Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2009, Article ID 273791, 11 pages
doi:10.1155/2009/273791

Research Article

An Interface for a Decentralized 2D Reconfiguration on Xilinx Virtex-FPGAs for Organic Computing

Christian Schuck, Bastian Haetzer, and Jürgen Becker

Institut für Technik der Informationsverarbeitung (ITIV), Universität Karlsruhe (TH), Vincenz-Prießnitz-Straße 1, 76131 Karlsruhe, Germany

Correspondence should be addressed to Christian Schuck, schuck@itiv.uni-karlsruhe.de

Received 31 December 2008; Accepted 3 August 2009

Recommended by Peter Zipf

Partial and dynamic online reconfiguration of Field Programmable Gate Arrays (FPGAs) is a promising approach to design high adaptive systems with lower power consumption, higher task specific performance, and even build-in fault tolerance. Different techniques and tool flows have been successfully developed. One of them, the two-dimensional partial reconfiguration, based on the Readback-Modify-Writeback method implemented on Xilinx Virtex devices, makes them ideally suited to be used as a hardware platform in future organic computing systems, where a highly adaptive hardware is necessary. In turn, decentralisation, the key property of an organic computing system, is in contradiction with the central nature of the FPGAs configuration port. Therefore, this paper presents an approach that connects the single ICAP port to a network on chip (NoC) to provide access for all clients of the network. Through this a virtual decentralisation of the ICAP is achieved. Further true 2-dimensional partial reconfiguration is raised to a higher level of abstraction through a lightweight Readback-Modify-Writeback hardware module with different configuration and addressing modes. Results show that configuration data as well as reconfiguration times could be significantly reduced.

1. Introduction

Modern VLSI circuits are getting more and more complex as CMOS technology scales down into deep submicron (DSM) domains. In near future it will be possible to integrate hundreds of predesigned IP-cores, such as general-purpose processors, DSP, reconfigurable arrays, memory subsystems, and so forth, on a single chip. State-of-the-art FPGAs, such as the Xilinx Virtex series, are growing in complexity almost in the same order of magnitude, opening new promising possibilities for future systems, but at the same time the resulting complexity of such systems leads to big challenges for both the system designer, system programmer, as well as the user of the device. Reliability, especially MTBF, power consumption, controllability, and programmability are the main drawbacks.

Therefore, a new paradigm of system design is necessary to efficiently utilize the available processing power of future chip generations. To address this issue in [1] the *Digital on Demand Computing Organism* (DodOrg), which is derived from a biological organism, was proposed. Decentralisation of all system instances is the key feature to reach the desired goals of self-organisation, self-adoption and self-healing, in short the self-x features. Hence, the hardware architecture of the DodOrg system consists of many heterogeneous cells, so-called *Organic Processing Cells* (OPCs), as shown in Figure 1. They provide services for computation, memory, IO, and monitoring. A completely decentralized acting middleware layer [2] is able to dynamically assign upcoming tasks to the OPCs and group OPCs together to form virtual organs that are working closely together to solve a given task. All cells are connected by the "*artNoC*" [3] router network, which efficiently fulfils the special communication requirements of the organic system.

In general, all OPCs are made of the same blueprint, see Figure 2. On the one side they contain the cell specific functionality and on the other side they contain several common units, that are responsible for the organic behaviour of the architecture.

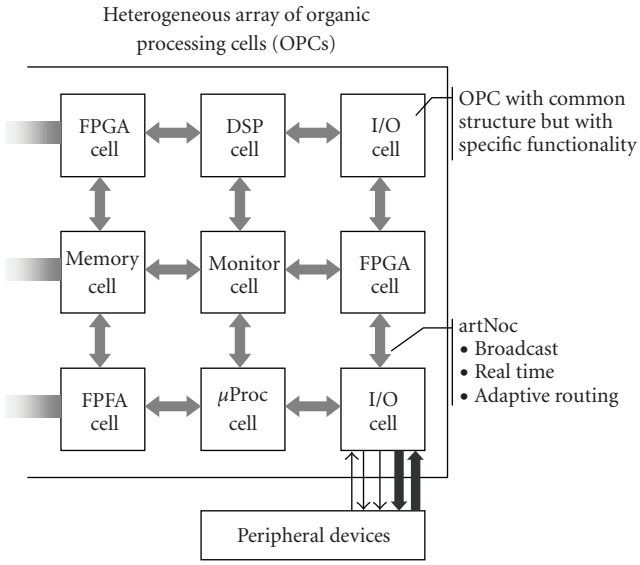

Figure 1: DodOrg organic hardware architecture.

Most OPCs of the array contain a highly adaptive datapath. Possible scenarios for the datapath range from μProc-cells, DSP-cells, coarse grained datapath units (DPUs), to completely flexible FPGA-like structures. This means, that some cells come prepacked with fixed functionality and others are more flexible. Therefore, it is possible to adapt the datapath during runtime to the needs of the current application tasks.

Within every OPC a *Configuration Management Unit* is responsible for the configuration of a single cell only and serves as a hardware protocol handler for the configuration exchange protocol with other OPCs.

During the distributed reconfiguration process, responsibilities might be shifted from one cell to another and configuration, as well as application data is exchanged. Each OPC forms its own local clock island, which is controlled by a *Clock and Power-Management Unit,* by applying dynamic frequency scaling (DFS). Therefore, it can control and adjust performance and power consumption of the cells datapath according to the actual computational demands of the application and the critical path accordingly. DFS has a high potential, as it decreases the dynamic power consumption by decreasing the switching activity of flip flops, gates in the fan-out of flip flops and the clock trees.

Xilinx Virtex-FPGAs offer the possibility of dynamic and partial online reconfiguration through an *internal reconfiguration access port* (ICAP). Therefore, they are ideally suited to serve as a test bed for the proposed organic cell based hardware architecture. However, three main requirements must be met the following:

(1) The OPC internal *Configuration Management Unit* needs to have access to the ICAP, in-order to realize a true decentralized reconfiguration of the cells datapath.

(2) For an effective floor plan, a fast two-dimensional reconfiguration must be possible.

(3) The FPGA area must be divided into clock islands that can be controlled by the *Clock and Power Management Unit*.

Therefore, in this paper we present the *artNoC-ICAP-Interface,* a lightweight hardware module that builds a bridge between the "*artNoC*" network and the ICAP. It is able to execute the *Readback-Modify-Writeback* (RMW) method in hardware, to fulfil the beforehand mentioned requirements. In the following sections the basics of Virtex-FPGAs and the RMW method are reviewed in brief and related work in the field of partial and dynamic reconfiguration and DFS on FPGAs is summarized. The rest of the paper is structured as follows: Section 4 introduces the basic architecture and the features of the implemented *artNoC-ICAP-Interface*. In Section 5 an overview of the DodOrg system architecture on Virtex-FPGAs is given, while Section 6 gives implementation results and performance and power figures. Finally, Section 7 concludes the work and gives an overview over future tasks.

2. Reconfiguration Basics

As this work is based on the Xilinx Virtex series reconfiguration architecture, the basics will be reviewed in brief.

2.1. Xilinx Reconfiguration Architecture. Xilinx FPGAs are configured by a so-called *bitstream*, which is written into the SRAM configuration memory. Basically, a Virtex-FPGA consists of two layers, the configuration memory layer and the hardware layer. The hardware layer contains the reconfigurable hardware structures, like configurable logic blocks (CLBs), RAM blocks (RAMBs), IO blocks (IOBs), digital clock managers DCMs and configurable wiring resources. The hardware units are aligned into a 2D grid.

The configuration memory layer is organized into columns, spanning from top to bottom of a device, see Figure 3. There are six different kinds of columns present, such as CLB-columns. Each column determines the function of the underlying hardware resources. Within such columns, several different hardware resources are included. For example, in every CLB-column, there are IOBs in the top and bottom of that column. For some hardware resources, like DCMs, no dedicated configuration columns exist. Instead, their configuration is included in one of the other columns.

Each column is further divided into subcolumns, called frames. One CLB-column for example is covered by 22 configuration frames. Within the Xilinx reconfiguration architecture, a frame is the smallest addressable unit [4].

Partial and dynamic online reconfiguration in this context means, that parts of the content of the configuration memory can be changed during runtime of the system, while the reminder of the configuration remains unchanged. However, as the smallest addressable unit is one frame, a single resource, for example, a CLB, cannot be addressed independently. Instead, several resources are accessed altogether. Furthermore, the division into large units, like CLBs or IOBs is too coarse grained—there are smaller units like

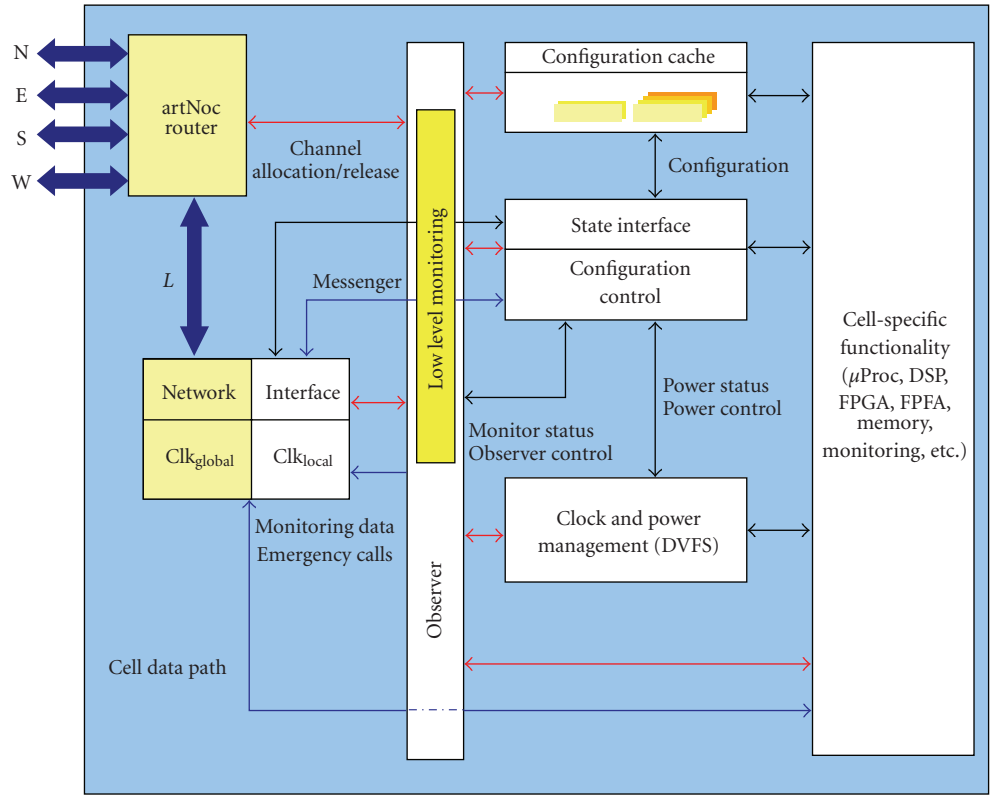

Figure 2: OPC block diagram view.

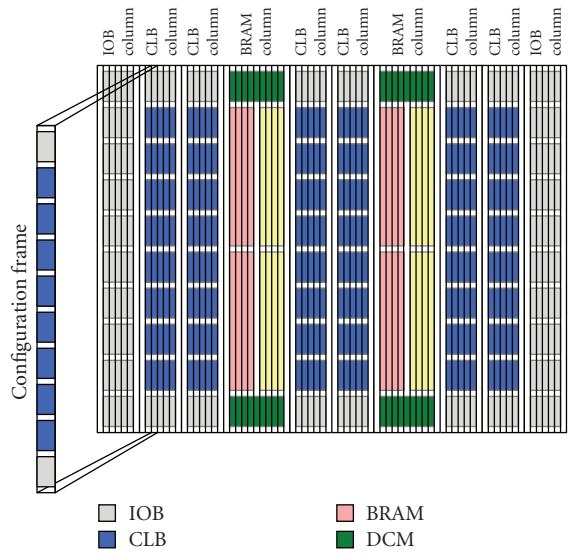

Figure 3: Xilinx virtex configuration architecture [4].

clock distribution buffer or look-up tables, which are not covered by the addressing scheme. A direct addressing of these units may be useful.

As discussed, the hardware units are arranged in a 2D grid, but the configuration architecture gives no direct access to the configuration space. However, for the proposed OC-application, a true 2D access is needed. Therefore, a special RMW [5] method has to be applied, as explained in the subsequent section.

2.2. Readback Modify Writeback Method. Besides writing configuration data to the configuration memory, Virtex-FPGAs allow that configuration can be read back. This feature, combined with the fact that a glitch less switching of configuration logic occurs, once a part of the configuration is overwritten with an equal content, can be used to alter every single bit of the configuration memory instead of whole frames. The procedure is applied as follows. First, the content of the frames to be modified is read back from the configuration memory and stored in a proper secondary memory. In a second step, the new configuration is merged into the read back frames. This entails, that parts of the frames stay the same, while other parts are overwritten by the new configuration. Finally, the content of the secondary memory, with the modified configuration, is written back to the configuration memory. The method significantly improves the flexibility and the area usage of the resulting system level design. However, the following two main drawbacks have to be taken:

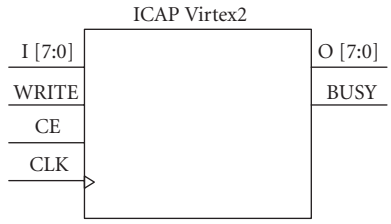

Figure 4: ICAP interface [4].

(1) The reconfiguration times for the same amount of chip area (e.g., measured in CLBs/second) are substantially increased compared to the classical one-dimensional approach [6].

(2) The interfacing with the configuration logic is considerably more complicated, as the merging of new configuration data into the read back data has to be controlled, as well as the single steps including generation of partial bitstream headers and addresses have to be accomplished [5].

Therefore, these major drawbacks are addressed by the presented approach in Section 4. Beforehand, the ICAP interface and related work will be introduced.

2.3. ICAP Interface. Besides other configuration interfaces Xilinx Virtex-FPGAs feature an internal configuration access port. It allows the on chip logic to have read, as well as write access to the *configuration control logic* (CCL) and to alter the current configuration during runtime. Combined with the presented methods, it is the key to a self-configuring and self-organizing hardware system.

Figure 4 shows its interface. Two separate 8-Bit data ports one for read (O) and one for write (I) are available. The WRITE signal indicates if data has to be written to or read from the ICAP. It has to be set by the controller. Once the ICAP is enabled with the CE line, it expects or rather provides valid data at every rising edge of the clock signal, given that the BUSY signal is held low. Once the ICAP interfaces raises the BUSY signal, it indicates, that it is no longer able to send/receive data at the data ports (O/I).

The CCL features a set of registers, which are used for communication. Each write access to the configuration memory is pipelined through an internal frame buffer. So, after a configuration frame is written to the CCL, an additional, so-called dummy frame, has to be written to flush the internal buffer to memory. The same is valid for read access.

The communication with the CCL consists of 5 basics steps.

(1) Sending of synchronisation sequence.

(2) Sending of operation commands to set the CCL registers.

(3) Sending/Receiving actual configuration data

(4) Sending/Reading of one dummy frame to flush CCL internal buffer.

(5) Sending desynchronisation sequence.

2.4. artNoC Local Port Interface. The local interface of the used artNoC [3] basically consists of two ports with mirrored signals: an input port for injecting data packets into the network, and an output port for receiving data packets from the network. The concept of virtual channels (VCs) allows a time interleaved simultaneous transmission of different data packets over the physical data input and output lines. The number of simultaneous virtual channels can be tailored during design time to fit the needs of the connected client. Further, for each VC a real-time feedback signal exists, that is routed in the opposite direction of the established datapath, from receiver to the sender, during VC reservation. Once the channel from sender to receiver is established, guaranties for throughput and latency can be given, and an in-order delivery of data can be assured. As we will see, this is important for the design of the proposed *artNoC-ICAP-Interface* module.

3. Related Work

3.1. Partial and Dynamic Online Reconfiguration. Most of the systems using partial and dynamic online reconfiguration make use of the Xilinx OPB-ICAP peripheral, which can be either connected to the "*Microblaze*" soft core processor or via a PLB to OPB-bridge to the Power PC cores available at the Virtex-II-Pro devices. Various systems have been implemented making use of 1-dimensional slot-based reconfiguration [6, 7], while others utilize the RMW method for a 2 dimensional placement [5]. Common to all these systems is that reconfiguration is controlled by a single processor. The reported configuration speeds are far away from the maximal possible speed provided by the ICAP. To overcome the speed limitations, in [8] an approach is presented that connects the ICAP to the PowerPC PLB bus with direct memory access (DMA) to a connected BRAM, lowering the processor load and speeds up writing of configuration data close to its maximum value. However, no support for the RMW method is provided, nor is it possible to access the ICAP in a virtual fashion. In [9, 10] the ICAP is connected to a NoC to allow a distributed access, but no support for the RMW method is provided, which is a key feature of the presented *artNoC-ICAP-Interface*. Further, the reported reconfiguration speeds suggest, that both interfaces do not operate at the optimal operation point.

3.2. Dynamic Frequency Scaling. Recently several works has been published dealing with power management and especially clock management on FPGAs. All authors agree, that there is a high potential for using DFS method in both ASIC and FPGA designs [11, 12].

In [13] the authors show, that even because of FPGA process variations and because of changing environmental conditions (hot, normal, cold temperature), dynamically clocking designs can lead to a speed improvement of up to 86% compared to using a fixed, statically, during design time estimated, clock. The authors use an external programmable

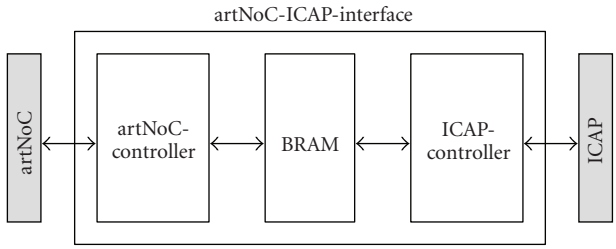

Figure 5: artNoC-ICAP-interface.

clock generator that is controlled by a host PC. However, in-order to enable the system to self-adapt its clock frequency, on chip solutions are required.

In [11] the authors proposed an online solution for clock gating. They use a feedback multiplexer with control logic in front of the registers. So it is possible to keep the register value and to prevent the combinatorial logic behind the register to toggle. But simultaneously, they highlight that clock gating on FPGAs could have a much higher power saving efficiency, if it would be possible to completely gate the FPGA clock tree. To overcome this drawback, in [14] the authors provide an architectural block, that is able to perform DFS. However, this approach leads to low speed designs and clock skew problems, as it is necessary to insert user logic into the clock network.

We show that on Xilinx Virtex-II no additional user logic is necessary to efficiently and reliably perform a fine grained self-adaptive DFS. This is possible because digital clock managers (DCMs) are available, that can be reconfigured during runtime with the presented interface. All advantages of the high speed clock distribution network could be maintained.

4. Architecture Details

To overcome the mentioned drawbacks and to realize a virtual decentralisation of the local ICAP interface an approach as shown in Figure 5 was taken.

The *artNoC-ICAP-Interface*, acts as a bridge between the artNoC router network and the ICAP interface of the Virtex-FPGAs.

The interface is able to receive configuration commands from all OPCs connected to the network. Further, from OPC view (client view) it provides different levels of abstraction. Clients can operate as close as with the natural ICAP commands, as well as sending their configuration on a higher level of abstraction, for example, on CLB basis. The actual interfacing in this case is taken over by *artNoC-ICAP-Interface*. This simplifies the handling of configuration significantly and the amount of configuration data to be sent to the interface is reduced (see Section 5.2) Furthermore, through a local handling of configuration data within the interface, important speedups could be achieved.

4.1. Basic Concepts. As mentioned before, the Xilinx Virtex configuration hardware works on a frame basis. When employing the RMW method every reconfiguration operation consists of a frame read and a frame write. To achieve this basic function, the interface consists of a BlockRAM, that can hold the content of one frame, and a unit that controls the ICAP, called *ICAP-Controller*. This unit implements all functionality to read a frame from ICAP, store it in BRAM and write it back to ICAP. To process the requests from network clients and to modify the configuration frame content, a second unit is implemented, the *artNoC-Controller*. To give this unit access to memory a dual ported BRAM is used. The advantage of this architecture is that the two controllers are decoupled by memory, making it possibly to operate both at different clock rates. Because the ICAP appears to be the bottleneck in the system the controller can work with the highest possible clock frequency (see Section 6.2). Further, a simultaneous loading of BRAM with both new configuration data and read back data (merging) can be performed resulting in remarkable speedups. The block diagram of the architecture is shown in Figure 5. The basic functions of the three units are summarized in the following listing.

artNoC-Controller

(i) processes configuration requests from clients,
(ii) controls the overall interface operation,
(iii) generation of frame address(es),
(iv) controls operation of ICAP-Controller,
(v) merging data into BRAM.

Dual Ported BRAM

(i) holds data of one configuration frame,
(ii) decouples artNoC-Controller and ICAP-Controller.

ICAP-Controller

(i) controls ICAP-Interface signals,
(ii) sends command sequences to ICAP,
(iii) reading/writing data from or to BRAM.

4.2. Design Considerations. FPGA resources, especially BRAMs, are strongly limited when thinking of a complete system with many OPCs that implement a certain functionality. As the interface implements the RMW method, at least data for one configuration frame has to be held in memory. There is a possibility to achieve an advantage by spending more memory than for one frame. As mentioned in Section 2.3, each read or write operation requires an additional dummy frame to be written. When consecutive frames have to be processed, the number of dummy frames can be reduced by reading more than one frame into BRAM, modify the frames and write them back to ICAP with only one dummy frame at the end of the read and write operation. Through this, the number of dummy frames can be reduced from n to [n/m] (with n # frames

to modify, m # memory for one frame), which leads to a higher reconfiguration performance. Unfortunately this is only possible for consecutive frames, with the cost of more memory and advanced control logic.

The used technique that configuration data from clients is directly merged in the BRAM saves resources, because no requests have to be stored and the control logic is simple (see Section 6.1). The disadvantage is that clients have to wait till complete configuration operations have finished, before another request can be accepted.

In this implementation at one time only requests from one client can be processed. The reason for this restriction is motivated as follows. Every configuration operation must do a complete RMW sequence as the ICAP cannot be interrupted. Only requests from other clients which would operate at the same frame have a chance to be processed in parallel. When expecting that requests from different clients are statistically independent, the possibility that requests belong to the same configuration frame is very small, due to the big FPGA area. Even if this occurs, special care has to be taken, as the configuration regions may overlap, which leads to a more complex control logic. In scenarios, where some clients are correlated with each other, an enhanced approach may be used. Otherwise, it is not worth to process more than one client request simultaneously.

The implementation presented here uses the simplest and least resource consuming approach.

4.3. Operation Modes. The *artNoC-ICAP-Interface* features five different operation modes, which allow the client cells to access the interface according to their current needs. One OPC, for instance, wants to configure its whole cell internal datapath, spanning several CLB blocks, the other cell just wants to change a few bits within a LUT, to adjust the coefficients of an implemented filter task. The different operation modes both insure that a minimum of configuration data has to be sent over the network, as well as the actions for controlling the reconfiguration, that have to be taken by the client cells, are kept low. Hence, the amount of additional memory needed to store the new partial configuration data as well as the complexity of the configuration managers within each OPC can be reduced.

The following listing describes the different modes and explains their purpose.

(1) Transparent-Operation-Mode. The interface works as a virtual extension of the physical ICAP hardware. Clients can send native ICAP commands [4], which are tunnelled through to the ICAP. The clients have full access to all ICAP features. This mode is intended to serve as a debugging feature or in case later system enhancements need this low level access.

(2) Read/Modify Frame-Operation-Mode. The client can read back or modify configuration data of one frame. This mode is suited, if just parts of modules have to be reconfigured, for example, changing a LUT.

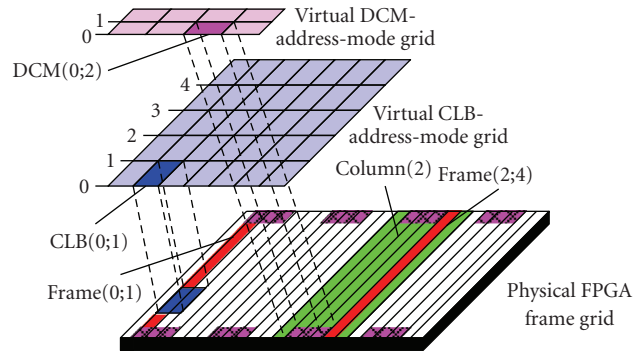

Figure 6: Virtual address grids.

Different address modes are available to select resources within a frame (see Section 4.4). Every desired frame can be addressed and several units within the frame can be processed.

(3) Read/Modify Module-Operation-Mode. The configuration data of whole modules can be read or modified. The module width can span several columns and the height can range from one unit to the maximum count a frame can hold (see Section 4.4). All frames within these columns are processed (e.g., CLB column: 22 frames/column).

4.4. Address Modes. As shown in Figure 6 the physical FPGA hardware resources are covered by a frame grid. Thereby, every single Virtex configuration frame contains the configuration information for different FPGA resources, such as CLBs, IOBs or DCMs. To take the burden from a client who, for example, wants to operate on CLBs, to calculate the addresses of the frames that have to be processed and the positions within these frames, where the configuration lies, the *artNoC-ICAP-Interface* features different addressing modes.

In this example the client can operate in CLB address mode and just needs to transmit the CLB coordinates of the desired CLB (cf. Figure 6, e.g., CLB $(0;1)$) where the new configuration has to be inserted. The interface automatically performs the mapping down to the physical FPGA frame grid. As with the RMW method every single bit can be changed independently, the access to the configuration memory can be more fine granular. This means, that new addressing modes can easily be added to give individual access to smaller hardware resources like clock distribution buffer or look-up tables, which is inherently not given. So, for each hardware resource an individual address grid can be introduced.

From a clients view with this method reconfiguration can be done on a much higher level of abstraction without knowing the exact underlying configuration architecture.

Moreover, with this approach complete hardware systems can be easily ported to different device families such as Virtex 4- or Virtex 5-FPGAs by just adjusting the address mapping parameters within the artNoC-ICAP-Interface package. At the moment CLB-, BRAM- and DCM-addressing modes are supported.

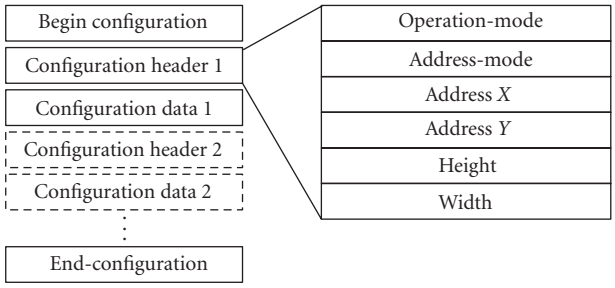

Figure 7: Configuration protocol.

4.5. Implemented Protocol. Each configuration request is started with the *begin-configuration* command. After that, different configuration operations are available, till an *end-configuration* command is received. Each configuration operation is preceded by a header followed by the data necessary for that operation (e.g., native ICAP command (transparent mode), module configuration data (module mode)). Figure 7 shows the protocol. The header main fields are *operation-mode* and *address-mode*. The *operation-mode* selects one of five different operations such as modify-module see Section 4.3. In every *operation-mode*, except transparent mode, all presented address modes are available, see Section 4.4. The *address-mode* determines on which FPGA resources the configuration works. The address fields are interpreted in related coordinates, so every resource is addressed with its own 2D grid. As discussed in the operational modes section, clients can configure whole modules (e.g., in CLB address mode 22 frames) or single frames. When a module is configured in module mode, one header has to be sent, followed with the complete information of that module. This especially includes data that has to be merged into all frames spanning the module, even if the data remains unchanged within several frames, so-called dummy data. When only few frames have to be changed it can be more efficient not to configure in module mode, but to address the frames in single frame mode. For every operation a header is needed followed by a data section. Dependent on the module size one of the two possibilities is more efficient. The client can decide which alternative is the appropriate for his needs. The following equation evaluates the number of frames n till the module operation is more efficient:

$$n = \frac{\text{header size} + \text{module height} \times 22}{\text{header size} + \text{module height}}. \quad (1)$$

If the number of frames to be modified is bigger than n, module mode is more efficient in terms of configuration data to be sent over the network, otherwise frame mode should be chosen.

4.6. RMW Configuration Procedure. The complete RMW configuration procedure is shown in Figures 8(a) and 8(b). First a network client establishes a real-time-channel with the *artNoC-Controller*. This ensures exclusive access for this

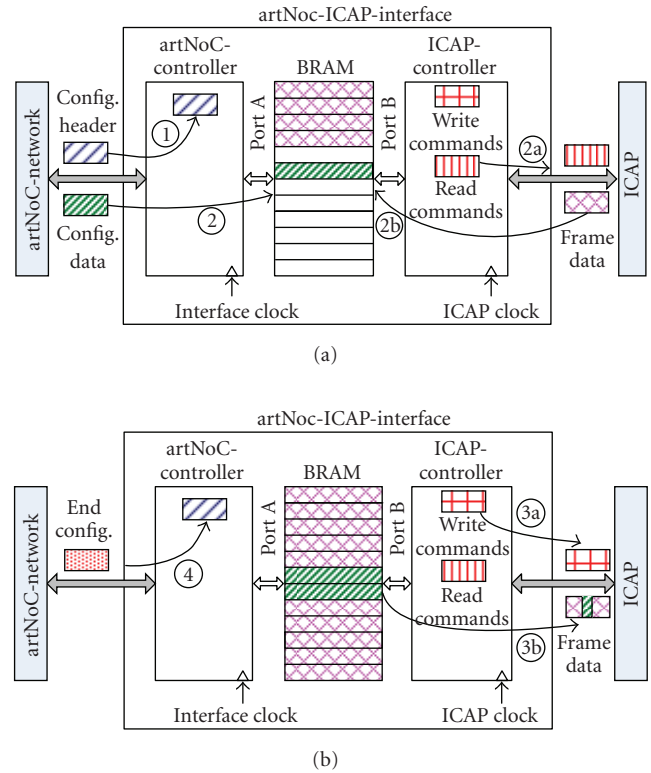

Figure 8: artNoC-ICAP-interface configuration procedure.

client and all other request from different clients will be rejected. Thereafter the *begin-configuration* command and the header are sent over the network (1). The *artNoC-Controller* processes the header to generate the proper frame-address(es) based on the chosen address mode, and triggers the *ICAP-Controller* to read back the frame from configuration memory. The readback consists of two steps: first a read command-sequence is written to the ICAP (2a) and the frame data from ICAP is continuously stored in BRAM (2b) on a byte basis (PORT B). Simultaneously, the configuration data, that has to be merged into the frame, is received from the network. It is directly written to the proper position in BRAM (2) over the second port (PORT A). At the same time PORT A has exclusive write access to the portion of BRAM it writes to, which results in the fact that the data written back from ICAP targeting this portion is discarded and the new configuration data is merged into the read back frame. When the whole frame is in memory, the write back procedure starts with the write command-sequence (3a) and the modified frame data (3b) are both send to ICAP.

This also includes automatic generation of the dummy frame after the frame has been written. Finally, the client sends an *end-configuration* command to the *artNoC-Controller* (4), which releases the established real-time channel and gives other clients the possibility to access the interface.

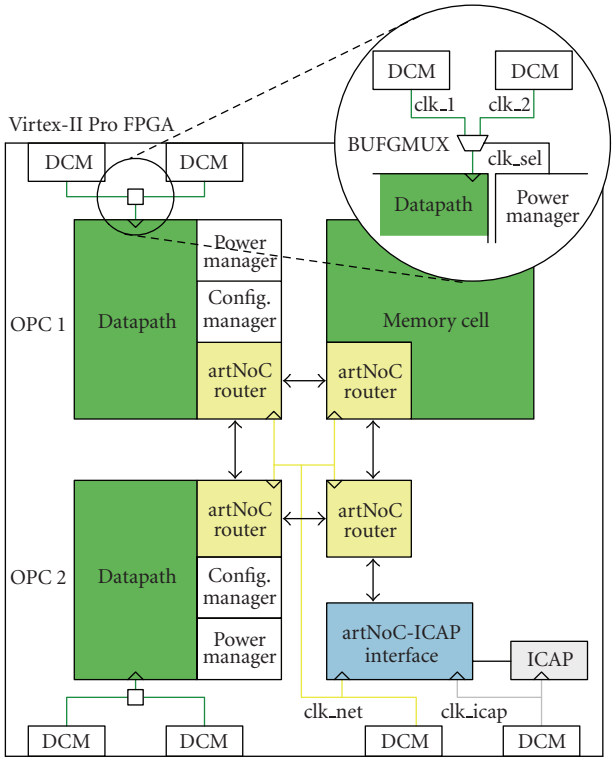

Figure 9: DodOrg FPGA floor plan/clock architecture.

5. Organic System Architecture

5.1. Overall Floorplan. As mentioned in the introduction, we propose to implement the OPC-based organic computing organisms on a Virtex-II Pro FPGA. The floor plan of the complete system is shown in Figure 9. Because of the two-dimensional reconfiguration capabilities, it is possible that every OPC occupies a self contained rectangular area. Depending on the device size several OPCs can be implemented onto a single FPGA. Every OPC is composed of a *configuration management unit* (CMU), a *power management unit* (PMU), a runtime reconfigurable datapath area (DPA) and the artNoC router. The artNoC router allows all OPC units to communicate with units of the other OPCs and the *artNoC-ICAP-Interface*. The *artNoC-ICAP-Interface* is located in the lower right corner of the FPGA and is connected to the artNoC as well as described in Section 4. All modules, except the datapath area, belong to the static design area and are not changed during runtime.

The clock net of the reconfigurable DP module of every OPC is connected to a BUFGMUX, which is driven by a pair of DCMs. The DP module clock is decoupled form the artNoC clock by using a dual ported, dual clock FIFO buffer. By using the features of the *artNoC-ICAP-Interface* every PMU can dynamically adapt the DCM output clock frequency, as described in Section 5.3.

5.2. Datapath Reconfiguration. The reconfiguration data of each DP module is stored in dedicated memory cells. Memory cells use the available on chip BRAM and provide an interface to be accessible over artNoC network. Every DP module is identified by a unique ID. A specific characteristic of the stored reconfiguration data is that it contains no absolute placement information and only the actual configuration data is stored. That means, that the represented DP module can be loaded into every OPCs reconfigurable datapath area. This is in contrast to the EAPR-flow, where the complete frames of all columns have to be stored. For example, the used "*Microblaze*" DP module requires 136 CLBs ($\rightarrow$ reconfigurable datapath area 4*34 CLBs) which results in a storage requirement of:

(i) 4Col*22Frames/Col*34Rows*10Byte/(Row*Frame) = *29920 Bytes*,
Compared to the partial bitstream size of:

(ii) 4Col*22Frames/Col*824Byte/Col= *72512 Bytes*
In case of the EAPR flow (for a XC2VP30 device). This is a reduction of *58.7%*.

Derived from the biological model the basic idea is the distributed organisation of the whole system. That means, in each OPC the CMU is responsible for the DPA of this cell. To carry out a reconfiguration process the cells CMU generates the configuration header and sends it to the *artNoC-ICAP-Interface*. It contains the placement information as explained in Section 4.5. Based on the ID, it then fetches the actual DP module configuration data from a memory cell and redirects it to the *artNoC-ICAP-Interface*. Finally, after the new DP module is configured the CMU triggers a reset and the module is ready to operate.

In general, each DP module has a specific maximum clock frequency according to its critical path. As a consequence, it is necessary to adjust the clock frequency after reconfiguration in-order to operate at maximum speed, or rather to operate in save conditions. This adjustment can be done by the PMU of the cell.

5.3. Power Management. Like DP reconfiguration, power management is performed on OPC basis. Therefore, the digital clock managers (DCMs) of Xilinx Virtex-FPGAs are used. Besides others, frequency synthesis is an important feature of the DCMs. Two main different programmable outputs are available. CLKDV provides an output frequency that is a fraction ($\div 1.5$, $\div 2$, $\div 2.5 \cdots \div 7$, $\div 7.5$, $\div 8$, $\div 9 \cdots \div 16$) of the input frequency CLKIN.

CLKFX is able to produce an output frequency that is synthesised by combination of a specified integer multiplier $M \in \{1 \cdots 32\}$ and a specified integer divisor $D \in \{1 \cdots 32\}$ by calculation CLKFX = M $\div$ D*CLKIN. By using the *artNoC-ICAP-Interface* both multiplier and divisor can be reconfigured during runtime and hence the clock frequency at the CLKFX output can be adjusted. However, reconfiguring a DCM involves a minimum delay of 60 μs (see Section 6.2) to change the clock frequency, if the *artNoC-ICAP-Interface* is not blocked by an active request.

This means, that the method is appropriate for reaching long term or intermediate term power management goals, that is, a new datapath is configured and the clock frequency is adapted to its critical path and then stays constant until a

new datapath is required. But if a frequent and immediate switching is necessary, for example, when data arrives in burst and between burst the OPC wants to toggle between shut off ($f_{DP} = 0$ Hz) and maximal performance ($f_{DP} = f_{max}$) the method needs to be extended.

In this case, a setup consisting of two DCMs and a BUFGMUX, as shown in Figure 9 can be chosen. The select input of the BUFGMUX is connected to the PMU of the OPC. Therefore, it is able to toggle between two frequencies immediately without any delay.

6. Result

6.1. Synthesis Results. The presented *artNoC-ICAP-Interface* has been synthesised and implemented targeting a Virtex2VP30 device by using the Xilinx ISE 9.1 toolchain. Table 1 shows the required resources. As we see only a small portion of FPGA resources are used. When considering complex multi core systems with many heterogeneous cores, like the DodOrg system, these figures can even get more neglected on bigger devices, but in turn the performance of the 2D reconfiguration can be significantly improved and the complexity of client's configuration logic, which is multiplied by the number of clients, can be reduced.

6.2. Reconfiguration Performance. One goal was to maximise reconfiguration performance of the RMW method. As a reference the time needed to perform a complete RMW cycle for one configuration frame on a Virtex2VP30 (824 Bytes) with 10 Byte configuration data was measured. As a test setup a hardware configuration client was connected to the artNoC, as shown in Figure 10. This client operates in *Modify-Frame-Mode* with *CLB-Address-Mode*. It repeatedly sends 10 Byte configuration data to toggle the function of a LUT from an AND- to an OR- gate. The correct function could be verified by applying two push buttons as input and a LED as an output to the gate.

Xilinx Chip Scope was connected to the signals representing the current internal state of the *artNoC-Controller*. So the number of clock cycles, starting with the receiving of the first byte of the config header until the RMW cycle was completed, could be measured. Thereby, the *artNoC-Controller,* as well as the "*artNoC*" where operated at a constant clock frequency of 50 MHz, whereas the clock frequency of the ICAP and the *ICAP-Controller* were altered from 50 to 110 MHz, as shown in Figure 11.

The measured data is divided into readback and write-back portion and is expressed in terms of clock cycles of the *artNoC-Controller*. As we can see, the maximum throughput is reached at an ICAP speed of 105 MHz with 2048 clock cycles which is equal to a configuration time of 40 μs/frame.

For example, the "*Microblaze*" DP module, spanning 4 clb-columns with an height of 34 clb-rows, can be modified within 3,5 ms. The width of the module determines the reconfiguration time, as it also determines the number of frames to be modified. The same "*Microblaze*" with a layout of 8 clb-columns and 17 clb-rows requires twice the time. This has to be considered during the design phase.

Table 1: Resource utilization.

Resource	Number	Percentage
Slices	387	2%
Slice FlipFlops	183	0%
4 input LUTs	709	2%
BRAMs	1	0%
MULT18x18s	2	1%

Table 2: Component power consumption.

	Passive power (mW)	Active power (mW)
static_offset	—	11
DCM	—	37
CMU	<1	<1
artNoC-ICAP-IF	<1	9
ICAP	69	76

One would expect that the performance can be further improved with increased ICAP clock frequency, but the measurements show, that the ICAP itself is limited in speed. With increased clock frequency the number of ICAP busy cycles also increases, due to ICAP internal processing. Especially in read-mode, the ICAP-busy cycles causes the *ICAP-Controller* to stall.

The speed of the *artNoC-Controller* itself is not critical, as in 1217 clock cycles, which are need for read-back, the new configuration of a whole frame can be easily merged into the read back frame (assuming 8 Bit router links).

6.3. Power Consumption. In the preceding section results for reconfiguration times and tradeoffs have been presented. This section evaluates the potential of power savings and performance enhancements in the context of module based partial online reconfiguration. Especially, the overhead in terms of area and power consumption introduced by the approach (CMU, *artNoC-ICAP-Interface*, DCM) is taken into account.

We calculated the power consumption by measuring the voltage drop over an external shunt resistor (0.4 Ohm) on the FPGA core voltage (FGPA_VINT). As a test system again the Xilinx XUP board, with a Virtex-II Pro (XC2VP30) device, was used. For all measurements the board source clock of 100 MHz was used as an input clock to the design.

To isolate the portions of power consumption, as shown in Table 2, several distinct designs have been synthesised. For DCM power measurement, an array of toggle flip-flops at 100 MHz with and without a DCM in the clock tree has been recorded and the difference of both values has been taken. For extracting ICAP power consumption, a system consisting of CMU, *artNoC-ICAP-Interface* and ICAP instance and a second identical system, but without ICAP instance, have been implemented. After activation, the CMU sends bursts of two complete alternating configuration frames, targeting the same frame in configuration memory. The ratio of toggling bits between the two frames is 80% and is considered to be

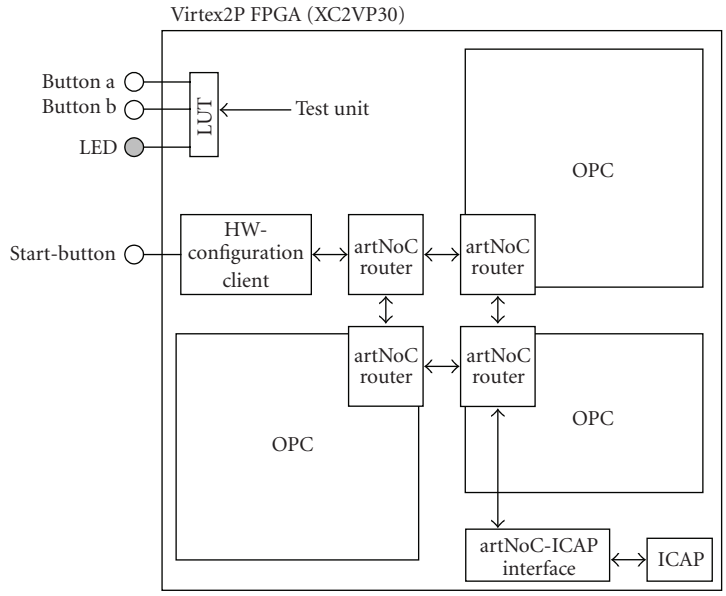

Figure 10: Measurement test setup.

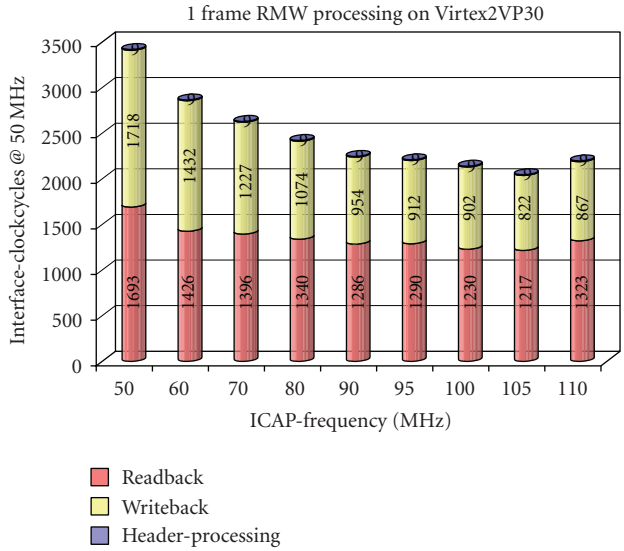

Figure 11: RMW performance.

representative for a partial reconfiguration. Therefore, before CMU activation, the "passive" power and after activation the "active" power could be measured. Again, the difference in power consumption of the two systems was taken to extract ICAP portion. The other components were measured with the same methodology. Therefore, for example, all components necessary to implement the approach presented in Section 5.3, with two DCMs + BUFGMUX consume 196 mW when active, that is, 180 mW when passive. But it has to be considered, that *artNoC-ICAP-Interface,* as well as ICAP is also used for partial 2D reconfiguration.

7. Summary and Future Work

In this paper we have presented the *artNoC-ICAP-Interface,* which is a small and lightweight hardware module to support a fast RMW method for a true 2 dimensional online reconfiguration of Xilinx Virtex-FPGAs. The reconfiguration time could be reduced to 40 μs per frame, which at the same time determines the physical limit of the ICAP. Different operation modes, which can be combined with various addressing modes, optimize the amount of traffic that has to be sent over the network for reconfiguration and raise the abstraction level of the reconfiguration interface to a higher level. The innovative concept can be easily ported to different device families, also featuring an internal configuration access port. With slight changes, it should also be possible, to modify the network interface in-order to suit the needs of different NoC architectures, so they can also benefit from the achieved virtual decentralisation of the ICAP. Similarly, not only distributed reconfiguration systems can take effort from the presented concepts. Because of the FIFO based interface, the *artNoC-ICAP-Interface* can also be easily connected to single configuration controller, like the "*Microblaze*" processor (e.g., over the FSL-bus). Hence, it can take advantage of the addressing modes and performance improvements in 2D reconfiguration.

In this work we determined the optimal operation point to reach a throughput for RMW, which is just limited by the physical ICAP instance.

Further we showed how the *artNoC-ICAP-Interface* can be used to reconfigure the Digital Clock Managers during runtime, to perform a Dynamic Frequency Scaling. A concept for clock net partitioning, using the example of the organic system architecture, was shown. Both long term as well as short term power management goals can be achieved.

All advantages of the high speed clock distribution network could be maintained.

Finally, we provide measurement results for power consumption of all relevant components involved.

To the best of our knowledge the *artNoC-ICAP-Interface* is the first approach to support the RMW method with all properties described above. Future work is targeted towards the self-adaptation and self-healing capabilities of the presented organic hardware architecture by employing suitable monitoring or observer techniques.

References

[1] J. Becker, et al., "Digital on-demand computing organism for real-time systems," in *Proceedings of the 19th International Conference on Architecture of Computing Systems (ARCS '06)*, W. Karl, et al., Ed., Karlsruhe, Germany, February 2006.

[2] U. Brinkschulte, M. Pacher, and A. von Renteln, *An Artificial Hormone System for Self-Organizing Real-Time Task Allocation in Organic Middleware*, Springer, Berlin, Germany, 2007.

[3] C. Schuck, S. Lamparth, and J. Becker, "artNoC—a novel multi-functional router architecture for organic computing," in *Proceedings of the 17th International Conference on Field Programmable Logic and Applications (FPL '07)*, pp. 371–376, Amsterdam, Netherlands, August 2007.

[4] "Xilinx Virtex-II Platform FPGA User Guide," UG002(V1.3).

[5] M. Hübner, C. Schuck, and J. Becker, "Elementary block based 2-dimensional dynamic and partial reconfiguration for Virtex-II FPGAs," in *Proceedings of the 20th International Parallel and Distributed Processing Symposium (IPDPS '06)*, p. 8, Rhodes Island, Greece, April 2006.

[6] C. Bobda, M. Majer, A. Ahmadinia, T. Haller, A. Linarth, and J. Teich, "The Erlangen slot machine: increasing flexibility in FPGA-based reconfigurable platforms," in *Proceedings of the IEEE International Conference on Field Programmable Technology*, pp. 37–42, December 2005.

[7] M. Ullmann, M. Hübner, B. Grimm, and J. Becker, "An FPGA run-time system for dynamical on-demand reconfiguration," in *Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS '04)*, vol. 18, pp. 1841–1848, Santa Fe, NM, USA, April 2004.

[8] C. Claus, F. H. Müller, J. Zeppenfeld, and W. Stechele, "A new framework to accelerate Virtex-II Pro dynamic partial self-reconfiguration," in *Proceedings of the 21st International Parallel and Distributed Processing Symposium (IPDPS '07)*, pp. 1–7, Long Beach, Calif, USA, March 2007.

[9] R. Koch, T. Pionteck, C. Albrecht, and E. Maehle, "An adaptive system-on-chip for network applications," in *Proceedings of the 20th International Parallel and Distributed Processing Symposium (IPDPS '06)*, p. 8, April 2006.

[10] L. Möller, I. Grehs, E. Carvalho, et al., "A NoC-based infrastructure to enable dynamic self reconfigurable systems," in *Proceedings of the 3rd International Workshop on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC '07)*, Montpellier, France, June 2007.

[11] Y. Zhang, J. Roivainen, and A. Mämmelä, "Clock-gating in FPGAs: a novel and comparative evaluation," in *Proceedings of the 9th EUROMICRO Conference on Digital System Design: Architectures, Methods and Tools (DSD '06)*, pp. 584–588, August-September 2006.

[12] I. Brynjolfson and Z. Zilic, "FPGA clock management for low power," in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGAs '00)*, Monterey, Calif, USA, February 2000.

[13] J. A. Bower, W. Luk, O. Mencer, M. J. Flynn, and M. Morf, "Dynamic clock-frequencies for FPGAs," *Microprocessors and Microsystems*, vol. 30, no. 6, pp. 388–397, 2006.

[14] I. Brynjolfson and Z. Zilic, "Dynamic clock management for low power applications in FPGAs," in *Proceedings of the Custom Integrated Circuits Conference (CICC '00)*, pp. 139–142, Orlando, Fla, USA, May 2000.