Hindawi
International Journal of Reconfigurable Computing
Volume 2018, Article ID 2843582, 11 pages
https://doi.org/10.1155/2018/2843582

Research Article

Toward the Implementation of an ASIC-Like System on FPGA for Real-Time Video Processing with Power Reduction

Lilia Kechiche,[1] **Lamjed Touil**,[2] **and Bouraoui Ouni**[2]

[1] *Laboratory of Electronics and Microelectronics, University of Monastir, Monastir, Tunisia*
[2] *Networked Objects Control and Communication Systems Lab, Sousse, Tunisia*

Correspondence should be addressed to Lilia Kechiche; l.kechiche@gmail.com

Received 4 January 2018; Revised 15 February 2018; Accepted 1 March 2018; Published 22 April 2018

Academic Editor: Michael Hübner

Driven by the importance of energy consumption in system-on-chip design as an evaluation factor, this paper presents a design methodology at the system level to optimize power consumption on ARM-based architecture for real-time video processing. The proposed design flow is based on the interaction between the tool and user optimizations. The tool optimizations are the options and best practices available on the integrated design environment for the Xilinx technology and the target Zynq-7000 architecture. The user methods present methods proposed by the user to optimize power consumption. We used the principles of voltage scaling and frequency scaling techniques for user methods. These two techniques allow energy to be consumed in the proportion of work to be done. The suggested flow is applied on real-time video processing system. The results show power savings for up to 60% with respect to performance and real-time constraints.

1. Introduction

Embedded real-time video applications are becoming widely spread and more and more used in a lot of systems. They have important applications in various domains such as segmentation [1], object tracking [2], visual detection and matching [3], and stereo vision [4]. These systems are generally executed in an embedded environment and are subjected to many constraints like power consumption, time, and resource constraints.

To develop and implement real-time high-quality applications, a combination of dedicated technologies and methodologies has been proposed in order to obtain optimized systems with required performance. The used technologies vary from specific processors such as General Purpose Processors (GPP), Graphic Processing Unit (GPU), and Digital Signal Processors (DSP) to parallel architectures like Application Specific Integrated Circuits (ASICs) or even programmable logic devices (FPGAs).

Today, FPGAs are more used to build complex video processing applications. They give real-time performance, hard to achieve with GPP or DSP [5, 6], while limiting the extensive design work required for ASICs. Furthermore, FPGAs enable implementing highly parallel architectures thanks to the huge amount of programmable logic available on a chip [7].

One of the major problems that face FPGA compared with ASICs is the power consumption, which becomes a limiting factor [8]. Therefore, more effort is being spent to propose a design with low-power dissipation.

Since FPGAs are based on CMOS transistors, power consumption can be divided into two main categories, static and dynamic. The static power is dissipated when the circuit is in a quiescent state. It is caused by leakage currents of CMOS transistors. These currents are the subthreshold leakage current, the gate leakage current, and the junction leakage current [9]. Dynamic power consumption is given by

$$P_d = \alpha \cdot C \cdot V^2 \cdot f, \tag{1}$$

with the following: C is the capacitance, α is the charging rate (depends on clock frequency), V is the voltage supply, and f is clock frequency.

This power is highly related to the technology and it has become a concern with modern FPGAs, which are being implemented in a 24 nm technology.

To reduce power consumption, the principle causes of power dissipation have to be investigated during all the steps of the design process of integrated circuits from the algorithmic level down to the transistor level. Designers should consider the latest methods in low-power technology at all design levels. Power savings can be realized by making the right choices during all abstraction levels.

In this paper, we propose a design methodology for real-time video processing to optimize power consumption and to give an ASIC-like system on FPGA. This methodology is applied on a high performance video system of 1920 × 1080 pictures at 60 frames/sec. The paper is divided as follows: the second part contains the state of the art of some methods used to reduce static and dynamic power for CMOS devices. The third part presents a real-time video processing system. The fourth part gives the suggested design methodology to end up with an ASIC-like system. The fifth part contains the experimental results and discussion and the sixth part concludes the work.

2. FPGA Power Reduction: State of the Art

Most of modern FPGA boards are computing platforms including programmable hardware element, memory resources, configurable I/O, embedded processors, and even embedded operating systems running within FPGA. These platforms allow a high level of system integration, with the ability to combine related peripheral devices and embedded processor cores with hardware functions. The challenge in giving such complete platforms with hardware and software functionalities is to permit the combination of hardware performance and software flexibility. This combination is at the expense of high power consumption, which is the major limit of FPGA platforms.

To overcome this limit, researchers continually try to find new methods to optimize power consumption. The proposed methods have explored all the abstraction levels of the design process from the system level to the circuit and the technological level. This paragraph will be dedicated for some works and methods for power reduction at various design levels.

Some works [10–12] have used HW/SW partitioning to put forward optimal systems with low-power consumption as power-aware decisions at a very early stage of the design process. HW/SW partitioning is the problem of assigning application tasks to the existing computational cores under defined constraints such as area and power. It is formalized as an optimization problem aiming to minimize an objective function under defined constraints. The authors in [10] proposed an algorithm for HW/SW partitioning to find the best tradeoff between power and latency. They modeled the application as a data flow graph and computed the latency and power consumption for every suggested partitioning. The proposed algorithm made a heuristic search for the best solution which respected the defined constraints.

The authors in [12] modeled the system using data flow graph using the bees' colony to solve the optimization problem of HW/SW partitioning under power and time constraints. The heuristic algorithms were utilized as the optimization problem was NP-Hard and the exact resolution might take a longer time. To adjust the constraints according to the user wishes, weighting coefficients were added to the constraints to specify which of the two conflicting terms would be more important for the final partitioning result.

As the HW/SW problem is a high-level method used at the system level, the metrics of the problem are computed utilizing estimations and assumptions. These estimations make the proposed solution very abstract with a low accuracy when confronted with the constraints of real world implementation.

To move deeper into the design process, other researchers have used methods at the architecture level like the dynamic voltage scaling (DVS) and dynamic frequency scaling (DFS). These methods use the dynamic adjustment of voltage and frequency at run time. They aim to save power by enabling the device regulation of its frequency and voltage based on operating conditions. These methods need deep knowledge of the working blocks and their frequencies.

The DVS, the DFS, or even Dynamic Voltage and Frequency Scaling (DVFS) was first proposed to reduce power consumption of microprocessors [15–17] and as they were successful, they were generalized and used on FPGAs [18, 19]. The authors in [15] propose a method for static timing analysis for dynamic scheduling schemes in order to use the DFS. They propose a safe timing analysis for systems with off-chip and hierarchical memories where memory latency would not scale with processor frequency. This method, called "frequency-aware," replaced the Worst Case Execution Time (WCET) obtained by static timing analysis. It expressed WCET bounds with frequency-sensitive parameters where cycles were interpreted in terms of the processor frequency and where memory access is expressed in terms of the memory latency overhead. The new suggested WCET is determined on the fly for a given frequency.

The authors of [16] addressed the problem of real-time systems with time critical applications. They put forward a new algorithm for the DFS, which is directly applied to the scheduler to modify the OS's scheduler and task management service. First, a static frequency was assigned to every task. This frequency was the lowest possible frequency that allows the scheduler to meet deadlines for a given task set.

If a task finished before the worst case specification, the frequency would be recomputed using the latest information. The new value would be utilized until the release of the task for a future invocation. Therefore, the frequency is recomputed at every new scheduling time using the real computed time for accomplished tasks and the specified worst case for others. This method provided a power reduction from 20 to 40%.

The authors of [17] proposed a DVFS technique for non-real-time applications. The main idea reposes on lowering the CPU frequency when accessing off-chip peripherals. The temporal distribution of the on-chip and the off-chip workload is computed with different scenarios to determine the CPU frequency during idle periods. The energy savings went

up to 70% with a variable performance penalty depending on the saving value.

In [18] the authors described a methodology for DVS on commercial FPGAs. The authors used a circuit to measure the logic delay, which would be used by the voltage controller block to dynamically adjust the supply voltage using a closed loop. The voltage controller was an external module implemented on a PC. The experimental results using a Xilinx Virtex XCV300E FPGA were presented and the power savings were between 4% and 54%. Although the authors said that implementing the voltage controller as an internal module is simple, giving results and statistics based on the external module is disputable since it is possible to cause delay violation or more resource utilization.

The author in [19] suggested a method for dynamic voltage scaling. The critical path is used to locate the in situ detectors, which would be utilized by the logic delay measurement circuit to identify the critical point of operation. The voltage was adjusted, and then a suitable corresponding frequency for operation is identified in a closed loop configuration. As the used Virtex-5 XC5VLX110T platform did not allow voltage scaling, the authors use a redesign of DC-to-DC module that supplies the Vccint voltage to the FPGA.

In the next paragraphs we present the proposed HW/SW architecture for real-time video platform and the methods used for optimization.

3. Hardware Software Architecture for Real-Time Video Processing

Real-time video processing plays a key role in industrial systems as it is being expanded to a lot of fields like multimedia-based electronic products as video surveillance systems and cell-phone cameras.

Real-time video processing systems consist of processing huge amounts of image data in a short time. The purpose varies from a simple display to the extraction of useful information for intelligent scene analysis. Digital videos are multidimensional signals, which make them data intensive and resource demanding as they need an important amount of resources for computations and memory operations. For example, for a full HDTV resolution of a 1920 × 1080 digital image frame with 3 bytes of pixels at 60 frames/sec, such an image contains 1920 × 1080 × 24 bits of data. Furthermore, the large dimension of digital video demands the processing of huge amounts of data of approximately 49 million pixels per second.

While designing FPGA systems for real-time video processing, the following issues have to be considered:

(i) How to propose an architecture that can fulfill the performances required by different communicating blocks: blocks in relation to the video processing systems are subjected to timing constraints for high bandwidth and data intensive operations with the need for accessing the memory at the video rate.

(ii) How to propose an architecture that can be extended when needed: the extension is the ability to add any block in relation to the video processing system in the limit of available resources.

(iii) How to get an optimal use of available resources: the problem of partitioning the system between HW/SW resources allows permitting the high performance of hardware and the flexibility of software.

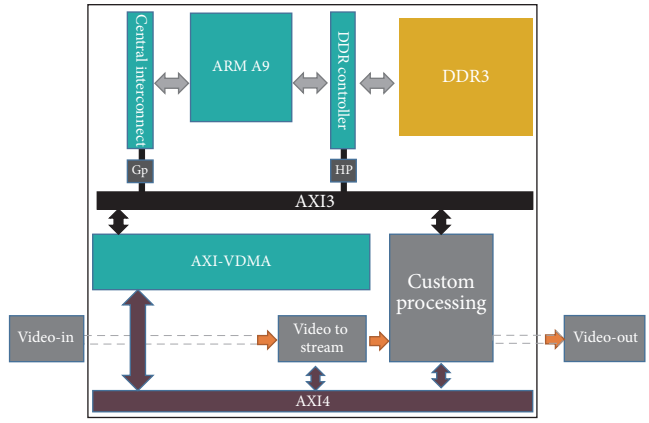


Figure 1: Block diagram of implemented design.

In previous works [20, 21], we suggested various real-time video processing architecture with a variety of power saving techniques. In [20] the authors put forward a hardware architecture for video zoom-in processing with power reduction. The proposed hardware architecture was based on the MicroBlaze 32-bit microprocessor and uses the Processor Local Bus (PLB), an ancient protocol adopted by Xilinx before moving to the Advanced eXtensible Interface (AXI) protocol. To reduce power consumption, the authors proposed a clock controller block, which would verify every input before processing. The new input would go through processing only if it differs from the previous one. Otherwise, the blocks would be deactivated. In [21], the authors explored different design methodologies at different levels of abstraction. They discussed how the choice of the design methodology has an impact on the proposed architecture. The platform-based design was proposed as a design methodology and was tested on real-time video processing system. Both works are implemented on Virtex 5 platform.

The choice of the appropriate target platform is very essential for real-time systems. For the existing FPGA platforms, many are suitable for video processing at real-time as they can work at a frequency that can exceed 150 MHz hence the ability to support HD resolutions. In addition, for the FPGA vendors, they have incorporated various hardware and software IP cores for the most important functions needed for video processing like timing generation and RGB conversions.

Figure 1 shows the block diagram for the proposed architecture implementation. The hardware or programmable logic (PL) is used to implement video-related blocks like video-in, video processing subsystems, and video-out. The software part or processing system (PS) is responsible for system-level-control registers, DMA controllers, and accelerator coherency port (ACP). The memory interface enables

the PS and blocks implemented in the PL to access the memory. The exploitation of the PS and the PL to implement video processing modules is done according to the available resources, performance constraints, power constraints, and other issues like flexibility and time to market.

Choosing the best hardware platform to implement a real-time video processing engine is challenging. In this work, the Zynq ZC702 based Xilinx evaluation kit is chosen as a target platform. It includes software, hardware, and IP components facilitating the development of custom video applications. In this design, we use the AXI protocol [22], which is a part of ARM AMBA microcontrollers that was adopted by Xilinx with Spartan 6 and future platforms.

The AXI interconnect, AXI3, and AXI Video DMA IP cores can form the basis of video systems capable of allowing handling video frame buffers and giving access to a shared DDR3 SDRAM [23–25]. The AXI Video Direct Memory Access (VDMA) core implements a video optimized direct memory access engine with frame buffering. The AXI VDMA core transfers video data to and from memory under a dynamic software control. The processor can access the PL using the AXI3 master General Purpose (GP) interfaces. The AXI_HP interface allows the PL to access the DDR memory through high bandwidth data path bus masters.

The PS is used to initialize blocks and master memory access. The custom processing block presents any kind of processing that can be added to the video system. This processing can be a low-level treatment, which makes changes on the video directly, or a high-level treatment which aims to extract semantic information. This block represents the extensible part of the video system.

The video-out block is responsible for displaying the video on a monitor.

The next paragraph will present the used methods to optimize power consumption on the proposed architecture.

4. Real-Time Video Processing System Design for Power Optimization

As demonstrated in the introduction, the power consumption is a key factor in the evaluation of embedded systems. A challenging idea for researchers as well as for system designers is to propose architecture with minimal power consumption. The design of an ASIC-like system needs tailored power optimization decisions that will verify the system requirements and performance. The design process has to start by defining different application requirements. Figure 2 shows the steps followed during the design process in order to propose an ASIC-like system. The figure shows a typical FPGA design process and the interaction with the power-aware decisions. At the specification step, the device is selected and the system requirements are well defined. A thermal power and supply current allocation are done at this step. After the proposition of the architecture and the definition of the hardware and the software tasks, the designer adds the appropriate power-aware decisions. These decisions can be tool and user methods. The tool methods are high-level decisions available within the design tool and can be selected and added by user as optimization decisions. When

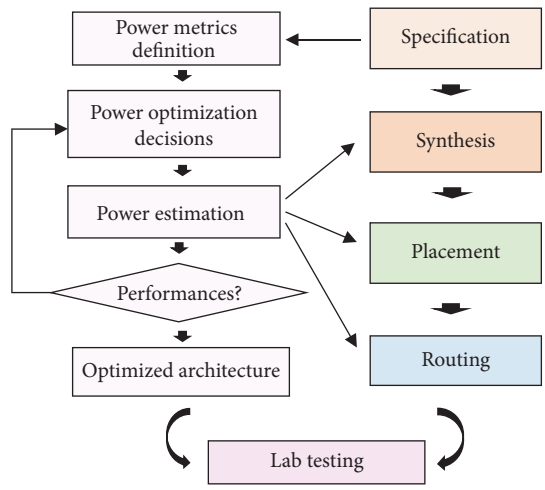


Figure 2: Power in the FPGA design process.

chosen, these decisions will be added automatically to the design during the synthesis and implementation step. The user methods are optimization strategies that are suggested and implemented by the user and which can be application specific or generalized methods. The power-aware decisions will be followed by a test and verification to check that the system performance is not violated. This verification uses the physical information, which is added to the dimension of the problem after the synthesis, placement, and routing steps. At these stages, a more accurate estimation for power can be added, and so the designer can change the decision if the budget exceeded or performance is violated.

The validation of the power-aware methods is based on power estimations at every design stage. Xilinx provides two different tools for power estimation, which are Vivado Power Analysis (VPA) and Xilinx Power Estimator (XPE). XPE is a power estimation tool used in the predesign and preimplementation phases. Based on architecture and device, it helps select power supply and to specify I/O loading, design resource usage, and activity rates. The VPA allows performing power estimation at all stages of the design flow: synthesis, placement, and routing. It is more accurate at the postroute since it can read from the implemented design database the exact logic and routing resources used.

4.1. Tool Optimization Method. The FPGA market is almost dominated with two top FPGA companies, which are Xilinx and Altera. Every company provides its specific platforms and specific design tools used for these platforms. As a result, the tool optimizations are platform-specific and tool-specific. They are applied on specific details of the design. In this work, we target the Xilinx FPGA and hence we use the Vivado design tool. The Vivado Integrated Design Environment (IDE) is a tool that offers many options to help designers during the design process

The available optimization options are as follows:

(i) Intelligent clock gating: this method is used to reduce dynamic power consumption through turning off the clock. There is a general clock gating or specific block

clock gating. General clock gating is provided by Vivado IDE to automatically reduce the switching factor α defined in (1). This option can be selected whether during the preplacement savings or the postplacement savings.

(ii) BRAM savings: they are a configurable memory module compatible with a variety of BRAM interface controllers. The Vivado IDE offers power optimization for available Xilinx 7 series devices for BRAMs in true dual port mode. The writing mode can be changed safely for both reading and writing operations if its output is not connected or not needed during writing or reading operations. These optimizations will be performed as long as they do not affect user functionality and performance. This option can be explicitly enabled and disabled during the preplacement savings. The designer can exclude a part of the design from the specified optimizations when it is necessary to protect a defined critical path that may be changed after power savings.

(iii) Standby mode: this option puts the CPU in the standby mode and starts it up when an event or interrupt is detected. In this mode of operation, the device is powered up, but the majority of its clocks are gated off. This puts the processor in a static state where the only consumption will be caused by leakage currents and some logic which are responsible for the detection of wakeup conditions.

4.2. User Methods for Power Optimization. User methods are all the methods that can be added to optimize the power consumption of the whole system. Some of the famous methods used are voltage and frequency scaling.

The DVS or the DFS is the process of varying the voltage or frequency for a target processor voltage domain and frequency domain. As the voltage and frequency are directly related to power consumption (1), reducing the voltage permits a quadric reduction while reducing frequency allows a linear reduction of power consumption.

To use the DVS and the DFS on a target FPGA, the following definitions have to be considered [26]:

(i) The *process strength* of a device is defined as the ability and degree of variation in the attributes of integrated transistors of a device. There are three classes, weak, nominal, and strong. A weak device is able to operate with the lowest acceptable frequency at nominal voltages. Strong devices can run at faster frequencies than required at nominal voltages and can function at voltages lower than nominal voltage values at the minimum specified frequency.

(ii) The *voltage domain* is defined as the group of modules sharing the same power supply voltage for the core logic of each device.

(iii) The *operating performance point* is the voltage required for every device to be able to operate at a desired processor clocking frequency. For example, for the processing system DDR I/O supply voltage, the minimum value is 1.14 v and the maximum is 1.89 v [27]. This information can be found in the technical report of the target FPGA device.

(iv) The *critical path* of the system is defined as the longest path for achieving the execution of a program from the source to the sink of a dataflow graph. This information is used to track the critical tasks when scaling the voltage and the frequency to ensure performance of the system.

```
Inputs: V_min_x, V_op_x, st;
Begin
Identify voltage bounds for the target block;
Do
   V_op = V_op − st;
   Test performance;
   if test failed
       V_op = V_op − st;
       Break;
   End if
While (V_op > V_min_x)
End
```

Algorithm 1: Voltage scaling algorithm.

4.2.1. Dynamic Voltage Scaling. In this paper, the DVS is used for power saving. Let $V_{\min\_x}$ be the minimum voltage required for the block x to operate in normal conditions and under which the system fails. These values are characteristics fixed at the manufacturing of the device. Let V_{op_x} be the operating voltage of the block x. The algorithm used to scale the voltage is as in Algorithm 1, where st is a float representing the step of incrementing and decrementing the voltage.

In the ZC702 board, the power is supplied to the components through a number of independent rails using programmable power regulators (UCD9248) and Power Management Bus (PMBus) compliant system controller from Texas Instruments. The PMBus is an open standard protocol that defines communication with power converters. The PMBus allows writing and reading power, current, and voltage information. The processors can access the PMBus using 1-to-8 I2C switch present on the board. The Zynq ZC7020 device is divided into several power domains. These power domains are generated by 6 regulators which generate different voltages required by the Zynq and the onboard components. The supplied voltage and currents are continuously measured and monitored by three (UCD9248) power controllers available on the ZC702 board.

In this work, scaling the voltage of hard blocks is done using the PMBus protocol, which simplifies the communication with power converters in a power system. It enables software or hardware in the device to access to a manageable power supply [28–30].

The optimal operating voltages are computed using algorithm in Algorithm 1. Then the system is configured only one time with these values at every start of the system. The configuration and the implementation of the proposed

algorithm are done by the processor through soft code. More details about the scaled blocks will be found in experimental results.

4.2.2. Frequency Scaling. Frequency scaling is the ability to change the frequency of a block without degrading the performance of the system.

As we target a defined reconfigurable platform, which is the Zynq, the analysis of technical details about the clocking system is essential to determine to which level the scaling is possible.

In this work, we test two methods for frequency scaling: scaling the frequency of the processor, which we call the DVS, and scaling the frequency of the blocks implemented on the PL, which we call adaptive frequency scaling (AFS). Whether applying the DVS or the AFS, an analysis of different execution paths is essential to ensure the nonviolation of the system performance.

Let $O = \{o_1, o_2, \ldots, o_n\}$ be a set of n tasks defining an application A. The meaning of task varies according to the chosen granularity; it can be an instruction, a portion of code, or a block. Let G be a Control Data Flow Graph (CDFG). G is defined as $G(E, V)$ with

$$V = \{v_1; v_2; \ldots; v_n\}$$
$$E = \{e_{ij} \mid 1 < i, j < n\}. \tag{2}$$

$G(E, V)$ is an oriented graph that describes the dependencies between the tasks o_i of an application, V is the set of nodes, and E is a set of edges.

Every task o_i is mapped to a node v_i. e_{ij} is the edge linking task o_i to task o_j.

Let $S = \{s_1, s_2, \ldots, s_m\}$ be a set of paths from the source to the sink, $s_i = \{v_1 - v_j - v_k \cdots - v_n\}$, $1 < i < m$. Every path starts with a source node $v_1 \in V$ and ends with a sink node $v_n \in V$.

Every task o_i has a Best Case Execution Time (BCET) and Worst Case Execution Time (WCET). The WCET of a program is the WCET of the longest path from the source to the sink, WCET $= \{\max \mathrm{wcet}_i / 1 < i < m\}$, and the BCET of a an application is the BCET of the longest path, BCET $= \{\max \mathrm{bcet}_i / 1 < i < m\}$.

BCET is when everything goes as expected without exceptions and WCET is due to unexpected scenarios like resource occupation, memory refreshment, or pipeline hazards.

Using timing analysis and the computation of WCET helps define the bounds of the execution time of a task on a particular hardware or software platform.

There exist two methods to perform WCET analysis [31]:

(i) Static methods: they take the task code with some annotations, analyze the set of possible paths, and combine control flow with a model of the target architecture to obtain upper bounds for this combination.

(ii) Measurement-based methods: they execute the task on the target hardware or a simulator for a vector of inputs.

In this work we use a measurement-based method to determine the bounds of the system. The computed WCETs are used as input information for the scaling algorithm.

As we target a defined reconfigurable platform which is the Zynq, the analysis of technical details about the clocking system is essential to determine to which level the scaling is possible.

The 7 series FPGAs clocking resources [32] manage clocking using clock management tiles (CMT) which provide clock frequency deskew, synthesis, and jitter filtering. Each CMT consists of one phase-locked loop (PLL) and one mixed-mode clock manager (MMCM). The PLLs and MMCMs used to synthesize frequencies with a wide range. The PS clocks are derived from one of three programmable PLLs. These 3 PLLs drive the clocks of the CPU, DDR, and I/O. Each of these PLLs is loosely associated with the clocks in the CPU, DDR, and peripheral subsystems.

The PL has its independent clock management generation and distribution structures. It also receives four clock signals from the PS side. These signals are completely asynchronous to each other and do not have a relation with other PL clocks. The communication between PL and PS side is possible through functional interfaces like AXI interfaces, interrupts, and clocks and through configuration signals [27]. The Dynamic Configuration Port (DRP) allows software to configure hardware using a DRP interface, which is an AXI4-Lite interface. The DRP also enables generating custom clocks on the fly through a dynamic partial reconfiguration of PLL or MMCM modules. The relation between the input frequency and the output is outlined using

$$\mathrm{Fclkout} = \mathrm{Fclkin} * \frac{D}{M} \tag{3}$$

with D, M being two programmable frequency dividers by configuration through DRP.

The DFS is performed at run time with frequencies that conform to the used processor mode, whereas the AFS is done according to the data input size. As the target is a real-time video processing system, the input varies according to the size of the video. The operating frequency of the PL blocks varies also with the video size: hence, a HD video of 1360/738 does not require the same processing frequency as full HD video of 1920/1080.

The AFS allows the system to operate under the minimum frequency needed to guarantee the nonviolation of performance.

5. Experimental Results

The proposed architecture is prototyped on the Zynq ZC702 evaluation board. The ZC702 provides a hardware environment for developing and evaluating designs targeting the Zynq XC7Z020 device. It includes 1 GB DDR3 component memory, 128 Mb Quad SPI flash memory, USB 2.0, Secure Digital (SD) connector, a HDMI codec, I2C bus, and 2 UART interfaces.

This paragraph is dedicated to give experimental results and show the benefits of the proposed design flow in power

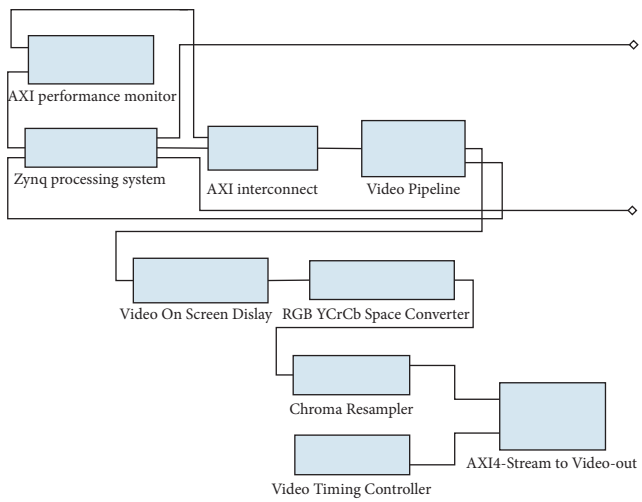


Figure 3: Simplified block diagram of proposed system.

reduction. A performance comparison is also held to verify the performance of the whole system.

5.1. Hardware and Software Architecture. The design of the target system is carried out using the Vivado Design Suite tool. It is utilized to design, integrate, and implement with the Zynq-7000 All Programmable, Xilinx 7 series, and UltraScale devices. Working with this design suite, the design implementation can be accelerated with place and route tools that analytically optimize multiple and concurrent design metrics, such as timing and power. The Vivado gives the ability to analyze the design at each design stage, which allows earlier modifications in the design processes. It provides also timing and power estimations after synthesis, placement, or routing.

Figure 3 illustrates a simplified block diagram of the hardware implementation of the video processing architecture. This figure is limited to interface connection between blocks. It contains the Zynq7 processing system, a video pipeline, AXI interconnects, AXI performance monitor, and video-related blocks.

(i) The processing system is used to initialize blocks and master memory access. The frequency and voltage scaling modules are also implemented on the processing system.

(ii) The AXI interconnects are responsible for handling information to and from the processing system.

(iii) The AXI performance monitor is responsible for different statistics with the AXI protocol interfaces. It is used for transactions, external system events, and performance measurement for AXI4-, AXI3-, and AXI4-Stream and AXI4-Lite interfaces. It captures real-time performance metrics for the throughput and the latency. In this work we used the performance monitor IP to perform real-time profiling using the SDK.

(iv) The video pipeline contains the Video Direct Memory Access (VDMA), which is a soft IP core that provides high bandwidth for direct access to the memory using AXI4-Stream-video peripherals. The AXI4-Lite slave interface is used to perform initialization, registers, and status.

(v) The video-related blocks are standard blocks used in a video chain and transfer the video stream after making the necessary conversions. The RGB to YCrCb converts the design pixels to from RGB used by the AXI4-Stream to the 16-bit YUV 4 : 4 : 4 signal format. The Chroma Resampler block takes the output of the RGB to YCrCb and converts it to YUV 4 : 2 : 2 which is the format required by HDMI output.

Table 1 gives the resource utilization of the proposed architecture and some blocks.

The software part running on the PS is built using the Xilinx Software Development Kit (SDK). The SDK is an embedded environment to create and test software platforms and applications targeting Xilinx embedded processors. This software environment works with hardware designs created with Vivado. In this work, the software part running on the ARM processor with standalone operating system has the role controlling hardware.

5.2. Tool Optimizations Results. Figures 4(a) and 4(b) show, respectively, the power consumption and the resource utilization for various optimization approaches using tool optimizations. The proposed groups of values are named, respectively, as follows:

(i) No optimization (no-opt) for synthesis and implementation without any additional optimization

(ii) Preplacement power gating (pre-opt), which is a power optimization method added by the tool and performed after the placement step

(iii) Postplacement clock gating (post-opt), which is a power optimization added by the tool after the placement step

(iv) Low DDR mode (low-ddr).

The maximum of power savings is obtained with a low DDR mode. The tool optimizations allow power savings up to 7%.

5.3. Voltage Scaling Results. The voltage scaling method is implemented as a software code running on the processor. The communication with voltage rails is done using the I2C bus. To avoid the unnecessary additional resources, the optimal values are computed using excessive test scenarios. This analysis defines the optimal values that will be used by the system without need to collect them at run time using additional resources.

The DVS method requires the knowledge of the voltage bounds of different blocks. Table 2 [27] shows the maximum and minimum of recommended operating voltage values for some blocks of the ZC702 device.

Although the voltage value can reach more inferior values than those indicated in the safety bounds (e.g., the V_{ccpint} minimal value can reach 0.5 v) [27], the safety of a functioning of the device outside the indicated bounds is not

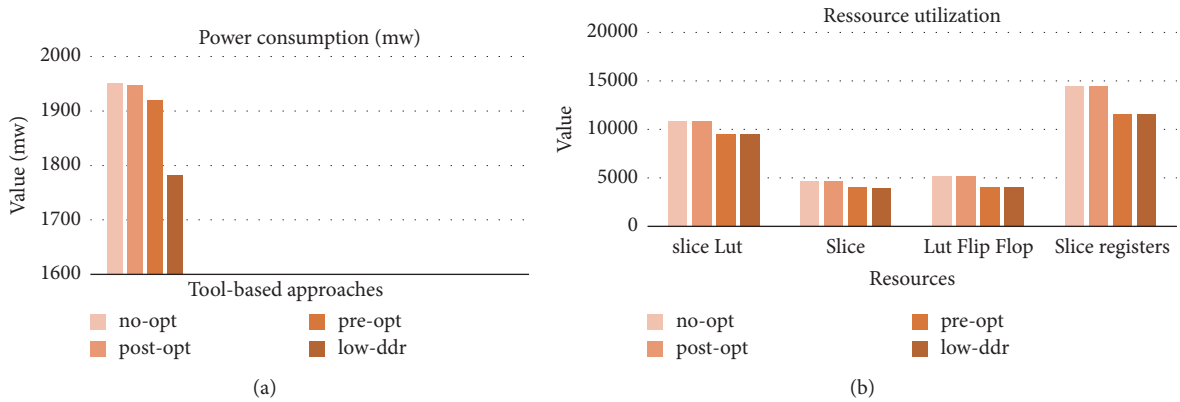


Figure 4: Resource utilization for different saving approaches.

Table 1: Resource utilization.

	Slice LUTs	Slice registers	Slice	Lut Flip Flop	Block RAM	DSPs
The total system	10823	14524	4689	5121	10.5	19
AXI performance Monitor	2787	3504	1135	1079	0	0
Clock reset	19	40	14	15	0	0
Processing system	538	655	226	284	0	0
Video-out	239	355	126	103	1	0
RGB to YCrCb	365	335	144	163	0	4
Video Timing Controller	131	212	78	39	0	0
Video pipe (VDMA)	5886	8371	2514	3042	9.5	12

Table 2: Voltage recommended values for some blocks of the target Zynq ZC702 device.

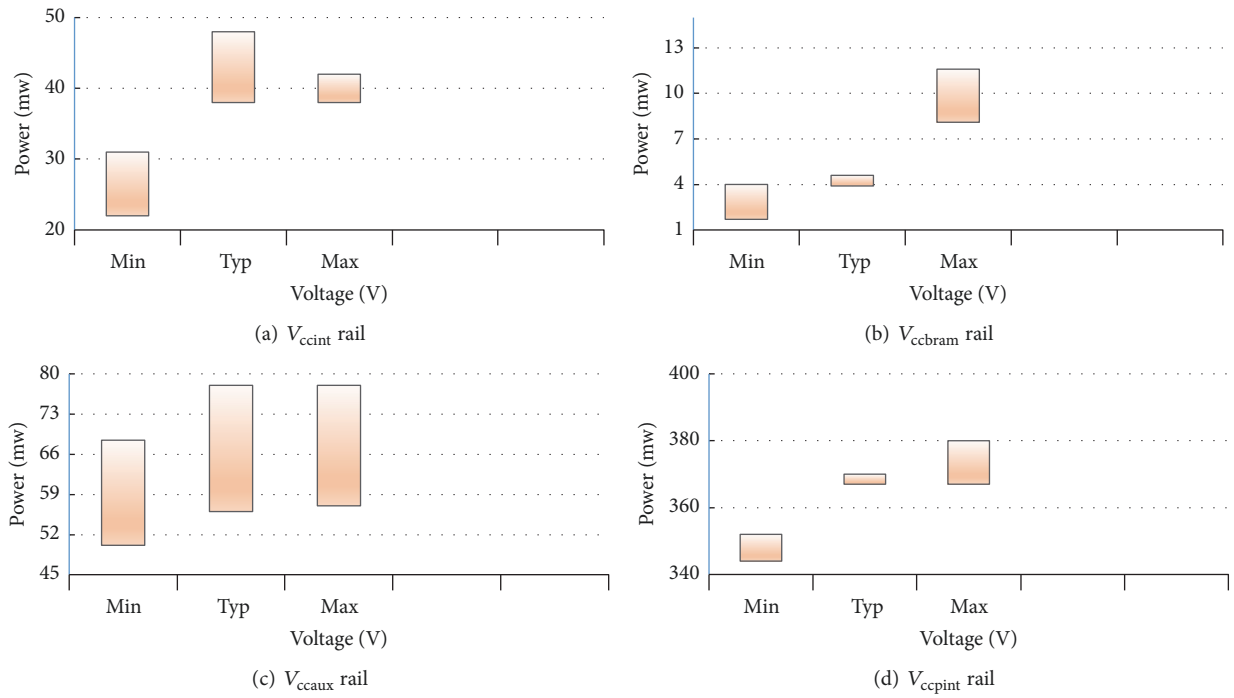
Blocks	Description	Minimum operating voltage (v)	Typical operating voltage (v)	Maximum operating voltage (v)
v_{ccpint}	PS internal supply voltage	0.95	1	1.05
v_{ccint}	PL internal supply voltage	0.95	1	1.05
v_{ccpaux}	PS auxiliary supply voltage	1.71	1.80	1.89
v_{ccaux}	PL auxiliary supply voltage	1.71	1.80	1.89
v_{ccbram}	PL block RAM supply voltage	0.95	1.00	1.05

tested. The technical report [27] indicates that an exposure to maximum values for extended periods of time might affect device reliability.

Figure 5 highlights the voltage scaling results of some V_{cc} rails. Each figure indicates the power margin for every chosen value. For example, for the V_{ccint} rail, when the minimum operating value is chosen (0.95) the power consumption of this block varies between 22 and 31 mw.

5.4. Frequency Scaling and Design Performances. The frequency scaling varies according to the target architecture. There is the frequency of the PL blocks and the frequency of the PS. Defining the scaling values needs an analysis of the functioning of the system. To do so, we use the System Performance Analysis (SPA) toolbox available under the SDK development tool. This toolbox allows exploring the performances of hardware and software at run time. The observation of the system performance at critical stages allows taking the right optimization decisions in order to refine the system performances without degradation.

Figure 6 illustrates the CPU utilization rate. We use only one core in the proposed architecture. The graph shows that the utilization rate is between 91% and 100%. This information shows that applying DFS on a processor executing a real-time video system does not allow for a noticeable optimization as the CPU utilization is usually at its maximum values.



(a) V_{ccint} rail

(b) V_{ccbram} rail

(c) V_{ccaux} rail

(d) V_{ccpint} rail

Figure 5: Voltage scaling results on different rails.

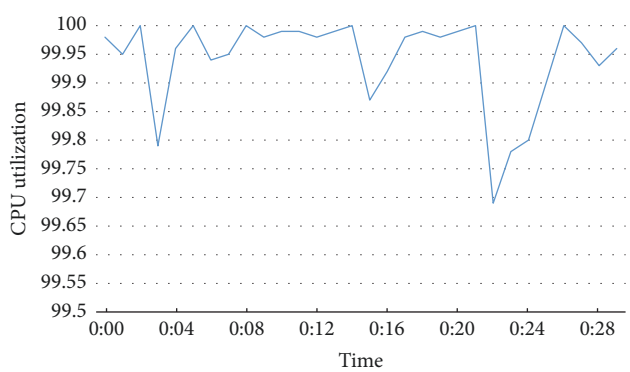


Figure 6: CPU utilization.

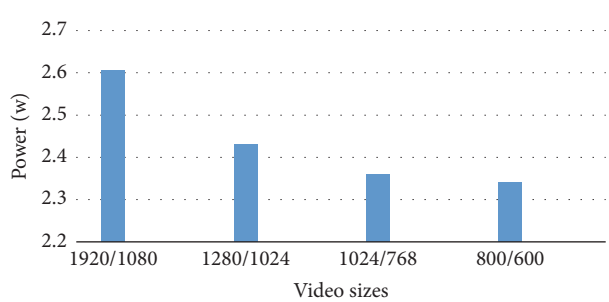


Figure 7: AFS power results.

Adaptively, we propose the AFS, which allows scaling frequency according to the input size.

Figure 7 shows the result of AFS on different video input sizes. Power savings are up to 12%.

5.5. Comparison with Other Works. Voltage and frequency scaling, for both hard and soft parts for commercial FPGAs, are implemented by several researchers using different methods. In this part, we compare our work with two existing works, which are the main works in the literature that focus on frequency and voltage for commercial 28 nm FPGAs. The authors used voltage and frequency scaling [13] and added logic scaling in [14] to optimize power consumption. The voltage scaling and frequency scaling were performed through two separate units for each of them. In [13], the DVS unit was composed of a MicroBlaze processor, a Dual Port RAM, and an I2C IP core. It allows accessing the configuration and monitoring the PMBus power rails. The control and record of power and voltage values were done at run time. The MicroBlaze received the power and the voltage values and communicated with the PMBUS via the I2C to write new values. The dialog between the MicroBlaze and the power rails was done through commands written in C.

The frequency scaling unit utilized a PicoBlaze™ 8-bit microcontroller. Frequency scaling was made through the communication with the off-chip Silicon Labs Si570, which was an oscillator programmable at run time to scale the frequency to the wanted values. The communication was done through I2C protocol.

In [14] the authors used voltage, frequency, and logic scaling to optimize power consumption. The voltage scaling unit is identical to the one used in [13]. The frequency scaling unit utilized a ROM containing the configuration parameters used by the MMCM to generate the clock for the user logic. The frequency would be decreased continually at run time and stopped when a value causing timing violations is detected.

Table 3: Comparison with different works.

	Additional resources	Frequency scaling method	Voltage scaling method	Test method	Power achievements
Work 1 [13]	MicroBlaze PicoBlaze I2C IP core Dual Port RAM	I2C communication with Si570 oscilator	I2C communication with PMBus	Random functions with different sizes	Up to 64.98% (using voltage and frequency scaling)
Work 2 [14]	MicroBlaze I2C IP core Dual Port RAM ROM	Configuration of MMCM ROM	I2C communication with PMBus	Random functions with different sizes	Up to 60% (using frequency, voltage, and logic scaling)
Our work	I2C IP core	Soft configuration of the MMCM	Soft configuration of PMBus	Real-time video application With real-time constraints	Up to 60% (using frequency and voltage scaling and tool optimizations)

Compared to the previous cited works, we utilize adaptive frequency and voltage scaling as user methods in addition to the tool ones. The two previous works collected information such as the frequency and voltage of functioning blocks at run time and scale them accordingly. Collecting information at run time would result in extra resources and additional complexity in the algorithms used to manage the information. The two works applied the proposed methods on test modules, which they called Power Consuming and Speed Testing Modules (PCASTMs). These PCASTMs were proposed with various numbers of modules to occupy different percentages of the device. The proposed tests were tasks without real-time requirements. In addition, for the voltage scaling, they tested voltage values under the recommended values available in the datasheet. Testing with values under the minimum recommended could give more power savings up to 70% [33] but with doubtful safety for long term functioning with real-time constraints.

In our work, the idea is simple: as the different execution scenarios are known in advance, the collection of the information is done at the design level using tests and timing analysis to determine the optimal operating points. The proposed method necessitates only I2C IP core and its implementation presents a low complexity. In addition, scaling the voltage many times at run time is not useful in our opinion as it leads to power and heat dissipation while accessing the power rails. Configuring the device blocks with the optimal voltage at the start is more efficient. Table 3 gives a summary about the three works.

6. Conclusion

Driven by the complexity of the SOC design and the big concurrence between SOC vendors, the design of an optimized system is very challenging. With the increasing consumers' demands and the saturation that faces Moore's law, the power consumption has become a struggle. Designers have to find new solutions for the power consumption problem. In this work, we have proposed a design methodology that brings together tool and user optimizations. These methods have been used to design a real-time video processing system. The work has been implemented on Zynq ZC702 board with ARM 9 target processor. The tool methods have been specific to the design tool and work with a target platform. The user methods have been general and can be applied on other real-time video processing systems. The adaptive frequency scaling and the dynamic voltage scaling have been controlled by the ARM processor. Compared to the existing works, this methodology has allowed up to 60% of power savings with minimal additional resources. Future work will be on the application of other methods like clock gating and its integration into the design of real-time video systems.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] M. Genovese and E. Napoli, "ASIC and FPGA implementation of the gaussian mixture model algorithm for real-time segmentation of high definition video," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 3, pp. 537–547, 2014.

[2] M. Happe, E. Lübbers, and M. Platzner, "A self-adaptive heterogeneous multi-core architecture for embedded real-time video object tracking," *Journal of Real-Time Image Processing*, vol. 8, no. 1, pp. 95–110, 2013.

[3] J. Wang, S. Zhong, L. Yan, and Z. Cao, "An embedded system-on-chip architecture for real-time visual detection and matching," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 3, pp. 525–538, 2014.

[4] W. Wang, J. Yan, N. Xu, Y. Wang, and F.-H. Hsu, "Real-Time High-Quality Stereo Vision System in FPGA," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 10, pp. 1696–1708, 2015.

[5] J. Fowers, G. Brown, P. Cooke, and G. Stitt, "A performance and energy comparison of FPGAs, GPUs, and multicores for sliding-window applications," in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA '12)*, pp. 47–56, ACM, Monterey, Calif, USA, February 2012.

[6] K. Pauwels, M. Tomasi, J. Díaz Alonso, E. Ros, and M. M. Van Hulle, "A Comparison of FPGA and GPU for real-time

phase-based optical flow, stereo, and local image features," *IEEE Transactions on Computers*, vol. 61, no. 7, pp. 999–1012, 2012.

[7] M. Baklouti, Y. Aydi, P. Marquet, J. L. Dekeyser, and M. Abid, "Scalable mpNoC for massively parallel systems-Design and implementation on FPGA," *Journal of Systems Architecture*, vol. 56, no. 7, pp. 278–292, 2010.

[8] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 2, pp. 203–215, 2007.

[9] Altera, *40-nm FPGA Power Management and Advantages*, Altera Corporation, 2008.

[10] S. Ben Haj Hassine, M. Jemai, and B. Ouni, "Power and execution time optimization through hardware software partitioning algorithm for core based embedded system," *Journal of Optimization*, vol. 2017, Article ID 8624021, 11 pages, 2017.

[11] H. Han, W. Liu, W. Jigang, and G. Jiang, "Efficient algorithm for hardware/software partitioning and scheduling on MPSoC," *Journal of Computers (Finland)*, vol. 8, no. 1, pp. 61–68, 2013.

[12] L. Kechiche, L. Touil, and B. Ouni, "SOPC for real time multi-video treatments with QoS requirements," *International Journal of Advanced Intelligence Paradigms*, 2017.

[13] A. F. Beldachi and J. L. Nunez-Yanez, "Run-time power and performance scaling in 28 nm FPGAs," *IET Computers & Digital Techniques*, vol. 8, no. 4, pp. 178–186, 2014.

[14] J. Luis Nunez-Yanez, M. Hosseinabady, and A. Beldachi, "Energy Optimization in Commercial FPGAs with Voltage, Frequency and Logic Scaling," *IEEE Transactions on Computers*, vol. 65, no. 5, pp. 1484–1493, 2016.

[15] K. Seth, A. Anantaraman, F. Mueller, and E. Rotenberg, "FAST: frequency-aware static timing analysis," in *Proceedings of the 24th IEEE International Real-Time Systems Symposium*, pp. 40–51, Cancun, Mexico.

[16] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low-power," in *Proceedings of the eighteenth ACM symposium on Operating systems principles*, pp. 89–102, Alberta, Canada, October 2001.

[17] K. Choi, R. Soma, and M. Pedram, "Fine-grained dynamic voltage and frequency scaling for precise energy and performance tradeoff based on the ratio of off-chip access to on-chip computation times," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 1, pp. 18–28, 2005.

[18] C. T. Chow, L. S. M. Tsui, P. H. W. Leong, W. Luk, and S. J. E. Wilton, "Dynamic voltage scaling for commercial FPGAs," in *Proceedings of the 2005 IEEE International Conference on Field Programmable Technology*, pp. 173–180, Singapore, December 2005.

[19] J. L. Nunez-Yanez, "Adaptive voltage scaling with in-situ detectors in commercial FPGAs," *IEEE Transactions on Computers*, vol. 64, no. 1, pp. 45–53, 2015.

[20] L. Touil, L. Kechiche, and B. Ouni, "Design of low power system on programmable chip for video zoom-in processing," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 24, no. 5, pp. 3405–3418, 2016.

[21] L. Kechiche, L. Touil, and B. Ouni, "Real-time image and video processing: Method and architecture," in *Proceedings of the 2nd International Conference on Advanced Technologies for Signal and Image Processing, ATSIP 2016*, pp. 194–199, Tunisia, March 2016.

[22] XILINX, *AXI Reference Guide UG761 (v13.4)*, XILINX, 2012.

[23] XILINX, *ZC702 Evaluation Board for the Zynq-7000 XC7Z020 all programmable SoC, UG850*, Xilinx, 2015.

[24] XILINX, *Zynq- 7000 all Programmable SoC technical reference manual, UG 585*, XILINX, 2016.

[25] J. Lucero and B. Slous, *Designing High-Performance Video Systems with the Zynq-7000 All Programmable SoC Using IP Integrator, XAPP1205*, XILINX, 2014.

[26] F. Dehmelt, *Adaptive (Dynamic) Voltage (Frequency) Scaling—Motivation and Implementation*, Texas Instruments, 2014.

[27] XILINX, "Zynq-7000 all programmable SoC (Z-7007S, Z-7012S, Z-7014S, Z-7010, Z-7015, and Z-7020)," in *DC and AC Switching Characteristics*, XILINx INC, 2017.

[28] A. F. Beldachi and J. L. Nunez-Yanez, "Accurate power control and monitoring in ZYNQ boards," in *Proceedings of the 24th International Conference on Field Programmable Logic and Applications, FPL 2014*, Germany, September 2014.

[29] J. Nunez-Yanez, "Adaptive voltage scaling in a heterogeneous FPGA device with memory and logic in-situ detectors," *Microprocessors and Microsystems*, vol. 51, pp. 227–238, 2017.

[30] Xilinx, *Zynq 7000 All Programmable SOC PCB Design Guide (UG933)*, Xilinx, 2016.

[31] R. Wilhelm, J. Engblom, A. Ermedahl et al., "The worst-case execution-time problem-overview of methods and survey of tools," *ACM Transactions on Embedded Computing Systems*, vol. 7, no. 3, article no. 36, 2008.

[32] Xilinx, *7 Series FPGAs Clocking Ressources user guide (UG 472)*, Xilinx, 2017.

[33] M. Hosseinabady and J. L. Nunez-Yanez, "Run-time power gating in hybrid ARM-FPGA devices," in *Proceedings of the 24th International Conference on Field Programmable Logic and Applications (FPL'14)*, 6, 1 pages, Germany, September 2014.