[1] *Microelectronics Laboratory, Polytechnic School, University of São Paulo, 05508900 São Paulo, SP, Brazil*
[2] *Federal Institute of Education, Science and Technology of São Paulo, 01109010 São Paulo, SP, Brazil*
[3] *Information and Communication Science and Technology Laboratory, University of South Brittanny, 56100 Lorient, France*

Correspondence should be addressed to Johanna Sepulveda, jsepulveda@lme.usp.br

Received 20 January 2012; Revised 15 May 2012; Accepted 15 May 2012

Academic Editor: Elmar Melcher

As electronic systems are pervading our lives, MPSoC (multiprocessor system-on-chip) security is becoming an important requirement. MPSoCs are able to support multiple applications on the same chip. The challenge is to provide MPSoC security that makes possible a trustworthy system that meets the performance and security requirements of all the applications. The network-on-chip (NoC) can be used to efficiently incorporate security. Our work proposes the implementation of QoSS (quality of security service) to overcome present MPSoC vulnerabilities. QoSS is a novel concept for data protection that introduces security as a dimension of QoS. QoSS takes advantage of the NoC wide system visibility and critical role in enabling system operation, exploiting the NoC components to detect and prevent a wide range of attacks. In this paper, we present the implementation of a layered dynamic security NoC architecture that integrates agile and dynamic security firewalls in order to detect attacks based on different security rules. We evaluate the effectiveness of our approach over several MPSoCs scenarios and estimate their impact on the overall performance. We show that our architecture can perform a fast detection of a wide range of attacks and a fast configuration of different security policies for several MPSoC applications.

1. Introduction

SoC designers have to face up tight development times as well as the rapid evolution of current applications [1]. To be cost effective, SoCs are often programmable and integrate different applications on the same chip (i.e., cell-phone, personal digital assistant) [1]. Although sharing many of the hardware components on the SoC, different applications executed on a single chip may present very different requirements and design constraints. Such type of system is called multiapplication [2]. MPSoCs have been proposed as a promising architecture choice to overcome the new challenging application requirements. A MPSoC integrates multiple programmable processor cores, specialized memories, and other intellectual property (IP) components into a single chip [1]. MPSoC platforms allow simultaneous execution of several applications in the same structure. Current ubiquitous computing and flexibility in SoC design trends promote the resource sharing and upgrading capabilities. As MPSoCs are pervading our lives, security is emerging as an extremely important design requirement. Many of the current electronic systems embedded into an MPSoC are used to capture, store, manipulate and access sensitive data, and perform several critical functions without security guarantee [3]. Due to the increasing complexity, flexibility, intrinsic embedded constraints, and strict requirements, the implementation of security is considered as a challenging task. The security at MPSoC is specially challenging, as the flexibility offered by the platform also causes vulnerability. Each application supported by the MPSoC is characterized by a different set of security rules, called security policy. The set of applications can be mapped dynamically on the MPSoC. Therefore, there is no a single and static security requirement, but a set of ever changing security policies that must be satisfied. The MPSoCs security policy must be able to supply different levels of security and be capable

of changes during operation time. A McAfee report [4] estimates an exorbitant increasing of embedded attacks in the last 5 years and loses of billions of dollars [3]. MPSoC can be attacked via hardware/software [3]. Software attacks are responsible for 80% of security incidents [4]. All software attacks start with an abnormal communication. In this paper, we address protection of the MPSoC against software attacks by implementation of security policies at the NoC-based communication structure.

In order to support the MPSoC high communication requirements, the network-on-chip (NoC) approach is employed [5–7]. An NoC is an integrated network that uses routers to allow the communication among computation components. Data flows through the NoC as packets. The integration of security at the NoC is naturally advantageous [8]. The NoC may contribute to the overall security of the system, providing the ideal mean for monitoring systems behavior and detecting specific attacks [8]. The communication structure is becoming the heart of the MPSoC [7]. It has a significant impact on the overall MPSoC performance. To make feasible MPSoC protection by NoCs, the security policy must be customized in order to provide a better tradeoff between system performance and security. Our work proposes the implementation of QoSS (quality of security service) to overcome present SoC vulnerabilities. QoSS is a novel concept for data protection that introduces security as a dimension of QoS (quality-of-service). In contrast with previous works, different security levels deployment allows the best tradeoff between system security and performance requirements. In our work, QoSS is implemented at the network-on-chip (NoC) to provide predictable security levels of the system by adding functionality to the routers of the network and consequently changing some local configuration parameters or modifying the network interfaces. The goal of our work is to present a layered dynamical NoC-based architecture to efficiently meet the changeable MPSoC security requirements. Our hierarchical architecture distributes the security policy management by partitioning the NoC topology into different *security zones* (low NoC), ruled by a *local* security policy and connected through a global interconnect (high NoC), which implements a *global* security policy. Our hierarchical approach improves system's performance while effectively handling the security policy changes. Each zone integrates a set of mechanisms capable of being configured according to QoSS needs of each application. We show that our architecture can perform a fast detection of a wide range of attacks and a fast configuration of the different security policies for several MPSoC applications. We also show that penalties due to the integration of the dynamic NoC-based security architecture are limited to a fraction of time and space of the system. The different levels of security of each *security zone* arise from the configuration of the parameters corresponding to the NoC security mechanisms. Two techniques are employed in order to achieve an efficient configuration: (1) security mechanisms are implemented hierarchically, therefore, avoiding interruption in NoC execution; (2) QoS (quality-of-service) mechanisms are employed to provide predictable penalties while the network interfaces are modified. The experiments were performed using a SystemC-TLM-timed simulation framework. It automatically carries out performance evaluations for a wide variety of MPSoC scenarios.

In summary, the novelties of our work are as follows:

(i) implementation of a layered dynamical MPSoC security by the use of a hierarchical NoC, we divide the MPSoC security policy into global and local ones, which is suitable for dynamic systems;

(ii) creation of security zones in the MPSoC;

(iii) implementation of QoSS for hierarchical NoCs.

2. Previous Works

Security integration at the NoC level was addressed in the works of [8, 9]. They take advantage of the NoC wide system visibility and critical role in enabling system operation, exploiting the NoC to detect and prevent attacks. However, they implement a single-level static security, which may be inefficient as a solution for the highly changeable MPSoC security requirements. The work presented in [6] proposes the integration of ciphering techniques at the network interface in order to ensure the secrecy of the exchanged information through the NoC. The proposed mechanism ensures that no unencrypted data leaves the NoC. A key-keeper secure core is responsible for the key distribution in the NoC. New keys can be downloaded and stored in the key-keeper core by using encryption techniques. The papers [8–10] integrate a table at the network interface containing the access control rules of each IP. They specify how a component of the NoC can access the protected device. Packets that do not satisfy these access control rules are discarded. The purpose of [9] is to prevent attacks by verifying the source and size of the packets. The packets that do not obey the communication rules are discarded. This work also integrates a secure network manager component to monitor the NoC behavior. Its purpose is to prevent four common NoC attacks: denial-of-service (DoS), draining, extraction of secret information, and modification. The filter of [10, 11], called data protection unit (DPU), enables the communication only if the type of operation requested by the initiator of the communication is authorized. The work of [12] proposes an architecture composed by three modules in order to avoid code injection attacks: (1) SPU (stack protection unit), to track the transactions; (2) ITU (instruction trace unit), to track the instructions; (3) LSM (local security manager), to react upon the reception of an alert signal of SPU or ITU. These previous works show the main role of the NoC in the security of the system. It is a powerful structure for the surveillance and detection of attacks in SoCs. However, the adoption of these previous solutions to address MPSoC security challenges present three main limitations. (1) They implement a single NoC security level for the entire SoC; (2) they implement static security policies; (3) they are not appropriate for multithreaded systems. In our previous work [5], we developed a dynamical NoC-based protection for SoCs. However, it uses central control blocks that present a strong impact on the overall

area due to link overhead. The purpose of the present work is to overcome these limitations.

3. Security

Many SoCs interact with other electronic devices, in many cases wirelessly. By interacting with other digital devices, an SoC may receive viruses (or other similar malicious pieces of code). Among the motivations for someone to attack a SoC, we underline three examples: (1) economical gain by obtaining confidential information (e.g., passwords, IP bitstreams) stored in an SoC; (2) reputation: a hacker may attack a SoC by viewing this action as a personal challenge; (3) vandalism: the purpose is to cause loss or damage to a SoC. Viruses may be used for this purpose.

Attacks exploit different system vulnerabilities. An attack can be defined as any unauthorized attempt to access or to use the system resources [3]. An attack can be conducted through three different ways: (1) software: tampering with executable instructions through the communication structure [3]; (2) microprobing: an invasive technique that involves physical manipulation of the system [5]; and (3) side-channel: based on information gained from the physical implementation, including timing analysis, power analysis, electromagnetic analysis, and fault induction [6]. According to the purpose of the attacks, they can be classified into three categories: (1) extraction: unauthorized reading of critical data that is being exchanged through the network from/to a secure target; (2) modification: unauthorized change of critical data that may be done through writing actions, state modifications, data creation or removal; (3) denial of Service, whose aim is to bring down the system performance.

In order to prevent and to mitigate attacks to the NoC, security services can be implemented. The main function of security services is to protect network resources and data exchanges by means of communication management [9, 11]. There are six security services [10]: (1) confidentiality which ensures the data secrecy; (2) integrity which assures that data is kept unchanged during any operation; (3) authentication which validates the sender IP integrity; (4) access control which allows or denies the use of a particular resource; (5) availability which ensures the use of the network resources; (6) nonrepudiation that maintains evidence of NoC communication events. It has been shown that the most successful attacks are the software-based ones [3]. One of the most obvious threats to MPSoC security during its normal operation occurs at its interface to external devices, frequently involving reconfigurable devices or wireless communication IPs embedded onto the SoC. In such cases, the vulnerable IPs fall under control of an external attacker. Thus, these IPs may become malicious. Under the attacker's control, the infected IPs may try, for example, to obtain sensitive information, like passwords or FPGA bitstreams, stored inside the SoC, and send it to the external world. An interface IP may also become a door by which viruses enter the SoC. There are known cases of SoC attacks that have succeeded [4]. In our work, we consider attacks that take advantage of the lack of MPSoC security upgrading. However, our architecture can defend against a broader range of attacks. MPSoC's characteristics demand that security policy must be upgraded as a response of the execution of a new application on it. We consider, for example, that there is an MPSoC designed to support three applications (*A1*, *A2,* and *A3*). Initially, the set of applications *A1* and *A2* is being executed in the MPSoC when, at some instant, the application *A3* must be executed. The tasks of *A3* are mapped onto the components of the MPSoC. The task that performs critical functions and the sensitive information (i.e., a ciphering key or personal data) of *A3* is mapped together with the tasks of *A1* and *A2*. The tasks of *A3* will be unprotected, if the security policy that defines the access control rights is not upgraded for the new scenario. An attacker can take advantage of this threat and use the rights over *A1* and *A2* to expose/modify the sensitive information of *A3* or/and avoid the utilization of the MPSoC resources by the tasks of *A3* by keeping busy the component with a never-ending *A1*/*A2* task execution.

4. Quality of Security Service

4.1. Security Mechanisms. The implementation of security services is carried out through security mechanisms which increase the complexity of the NoC. Optimal NoC configuration demands a deep exploration of the wide NoC design space. Recently, the work in [13] proposes the quality-of-security-service QoSS (quality-of-security-service) for extrachip (conventional) networks. It explores the tradeoff between the system trustfulness and its performance. The traffic of a single embedded application may integrate several flows, each of which characterized by different security requirements. The QoSS concept allows differentiated treatment for the data exchange carried out through the NoC. The QoSS can be implemented either by adding functionality to NoC routers and consequently changing some local configuration parameters or by modifying network interfaces. Different security levels are implemented through security mechanisms. The security choices represent a special configuration of the *security mechanisms*. The security mechanisms use the embodied information within the packet to perform access control and authentication on the packet arriving at (1) the NoC interface or (2) the router. For comparison reasons, we addressed access control and authentication services in order to neutralize extraction and modification attacks. Additional security service can be handled with complementary cyphering techniques [8]. Access control and authentication security mechanisms are implemented as *firewalls*. Each firewall stores the security policy information of each resource in a security table. Whenever a transaction takes place, the information embodied in the packet is checked against the security table information.

4.2. Access Control. It regulates NoC traffic, allowing or denying data exchanges between a master-slave pair based upon a set of rules. The communication control flow was modified to manage packet accesses by using the data stored in the *security table* of the firewall which contains the access rights of each *MPSoC* computation component. The rights change according to the applications that are mapped on the MPSoC at that time. Our access control service implements four

Table 1: Access control levels.

Level	SV	OV	RV
L0			
L1	×		
L2	×	×	
L3	×	×	×

Table 2: Authentication levels.

Level	SV	PV	CV
L0			
L1	×		
L2	×	×	
L3	×	×	×

security levels, which arise from the combination of three security mechanisms. Note that our method can support any number of security levels. Table 1 shows the different access control levels. A higher security level compares a larger number of information embodied in the packet (source, type of operation, and master role). Unauthorized packets are discarded.

SV: it verifies the source/target of the transaction that is, the rights that the master has over the target slave component. It identifies the task and the thread authorized for each transaction.

OV: it verifies that the authorized operations are executed in the correct MPSoC resources. It also regulates the memory access through the verification of the fields: *address*, specifying the memory addresses involved in the operation; *size*, corresponding to the maximal range of memory addresses capable of being modified by the operation; *time*,stating the number of memory modifications allowed by the transaction.

RV: it verifies the role of the initiator component of the transaction.

The *SV* offers a basic access control. It verifies the existence of the message destination and that the master and slave components have not identical NoC addresses. Such characteristic avoid possible DoS (Denial of Service) attacks through *livelock*, characterized by the insertion of a packet that cannot reach its destination, and draining attacks, which is characterized by the intentional wasting of NoC resources. The *OV* can be used to detect *buffer overflow* attacks, one of the common embedded attacks [3]. It explores the rights of an authenticated component in order to perform unauthorized operations [3]. The RV mechanism includes the operation context of the master of the transaction.

4.3. Authentication. The authentication security service verifies the integrity of the source of critical data. It does this by checking if the route taken by the packet is consistent with the source IP field contents.

Our authentication service implements four security levels, which arise from the combination of 3 security mechanisms. Table 2 shows the different authentication levels. These strategies make very difficult for a malicious master to successfully send a packet as it would be another master.

SV: it is the same mechanism described at Section 4.2.

PV: in order to perform the authentication, a routing trace is embodied in the header field of the packet. This field content is modified by the routers along the communication path of each packet. In this process, when a packet traverses a router, the packet header receives this router signature. A simple strategy to do this, adopted in this work, is to use a trace field containing R bits, where R is the number of routers in the NoC. The routers are numbered from 0 to $(R-1)$. To each router, r corresponds the bit in the trace field whose position inside the field is also r. When a packet enters the NoC, all the bits of its trace field are equal to 0. Each time the packet crosses a router, this router changes the corresponding bit in the trace field to 1. Then, at the end of the route, the packet terminator reveals the complete path that has been taken by the packet, indicating with 1 which routers have been crossed by the packet and with 0 which have not. By knowing the NoC topology and routing algorithm, the slave can deduce what is the true packet sender and thus it can verify if the alleged source is in fact this sender. For this purpose, a *security table* in each destination contains the expected value for the trace field coming from each possible master. In the case of a mismatch (i.e., the trace field does not correspond to the expected one) the packet is discarded.

CV: each master-slave pair may also keep track of the sequential number of its transactions. In this case, this sequential number is also included in the header. The slave then verifies if the transactions occur according to the expected numbering. To overcome this feature, an intruder would have to know the current expected sequential number of the transaction of the master-slave pair it intends to attack. A packet whose sequential number differs from the expected one is discarded.

5. Our Approach

The goal of our work is to design a layered dynamic NoC-based architecture for the MPSoC protection. In such a scenario, all the applications that are going to be mapped on the MPSoC are known in advance. Our security architecture is based on a hierarchical NoC (HNoC), combining a *low* and *high* NoC, and on the configuration of the security mechanism embodied at the network interface of the NoCs. Such configuration is carried out through the establishment of a set of parameters (upgrade of the *security table*), avoiding the high-cost of dynamic reconfiguration. There is an IP that stores all the security policies that the MPSoC must obey. The HNoC is notated as $n(S_1)/(S_2)$, where n is the number of security zones, S_1 is the *low* NoC size, and S_2 is the *high* NoC size. Figure 1 shows an HNoC, whose size is $4(2\times 2)/(2\times 2)$. The security zones are identified as *Zone I to Zone IV*.

The IP components are mapped on the HNoC according to their communication and security characteristics by using the MAIAS algorithm [14]. The purpose is to allocate in the same security zone the IPs with higher communication frequency and similar security requirements. As a consequence,

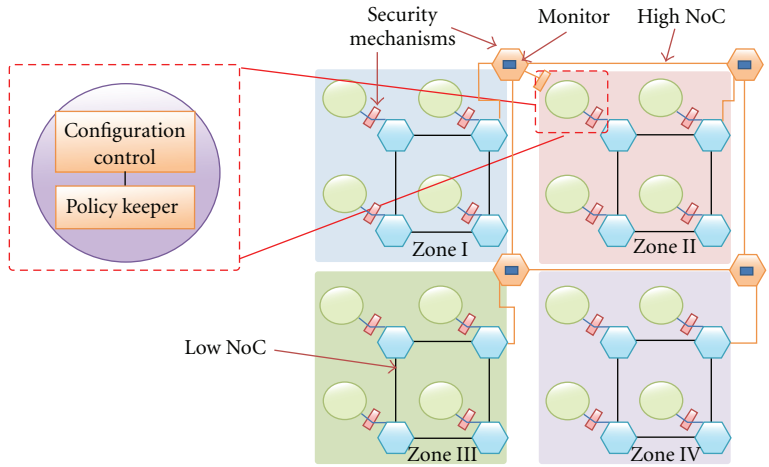


Figure 1: Layered dynamical architecture.

the inter-cluster communication is reduced. The HNoC is a scalable solution [15]. It integrates low-diameter topologies, that aided by efficient flow control and routing mechanisms, minimizes the overall power consumption and NoC latency [16].

Our architecture integrates five key components: (1) *low NoC/high NoC,* (2) *policy keeper*, (3) *configuration control,* (4) *security mechanisms,* and (5) *monitors*. Our work supposes that the task allocation of the applications has been previously defined.

5.1. Low NoC and High NoC. The HNoC is a layered communication structure composed by a set of networks. It can be divided into *low* and *high NoC* [14]. The IP cores of the MPSoC can be organized into clusters by the *low NoC*. Each cluster is completely separated from the others. The *low* 2D-mesh NoC has its own connection to a router of the *high NoC*. The *high NoC* collects the traffic coming from different clusters. Therefore, the *high NoC* must provide a larger bandwidth than the links in the *low NoC*. Higher frequency or additional links can be used at the high NoC. We use additional links. The *high NoC* links are physically longer than links at the *low NoC,* but they can be implemented on the higher metal layers with reduced RC-delay. This hierarchical structure allows the implementation of security zones, composed of the IPs at each cluster. The components at each security zone have similar security characteristics. The security policy, that rules the interaction among all the system components, is divided into local and global policies. The IPs of a security zone are ruled by the *local security policy*. The intercluster communication is ruled by the *global security policy*. Such hierarchical approach provides three advantages: (1) it facilitates the security management of the MPSoC; (2) it produces smaller security tables; (3) it improves system performance. The security policy of the MPSoC can change over time. Such behaviour is the consequence of two factors: (1) MPSoCs are used as a platform to support different applications, each one characterized by a different set of security requirements; (2) MPSoC status could influence the security policy, that is, under certain conditions, the security level of the system may be dynamically reinforced or decreased. Therefore, the security configuration of the MPSoC must be modified in order to satisfy the new requirements.

5.2. Policy Keeper. As mentioned earlier, secure mobile computer security systems need policies that are flexible and expressive. But traditionally, security systems are designed to enforce one particular security policy. To encompass a wide variety of policies, the *policy keeper* component stores a thread-oriented policy representation which allows the security specification tailored to the threads set of the applications being executed at each security zone of the MPSoC. The *policy keeper* is a safe component that stores the security policies (*local* and *global*). It integrates the information of the MPSoC thread scheduler, the security zone, and the access rules (rights) of each thread being executed on the MPSoC over the system resources. The local security policy configures the *low NoC* security mechanisms of each *security zone* and the global security policy configures the security mechanisms embodied at the high NoC. The security policy can express different levels of security. Data can be loaded on the *policy keeper* at two moments: (1) power up time, when all the applications that will be executed on the MPSoC are previously known; (2) run time, otherwise. At run time, the loaded security policy must come from a trusted third-party authority. For some applications, like military applications, the storage of the security requirements in the system is not desirable. The size of the table stored by the policy keeper component depends on the number of applications, tasks, threads, and operation modes supported by the MPSoC.

5.3. Configuration Control. The *configuration control* component focuses on establishing the conditions to guarantee the security requirements of each application and for all the operation modes throughout the operation of the MPSoC. Its main function is the coordination and configuration of the security mechanisms of the MPSoC according to the local

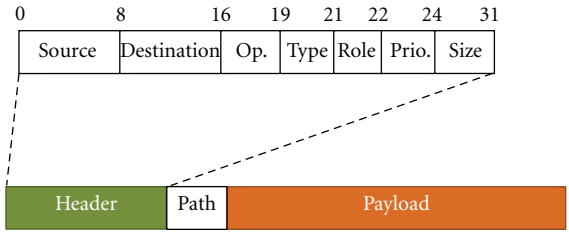


Figure 2: OCP compliant NoC packet format.

and global security policies of each application. In this work, the security mechanisms are composed of firewalls embodied in the NoC (*low* and *high*), the communication structure of the MPSoC. It uses the NoC interface ID source/destination information embodied in the *policy keeper* component to block the communication and start the reconfiguration process of the security mechanisms.

5.4. Security Mechanisms. The main function of the *security mechanism* components is to defend the MPSoC against possible attacks. The NoC security implementation allows MPSoC protection by means of communication management. NoC firewalls are implemented at the network interface (*secure network interface*) at the *low NoC* and at the *high NoC*. The mechanisms implemented in this work are explained in Section 3. Security mechanisms at the *low NoC* are implemented at the network interface. The network interface is the point of interconnection between NoC routers and processing components of the MPSoC.

Security mechanisms at the *high NoC* are implemented at NoC routers. The security mechanism uses information embodied in the packets that flow through the NoC to enforce the different security policies. Our work assumes a network interface compliant with the specifications of the OCP/IP (open core protocol) interface. Messages coming from an MPSoC processing component are translated by the interface into packets compliant to the protocol used within the NoC. The adopted OCP compliant NoC packet format (see Figure 2) is composed of 9 fields.

(1) Source: it identifies the task, the thread, and the master component. It is the initiator of the communication.

(2) Destination: it identifies the slave component. It is the target of the communication.

(3) Operation: it codes the transaction type, that is, a read, read linked, read-exclusive, write, write-non-post, conditional write, and broadcast.

(4) Type: it defines the information type that is being exchanged, that is, data, instruction, or signal types.

(5) Role: it represents the role of the initiator component. That is, user, root. The roles are defined by the security policy of the system.

(6) Priority: it allows traffic priority classification.

(7) Size: it defines the number of bytes contained in the packet payload.

(8) Path: it registers the path of the packet through the NoC and the sequential number of this packet in the current transaction between this master-slave pair.

(9) Payload: it embodies the information generated by the master.

Our firewall differs from those proposed by [8, 10] in the *source*, *type* and *role* fields. Such characteristics allow that our security mechanism avoid a wider set of attacks as the DoS (Denial of service), by using the *source* field, and buffer overflow by the verification of the *address*, *type* and *size* fields. Moreover, our approach supports the multithreading characteristic of the MPSoC. It allows the safe execution of multiple tasks of different applications, and thus multiple security policies implementations, at the same time. Note that our architecture is also feasible for different security mechanisms whose security characteristics are capable of being changed during system operation.

5.5. Monitors. The *monitor* component is implemented at the *high NoC* and *low NoC* routers and is permanently auditing the MPSoC communication behaviour. It detects NoC activity in order to determine the completion of each communication event between different master/slave pairs of the MPSoC. A master is defined as any component that initiates a communication transaction. A slave is any component that receives a request of a communication transaction of a master. The monitor also has the ability to record and report on the security mechanism configuration at any moment.

5.6. Execution Mode. The functionality of our layered security system can be summarized as in Figure 3. When the MPSoC security policy must be upgraded, because, for example, a new application must be executed or the MPSoC operation mode is changed, the *configuration control* starts six procedures: *lookup policy keeper*, *block, look-up monitors, global configuration (high NoC), local configuration (low NoC)*, and *unblock*. At the *lookup policy keeper* step, the configuration control downloads the proper *local* and *global* security policies, stored in the *policy keeper* component. It uses the MPSoC tasks mapping information to identify which *security mechanism* and which *security zones* must be modified as well as the new security tables information.

The *block* procedure interrupts the injection of packets whose final destination is the processing component linked to the security zone that is going to be reconfigured (called *target interface*). Such packets are stored on the interface linked to the master processing component. The reconfiguration manager also starts a QoS (quality-of-service) mechanism that raises the priority of the communication of the packets whose final destination is the component linked to the *target interface*; modifying the *Priority* packet field. The QoS mechanism *modifies the arbitration* of the NoC routers, so that, the communication of the packets with higher priorities is performed first.

Once the communication of all the packets flowing to the target interface is finished, the reconfiguration of the *target interface* can start. In order to configure the HNoC,

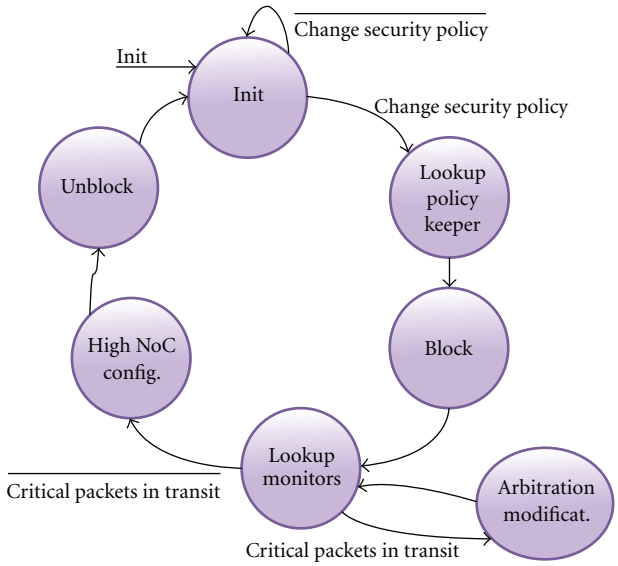


Figure 3: Functionality of the layered security system.

the *configuration control* component sends new information that must be stored at the *security tables* of the firewalls. The *high NoC configuration* modifies the security tables at the routers. The *low NoC configuration* modifies the security tables at the network interfaces of the selected security zone. Note that as the *security mechanisms* are implemented at the NoC interface, the communication is not interrupted at the NoC during the reconfiguration. The NoC routers continue the communication of the remaining packets through the network. This characteristic of our architecture can reduce the latency penalties due to the reconfiguration. When the reconfiguration is finished, the final *unblock procedure* starts. The *configuration control* frees the injection of the packets that were being blocked during the reconfiguration. The normal NoC operation is achieved when all the target interfaces are reconfigured.

6. Results

6.1. Experimental Setup. We have developed a SystemC-TLM cycle-accurate model of our architecture. Its evaluation was performed by the SystemC-TLM framework as described in [5]. We employ a $4(2 \times 2)/(2 \times 2)$ HNoC characterized by an XY routing scheme, round-robin (RR) arbiter, and FIFO memory organization. The proposed solution has been verified against three types of attack scenarios: (1) extraction: characterized by unauthorized attempts to access data; (2) modification: using the buffer overflow technique; and (3) DoS: repeating several times the same transaction. The performance evaluation of our approach was based on the MiBench benchmark suite [17]. We select three applications: auto/industrial (*A1*), consumer electronics (*A2*), and telecommunication (*A3*), see Table 3. Each application is characterized by a security policy that establishes different levels of authentication and access control security mechanisms, as shown in Table 4.

Table 3: Benchmarks.

Auto/Industrial (*A1*)	Consumer (*A2*)	Telecomm. (*A3*)
Basicmath	Jpeg	CRC32
Bitcount	lame	FFT
Qsort	mad	IFFT
susan (edges)	tiff2bw	ADPCM enc.
susan (corners)	tiff2rgba	ADPCM dec.
susan (smoothing)	tiffdither	GSM enc.
	tiffmedian	GSM dec.
	typeset	

Table 4: Security levels.

Application	Function	Authentic.	Access control
Automotive (*A1*)	Basicmath	Level 0	Level 2
	Bitcount	Level 0	Level 0
	Qsort	Level 3	Level 3
	susan (edges)	Level 2	Level 2
	susan (corners)	Level 2	Level 2
	susan (smoothing)	Level 2	Level 2
Consummer Electronics (*A2*)	Jpeg	Level 2	Level 2
	Lame	Level 2	Level 2
	Mad	Level 0	Level 0
	tiff2bw	Level 0	Level 0
	tiff2rgba	Level 0	Level 0
	tiffdither	Level 0	Level 0
	tiffmedian	Level 0	Level 0
	typeset	Level 0	Level 0
Telecomm. (*A3*)	CRC32	Level 2	Level 2
	FFT	Level 1	Level 1
	IFFT	Level 1	Level 1
	ADPCM enc.	Level 0	Level 0
	ADPCM dec.	Level 0	Level 0
	GSM enc.	Level 3	Level 3
	GSM dec.	Level 3	Level 3

Our experimental work considers six mapping combinations resulting from the execution of these three applications on the MPSoC (*A1*, *A2*, *A3*, *A1-A2*, *A1-A3*, *A3-A2*). These patterns were used as NoC benchmarks in previous works. Our MPSoC traffic is composed of a set of heterogeneous tasks arriving at different rates in the system. The tasks are randomly allocated in the system processing components. For comparison reasons, each traffic pattern is composed of five flit size packets of three types: real-time, write or read and signaling, characterized by a different generation rate. The intercluster communication varies from 20% to 50% of the overall traffic. We compared the communication performance of the HNoC-based dynamic security architecture against the simple NoC-based dynamic security and the best-effort NoC (without security).

Table 5: Security efficacy.

Attack scenario	Authentication efficacy	Acces Control efficacy
Send critical information	87%	100%
Read critical information	83%	100%
Write not authorized areas	100%	100%
Nonexisting target	100%	100%
Repeated information	89%	100%
Communication target = source	100%	100%

6.2. Security Efficacy. Table 5 shows the results of the efficacy of our security implementation. It represents the percentage of attacks detected by the security mechanisms. The percentages of the efficacy of authentication and access control security services were the same as our previous dynamic approach. That is because both approaches use the same security mechanisms. In some cases a few attacks were detected. This means that the security level should be increased in order to achieve 100% of security.

6.3. Security Performance. For comparison reasons, the performance of the security mechanisms is expressed in this work in terms of power consumption and latency. The power consumption (P_{DYN}), given by (1) is the sum of P_{NoC} (*low NoC* and *high NoC* power), P_{Keep} (*policy keeper* power), P_{Con} (*configuration control* power), and P_{Mon} (*monitor* power):

$$P_{DYN} = P_{NoC} + P_{Keep} + P_{Con} + P_{Mon}. \quad (1)$$

The *NoC* power is given by (2). It is the sum of P_{Li} (sum of the links power of the *low NoC* and the *high NoC*), P_{int} (interfaces power), and P_{Ri} (*low NoC* and *high NoC* routers power) due to transaction completion [14]. P_{Li} and P_{Ri} are proportional to the *channel utilization rate* and *router utilization rate,* respectively,

$$P_{NoC} = P_{Li} + P_{Int} + P_{Ri}. \quad (2)$$

We developed power models for the main components in the HNoC architecture. We integrated these models into the simulator, taking the architectural and technological parameters into account. The characterization was made under the 65 nm process constraints, 1 volt as a power supply, and a 25°C temperature. Our power estimation strategy is based on identifying the activity of each component of our dynamic system. In order to fulfill this task, the monitor annotates the communication events on the *NoC*. Figures 4 and 5 show the latency and power distributions over all the components of our hierarchical architecture. They show that the security interfaces and the policy keeper are the components that consume more time and power, respectively.

Figure 6 represents the average relative execution time of our architecture after completing 4, 8, 16, 32, and 64 tasks of the MiBench benchmarks. The number of processors that execute these tasks varies from 1 to 16. The comparison among the best effort NoC (without security), the simple

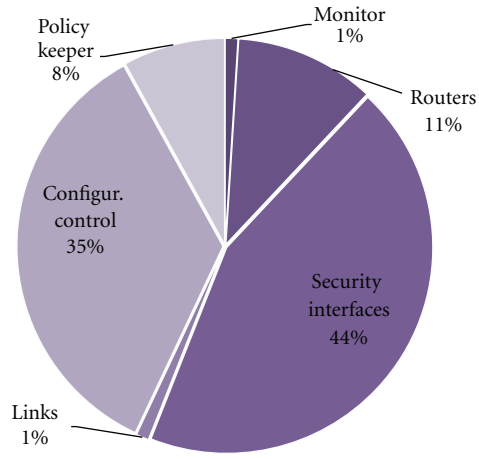


Figure 4: Latency distribution in our architecture.

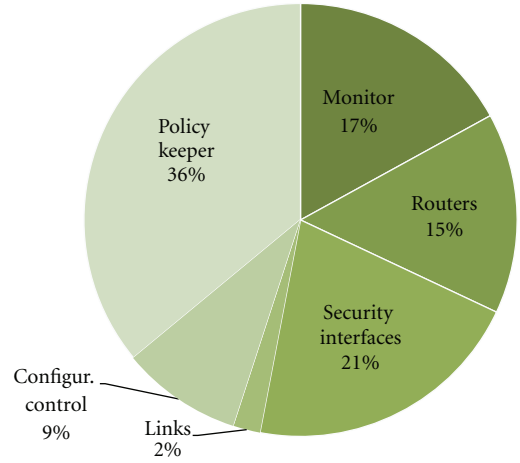


Figure 5: Power consumption distribution in our architecture.

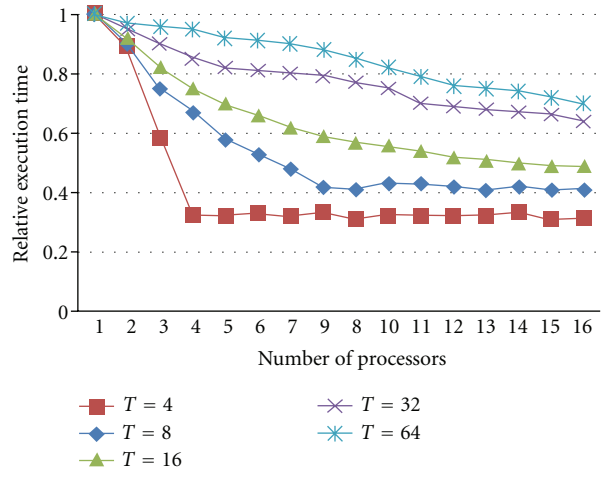


Figure 6: Relative execution time.

Table 6: Implementation penalties.

Parameter	Dynamical approach	Our HNoC approach
Latency increment	4.1%	3.8%
power overhead	19.6%	7.6%
area overhead	26.7%	5.2%

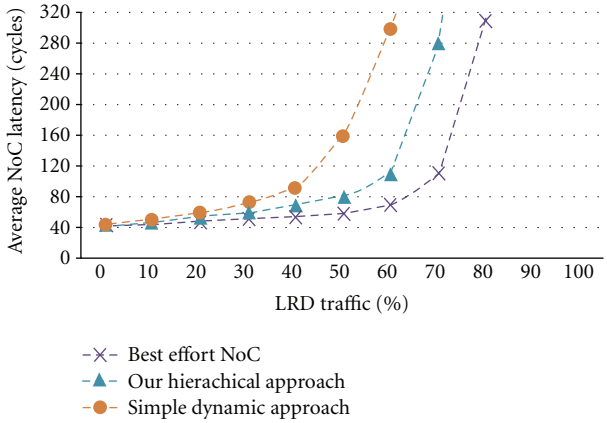


Figure 7: Average NoC latency.

dynamic security approach [5] and the layered security architecture, is shown in Figure 7 and Table 6. The results of Figure 7 are obtained after injecting a percentage of long-range-dependence (LRD) traffic in the MPSoC. Such traffic is typical for the three MiBench applications that we selected [17]. Our layered approach performs better than the simple dynamic security architecture for all percentages of LRD traffic. Table 6 shows the implementation penalties of the simple dynamic security architecture and our layered architecture when compared to the best effort NoC, stressed by uniform traffic. Our approach always achieves the best results.

7. Conclusion

In this work, we proposed a layered NoC security architecture able to support dynamic protection for MPSoC. We implement two security services: access control and authentication. We adopted the QoSS concept that allows the implementation of different security levels. Our work shows that NoC-centric security may take advantage of the distributed property of the NoC. Results show that the inclusion of security issues in the hierarchical NoC performs better that a simple NoC architecture. It reduces the penalties of the latency, power, and area up to 0.3%, 12%, and 21%, respectively, when compared to the simple dynamic solution. Inclusion of the QoSS concept allows the designer to select the most suited among different security levels in order to satisfy both security and performance requirements. In our current architecture, all the applications that are going to be mapped on the MPSoC are known in advance. As a future work, we plan to create virtual security zones, whose size will be defined by the security characteristics of the unkown applications.

References

[1] M. Loghi, F. Angiolini, D. Bertozzi, L. Benini, and R. Zafalon, "Analyzing on-chip communication in a MPSoC environment," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '04)*, pp. 752–757, February 2004.

[2] L. Benini, "Application specific NoC design," in *Proceedings of the Design, Automation and Test in Europe (DATE '06)*, vol. 1, pp. 1–5, March 2006.

[3] P. Kocher, R. Lee, G. McGraw, A. Raghunathan, and S. Ravi, "Security as a new dimension in embedded system design," in *Proceedings of the 41st Design Automation Conference (DAC '04)*, pp. 753–760, June 2004.

[4] McAfee, "Annual report 2011," http://www.mcafee.com/.

[5] J. Sepulveda, G. Gogniat, J. C. Wang, and M. Strum, "Dynamic NoC-Based architecture for MPSoC security implementation," in *Proceedings of the 24th Symposium on Integrated Circuits and Systems (SBCCI '11)*, pp. 197–202, 2011.

[6] C. Gebotys and Y. Zhang, "Security wrappers and power analysisforSoC technologies," in *Proceedings of the 1st IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES '03)*, pp. 162–167, 2003.

[7] U. Y. Ogras, J. Hu, and R. Marculescu, "Communication-centric SoC design for nanoscale domain," in *Proceedings of the IEEE 16th International Conference on Application-Specific Systems, Architectures, and Processors (ASAP '05)*, pp. 73–78, July 2005.

[8] L. Fiorin, C. Silvano, and M. Sami, "Security aspects in networks-on-chips: Overview and proposals for secure implementations," in *Proceedings of the 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD '07)*, pp. 539–542, August 2007.

[9] S. Evain and J. Diguet, "From NoC security analysis to design solutions," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '06)*, 2006.

[10] L. Fiorin, S. Lukovic, and G. Palermo, "Implementation of a reconfigurable data protection module for NoC-based MPSoCs," in *Proceedings od the 22nd IEEE International Parallel and Distributed Processing Symposium (IPDPS '08)*, April 2008.

[11] L. Fiorin, G. Palermo, S. Lukovic, V. Catalano, and C. Silvano, "Secure memory accesses on networks-on-chip," *IEEE Transactions on Computers*, vol. 57, no. 9, pp. 1216–1229, 2008.

[12] S. Lukovic and N. Christianos, "Enhancing network-on-chip components to support security of processing elements," in *Proceedings of the 5th Workshop on Embedded Systems Security, (WESS '10)*, October 2010.

[13] C. Irvine and T. Levin, "Security as a dimension of quality of service in active service environments," in *Proceedings of the 3rd Annual International Workshop on Active Middleware Services*, 2001.

[14] J. Sepulveda, G. Gogniat, R. Pires, C. Pedraza, W. Chau, and M. Strum, "Multi-objective artificial immune algorithm for security-constrained multi-application NoC mapping," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2012.

[15] B. Grot, J. Hestness, S. Keckler, and O. Multu, "Kilo-NoC: a heterogeneous network-on-chip architecture for scalability

and service guarantees," in *Proceedings of the 38th International Symposium on Computer Architecture*, 2012.

[16] M. Winter, S. Prusseit, and G. P. Fettweis, "Hierarchical routing architectures in clustered 2D-mesh networks-on-chip," in *Proceedings of the International SoC Design Conference (ISOCC '10)*, pp. 388–391, kor, November 2010.

[17] MiBench Version 1.0., http://www.eecs.umich.edu/mibench/.

Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2012, Article ID 473725, 10 pages
doi:10.1155/2012/473725

Research Article

A Memory Hierarchy Model Based on Data Reuse for Full-Search Motion Estimation on High-Definition Digital Videos

Alba Sandyra Bezerra Lopes,[1] Ivan Saraiva Silva,[2] and Luciano Volcan Agostini[3]

[1] *Federal Institute of Education, Science and Technology of Rio Grande do Norte, Campus João Câmara, 59550-000 João Câmara, RN, Brazil*
[2] *Department of Computer Science and Statistics, Campus Ministro Petronio Portela, Federal University of Piauí, 64049-550 Teresina, PI, Brazil*
[3] *Group of Architectures and Integrated Circuits-GACI, Federal University of Pelotas Pelotas, RS, Brazil*

Correspondence should be addressed to Alba Sandyra Bezerra Lopes, alba.lopes@ifrn.edu.br

Received 20 January 2012; Accepted 19 April 2012

Academic Editor: Alisson Brito

The motion estimation is the most complex module in a video encoder requiring a high processing throughput and high memory bandwidth, mainly when the focus is high-definition videos. The throughput problem can be solved increasing the parallelism in the internal operations. The external memory bandwidth may be reduced using a memory hierarchy. This work presents a memory hierarchy model for a full-search motion estimation core. The proposed memory hierarchy model is based on a data reuse scheme considering the full search algorithm features. The proposed memory hierarchy expressively reduces the external memory bandwidth required for the motion estimation process, and it provides a very high data throughput for the ME core. This throughput is necessary to achieve real time when processing high-definition videos. When considering the worst bandwidth scenario, this memory hierarchy is able to reduce the external memory bandwidth in 578 times. A case study for the proposed hierarchy, using 32×32 search window and 8×8 block size, was implemented and prototyped on a Virtex 4 FPGA. The results show that it is possible to reach 38 frames per second when processing full HD frames (1920×1080 pixels) using nearly 299 Mbytes per second of external memory bandwidth.

1. Introduction

Nowadays, several electronic devices support high-definition digital videos. Applications like internet and digital television broadcasting are also massively supporting this kind of media. In this scenario, the video coding becomes an essential area to make possible the storage and principally the transmission of these videos, mainly when the focus is in high definition.

The most recent and advanced video coding standard is the H.264/AVC (advanced video coding) [1]. This standard includes high complexity on its modules, aiming to achieve high compression rates. This high complexity makes difficult to achieve real time (e.g. 30 frames per second) though software implementations, especially when high definition videos, like 1920×1080 pixels, are considered.

A digital video is a sequence of still images, called frames, typically sampled at a rate of 30 frames per second. In a video sequence, there is a considerable amount of redundant elements, like background scenes or objects that do not have any motion from a frame to another, that are not really essential for the construction of new images. These elements are usually called redundant information [2]. There are three types of redundancy: spatial redundancy (similarity in homogeneous texture areas), temporal redundancy (similarity between sequential frames) and entropic redundancy (redundancy in the bit stream representation). Those redundancies can be removed in order to achieve high compression rates.

The motion estimation (ME) is the most computationally intensive module of a video encoder. This module explores the temporal redundancy to reduce the amount of data needed to represent the video sequence.

A feature of the H.264/AVC and other video coding standards is the use of asymmetric compression algorithms. In this case, the encoder and the decoder have different

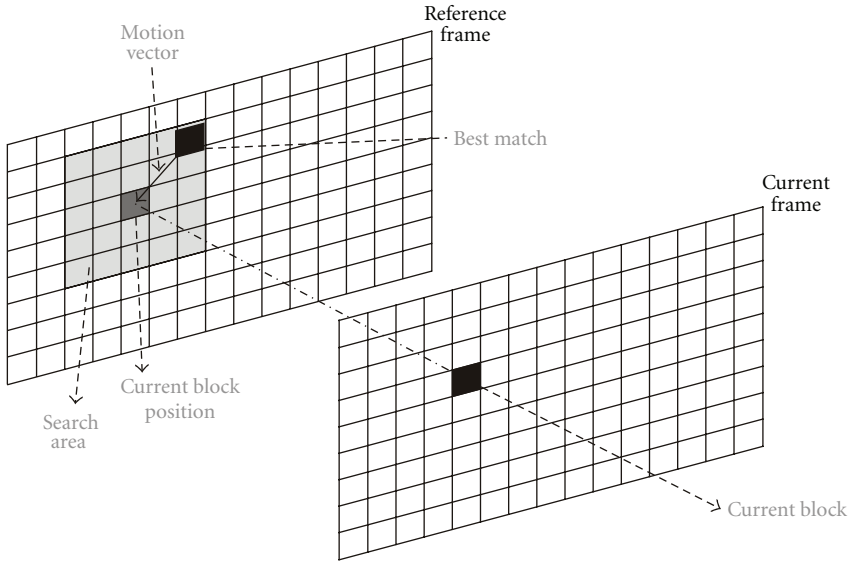


Figure 1: Motion estimation process.

definitions and the decoder, which is used in the higher number of devices, is less complex and cheaper than the encoder. The H.264/AVC standard introduces several new features when compared with others video compression standards. But the implementation of many of these new features is not mandatory [1]. Although the motion estimation is the most complex module in a digital video codec, it is presented only in the encoder side.

The motion estimation requires, besides the high processing throughput, also a very high bandwidth of external memory to realize its operations in real time when considering full HD videos (frames with 1920×1080 pixels) [3]. The throughput can be solved increasing the parallelism in the internal operations. The external memory bandwidth may be reduced using a memory hierarchy. Besides, the increase of the parallelism implies increasing the bandwidth. If more calculations must be performed at the same time, more data must be available to perform these operations.

The motion estimation presents the highest demand for external memory bandwidth, and it shares the external memory interconnection subsystem with others encoder modules like the motion compensation, the intraframe prediction and the deblocking filter. In this context, it is important to explore methods and architectural solutions which minimize the number of external memory accesses. An efficient memory hierarchy is the key point to respect the demands of a video encoder, mainly when high-definition videos are being processed.

This paper presents a memory hierarchy model for a full-search motion estimation core. The proposed memory hierarchy model is based on a data reuse scheme, and it aims to minimize the memory bandwidth and to maximize the data throughput delivered to the motion estimation core. The paper is structured as follows. Section 2 introduces the motion estimation process. Section 3 presents the data reuse exploration scheme. Section 4 presents the proposed memory hierarchy model. Section 5 presents some results and discussions. Section 6 presents the implemented core using the proposed memory hierarchy which was used as a case study. Section 7 presents comparisons with some related works. Finally, Section 8 presents conclusions and future works.

2. Motion Estimation

The ME process (illustrated on Figure 1) uses at least two frames: one current frame and one or more reference frames. The current frame is the frame that is being encoded. The reference frame is a previously encoded frame in the video sequence.

The current frame is divided into nonoverlapped blocks. For each block, a search window is defined in the reference frame. A search process is performed to find the better match for each block in its respective search window. To determinate the better match, a similarity criteria like Sum of absolute differences (SAD) [4] are used. A motion vector is generated for each block to represent where the better match was found. In the H.264/AVC, multiple reference frames can be used. In this case, the same search is done for all available reference frames and the candidate block that presents the better value of similarity among all reference frames is selected.

There are several search algorithms that define how the search process is done inside a search window. The full search (FS) is an algorithm that generates optimal results in accuracy, with a high cost in complexity [5]. Fast algorithms, like diamond search (DS), provide a great complexity reduction with acceptable lossless in accuracy [5].

Besides to provide the best possible motion vector, another important feature in FS algorithm is that it has a regular pattern with no data dependencies.

The FS algorithm looks for the best match in all possible candidate blocks in the search window, from the superior left

border until the inferior right border, shifting pixel by pixel in the search window. For each candidate block, a similarity calculation must be done to measure how similar is the candidate block to the current block. Once the similarity value is computed for all candidate blocks, the best match is the most similar candidate block (for instance, the lowest sum of absolute differences if SAD is used as similarity value). The FS process increases its complexity proportionally to the increase of search window range [6]. For a 32×32 search window and a 16×16 block size, there are 289 candidate blocks. A similarity calculation should be performed for each one of these blocks candidates. If a 64×64 is used, the number of candidate blocks increases to 1089.

Despite the complexity, in the FS motion estimation (FSME), there is no data dependency during the process of similarity calculation. So it is possible to design hardware architectures that provide the necessary resources in order to achieve high performance rates. It is possible to use systolic arrays [7] or tree structures to solve computational problems providing enough processing elements to compute the similarity between samples from the current block and the candidate block in parallel. However, the design of high performance motion estimation hardware is not an easy task. The parallelism exploitation increases the chip area proportionally to the parallelism level. The memory bandwidth also increases with the number of parallel similarity computation. Some criteria must be observed in order to achieve a good tradeoff between some parameters like. the degree of parallelism, the chip area, the memory bandwidth, and the power consumption.

On other hand, software solutions based on current general-purpose processors are not able to encode1080 HD digital video in real time (at a rate of 30 frames per second) when all H.264/AVC coding features are used. Then, the use of dedicated hardware architectures is mandatory to provide the required performance.

3. Data Reuse in Full-Search Motion Estimation

On FSME, data reuse refers to the use of each pixel of the search window (one or more n-bits words) to the similarity calculation of all candidate blocks that share the pixel without additional fetches in memory. A sample can be shared by neighboring candidate blocks of a same search window or by neighboring search windows in a reference frame. The number of candidate blocks that share a single sample depends on the size of the block, the size of the search window, and the position of the sample in the search window. For instance, a sample on the superior left corner of a search window is not shared by other candidates blocks, a single candidate block of a single search window have has sample, while a sample on the center of a search window is shared by many blocks of many search windows. Typically a sample can be shared by as many candidate blocks as the number of samples in a candidate blocks.

The regular pattern of FS algorithm allows the exploration of data reuse along with the exploration of parallelism in high degrees. The data reuse approach provides a way to reduce the bandwidth at the same time that the complexity of FS similarity calculation is reduced to the calculation of the similarity of a pair of block. In this context, data reuse is essential to reduce the bandwidth and the high parallelism is essential to achieve high performance.

As higher is the required video quality and definition, as higher is the required external memory bandwidth. The bandwidth increases with the frame size increase, with the search window increase, and with the frame rate increase. The bandwidth problem can be reduced using an efficient memory hierarchy.

As mentioned before, motion estimation uses at least two frames: one current frame and one reference frame.

The current frame is divided into several nonoverlapped $N \times N$ blocks. In the FSME, the search for the best match of one current block requires that its samples are repeatedly compared with samples from the reference frame. The current block samples lifetime is the time period of motion-estimating one current block [3]. Keeping the $N \times N$ samples from the current block in a local memory during the search reduces the data access from the current frame to the maximum possible, minimizing currant frame bandwidth. Doing this, each sample of the current frame will be fetched from the main memory just once in the whole motion estimation process.

Although adjacent current blocks do not overlap, adjacent search windows are no longer independent and overlap. Each search window in the reference frame is a rectangle ($SW_W * SW_H$) centered on the current block. This generally means that the search window of the current block shares data with the search window of the neighboring block.

In [3], four data reuse levels in the reference frame were defined according to the degree of data reuse: from level A to level D, where level A is the weakest reuse degree and the level D is the strongest reuse degree. Level A and Level B cover data reuse within a single search window. Level C and Level D cover data reuse among different search windows.

The data reuse level is an important factor in dealing with memory bandwidth requirements of ME. As stronger is the reuse level, less memory bandwidth is required.

In the search window, a row of candidate blocks is called a candidate block strip. Adjacent candidate blocks in a candidate block strip only differ by a column of samples. The Level A of data reuse take advantage of this feature to reuse all data loaded from external memory to the calculation of the previously candidate block and load just the column of samples which is necessary for the next candidate block calculation.

Figure 2 shows the level A data reuse scheme, where N represents the block dimension, SW_H and SW_W represents the height and the width of the search window, respectively. At this figure, two adjacent candidate blocks are showed (candidate block 1 and candidate block 2). The gray area is all samples within the search window that can be reused from a candidate block to another.

The level B of data reuse considers that adjacent vertical strips of candidate blocks within the search window also overlap significantly, only differing by a row of samples. All

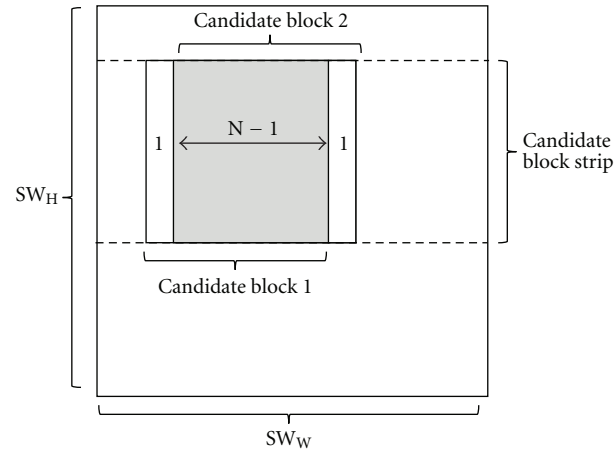


Figure 2: Level A data reuse scheme.

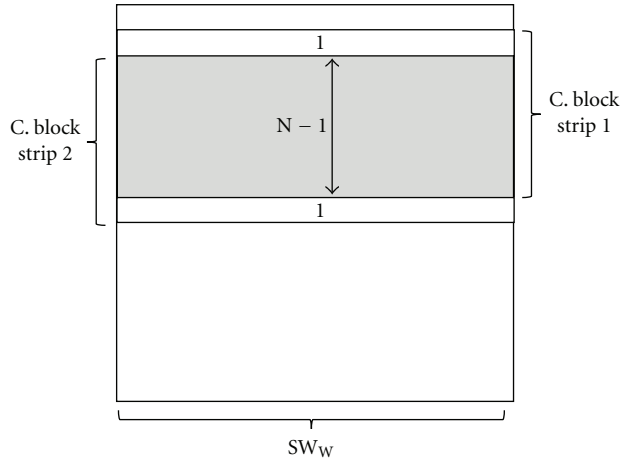


Figure 3: Level B data reuse scheme.

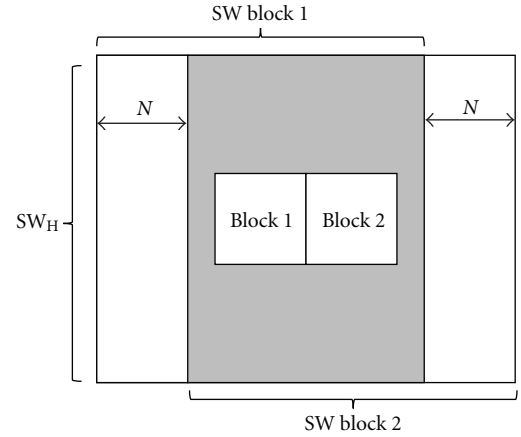


Figure 4: Level C on data reuse scheme.

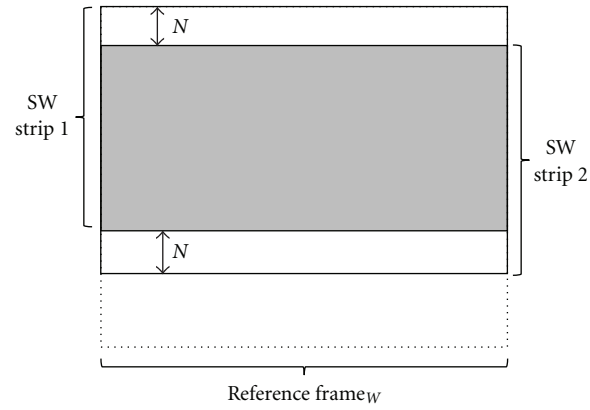


Figure 5: Level D data reuse scheme.

sample loaded for the previously strip of candidate blocks, with one row exception, can be reused.

Figure 3 represent two vertically adjacent candidate block strips (C. block Strip 1 and C. block strip 2). All the gray area represents data that can be reused while processing the next candidate block strip.

The level C refers to data reuse among different search windows. Search windows from neighboring blocks have several samples in common. in Figure 4, two blocks (block 1 and block 2) and their respective search windows (SW block 1 and SW block 2) are represented. The data of the two search windows differ by N columns of samples (where N represents the block dimension). All gray area is data that can be reused in the next block processing.

Finally, level D showed in Figure 5 resembles to level B of reuse data, except that it applies to reuses of samples in the entire search window strip instead of candidate blocks strip. With this level application, each sample of the reference frame is loaded just once during the entire motion estimation process.

The level A scheme uses the smallest size of on-chip memory, but it consumes more off-chip memory bandwidth. On the other hand, the level D uses more on-chip memory but it archives minimal off-chip memory bandwidth.

This work adopts Level C on data reuse scheme to balance on-chip memory size and off-chip memory bandwidth.

4. Proposed Memory Hierarchy

Generally, a memory hierarchy is composed of registers, local memories, and external memories. External memories are cheaper and have high storage capacity. But they are also slower. Local memories are faster than external memories, and their cost is also higher. Registers are the fastest option but with the highest cost among all solutions.

Considering these features, the proposition of a memory hierarchy for the motion estimation process must provide: (i) memory capacity to store at least the data that can be reused, considering the adopted reuse level (in this paper the Level C was chosen); (ii) a combination of high-level and low level memories in the hierarchy aiming to balance memory hierarchy cost, bandwidth, and throughput.

Data that can be reused in the ME in general was previously loaded from external memory. So it must be appropriately distributed between memory modules in the memory hierarchy to meet the goals described above.

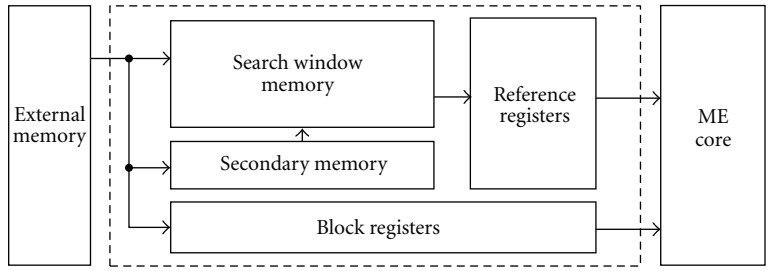


Figure 6: Proposed memory hierarchy model.

According to level C, data reuse scheme presented on [8] and illustrated in Figure 6 was proposed. This model aims to reduce the required memory bandwidth, while it balances cost and throughput. This way, the memory hierarchy will be able to provide the necessary data to the ME core that achieves the required high processing rates.

The external memory stores frames (current and reference frames) that will be used in the motion estimation process. The memory hierarchy levels store data that are recurrently used, avoiding redundant accesses to the external memory. These levels are composed by a search window memory, a secondary memory, and registers (block and reference registers). Each one will be described below.

In the proposed hierarchy, the block registers bank (BRB) contains $N \times N$ registers, each one storing a sample of the current block. The data from current block is used in the similarity calculation of all candidate blocks in the search window. It is important to use registers to store this block because it will be accessed continuously during all motion estimation process so the access to these data must be fast. The data at this bank is loaded just once per each block in the current frame and all values are used at each cycle during all ME process of the current block. Using this block register bank reduces to the minimum possible the external memory access of the current frame.

The reference register bank (RRB) has $N * (\mathrm{SW_W} - N)$ registers, where $\mathrm{SW_W}$ is the search window width. It is loaded with data from search window memory. These data refers to candidate blocks in the search window which are being used in the similarity calculation in relation to the current block. This bank in the way it was designed is sufficient to provide a high throughput to the ME core process. A fast access to these data is required to allow a high throughput in the ME core; thus, registers are also needed to store them.

At the reference register bank, three types of data shifts are allowed: shift right, shift left, and shift up. It was planned to minimize local memory access when the procedure of full search scan is performed as proposed in Figure 7.

In the proposed scheme, the data in RRB are shifted N times left until reaching the search window right border. One shift down is made when borders are reached and N shifts right are done to reach the left border of the search window. In each shift, only one column or row of new data must be stored in the RRB. Each shift is performed in one clock cycle. Every time a search window border (left or right) is reached, a complete data of candidate blocks strip has been provided to the ME core.

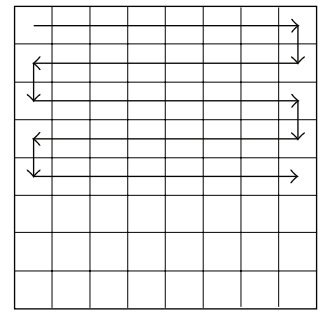


Figure 7: Proposed full-search scanning order.

The search window memory stores the complete search window of the current block. Sets of data from this memory are accessed at each moment from the motion estimation process. Since these data do not need to be simultaneously accessed, it can be stored in a local memory (on-chip memory). So the search window memory has a storage limit of $\mathrm{SR_W} * \mathrm{SR_H}$ samples. This is the memory that provides data to the RRB. After the initial RRB fill, at each clock cycle, the search window memory provides to the RRB on row or column of data.

Together, the RRB and the search window memory cover data reuse within a single search window, which represents level A and level B on the data reuse scheme.

The secondary memory stores the difference of search window data from two adjacent current blocks. So the size of this memory is $N * \mathrm{SR_H}$. These data are loaded from external memory, while the full search scan is being done. Once immediate accesses to them are not required, they can be stored in the local memory. When it is time to process a new current block, no extra clock cycle is expended to request data from external memory. The search window memory keeps the data that can be reused from the search window of the previously current block, and the data at the secondary memory overwrites the data that are no longer needed in the search window memory. So the complete search window from the next current block is available in the memory hierarchy modules.

The search window memory and secondary memory cover the level C on the data reuse scheme that refers to data that can be reused among different search windows.

A complete search window of a current block just need to be loaded from external memory when the current block

is the first one in the strip of current blocks in the current frame.

5. Experimental Results

Table 1 presents the experimental results using the proposed memory hierarchy for three sizes of search window (32×32, 64×64, and 96×96) and two block sizes (8×8 and 16×16).

The first and second columns on the table present the search windows and block sizes, respectively, used in this evaluation. The third column presents the number of candidate blocks existent in the search window. The number of candidate blocks in a $k \times k$ search window can be calculated by the formula $(k - n + 1)^2$, where n represents the block dimension.

The fourth column presents the bandwidth required (in megabytes) to bring data from external memory to process 30 frames in 1080 HD using our memory hierarchy model. The fifth column shows the bandwidth per second (in gigabytes) provided, using the proposed model, to the ME Core. The presented numbers consider the way of our memory hierarchy reuse data, keeping data previously loaded into local memories and registers.

The sixth column presents the number of cycles needed to fill a complete local memory with data from a whole search window. The next one shows the size (in Kbytes) of on-chip memory needed for each pair "search window/block size." This number is the sum of search window memory size and the secondary memory size. The eighth column presents the size (in Kbytes) of used registers (block registers and reference registers).

Finally, the last column shows the PSNR (Peak Signal Noise Rate) reached using the specific parameters of search window and block size. The PSRN results were obtained by software evaluations, using the H.264/AVC reference software [9], testing 10 video sequences (blue sky, man in Car, pedestrian area, riverbed, rolling tomatoes, rush hour, station, sunflower, traffic, and tractor). The PSNR values presented are the average of the values of all 10 video sequences because the values of PSNR may vary according to the video characteristics, like the amount of movement present in each video sequence.

An important relation that must be observed is the increase in the number of candidate blocks. A smaller block size, for a same search window, implies in a higher number of candidate blocks. Considering the search window variation, the number of candidate blocks increases proportionally with the search window increase. These aspects affect directly the computation necessary to the ME, and, consequently, the properly amount of data must be provided to ME core.

Using the proposed memory hierarchy, the external memory bandwidth to the ME is reduced, as shown in Figure 8. While the number of candidate blocks increases exponentially, the bandwidth maintains a linear growth. Comparing the best and the worst case presented on Table 1, while the number of candidate blocks increases approximately 27 times, the bandwidth required is only 4.4 times bigger. Without the proposed hierarchy, both curves presented on Figure 6 would have the same exponential behavior.

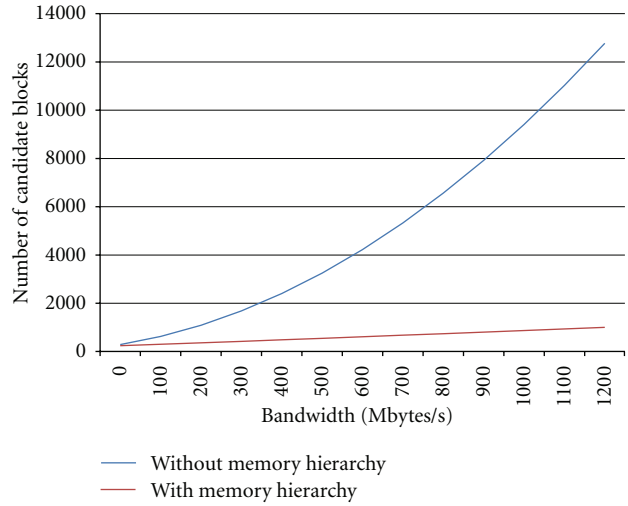


Figure 8: Candidate blocks versus bandwidth required.

According to a DDR SDRAM behavior, the number of clock cycles needed to fill the complete memory hierarchy with data from external memory was estimated.

A complete search window must be loaded from external memory only when the current block is the first one in a strip of current blocks. Considering a 1080 HD frame and a block size 8×8, there are 135 strips of current blocks. Using a 32×32 search window, 74 clock cycles are needed to fill the local memory. So only 9.990 clock cycles are expended with external memory accesses per frame. The fill of secondary memory can be done without introducing any extra clock cycles in the critical path, since this filling is done in parallel with the ME core process.

The on-chip memory presented in Table 1 is the sum of search window memory size and secondary memory size. The search window memory has a fixed size according to the size of search window. Once motion estimation uses only luminance samples (the motion vector to chrominance blocks are calculated based on the luminance ones), with a 32×32 search window and using 1 byte per luminance sample, the size of local memory is exactly 1 Kbyte. But the size of secondary memory depends on the block width and on the search window height. So using an 8×8 block size and the same 32×32 search window, the secondary memory uses 0.25 Kbytes of on-chip memory, totalizing 1.25 Kbytes. This way, the increase of search window memory is proportional to the square size of search window, when the secondary memory increase is proportional to both, the search window and the block size. When comparing the size of secondary memory with search window memory, the search window memory size increases 9 times between the best and worst case, and the secondary memory increases only 1.5 times.

The number of registers used is proportional to the block size and the search window. As bigger is the block size, as bigger is the block register size. Analogously, bigger search windows imply in bigger reference registers banks.

Table 1: Experimental results.

Search window	Block size	Candidate blocks	Bandwidth Ext. mem. (Mbytes/sec)	Bandwidth ME core (Gbytes/sec)	Clock cycles	On-chip memory (Kbytes)	Registers (Kbytes)	PSNR (dB)
32 × 32	16 × 16	289	179.0	15.8	74	1.50	0.5	34.00
	8 × 8	625	299.6	34.8		1.25	0.25	35.90
64 × 64	16 × 16	2,401	302.6	136.3	266	5.00	1	35.41
	8 × 8	3,249	547.8	184.9		4.50	0.5	37.64
96 × 96	16 × 16	6,561	430.1	375.4	586	10.50	1.5	35.90
	8 × 8	7,921	803.9	453.8		9.75	0.75	38.26

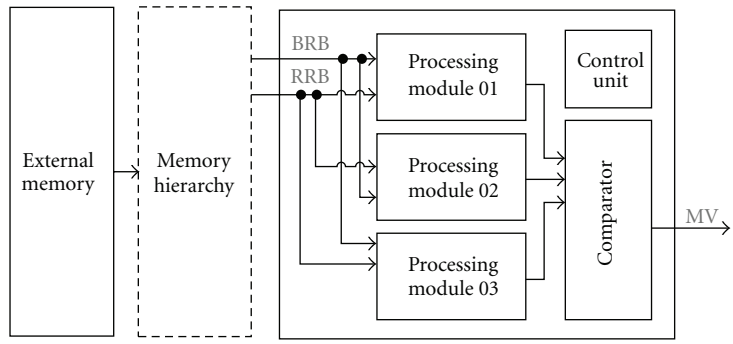


Figure 9: ME core block diagram.

So, according to Table 1, the configuration that spends more registers is that considering a 96 × 96 search window and a 16 × 16 block size.

One important factor that must also be considered when the parameters presented Table 1 in are evaluated is the quality of generated matches. The PSNR (peak signal noise rate) is one of the most used parameters to compare the video quality [6]. It is a quantitative measure that evaluates how much noise exists in the encoded video when compared to the original one. As biggest the PSNR value is, smaller is the noise and better is the video quality.

The PSNR results presented on Table 1 are the average of PSNR results from ten 1080 HD video sequences. As larger is the size of the search window, as higher is the ME chance to find a best match and a best motion vector. On the other hand, as smaller is block size, as higher is the ME chance to find a best match. Analyzing the PSNR results, the variation between worst and best case is more than 4 dB.

Furthermore, as bigger is the search window size and as smaller is the block size, as bigger is the number of computation needed to process the ME and also higher is the memory bandwidth required, as shown on Table 1.

Motion estimation architecture decisions always imply in a tradeoff between the numbers of needed calculations, the quality of ME results, the chip area, and the required external memory bandwidth. So it is important to balance all these criteria when designing a new architecture for motion estimation. But, in all cases, the use of the proposed memory hierarchy will decrease a lot the external memory bandwidth, allowing the ME to reach high throughputs, even for high-definition videos.

6. Case Study

The proposed memory hierarchy model and a functional motion estimation core were implemented and coupled as shown, in Figure 9 to demonstrate the qualities of the proposed memory hierarchy. The complete architecture, including the memory hierarchy and the ME core, was described in VHDL and synthesized to a Xilinx Virtex 4 XC4VLX25 FPGA [10].

The local memories (search window memory and secondary memory) were mapped to FPGA BRAMs.

A module to control the accesses to memory hierarchy was designed. It was designed to make read and write requests at the right moment: write data from external memory into the search window memory, secondary memory or block register; read data from search window memory, and writes in the reference register bank; also read data from secondary memory and writes into the search window memory.

These ME core was implemented using a fixed 8 × 8 block size and the 32 × 32 search window. This parameters were defined after analyzing some of the criteria presented in Table 1, and it is realized that these parameters presents good tradeoff between numbers of needed calculations, the quality of ME results, the chip area, and the required external memory bandwidth.

The ME core uses SAD [4] as similarity criterion. This is the most used criteria for hardware implementations because it executes simple calculations and it offers a good quality answer. It performs the absolute difference between samples from current block and samples from candidate block. The ME core was implemented intending to achieve

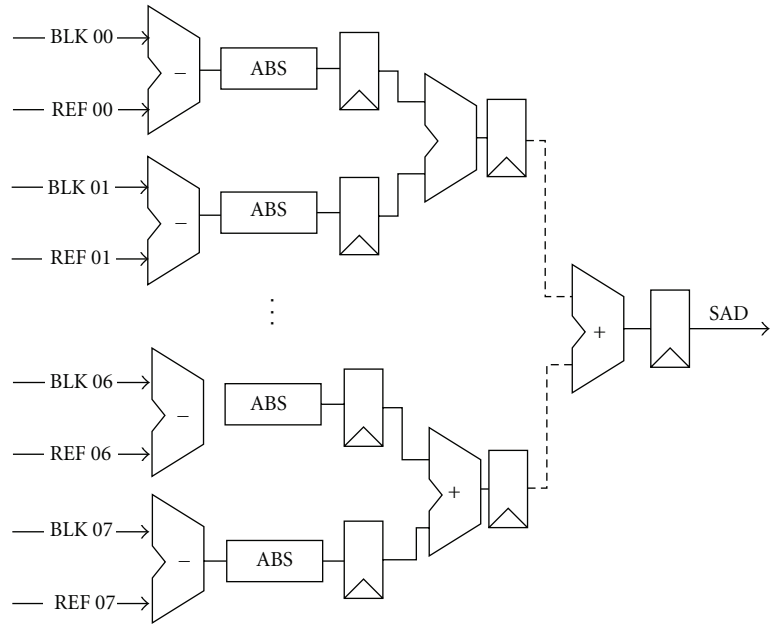


Figure 10: Processing unit RTL scheme.

high processing rates. So it takes all bandwidth provided by the memory hierarchy.

The ME architecture is composed by three processing modules (PMs), each one including units responsible for arithmetic operations; a comparator to decide which block represents the best match; a control unit that is responsible to manage the data flow with control signals.

Each PM is composed by eight processing units (PUs). The processing units (PUs) are responsible to give the results of similarity calculation between the current block and candidate block. Figure 10 presents the PU RTL scheme.

Once there are eight processing units per PM, each PU calculates the absolute difference between one row of the current block and one row of the candidate block. The data of the current block are obtained from the BRB, and the data from the candidate block are obtained from the RRB. In the PM there are pipelined adders that sum the results of the eight PUs and compute the complete SAD for a candidate block.

The comparator (Figure 11) receives the three SAD values and their respective motion vectors (MVs), each one generated for one PM. The three values are compared and the small one is stored in best motion vector register (best vector in Figure 11).

Once the PMs generate SAD values in pipeline, three SAD values are received for the comparator module at each clock cycle. After the comparator's pipeline is full, the comparison of these three values is delivered at each clock cycle. The best value of the three SADs is compared with the SAD stored at the best SAD register. At the end of the ME process, the best motion vector referent to the best SAD will be available at the best vector register.

The architecture control was developed in a decentralized way. Each module has its own manager, and the control unit sends signals for all the modules managers. The control unit has signals to initialize the complete architecture and output signals that signalize when the process of ME has finished.

Table 2: Synthesis results.

	Used	Available	Percent of usage
Slices	6,895	10,752	64%
Slice flip flops	6,469	21,504	30%
4 input LUTs	11,893	21,504	55%
BRAMs	1	72	1%

The synthesized results of the complete architecture to the Virtex 4 XC4VLX25 FPGA are showed on Table 2.

As shown in Table 2 the complete architecture spends 64% of available slices, 30% of flip-flop slices, and 55% of LUTs. Once this device is not a huge one, these values are acceptable for ME cores.

Focusing on BRAM results showed in Table 2, it is possible to realize that, from the total of BRAMs available on the target FPGA device, only 1% was used for the complete architecture. This means that the proposed memory hierarchy is very efficient to reduce the external bandwidth, to feed a high throughput ME core, and presents all these features using a very low amount of hardware resources. This result is also a stimulus to design a new hierarchy model considering the level D of data reuse, as proposed in [3].

Table 3 presents information about the minimum operation frequency required to achieve real time for different video resolutions, considering this implementations. The achieved frequency in this case study using a Virtex 4 was 292 MHz. The complete architecture is capable to process, in real time, since small resolution videos until high-definition videos. With this frequency, the architecture is able to process more than 86 720 HD frames per second and 38 1080 HD frames per second. Considering 1080 HD resolution at a

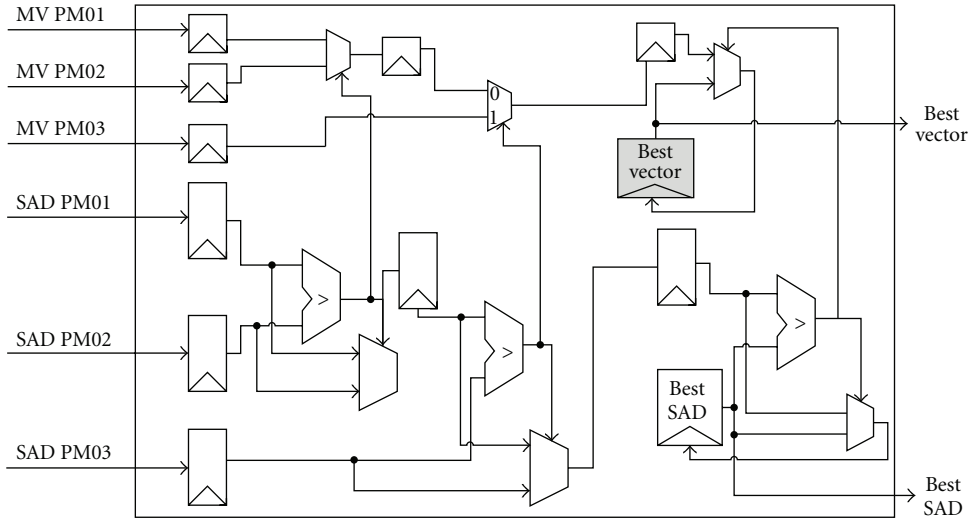


Figure 11: RTL scheme for the comparator module.

Table 3: Frequency and bandwidth required.

Resolution	Min. freq. (MHz)	Max. frames/sec.	Bandwidth (Mbytes/s)
CIF (352 × 288)	11.20	784.10	15.29
VGA (640 × 480)	33.83	259.59	45.26
SD (720 × 480)	38.04	230.84	50.75
720 HD (1280 × 720)	101.29	86.70	133.81
1080 HD (1920 × 1080)	227.75	38.56	299.59

frame rate of 30 fps, the required bandwidth is 299 Mbytes per second.

Using the proposed memory hierarchy, it is possible to develop a functional architecture for motion estimation with good quality results, high processing rates, and mainly with a reduced bandwidth.

7. Related Works

There are many works that focus on motion estimation architectures. But few of them focus on the memory bandwidth reduction. All works presented in this section uses full search as block matching algorithm. Table 4 presents a comparison with this related works.

The work presented in [11] uses variable block size, doing the SAD calculation for 4 × 4 blocks, and reusing these data to generate the SAD for the other block sizes. The architecture stores just the current block and the 4 lines of the search window. No local memory are used, only registers. This architecture attends to the level B data reuse scheme. Besides the use of small amount of registers and no local memory, it requires more external memory bandwidth than our solution.

In [12] is presented an architecture that allows variable block size and uses a 16 × 16 search window. It has a computation mechanism that processes the search window line per line, processing the first line of all candidate blocks in that line, and then, starting the next line, and so on, until the last line of the search window is reached. This architecture attends to the level B data reuse scheme. Although using a small size of search window, this architecture requires a high bandwidth because it does not reuse search window data from neighboring blocks.

Table 4: Comparison with related works.

	[11]	[12]	[13]	This work
Block size	4 × 4 to 16 × 16	4 × 4 to 16 × 16	4 × 4 to 16 × 16	8 × 8
Search window	19 × 19	16 × 16	65 × 65	32 × 32
Level of data reuse	B	B	C	C
Memory (Kbytes)	—	0.25	7.75	1.25
Register (Kbytes)	0.08	0.06	0.5	0.25
Bandwidth (MB/s)	1,397.8	1,008.5	1,054.0	299.5

The work presented in [13] attends to the level C data reuse scheme. It uses a 65 × 65 search window and variable block size. The size of the used search window is higher but the architecture attends to level C data reuse scheme, and then, this work also requires a very high bandwidth with external memory.

The architecture presented in this work using the proposed memory hierarchy model, presents a great tradeoff between search window size, block size and external memory bandwidth requirement.

Considering the external memory bandwidth required to process 30 frames per second, this work presents best result. Even using small search windows, the solutions presented in [11, 12] need a higher bandwidth to process 1080 HD videos in real time. This is because these architectures do not use a high level of data reuse. The work presented in [13] in spite of using level C of reuse of data uses a large search window, requiring a higher bandwidth than our work.

Develop hardware architecture for motion estimation implies always in a tradeoff between several parameters. So it is necessary to verify if the application priority is a better quality, a low cost in on-chip area, a reduction on external memory bandwidth or, an increase in the processing rate, for example. But, balancing all these criteria it is possible to develop efficient motion estimation architectures.

8. Conclusions and Future Works

This paper presented a memory hierarchy model for full-search motion estimation aiming to reduce the external memory bandwidth requirement. The proposed model, besides reducing memory bandwidth, it is also able to provide a high throughput to the ME core allowing the ME to process high-definition videos.

Considering the highest search window and the lowest block size evaluated, this architecture can reduce the external memory bandwidth in 578 times when compared to a solution without hierarchy.

The ME core designed and coupled to the memory hierarchy model proposed in this paper was described in detail. The complete architecture formed by the memory hierarchy and ME core can reach more than 38 frames per second when processing a 1080 HD video using only 299 Mb of external memory bandwidth.

As future works, we plan to evolve the memory hierarchy model to support motion estimation with multiple reference frames. It is also a future work to adopt the level D of data reuse, In this case, more on-chip memory will be necessary, but also the memory bandwidth requirement will be lower.

References

[1] J. V. Team, Draft ITU-T Rec. and Final Draft Int. Standard of Joint Video Spec. ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC, May 2003.

[2] A. Bovik, *Handbook of Image and Video Processing*, Academic Press, 2000.

[3] J. C. Tuan, T. S. Chang, and C. W. Jen, "On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 1, pp. 61–72, 2002.

[4] Y. Q. Shi and H. Sun, *Image and Video Compression for Multimedia Engineering*, CRC Press, 2nd edition, 2008.

[5] P. A. Kuhn, *Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation*, Kluwe Academic Plubisher, Boston, Mass, USA, 1999.

[6] I. Richardson, *H.264 and MPEG-4 Video Compression: Video Coding for Next-Generation Multimedia*, John Wiley & Sons, Chichester, UK, 2003.

[7] Y. H. Hu and S.-Y. Kung, *Handbook of Signal Processing Systems*, Springer, New York, NY, USA, 2010.

[8] A. S. B. Lopes, I. S. Silva, and L. V. Agostini, "An efficient memory hierarchy for full search motion estimation on high definition digital videos," in *Proceedings of the 24th Symposium on Integrated Circuits and Systems Design*, pp. 131–136, Joao Pessoa, Brazil, September 2011.

[9] JM15.1, "H.264/AVC JM Reference Software," 2011, http://iphome.hhi.de/suehring/tml/.

[10] Xilinx, "FPGA and CPLD Solutions from Xilinx, Inc," http://www.xilinx.com/.

[11] R. S. S. Dornelles, F. M. Sampaio, and L. V. Agostini, "Variable block size motion estimation architecture with a fast bottom-up Decision Mode and an integrated motion compensation targeting the H.264/AVC video coding standard," in *Proceedings of the 23rd Symposium on Integrated Circuits and Systems Design (SBCCI '10)*, pp. 186–191, September 2010.

[12] R. Porto, L. Agostini, and S. Bampi, "Hardware design of the H.264/AVC variable block size motion estimation for real-time 1080HD video encoding," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI '09)*, pp. 115–120, May 2009.

[13] L. Deng, W. Gao, M. Z. Hu, and Z. Z. Ji, "An efficient hardware implementation for motion estimation of AVC standard," *IEEE Transactions on Consumer Electronics*, vol. 51, no. 4, pp. 1360–1366, 2005.

Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2012, Article ID 148190, 16 pages
doi:10.1155/2012/148190

Research Article

An FPGA-Based Omnidirectional Vision Sensor for Motion Detection on Mobile Robots

Jones Y. Mori,[1] Janier Arias-Garcia,[1] Camilo Sánchez-Ferreira,[1] Daniel M. Muñoz,[2] Carlos H. Llanos,[1] and J. M. S. T. Motta[1]

[1] *Faculty of Technology, University of Brasilia, 70910-900, Brasilia, DF, Brazil*
[2] *Faculty of Gama, University of Brasilia, 72405-610, Brasilia, DF, Brazil*

Correspondence should be addressed to Jones Y. Mori, jonesyudi@unb.br

Received 20 February 2012; Accepted 4 April 2012

Academic Editor: Alisson Brito

This work presents the development of an integrated hardware/software sensor system for moving object detection and distance calculation, based on background subtraction algorithm. The sensor comprises a catadioptric system composed by a camera and a convex mirror that reflects the environment to the camera from all directions, obtaining a panoramic view. The sensor is used as an omnidirectional vision system, allowing for localization and navigation tasks of mobile robots. Several image processing operations such as filtering, segmentation and morphology have been included in the processing architecture. For achieving distance measurement, an algorithm to determine the center of mass of a detected object was implemented. The overall architecture has been mapped onto a commercial low-cost FPGA device, using a hardware/software co-design approach, which comprises a Nios II embedded microprocessor and specific image processing blocks, which have been implemented in hardware. The background subtraction algorithm was also used to calibrate the system, allowing for accurate results. Synthesis results show that the system can achieve a throughput of 26.6 processed frames per second and the performance analysis pointed out that the overall architecture achieves a speedup factor of 13.78 in comparison with a PC-based solution running on the real-time operating system xPC Target.

1. Introduction

Scientists predict that robots will play an important role in the future. In this scenario, robots will be able to assist humans in many tasks as domestic labors, elderly people care, cleaning, vehicles operation, and surveillance. Animals have mechanisms to interact with the environment provided by natural evolution. They are able to sense the surrounding environment and to move according to a defined objective, contouring obstacles and performing a dynamic path planning. In the robotic field, one of the major challenges is providing robots with sensorial and rational capabilities, allowing them to assist, and possibly substitute, humans in some activities requiring special skills.

Autonomous mobile robot navigation considers the execution of three stages: (a) mapping, (b) localization, and (c) decision making. The first stage uses information from sensors for creating a map of the environment. The second one relates the map with the sensor information, allowing the robot to self-localization in the environment. The third stage considers the path-planning problem [1].

Different kinds of sensors can be used for providing environment information to the mobile robot. Such sensors are classified in two main groups: (a) interoceptive and (b) exteroceptive. The interoceptive sensors perform internal robot parameters measurements without environment dependence. Encoders, gyroscopes, and accelerometers are some examples of interoceptive sensors. On the other hand, exteroceptive sensors perform external measurements, for instance, ultrasound, radar and infrared positioning systems as well as cameras, GPS and magnetometers. In humans, the vision sense is the one which provides more quantity of information about the environment. Through the sensorial fusion (provided by our stereo vision system) we are able to estimate efficiently the localization of surrounding objects.

The use of cameras jointly with image processing algorithms for implementing sensors (e.g., distance, movement, color, and presence sensors) is suitable solution for mobile robotic applications. Additionally, cameras with embedded image processing issues are the foundations of computer vision area. Catadioptric systems are realizations of omnidirectional vision, being mainly based on specially shaped mirrors (e.g., spherical, hyperbolic, parabolic, etc.) that reflect the environment to the camera from all directions, obtaining a panoramic view. Thus, these systems can provide information from a larger area than other vision sensors [2].

The task of processing the acquired images depends on the objective of the process itself. A common problem in mobile robotics is the localization of moving objects around the robot. For that, different methodologies can be used, such as motion detection, trajectory estimation, and tracking. Since the motion detection approach makes use of simple and easily implemented algorithms, this technique is suitable for real-time embedded applications. A common technique for implementing the motion detection is the background subtraction, in which an image is acquired at the beginning of the measurement process, and then each new image is subtracted pixel by pixel from the background. This technique is largely used in surveillance systems, since it acts as an automatic intrusion detection algorithm. In robotics, the localization of the differences between the background and the new frame provides the position estimation of the moving objects around the robot.

On the other hand, distance sensors are important for solving mobile robotic localization problems (namely, local and global localization tasks), and an important issue is the use of cameras for these tasks, providing (in real time) the robot with information about the distance to an obstacle. To accomplish this, the development of a mapping process among the actual scenario and the captured image is fundamental. This aspect introduces the calibration problem, which comprises the estimation of metrological values such as accuracy and precision (that are related to systematic and random errors, resp.), apart from the calculation of calibrations curves.

Otherwise, taking into account performance points, autonomous mobile robots must be able to acquire images from the environment, processing the information and making a decision in a short period of time. In order to avoid failures, autonomous mobile robots must perform the decision process as quickly as possible. This real-time constraints require the use of high-performance computational platforms for implementing image processing algorithms. In this context, the high computational cost of the involved algorithms is the main drawback, specifically when performing operations with high accuracy and high performance.

Common robotic platforms are based on desktop solutions executing complex algorithms for robot navigation. However, desktop platforms are not tailored for embedded applications with portability and low-power consumption requirements. Field programmable gate arrays (FPGAs) are a suitable solution for implementing image processing algorithms with a high performance. FPGAs allow the involved algorithms to be mapped directly in hardware in a parallel way. In addition, FPGAs allow software RISC processors to be implemented in order to execute parts of the algorithms with low performance requirements.

In [3] the authors proposed the development of a distance sensor based on an 800×480 pixels camera connected to an FPGA, a spatial convolution filter (for edge enhancement), a hardware architecture for estimating the distance of real objects and a touch-screen display as user interface (the camera image was addressed to the screen). In that system the screen was capable to detect the coordinates of a touched point, being used for calculating the distance (in pixels) from the robot to a defined object. In this approach the calibration parameters (errors and calibration curves) were calculated by comparing the actual distances, in a particular scenario, and the pixel distance in the screen.

The main contribution of this work is the design of an integrated hardware/software sensor system for both moving object detection and distance calculation, based on background subtraction algorithm. In this approach the calibration problem has been also treated, validating a proposed calibration process, which is suitable for this kind of application. Several image processing operations as filtering, segmentation, and erosion have been implemented in the architecture. The object's position was determined by computing its center of mass coordinates. The movement detection technique was also used to automate the omnidirectional vision system (achieved in a catadioptric implementation). As a result the uncertainties related to point objects in the touchscreen were eliminated.

The proposed pipeline image processing algorithm was mapped onto a Cyclone II FPGA device. Also, in this device, a NiosII soft processor was implemented for computing the distance and orientation as well as a simple user interface. Execution time comparisons among the proposed hardware architecture and a C-code implementation show that the hardware solution speeds up by 13.78 times a pc-based solution running in an Intel Pentium IV processor at 2.2 GHz, with 2.0 GB RAM and using a real-time operating system (the xPC Target OS from MathWorks).

The remainder of this paper is organized as follows. Section 2 outlines some computational vision techniques. Section 3 presents the related work. Section 4 describes the system and the calibration procedure. Section 5 shows the FPGA hardware and software implementations, and, before concluding, Section 6 presents synthesis, validation results, and a performance analysis.

2. Background

The use of cameras is a common solution for mobile robotics applications. Different works are related to the extraction of features from images. Monocular systems are able to provide only 2D information of the environment in front of the camera. In this case, in order to extract depth information it may be necessary to analyse vanishing points or to use

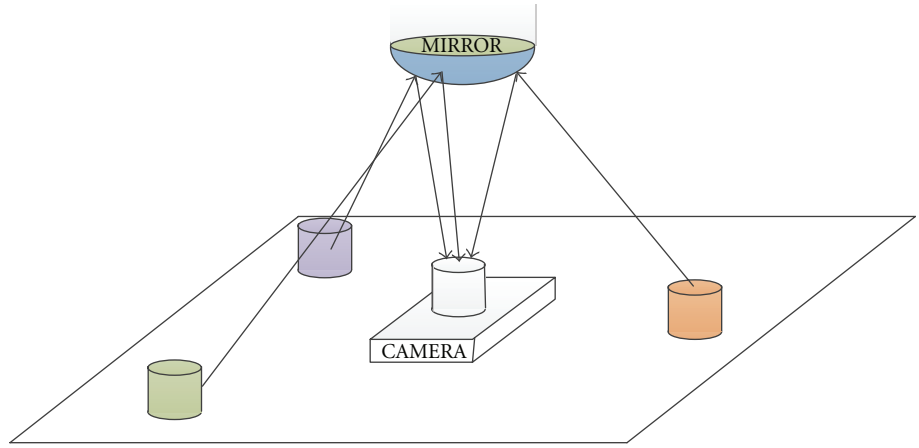
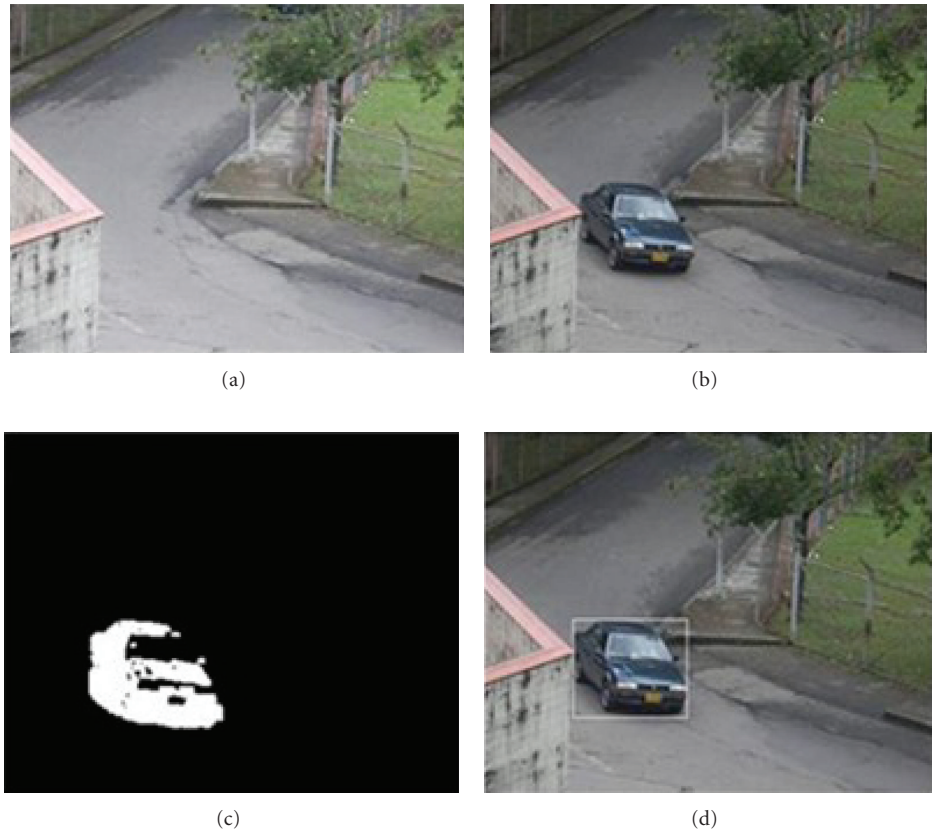


Figure 1: Catadioptric system.



(a) (b)

(c) (d)

Figure 2: Image chain of the background subtraction algorithm: (a) background image; (b) new image; (c) image subtraction; (d) overlap of new image and object's position.

perspective models with known object shapes. The use of a pair of cameras (stereo systems) allows the depth to be estimated through the epipolar geometry. Once the objects have been identified in each camera, it is possible to compute the depth of the image using simple geometric techniques. However, similar to monocular systems, stereo cameras provide information only in front of the cameras. In order to obtain information about the surrounding environment it is necessary to turn the system 360 degrees acquiring images in all the directions [4].

Omnidirectional vision systems are a suitable alternative to view the surrounding environment from one single image. Such a system can be built in two ways: (a) using panoramic lenses, for instance, fisheye lenses and (b) using catadioptric systems. Cameras equipped with fisheye lenses acquire images directly from the environment. On the other hand,

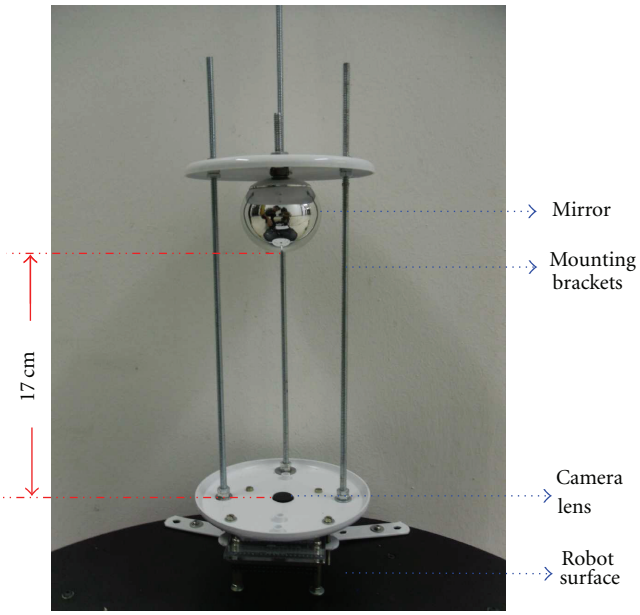


Figure 3: Main components of the proposed catadioptric system.

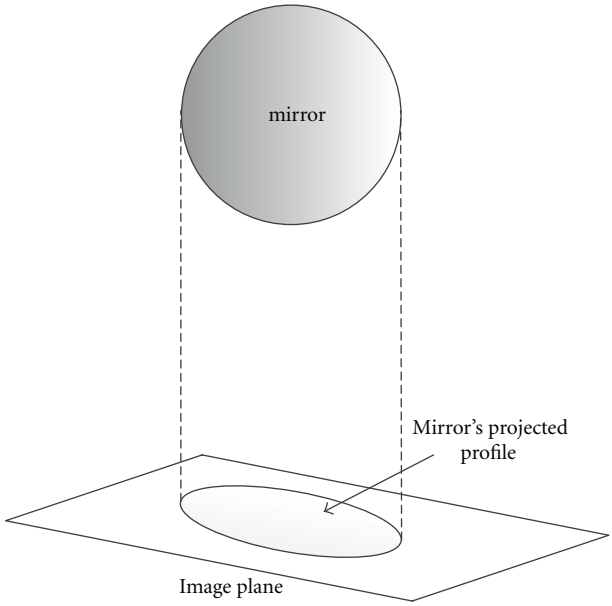


Figure 4: Mirror's profile projected over image plane.

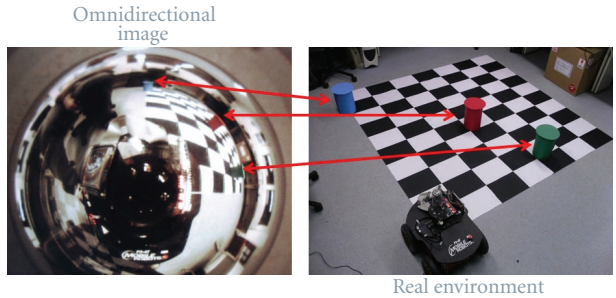


Figure 5: Correspondences among the omnidirectional image and the real environment.

Figure 6: The catadioptric system mounted over a calibration board.

catadioptric systems capture the image of the environment reflected on a special geometry mirror [5].

Figure 1 shows the principles of operation of a catadioptric system. It is composed of a camera and a convex mirror. The acquired image characteristics depend on the geometry of the mirror. Thus, knowing the geometry, reflection equations can be used in order to determine the environment geometric characteristics. Commonly, omnidirectional vision systems make use of hyperbolic, parabolic, spherical, and conical mirrors [5, 6].

In our previous work [3] the object's coordinates in the image were determined using a touch screen. Thus, by touching over the object in the image, the distance in pixels was calculated. Although this procedure is easily performed, its precision depends on several factors such as parallax effect, accuracy, and lighting variations. In order to minimize the human factor, in this work we make use of an identification method based on a background subtraction algorithm which allows the object to be automatically located and identified.

Figure 2 shows an image chain demonstrating the response of the background subtraction algorithm. In this method an image without objects is acquired (background image), and then each new image (with objects) is subtracted pixel by pixel from the background. After the subtraction (Figure 2(c)), the coordinates of the center of mass are determined using (1) and used as the object location. It is important to note that the system can determine the position of only a single object in the image, since all other moving objects will be considered noise and disregarded.

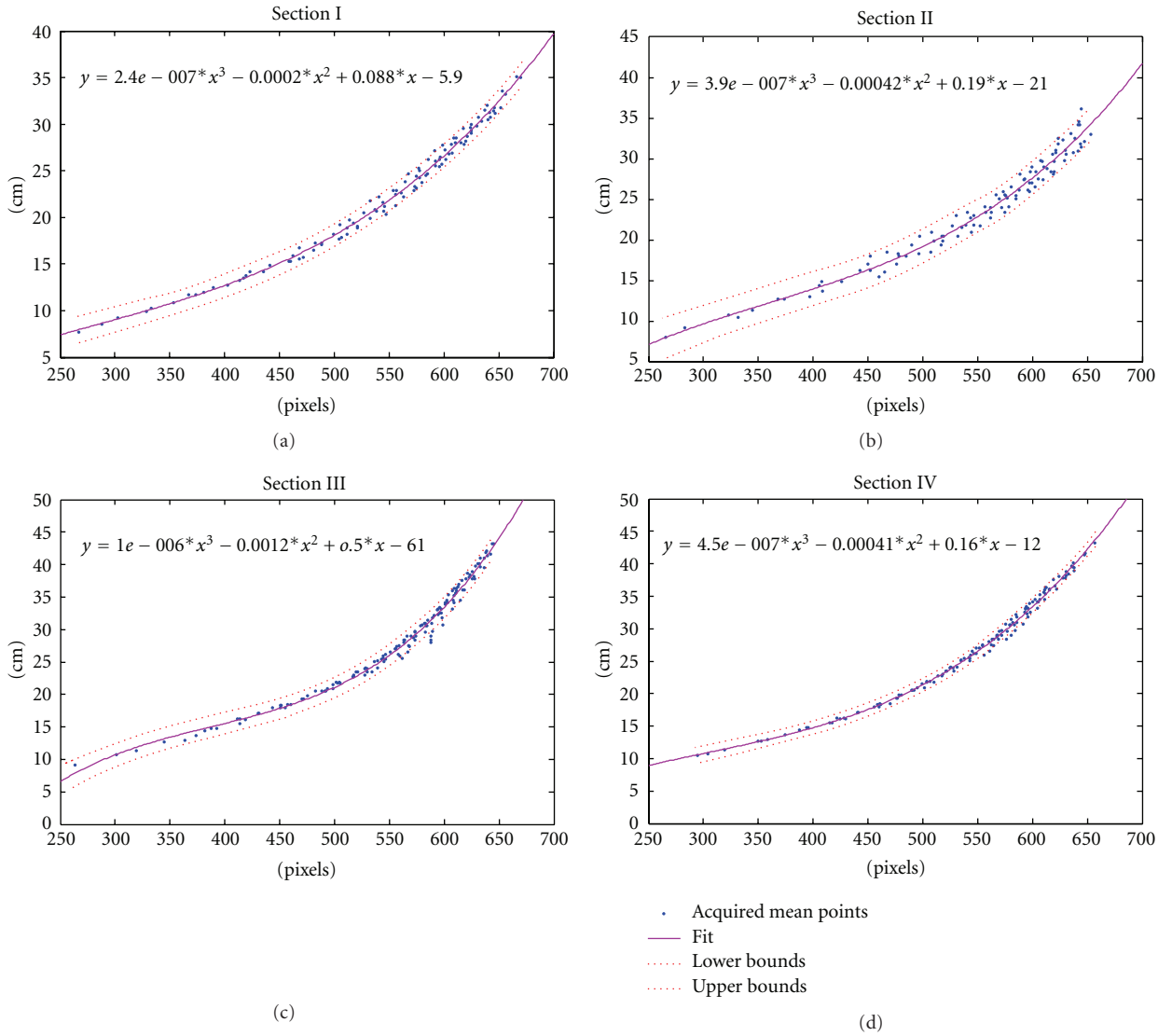


Figure 7: Polynomial fitting for distance estimation for sections I, II, III and IV.

Otherwise, the center of mass will be determined considering the distributed mass of all objects:

$$C(x \cdot y) = \frac{\sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (i,j) \cdot B(i,j)}{\sum_{i=0}^{M-1} \sum_{j=0}^{N-1} B(i,j)}. \quad (1)$$

The construction of catadioptric systems is a complex task taking into account that there are a lot of of geometrical uncertainties that must be precisely determined in order to assure the necessary accuracy of the system. In this context, it is essential to apply a calibration process to the catadioptric system for determining the errors (namely, systematic and random ones) related to defects in the catadioptric systems (e.g., mirror defects).

3. Related Works

Several works have been developed using FPGAs for speeding up image processing tasks, mainly for embedded systems applications with real-time constraints. In [7], a biological inspired architecture for motion estimation by optical flow was implemented. This approach is suitable for implementation in both FPGA and ASIC devices, achieving a processing rate of 177 frames per second (128×96 pixels). Also, in [8] an FPGA implementation of an embedded motion estimation sensor (that uses an optical flow algorithm achieving 15 frames per second for images with 640 × 480 pixels size) is proposed.

A vision system for visual feedback applied to control a mechanical system was proposed in [9]. In this approach a matched filter by correlation is used, which also determines the object's center of mass each 4,51 ms for small images (256 × 256 pixels). In [10], a system for image

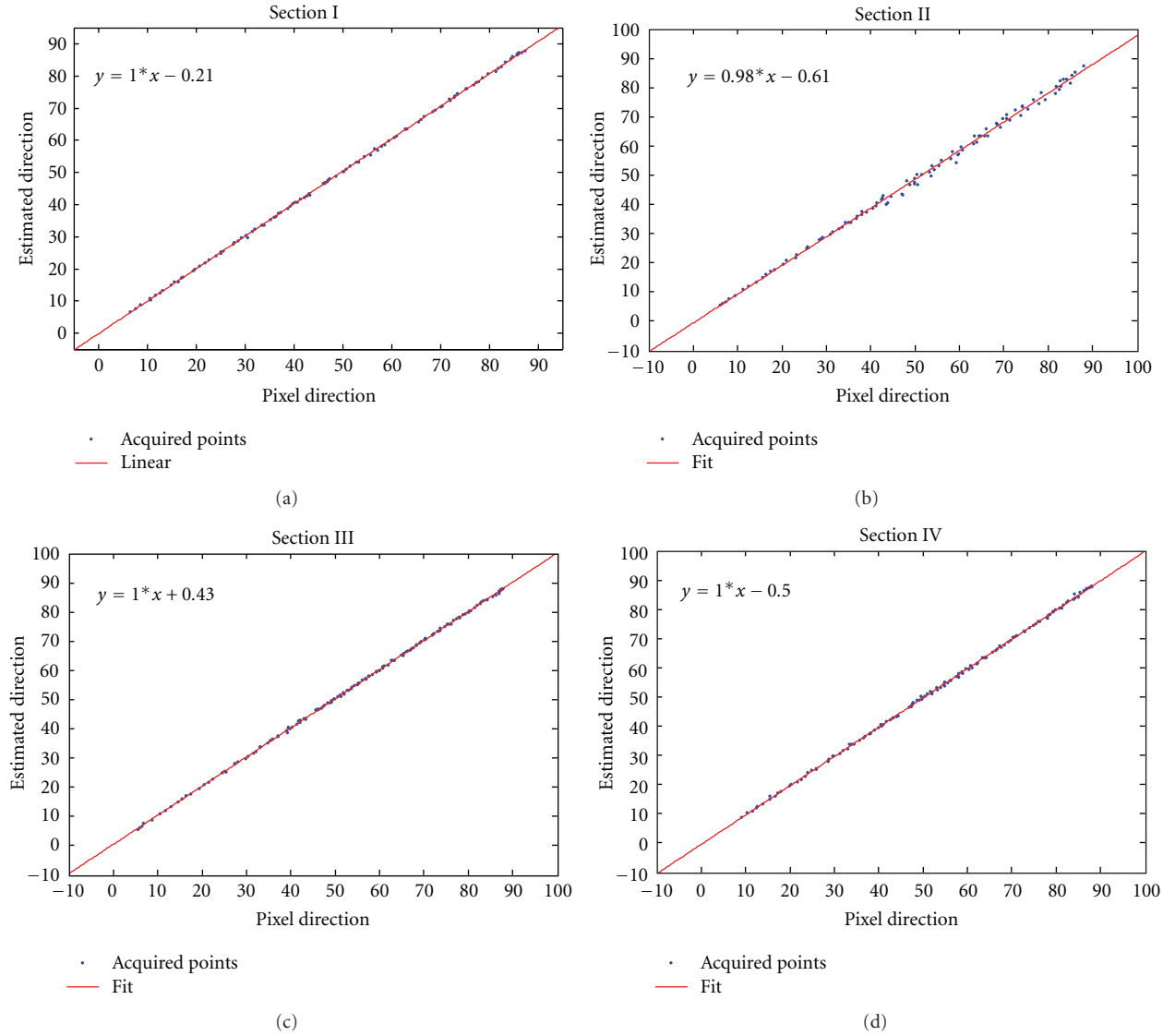


Figure 8: Polynomial fitting for direction estimation for sections I, II, III, and IV.

filtering and motion estimation using SAD (sum of absolute differences) is implemented using a systolic architecture suitable for estimating motion each 5 ms in images with 640×480 pixels.

The design and implementation of robust real-time visual servoing control, with an FPGA-based image coprocessor for a rotary inverted pendulum, are presented in [11]. In this approach the position of the pendulum is measured with a machine vision system whose image processing algorithms are pipelined and implemented on a FPGA device for achieving real-time constraints. Furthermore, it uses an edge enhancement algorithm to determine the center of mass of the detected object, reaching a throughput of 580 (128×101 pixels) processed frames per second.

In [12, 13] an FPGA-based video processing for surveillance systems is described. Reference [13] shows an FPGA implementation for real-time background subtraction. The implemented architecture reaches a performance of 32,8 frames per second with 1024×1024 images. In [12] a pipeline architecture for multimodal background generation algorithm is described, for colour video stream and moving objects segmentation based on brightness, colour, and textural information. In the later case, the overall throughput was about 25 frames per second, with a resolution of 720×576 pixels.

An implementation in a PC of an omnidirectional sensor for mobile robot navigation was presented in [14]. In this approach, a catadioptric system was calibrated by placing landmarks in the environment, using a polynomial interpolation to characterize the system. Additionally, the same uses an edge enhancement technique to create a polar map of the surrounding environment.

Reference [15] introduces an implementation of a framework for image processing speedup by using reconfigurable devices. Some of the most common image preprocessing algorithms were implemented achieving a high throughput.

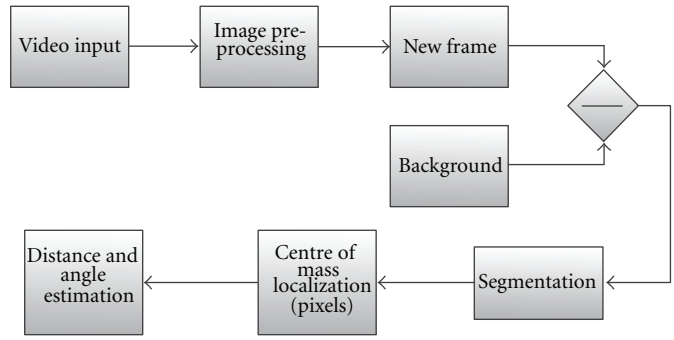


Figure 9: Flowchart of the general hardware architecture.

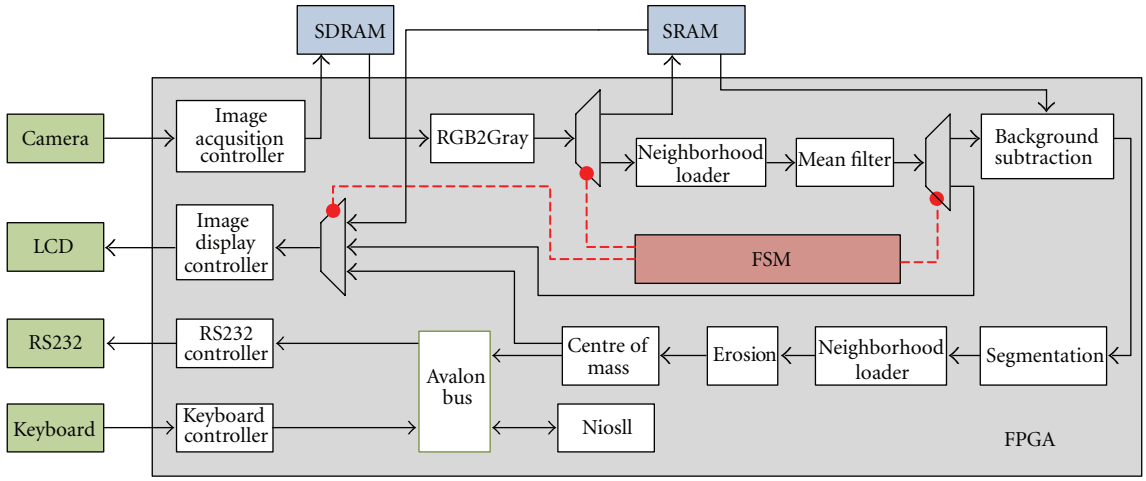


Figure 10: Hardware implementation of the image acquisition block.

Otherwise, [16] uses dynamic reconfiguration for color recognition and optical flow computation in FPGA, in which a throughput of 30 frames per second for 160 × 120 pixels images is achieved.

An approach that uses lookup tables for avoiding complex computations was proposed in [17], in which an architecture for real-time rectification of catadioptric images was implemented in FPGAs. This work has a good throughput but requires a large amount of memory block for storing all the distances, which were calculated off line.

In [18] an FPGA-based architecture (which calculates pixel by pixel the undistorted image from a polar frame) is proposed, thus providing a plane image as output. In that case, a pipeline architecture is used to organize the processing stages, achieving a throughput of one pixel per clock cycle. An FPGA for image reconstruction was used in [19]; however, differently from [17, 18], where the images are generated by an omnidirectional mirror, the system processes the images from a camera with a fisheye lens, although, in this case, the authors do not present a description of the architecture for the reconstruction process.

An architecture using a mixed FPGA/DSP to obtain large speedup factor for high-resolution images was presented in [20], while in [21] an embedded Nios II processor that makes use of several hardware coprocessors for image filtering and tasks related to the autoadjustment of the camera focal length was described.

In [6] a complete procedure for catadioptric systems calibration using line projections is presented. In this case, the system achieves a high accuracy for paraboloid mirrors. An approach that uses calibration patterns to determine the response of hyperbolic sections on a nonrevolute hyperbolic mirror was presented in [22]. The simple idea that the external and internal boundaries of the mirror can be used as a 3D calibration pattern was proposed in [23], allowing for a high-speed self-calibration procedure. [24] which developed a complete generic camera calibration procedure by overlapping calibration grids simultaneously. It is suitable for calibrating many kinds of cameras such as fisheye lens, catadioptric cameras with spherical and hyperbolic mirrors, and also multicamera setups.

Some of the cited works use FPGAs for accelerating omnidirectional vision processing and have mainly focused on image undistortion/reconstruction/rectification tasks. However, in mobile robot applications (such as localization, navigation, and multiagent robotics), it is not always necessarily a complete image reconstruction, but the correct, appropriated, and fast measurement of distances between the robot and the different environment objects. In this context, the main contribution of this work is to provide the robot

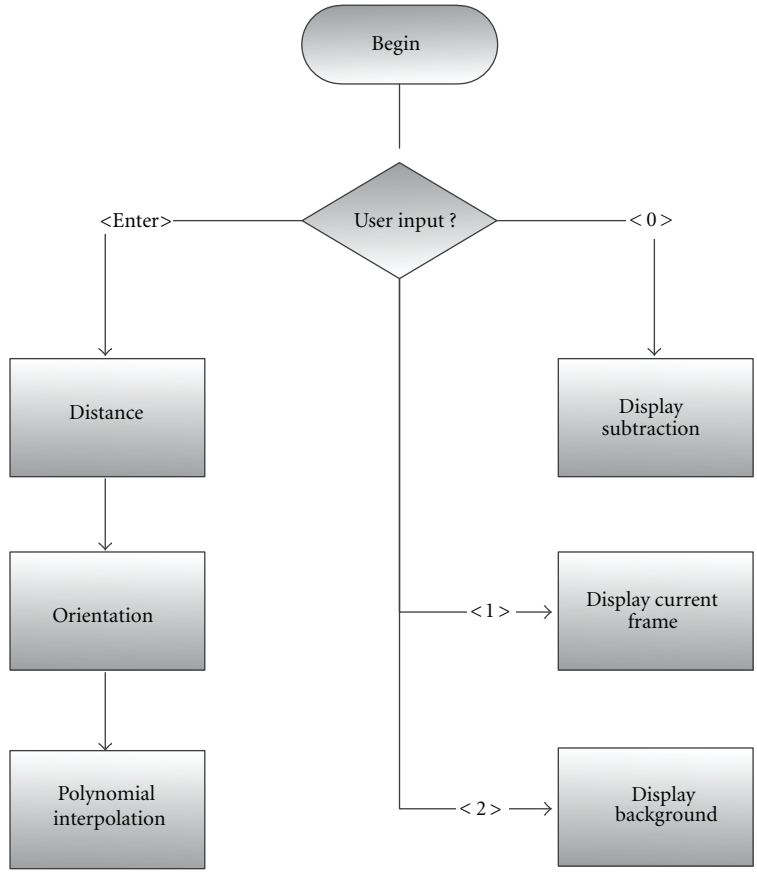


Figure 11: Flowchart of the NiosII coprocessor.

with a suitable and low-cost system for measuring distances automatically from the robot to the detected objects, using an appropriated image resolution (800 × 480 pixels) and achieving real-time characteristics.

4. Development of the Catadioptric System

This section describes the mechanical design of the catadioptric system as well as the calibration procedure description and its analysis.

4.1. The Proposed Catadioptric System. A catadioptric system allows the camera to capture reflected images in the mirror, obtaining a panoramic view. Figure 3 shows the system developed and its main components: (a) a convex mirror, (b) the mounting brackets, and (c) a CMOS camera with a maximum resolution of 2592 × 1944 pixels. In order to perform numerical comparisons with related works we have used a resolution of 800 × 480 pixels, which is appropriated for mobile robotics applications. The distance between the camera and the vertex of the mirror is approximately 17 cm.

In this case it is desirable that the center of the mirror is placed in a vertical line from the center of the camera lens. To achieve this, once mounted the system, an image was acquired and analyzed in order to determine (in pixels) the coordinates of the circle projected by the mirror over the image plane (Figure 4 shows this idea). Otherwise, in this approach, the mirror's projection was assumed to be circular.

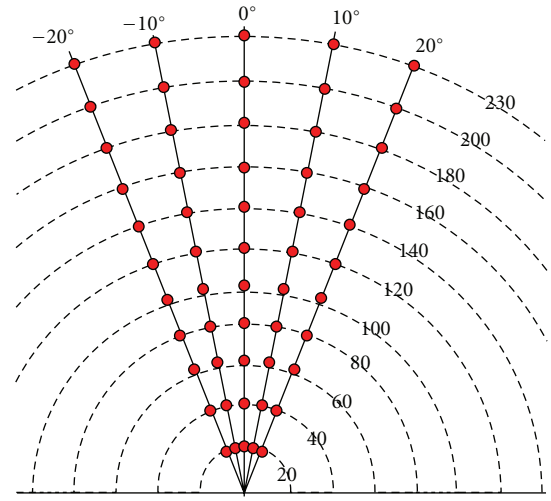


Figure 12: Calibration positions for a particular section in the scene.

For mobile robot applications, the catadioptric system provides a panoramic image in which the robot occupies

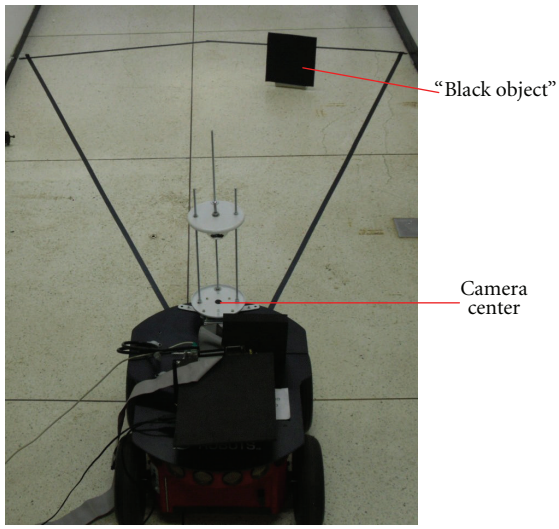


Figure 13: The overall system and the calibration environment.

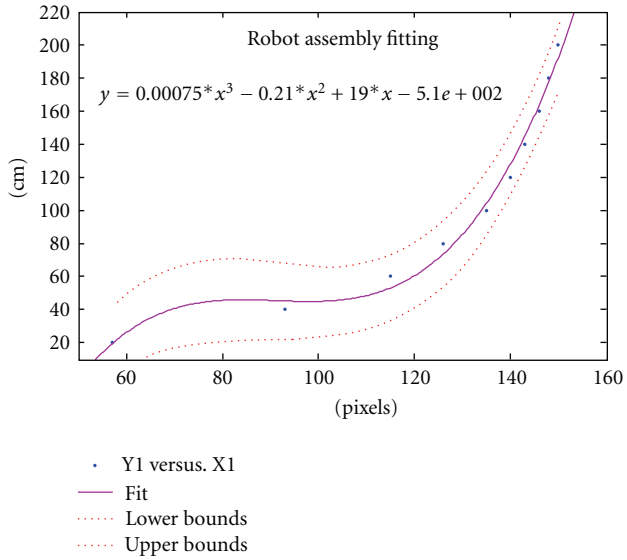


Figure 14: Polynomial interpolation functions.

the center of the image. Figure 5 shows the omnidirectional image captured and the respective environment, in which the robot is positioned with some objects around it, with arrows indicating the correspondence among objects and the image.

The omnidirectional vision system provides a panoramic image, which can be processed in order to extract the distance in pixels between the center of the mirror and the identified object. Once the system has been calibrated, the actual distance can be computed by using a mathematical model (explained in the following section). Additionally, by using omnidirectional vision it is possible to estimate the direction of the surrounding objects, providing to the robot a polar representation of the environment. We assume that the robot only detects objects within its viewing area, which corresponds to a circle with a radius of 2.0 meters. Objects outside this area are reflected on the mirror border, and then large distortions are produced. Additional distortions in the acquired image can be produced by small errors in the optical geometry of the vision system.

Table 1: RMSE for each section.

Section	Distance RMSE	Direction RMSE
I	0.9284	0.244
II	1.038	1.110
III	0.8353	0.273
IV	0.5398	0.388

4.2. Calibration Process. The quality of the data obtained from an omnidirectional vision system depends directly on several constructive parameters such as the optical geometry, curvature of the convex mirror, and quality of mirror's surface. For one to use the equations of the mirror surface profile (and afterward modeling the light reflection), the geometrical parameters of the system must be precisely characterized. However, in this work we have used a convex mirror with unknown geometry, and then it is not possible to use light reflection equations.

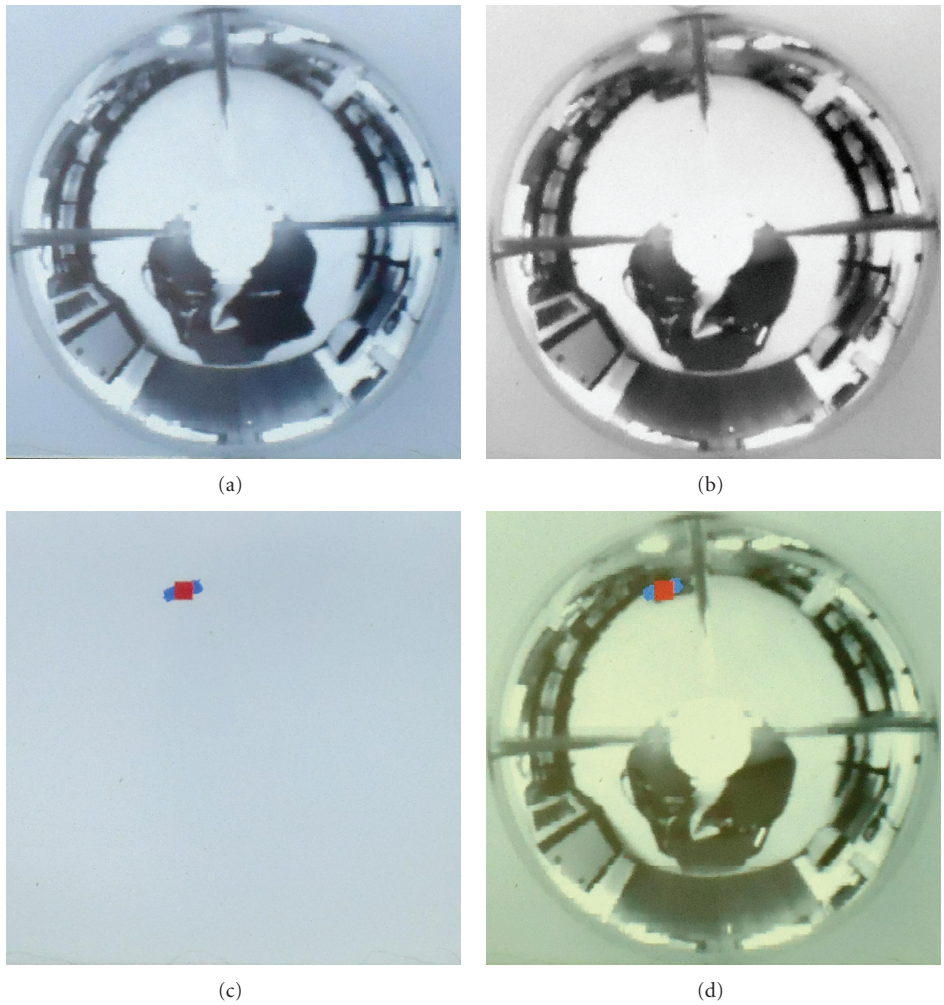
The proposed calibration process allows the whole vision system to be characterized providing a fitting function relating the distance in pixels with distances in world coordinates. The calibration procedure was performed by associating objects placed at measured distances with the distances estimated in the image in pixels, obtaining a polynomial fitting associated to a particular section of the convex mirror.

As in our previous work [3], the image was divided into four sections, and the same calibration procedure was executed to each one. In order to provide a better demonstration of the correctness of the calibration procedure, in this work a calibration board and a 14 megapixels camera have been used. The mounted system is depicted in Figure 6. The board used has a separation of 2,54 cm between each hole (in both horizontal and vertical directions). The catadioptric mount was positioned approximately in the center of the board, and the images are shown in Figure 6.

In this image, the holes were detected and their pixel coordinates were determined. The image was divided in four sections, and for each section the real distance and the distance in pixels were associated with a polynomial fitting. In order to allow the robot to identify the position of any object in the surrounding environment, the system was also calibrated to estimate the direction of the detected object. Figures 7 and 8 show the polynomial fitting for estimating both distances and directions, respectively, for each section of the mirror.

Table 1 shows the RMSE (root mean-square error) values of the fittings which characterize the quality of the obtained models. It can be observed that the polynomial fittings have a low value of the RMSE, which means that the calibration data can be well modeled by the polynomials.

In order to validate the precision of the measurement system, a new image (with the same assembly of Figure 6)



(a) (b) (c) (d)

Figure 15: Results of the image processing chain: (a) background, (b) new frame, (c) subtraction, and (d) overlap.

was shot, and five points were picked from each section. By using the polynomial fittings (used like calibration curves), their actual positions were estimated. Table 2 shows the actual and the estimated positions of the points (distance and direction).

The validation results (shown in Table 2) demonstrate that both the calibration procedure and the polynomial fitting can be effectively used to relate distances/directions in the image (in pixels) with the actual distances/directions values.

5. FPGA Implementation

The proposed algorithms for image processing were implemented in hardware, using both VHDL and Verilog hardware description languages. Figure 9 shows the general architecture for motion detection, which is composed of several hardware components (blocks) connected in a pipeline way. The first processing step receives from the camera an RGB 800 $\times$ 480 pixels image with a resolution of 8 bits per color channel. At the second step, namely, *image processing*, a gray-scale image is obtained and a *mean* filter is applied for eliminating noise. At the third step, the background subtraction is performed. To do this the background image has been previously stored in an SRAM. At the fourth step, a thresholding algorithm is applied for segmentation, obtaining a binary image (only one bit per pixel). Additionally, the obtained image is eroded in order to minimize noise. At the fifth stage the center of mass is computed, and, finally, at the sixth stage the object position (distance and orientation) is computed.

5.1. Image Acquisition and Color Conversion. The system uses a CMOS camera which provides synchronism and data signals in an RAW format. A color conversion process is performed by calculating the RGB data from RAW ones and storing it in an external SDRAM (see Figures 5 and 6).

5.2. Mean Filter Implementation. After the pixel conversion from RAW to RGB format, a gray-scale transformation is applied. Afterward, a *neighborhood loader* block provides a 3 $\times$ 3 neighborhood to a *mean filter*, which eliminates

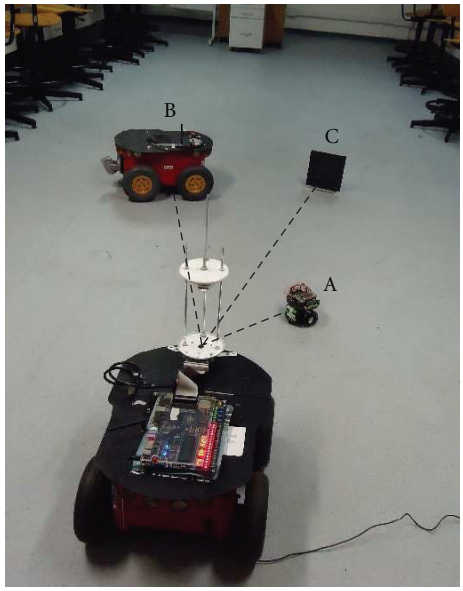


Figure 16: Objects used for testing the overall system.

high-frequency noises. The neighborhood loader operation requires an initial latency of 1603 clock cycles (69.62 μs) [25]. A convolution operation was used to implement the mean filter (see Figures 5 and 6), which is performed in one clock cycle by multiplying the mask with the neighborhood and then yielding the sum of the products, after an initial latency. More details on this implementation and the convolution architecture can be found in [25, 26].

5.3. The Background Storage and Image Subtraction. The background image is stored in an external 512 Kbyte SRAM memory (chip ISSI IS61LV25616AL) of the DE2 development kit. Once the mean filter is performed, the subtraction between the current frame and background is computed, providing one output pixel per clock cycle. Afterward, the absolute value of each pixel is calculated. Finally, the segmentation operation is performed by a simple thresholding operation (see Figures 5 and 6).

5.4. The Erosion Operation. The erosion computation is based on logic operations between the pixel of a binary image and a structuring element as shown in (2). The erosion block receives nine pixels from the neighborhood loader (f_i), as well as the structuring element (K_i) (a square mask was used like structuring element). Therefore, the e_i values are calculated in the first equation. Afterward, they are used in the next equation in order to perform a complete erosion operation. Both steps are performed in one clock cycle. In this work we have used a neighborhood of nine elements; therefore, $i = 1, \ldots, 9$:

$$e_i = \overline{K}_i \cdot \overline{f}_i + \overline{K}_i \cdot f_i + K_i \cdot f_i,$$
$$\text{Erosion} = e_1 \cdot e_2 \cdot e_3 \cdot e_4 \cdot e_5 \cdot e_6 \cdot e_7 \cdot e_8 \cdot e_9. \quad (2)$$

Table 2: Validation points.

Section	Actual distance (cm)	Estimated distance (cm)	Actual direction (°)	Estimated direction (°)
I	20.5	19.6	7.6	7.2
I	11.3	12.1	69.5	68.4
I	17.0	18.0	66.2	67.4
I	21.7	23.3	71.4	72.7
I	28.4	29.4	65.8	65.0
II	28.4	25.8	10.8	11.5
II	27.4	25.2	22.8	23.6
II	27.0	25.7	50.9	53.6
II	30.6	29.6	50.2	50.9
II	26.2	24.8	30.5	31.8
III	17.9	18.0	8.8	7.9
III	28.4	31.8	10.8	9.8
III	29.7	31.9	20.9	19.4
III	34.5	35.9	37.3	35.7
III	43.3	41.5	51.1	48.9
IV	15.5	16.1	10.3	10.2
IV	23.4	24.4	13.2	13.6
IV	29.6	30.3	32.3	32.8
IV	35.9	36.0	46.5	46.6
IV	43.3	43.7	51.1	51.8

5.5. The Center of Mass Calculation. In this work only the detection of a single object is performed at a time. In order to calculate the center of mass, $C(x, y)$, (1) was used, where $B(i, j)$ is a binary image and i and j are the positions of pixels on the image. Since the algorithm has to explore the overall image, the center of mass is calculated at each frame.

5.6. Distance and Orientation Estimation. In our previous work [3] the actual and pixel distances were computed using several floating-point arithmetic libraries [27]. However, a large consumption of hardware resources was observed, specially for embedded applications. In this work we have chosen an embedded software implementation for these computations allowing for cost reduction in logic area.

A Nios II soft processor (from Altera) has been used in order to execute the following tasks: (a) receiving commands from the user through a PS2 keyboard, (b) calculating the distance in pixels and orientation values, (c) calculating the actual distance using a polynomial function obtained from the calibration data (see Figure 14), and (d) sending to the host (PC or robot computer) both the estimated distance and orientation values. The communication with the PC is done through a RS232 communication standard. The image processing architecture and a keyboard are connected to the Nios II using the Avalon Bus from Altera, as shown in Figure 10.

Figure 11 shows the flowchart algorithm implemented in the NiosII processor. The processor receives the user input

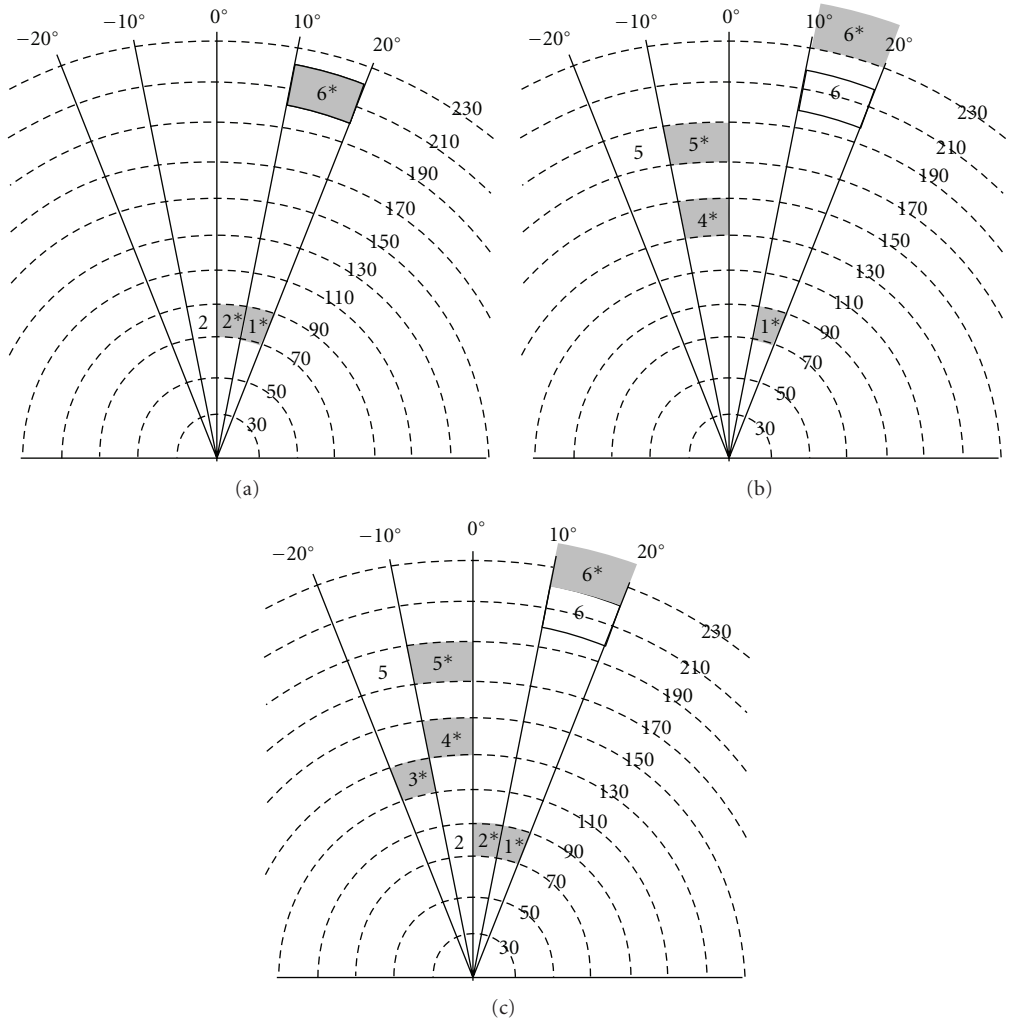


Figure 17: Estimation of distance and orientation. (a) object *A*, (b) object *B*, and (c) object *C*. White grids are the real positions and gray grids are the estimated position.

commands from a keyboard, allowing the selection of the following options.

(i) Send to the host the distance and orientation values. In this case the NiosII receives from the hardware architecture the center of mass coordinates $C(x, y)$ and computes the euclidean distance in pixels and the orientation θ using the atan() function. Finally, the polynomial interpolation is used to estimate the actual distance R, sending the polar coordinates (R, θ) to the host.

(ii) Capture a new background image. This option allows the user to upgrade the background image in the SRAM memory.

(iii) Subtract the background from current image. This option allows the user to manually execute the subtraction image step.

(iv) Display the subtraction result. This option allows the user to show the image substation result in the display.

(v) Display the current image. This option allows the user to address the current camera image to the display.

(vi) Display the background image. This option allows the user to see the stored background image in the display.

The last five commands make use of the Avalon bus to send the respective commands to the FSM (finite state machine) which controls the data flow and the LCD. The FSM was implemented in hardware for controlling the overall system operation according to the user inputs. The same controls several multiplexers for addressing the incoming pixels to the SRAM/SDRAM memories and the hardware modules previously described.

In our approach both the Euclidean distance as well as the arctangent function need to be computed (but only once time at each frame). In this case, although von Neumann-based architectures have serious restrictions (such as the NiosII case) for real-time image processing, specially for

Table 3: Synthesis results (chip ep2c35f672c6).

Implemented core	LC 33216	MB 483840	DSP18×18 35	Freq. 250 MHz
Entire Architecture	9953 (30.0%)	359352 (74.3%)	8 (22.8%)	10.2
Image Acquisition	2161 (6.5%)	57400 (11.86%)	0 (0%)	45.31
Gray scale Conversion	498 (1.5%)	384 (0.08%)	1 (1%)	45.31
Neighborhood Load	681 (2.05%)	16256 (3.36%)	0 (0%)	104.12
Spatial Convolution	1853 (5.0%)	16256 (3.36%)	3 (5%)	104.12
Background Subtraction	16 (0.05%)	0 (0%)	0 (0%)	250
Segmentation	15 (0.05%)	0 (0%)	0 (0%)	250
Erosion	722 (2.17%)	0 (0%)	0 (0%)	321.3
Center of mass	2521 (7.59%)	0 (0%)	0 (0%)	10.2
NiosII	3615 (10.9%)	285696 (59.0%)	4 (6%)	250.0

Table 4: Latency of the motion detection architecture.

Implemented core	Latency
Background Subtraction	2 clk
Segmentation	1 clk
Erostion	1603 clk
Center of mass	384.001 (1 frame)
Entire architecture	385.607

embedded systems, the software implementation of these tasks attends the real-time constraints in this work.

6. Results

The proposed architectures for processing the images from the omnidirectional vision system were effectively implemented in a Cyclone II FPGA device using the Quartus II development tool.

6.1. Synthesis Results. Table 3 presents the synthesis results of the overall architecture and its main components. The cost in logic area is presented in terms of logic cells (LCs), memory bits (MBs), and embedded DSP 18 × 18 blocks consumption. The performance of the architectures is presented in MHz.

It can be observed that the entire architecture consumes 30% of logic cells and around 74% of the memory bits. The image acquisition block requires the largest number of memory bits due to the fact that the camera provides pixels in an RAW format; therefore, this block needs to store several rows in order to convert the pixels to an RGB format. The neighborhood loader is composed of two line buffers, which are used for providing the 3 × 3 pixels for performing the neighborhood operations [25]. As described in Section 5, the spatial convolution block makes use of a neighborhood loader module leading to more memory bits consumption. As expected, the NiosII implementation requires a large

Table 5: Distance calibration data using mobile object detection.

Distance (cm)	Mean distance (pixels)	σ (pixels)
20	56.6	3.3
40	93.4	7.3
60	114.6	1.3
80	126.2	7.3
100	134.6	7.3
120	139.8	1.7
140	143.0	1.0
160	146.2	2.2
180	147.8	1.7
200	149.6	1.3
230	152.2	1.4

Table 6: Direction calibration data using mobile object detection.

Angle (°)	Mean angle (°)	σ (°)
−20	−24.8	4.8
−10	−14.4	11.6
0	−1.3	1.1
10	9.3	4.4
20	21.1	4.5

Table 7: Validation results (using the moving object detection technique) distance is given in cm.

Real distance	Estimated distance and error					
	A	e_A	B	e_B	C	e_C
$p_1 = 70$	70	0%	80	14%	83	19%
$p_2 = 74$	70	5%	—	—	83	12%
$p_3 = 115$	—	—	—	—	107	7%
$p_4 = 137$	—	—	132	4%	132	4%
$p_5 = 170$	—	—	170	0%	170	0%
$p_6 = 200$	200	0%	248	24%	235	17%

Table 8: Validation results for orientation in degrees (°), using mobile object detection.

Angle (°)	A	B	C
$p_1 = 20$	20	19	20
$p_2 = -1$	9	—	9
$p_3 = -18$	—	—	−11
$p_4 = -8$	—	−4	−3
$p_5 = -14$	—	−8	−7
$p_6 = 11$	16	15	16

Table 9: Performance comparison.

Author	Year	Main algorithm	Image resolution	Frames per second	Megapixels per second
[10]	2009	Sum of absolute differences (SAD)	640 × 480	200	61.44
[13]	2012	Background subtraction	1024 × 1024	32	33.55
[9]	2006	Correlation	256 × 256	221	14.48
[12]	2011	Background subtraction	720 × 576	25	10.37
This work	*2012*	*Background subtraction*	*800 × 480*	*26*	*9.98*
[11]	2011	Edge enhancement	128 × 101	580	7.50
[8]	2008	Optical flow	640 × 480	15	4.61
[7]	2008	Optical flow	128 × 96	177	2.17
[15]	2007	Optical flow	160 × 120	30	0.58

amount of memory bits for storing the program memory and its hardware architecture.

The background subtraction block is based on a pixel by pixel operation. The segmentation block operates using a comparator and a multiplexer. The erosion block is based on simple logic operations. Therefore, these blocks have a small hardware resources consumption.

It can be observed that the entire architecture operates at a maximum frequency of 10.2 MHz. Taking into account that the system is based on a pipeline architecture and the images resolution is 800 × 480 pixels, the system achieves a performance of 26.6 frames per second.

It is important to point out that the selected FPGA chip is not the largest device from the Cyclone II family; therefore, one can expect a performance improvement when using modern FPGA devices with more hardware resources.

Since the proposed system is based on a pipeline architecture, it is necessary for several clock cycles (*latency time*) before computing the first result (center of mass). According to Table 4 the latency of the proposed architecture is around 385.607 clock cycles. Note that the center of mass has the largest latency due to the fact that the entire image must be analyzed for computing the area.

6.2. The Calibration of the Overall System Using the Moving Object Detection Technique (Figure 13). The process of using a polynomial fitting for associating distances in pixels with actual distances was validated in Section 4. However, in real environments, mobile robots commonly operate with large uncertainties associated to the odometry sensors and nonholonomic restrictions. Therefore, we have again calibrated the catadioptric system including the mobile robot, using a precision lower than shown in Section 4. To do that, the moving object detection technique implemented in the FPGA has been used for calibration tasks. In this case, several objects were introduced in the scene, and the detection system automatically defined the distance between the robot and the detected object. The calculated distances can be compared with the actual ones in the scene, yielding new polynomial functions.

For this case, the mirror was divided into sections of 40° (nine sections cover the whole mirror surface). In the scene, a grid of radial distances were chosen to vary from 20 cm to 230 cm around all sections in a circle. Figure 11 shows the calibration positions for a particular section in the scene. A black object was positioned over the red dots. The calibration environment is shown in Figure 12, in which the robot, the catadioptric system, and the object are presented. It is important to point out that the environment does not suffer from natural light variations; only artificial illumination was used, and the white floor allows for a high contrast with the black object.

Tables 5 and 6 show a statistical analysis of the experimental data. According to Figure 4, each object located in the scene had its projected position on the image calculated (in pixels) for 5 different orientations in the chosen section, in the form of mean distance and its standard deviation, summing up 55 positions in a section.

As expected in Section 5, the interpolation function has a monotonically increasing behavior. It is important to notice that, as an effect of the reflection characteristics of this system, for greater distances the objects appear smaller than in common acquisition systems (e.g., nonomnidirectional ones). That occurs due to the reflection angle for objects placed far away from the mirror. In this case, it can be observed that the variance values for each distance do not follow the monotonic behavior. This is because the error in distance estimation is not a function of the distance but is mainly determined by mirror's surface quality.

The direction of the detected object is determined by calculating the arctangent function using the estimated coordinates of the center of mass (of the detected object). Notice that the uncertainty in direction estimation is related to the uncertainty in object detection. To calculate the direction, the values in pixels have been used; therefore, both the distance estimation and direction estimation are independent tasks.

Figure 14 depicts the behavior of the pixel/centimeter transformation by using a polynomial interpolation for each one of the regions.

6.3. Validation Results. In order to demonstrate the system running in an actual scene, Figure 15 shows an example of the processing chain, in which Figure 15(a) represents the background image. Figure 15(b) shows an object around the mobile robot. Figure 15(c) depicts the background

subtraction and the position of the center of mass. Finally, Figure 15(d) overlaps the current image and the detected center of mass.

Several experiments have been performed so as to evaluate the accuracy of the implemented system. To do that, three different objects (namely, *A*, *B*, and *C*) which corresponds to (a) a small cylindrical robot, (b) a pioneer mobile robot, and (c) the calibration object, respectively, were placed at different positions in front of the robot. Figure 16 depicts the objects used for testing the overall system. The distances and orientation between the center of the camera and the objects have been previously measured (the actual values) in the arena. Additionally, both the estimated distances and orientation were sent to the host (via RS 232 interface) and compared with the actual values.

Table 7 presents the location results and respective errors for each object. It can be observed that the proposed architecture achieves more accurate results when detecting the object *A*. It can be explained because the size and shape of the cylindrical robot produce a small shadow, leading to a better accurate. As expected, large approximation errors were achieved for large distances. For instance, results for localization of objects *B* and *C* show the largest errors (around 24% and 17%, resp.). This fact is explained given that the spherical mirror produces large distortions to the light rays reflected from the uppermost surface of the mirror. This distortion is produced by a compression effect, as the farther the object is, the smaller is its projection on the mirror surface.

Table 8 presents the orientation estimation for each object. Figure 17 uses occupation grids in a polar graph form for summarizing the achieved results of distance and direction estimation. The gray grid represents the estimated position of the object. As expected, the system produces large errors for large distances (see point 6 for objects *B* and *C*).

One can conclude that the omnidirectional system performs better for estimating distances than orientation. Notice that the calibration data (see Table 8) show large errors for orientation values. As explained in Section 4 the system requires a large contrast between background and objects. Therefore, when the system operates with a low contrast, some errors in the object borders are introduced. However, these errors can be overcome by using more efficient techniques for motion segmentation (e.g., optical flow).

6.4. Performance Analysis. Additionally, the same algorithm for motion detection was implemented in a PC, running at 2.2 GHz, 2.0 GB RAM using a real-time xPC Target OS from MathWorks. The average elapsed time for processing a 10×10 pixel image was around 138.1 μs, (value of the average TET (task execution time)). Thus, an output pixel is processed in 1.381 μs. Therefore, the proposed hardware architecture, operating at 10.2 MHz, achieves a speedup factor of 13.78 in comparison with the real-time software solution.

As cited in Section 3, several works have been developed to solve the problem of object's position estimation for real-time applications. Table 9 shows the comparison among some works and our approach. Each proposal uses different algorithms and image resolutions, leading to difficulty in the comparison task. Therefore, we have used the overall throughput (megapixels per second) as a comparison metric.

It is important to note that all systems listed in Table 9 have as output the estimated position of an object in the image. In this case, all listed systems have implemented FPGA-based hardware architectures to process the image and determine object's position.

7. Conclusions

This work has presented a FPGA-based omnidirectional vision system for mobile robotic applications. It takes advantage of a pipeline approach for processing the polar image, using a background subtraction algorithm. The overall latency of the motion detection architecture is 385.607 clock cycles, and after this latency, the system has a throughput of 26 frames per second (running at 10.2 MHz). The proposed architecture is suitable for robot localization, allowing to compute the distance between the robot and the surrounding objects.

The architectures were described in VHDL and Verilog and successfully implemented in a Cyclone II FPGA device. Synthesis results have demonstrated that the proposed hardware achieves an operational frequency around 10.2 MHz. In addition, the pipelined architecture allows the image to process one pixel per clock cycle after an initial delay. This fact demonstrated acceleration of 13, 78 times in comparison with the same algorithm in C using a xPC Target OS from MathWorks implementation running on a common desktop platform.

This work has also addressed several calibration problems related to omnidirectional vision systems (based on a catadioptric implementation) of mobile robotic applications, especially the application of a technique for detection of mobile objects for mirror calibration tasks. Experimental results show that the results are consistent with the expected ones.

Concerning our future work we intend to analyze the power consumption of the proposed architecture. It is an important issue in order to validate the effectiveness of the implemented algorithms for real-time image processing in portable applications.

Acknowledgments

The authors would like to thank CAPES Foundation and National Council of Scientific and Technological Development of Brazil-CNPq (processes 133501/2010-8 and 142033/2008-1) for the financial support of this work. Special thanks are to Altera Corp. for providing Quartus II Licenses and to DHW Engenharia e Representação for the partnership.

References

[1] R. Siegwart and I. Nourbakhsh, *Introduction to Autonomous Mobile Robots*, MIT Press, Cambridge, Mass, USA, 2004.

[2] L. Spacek and C. Burbridge, "Instantaneous robot self-localization and motion estimation with omnidirectional

vision," *Robotics and Autonomous Systems*, vol. 55, no. 9, pp. 667–674, 2007.

[3] J. Yudi Mori, D. Mũoz Arboleda, J. N. Arias Garcia, C. Llanos Quintero, and J. Motta, "FPGA-based image processing for omnidirectional vision on mobile robots," in *Proceedings of the 24th Symposium on Integrated Circuits and Systems Design*, pp. 113–118, João Pessoa, Brazil, 2011.

[4] E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision*, Prentice Hall, 1998.

[5] K. Daniilidis and C. Geyer, "Omnidirectional vision: theory and algorithms," in *Proceedings of the 15th International Conference on Pattern Recognition*, vol. 1, pp. 89–96, 2000.

[6] C. Geyer and K. Daniilidis, "Catadioptric camera calibration," in *Proceedings of the 17th IEEE International Conference on Computer Vision (ICCV '99)*, vol. 1, pp. 398–404, September 1999.

[7] G. Botella, M. Rodriguez, A. Garca, and E. Ros, "Neuromorphic configurable architecture for robust motion estimation," *International Journal of Reconfigurable Computing*, vol. 2008, Article ID 428265, 9 pages, 2008.

[8] Z. Wei, D. Lee, N. Brent, J. Archibald, and B. Edwards, "FPGA-based embedded motion estimation sensor," *International Journal of Reconfigurable Computing*, vol. 2008, Article ID 636145, 9 pages, 2008.

[9] K. Shimizu and S. Hirai, "CMOS+FPGA vision system for visual feedback of mechanical systems," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '06)*, pp. 2060–2065, May 2006.

[10] G. Saldaña-González and M. Arias-Estrada, "FPGA based acceleration for image processing applications," in *Image Processing*, 2009.

[11] Y. Tu and M. Ho, "Design and implementation of robust visual servoing control of an inverted pendulum with an FPGAbased image co-processor," *Mechatronics*, vol. 21, no. 7, pp. 1170–1182, 2011.

[12] T. Kryjak and M. Gorgoń, "Real-time implementation of moving object detection in video surveillance systems using FPGA," *Computer Science*, vol. 12, pp. 149–162, 2011.

[13] R. Rodriguez-Gomez, E. Fernandez-Sanchez, J. Diaz, and E. Ros, "FPGA implementation for real-time background subtraction based on horprasert model," *Sensors*, vol. 12, pp. 585–611, 2012.

[14] R. Chojecki and B. Siemiatkowska, "Mobile robot navigation based on omnidirectional sensor," in *Proceedings of the European Conference on Mobile Robots (ECMR '03)*, pp. 101–106, Radziejowice, Poland, September 2003.

[15] M. A. Vega-Rodríguez, A. Gómez-Iglesias, J. A. Gómez-Pulido, and J. M. Sánchez-Pérez, "Reconfigurable computing system for image processing via the internet," *Microprocessors and Microsystems* , vol. 31, pp. 498–515, 2007.

[16] F. Nava, D. Sciuto, M. D. Santambrogio et al., "Applying dynamic reconfiguration in the mobile robotics domain: a case study on computer vision algorithms," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 4, no. 3, 2011.

[17] L. Chen, M. Zhang, B. Wang, Z. Xiong, and G. Cheng, "Real-time FPGA-based panoramic unrolling of high-resolution catadioptric omnidirectional images," in *Proceedings of the International Conference on Measuring Technology and Mechatronics Automation (ICMTMA '09)*, pp. 502–505, Hunan, China, April 2009.

[18] A. Gardel, A. Hernández, R. Miota, I. Bravo, and R. Mateos, "Correction of omnidirectional camera images using reconfigurable hardware," in *Proceedings of the 32nd Annual Conference on IEEE Industrial Electronics*, pp. 3403–3407, Paris, France, November 2006.

[19] B. Zhang, Z. Qi, J. Zhu, and Z. Cao, "Omnidirection image restoration based on spherical perspective projection," in *Proceedings of the IEEE Asia Pacific Conference on Circuits and Systems*, pp. 922–925, Macao, China, December 2008.

[20] T. Shu-ren, Z. Mao-jun, X. Zhi-hui, L. Le, and C. L. Dong, "Design and implementation of high-resolution omnidirectional vision system," *Chinese Journal of Video Engineering*, vol. 10, no. 1, pp. 1–6, 2008.

[21] A. Maeder, H. Bistry, and J. Zhang, "Towards intelligent autonomous vision systems—smart image processing for robotic applications," in *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO '07)*, pp. 1081–1086, Sanya, China, December 2007.

[22] R. Benosman, E. Deforas, and J. Devars, "A new catadioptric sensor for the panoramic vision of mobile robots," in *Proceedings of the IEEE Workshop on Omnidirectional Vision*, pp. 112–116, 2000.

[23] J. Fabrizio, J.-P. Tarel, and R. Benosman, "Calibration of panoramic catadioptric sensors made easier," in *Proceedings of the 3rd Workshop on Omnidirectional Vision*, pp. 45–52, 2002.

[24] S. Ramalingam, P. Sturm, and S. K. Lodha, "Towards complete generic camera calibration," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '05)*, vol. 1, pp. 1093–1098, June 2005.

[25] J. Y. Mori, C. Sánchez-Ferreira, D. M. Munoz, C. H. Llanos, and P. Berger, "An unified approach for convolution-based image filtering on reconfigurable systems," in *Proceedings of the 7th Southern Conference on Programmable Logic (SPL '11)*, pp. 63–68, Crdoba, Argentina, April 2011.

[26] J. Mori, *Implementação de técnicas de processamento de imagens no domínio espacial em sistemas reconfiguráveis*, M.S. thesis, Universidade de Brasília, Brasília, Brazil, 2010.

[27] D. M. Muñoz, D. F. Sanchez, C. H. Llanos, and M. Ayala-Rincón, "Tradeoff of FPGA design of a floating-point library for arithmetic operators," *Journal of Integrated Circuits and Systems*, vol. 5, no. 1, pp. 42–52, 2010.