Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2011, Article ID 985931, 17 pages
doi:10.1155/2011/985931

Research Article

Exploration of the Power-Performance Tradeoff through Parameterization of FPGA-Based Multiprocessor Systems

Diana Göhringer,[1] Jonathan Obie,[1] André L. S. Braga,[2] Michael Hübner,[3] Carlos H. Llanos,[2] and Jürgen Becker[3]

[1] *Object Recognition Department, Fraunhofer IOSB, 76275 Ettlingen, Germany*
[2] *Department of Mechanical Engineering, University of Brasilia (UnB), 70910-900 Brasilia, DF, Brazil*
[3] *ITIV, Karlsruhe Institute of Technology (KIT), 76128 Karlsruhe, Germany*

Correspondence should be addressed to Diana Göhringer, diana.goehringer@iosb.fraunhofer.de

Received 6 September 2010; Accepted 10 February 2011

Academic Editor: Gilles Sassatelli

The design space of FPGA-based processor systems is huge, because many parameters can be modified at design- and runtime to achieve an efficient system solution in terms of performance, power and energy consumption. Such parameters are, for example, the number of processors and their configurations, the clock frequencies at design time, the use of dynamic frequency scaling at runtime, the application task distribution, and the FPGA type and size. The major contribution of this paper is the exploration of all these parameters and their impact on performance, power dissipation, and energy consumption for four different application scenarios. The goal is to introduce a first approach for a developer's guideline, supporting the choice of an optimized and specific system parameterization for a target application on FPGA-based multiprocessor systems-on-chip. The FPGAs used for these explorations were Xilinx Virtex-4 and Xilinx Virtex-5. The performance results were measured on the FPGA while the power consumption was estimated using the Xilinx XPower Analyzer tool. Finally, a novel runtime adaptive multiprocessor architecture for dynamic clock frequency scaling is introduced and used for the performance, power and energy consumption evaluations.

1. Introduction

Parameterizable function blocks used in FPGA-based system development, open a huge design space, which can only hardly be managed by the user. Examples for this are arithmetic blocks like divider, adder, and soft IP-multiplier, which are adjustable in terms of bit width and parallelism. Additional to arithmetic blocks, soft-IP processor cores provide a variety of parameters, which can be adapted to the requirements of the application to be realized with the system. Especially, Xilinx offers, with the MicroBlaze Soft-IP 32-bit RISC processor [1], a variety of options for characterizing the core individually. These options are, amongst others, the use and size of cache memory, the arithmetic unit, a memory management unit, and the number of pipeline stages. Furthermore, the tools offer to deploy semiautomatically up to two processor cores as multiprocessor on one FPGA. Certainly more cores are available for the system design by performing the custom tool chain. Every option as described above can be adjusted to find an optimized parameterization of the single processor core in relation to the target application. For example, a specific cache size can speed up the application tremendously, but also the optimal partition of functions onto the two cores has a strong impact on the speed and power consumption of the system. These examples show the huge design space, even if only one parameter is used. It is obvious that the deployment of multiple parameters for system adjustment leads to a multidimensional optimization problem, which is not, or at least very hardly, manageable by the designer. In order to gain experience regarding the impact of processor parameterization in relation to a specific application scenario, it is beneficial to evaluate, for example, the performance and power consumption of an FPGA-based system and compare the results to a standard design with a default set of parameter. The result of such an investigation is a first step for developing standard guidelines for designers

and an approach for an abstraction of the design space in FPGA-based system design. This paper presents first results of a parameterizable multiprocessor system on a Xilinx Virtex-4 FPGA, where the parameterization of the processor is evaluated in respect to power consumption and performance. Moreover, the varying partition of the different application scenarios is evaluated in terms of power consumption for a fixed performance. For this purpose, a tool flow for analyzing the power consumption through generating the switching activity interchange format (SAIF) file or the value change dump (VCD) file from the postplace and route simulation will be introduced. The presented flow enables to generate the most accurate power and energy consumption estimation from this level of abstraction. A further output of the presented work is an overview of the impact of parameterization to the performance, power and energy consumption for Xilinx Virtex-4 and Virtex-5 FPGAs. The results can be used as a basic guideline for designers who want to optimize their system performance and the power and/or energy consumption. This basic guideline can be a starting point for the analysis of wider application scenarios that can produce a more complete guideline for the definition of task escalation strategies and for the parameterization of FPGA-based MPSoCs. The paper is organized as follows. In Section 2, related work is presented. Section 3 describes the power estimation tool flow used for the presented approach. The novel system architecture deployed for analyzing the performance and the power consumption of the different applications is presented in Section 4. The application scenarios are described in Section 5. In Section 6, the application integration and the results of the performance and power consumption evaluation are provided. Finally, the paper is closed by presenting the conclusions and future work in Section 7.

2. Related Work

Reduction of the dynamic and static power consumption is very important especially for embedded systems, because they often use batteries as a power source.

Therefore, many researchers, for example, Meintanis and Papaefstathiou [2], explored the power consumption of Xilinx Virtex-II Pro, Xilinx Spartan-3, and Altera Cyclone-II FPGAs. They estimated the power consumption at design time using the commercial tools provided by Xilinx and Altera. They further explored the differences between the measured and estimated power consumption for these FPGAs. Becker et al. [3] explored the difference between measured and estimated power consumption for the Xilinx Virtex-2000E FPGA. Furthermore, they explored the behavior of the power consumption, when using dynamic reconfiguration to exchange the FPGA-system at runtime.

Other works focus on the development of own tools and models for efficient power estimation at design time for FPGA-based systems. Poon et al. [4] present a power model to estimate the dynamic, short circuit, and leakage power of island-style FPGA architectures. This power model has been integrated into the VPR CAD flow. It uses the transition density signal model [5] to determine signal activities within the FPGA. Weiss et al. [6] present an approach for design time power estimation for the Xilinx Virtex FPGA. This estimation method works well for control-flow-oriented applications but not so well for combinatorial logic. Degalahal and Tuan [7] present a methodology to estimate dynamic power consumption for FPGA-based system. They applied this methodology to explore the power consumption of the Xilinx Spartan-3 device and to compare the estimated results with the measured power consumption.

All these approaches focus either on the proposal of a new estimation model or tool for estimating the power consumption at design time, or they compare their own or commercial estimation models and tools with the real measured power consumption. The focus of the investigations presented in this paper is to show the impact of parameterization of IP cores, specifically the MicroBlaze soft processor, which differs from the approaches mentioned above where the topic is more on tool development for power estimation.

The novelty of our approach is to focus on the requirements of the target application and to propose a design guideline for system developers of processor-based FPGA systems. This means providing guidance in how to design a system to achieve a good tradeoff between performance and power and energy consumption for a target application. To develop such a guideline, the impact of the frequency, different processor configurations, and the task distribution in a processor-based design are investigated in this paper for different application scenarios.

3. Tool Flow for Power Measurement

Xilinx provides two types of tools for power consumption estimation: Xilinx Power Estimator (XPE) [8] and Xilinx Power Analyzer (XPower) [9].

The XPE tool is based on an excel spreadsheet. It receives information about the number and types of utilized resources via the report generated by the mapping process (MAP) of the Xilinx tool flow. Alternatively, the user can manually set the values for the number and type of used resources. The frequencies performed within the design have to be manually set by the user. The advantage of this method is that results are obtained very fast. The disadvantage is that the results are not very accurate, especially for the dynamic power consumption. This is because the different toggling rates of the signals are not taken into account. A further reason the results are not too accurate, is that they are based on the MAP report, and not on the postplace and route (PAR) report, which resembles the system used for generating the bitstream.

The XPower tool, the alternative to Xilinx Power Estimator, estimates the dynamic and static power consumption for submodules, different subcategories, and the whole system based on the results of a postplace and route (PAR) simulation. This makes the estimation results much more accurate compared to the XPE tool, because the physical place and route, as well as the utilized resources of the system, are taken into account for the power estimation. But even more important, due to the simulation of the PAR system with real input data, the toggling rates of the signals can

be extracted and used within the power estimation. For estimating the power consumption with the XPower tool, the following input files are required:

(i) native circuit description (NCD) file, which specifies the design resources,

(ii) physical constraint file (PCF), which specifies the design constraints,

(iii) switching activity interchange format (SAIF) or value change dump (VCD) file, which specify the simulated activity rates of the signals.

The NCD and the PCF files are obtained after the PAR phase of the Xilinx implementation tool flow. Both files provide the information needed for an accurate estimation of the total quiescent power of the device. To achieve a good estimation of the dynamic power consumption, an SAIF or VCD file is required, because they contain information about the toggling rates of all signals within the hardware realization on FPGA. Both files can be obtained after simulation of the PAR design with real input data using, for example, the ModelSim simulator. The XPower tool can also be used without an SAIF or VCD file. Then, default toggle rates are used to estimate the dynamic power of the design, which is not as accurate as using an SAIF or VCD file.

By choosing the specific tool, the user has to decide on a tradeoff between accuracy and design time. For a high accuracy, a longer design time is needed, by using XPower with the results from PAR and PAR simulation. If a rough estimation is sufficient, both the XPE tool and XPower, using only the PAR results, but no SAIF or VCD file, can be used.

Due to the higher accuracy, the XPower tool using PAR and PAR simulation results was used for the approach presented in this paper. As we wanted to estimate the power consumption for systems with one or two MicroBlaze processors, the hardware and the software executables of the different system were designed within the Xilinx Platform Studio (XPS) [10]. Figure 1 shows the flow diagram for doing power estimation with XPower for an XPS system.

After the hardware has been designed and implemented within the XPS environment, the SimGen [10] tool is used to generate the post-PAR timing simulation model of the system. This simulation model is used to do a post-PAR timing simulation of the design. Within this work, we used the newest Xilinx tool version 12.2. Here, two possible simulators exist: Xilinx ISIM and ModelSim simulator. Both can be used without any restrictions to generate the SAIF or VCD files. In this work, ISIM was used to generate the SAIF files for all the uniprocessor designs. ModelSim was used to generate the VCD files for the two processors design. In the last step, XPower is required to load the SAIF/VCD, the NCD, and the PCF files of the design and to estimate the dynamic and static power consumption. Care has to be taken, because, in a normal Xilinx implementation flow, the software executables are integrated into the memories of the processors after the bitstream has been generated. When using XPower and the post-PAR simulation, the memories of the processor have to be initialized in an earlier step. This means, into the post-PAR simulation model, otherwise the simulated system behavior and the VCD file would not be accurate. Using the Xilinx 12.2 tools, this memory initialization is done automatically by the tools, when the target devices are a Virtex-4 and Virtex-5 FPGA.

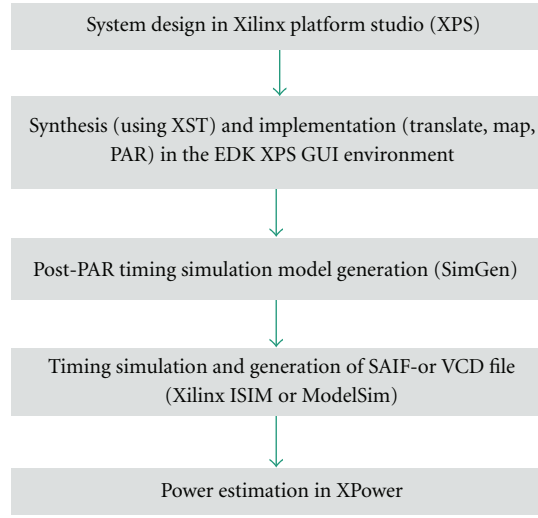


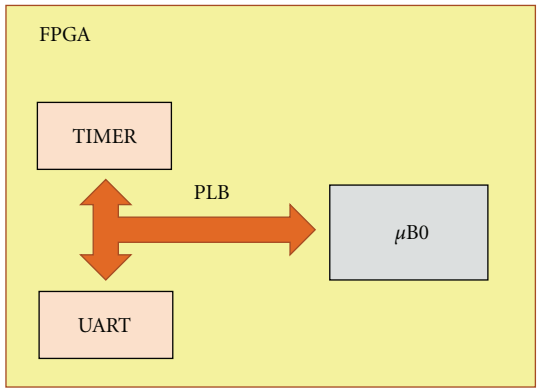
Figure 1: Diagram of the EDK XPower Flow.

4. Novel System Architecture

For evaluating the impact of the different design parameters onto the overall system performance, power, and energy consumption, three different processor designs have been built.

The first design is a uniprocessor, which is used to evaluate the impact of the clock frequency and the processor configurations onto the performance, power and energy consumption for two different Xilinx FPGAs: Virtex-4 and Virtex-5. Therefore, this system has been designed as shown in Figure 2. The number of I/O interfaces has been chosen in such a way that the same system can be used without changes for both FPGA boards. The only required changes are the physical I/O pin locations for the UART, clock, and reset pins. The system components remain unchanged, to allow a fair comparison between the two FPGAs. The input data for the applications is statically stored in the local on-chip memory of the processor, because an equal interface, for example PCI, was not present for both boards.

The system structure of the dual-processor system is shown in Figure 3. This system is used to evaluate the impact of different application partitions onto the overall performance, power dissipation, and energy consumption. This evaluation has been done on the Virtex-4 board only, because a PCI connection was needed to receive the input data for the applications from the host PC. Also, the differences between the power and energy consumption of the different FPGA families have been already done with the uniprocessor design, and, therefore, no differences are expected for the dual-processor design. Three new components have been designed and implemented: the virtual-IO, the bridge, and the reconfigurable clock unit. All three components have



PLB: processor local bus
UART: universal asynchronous receiver transmitter
μB: microblaze

Figure 2: Uniprocessor system.

been integrated into a library for the XPS tool. Therefore, they can be inserted and parameterized using the graphical user interface (GUI) of the XPS tool, which makes them easily reusable within other XPS designs.

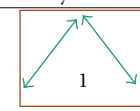
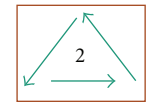
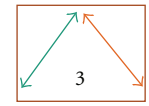
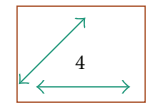
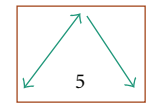
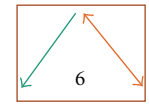
The virtual-IO receives data from the host PC and sends the results back to the host PC via the PCI bus. The virtual-IO communicates via the fast simplex links (FSLs) [11] with two MicroBlaze processors (μB0 and μB1). μB0 communicates with the user via the UART interface. It has a timer, which is used to measure the performance of the overall system. The two processors communicate with each other via FSLs over the bridge component. Depending on the fill level of the FIFOs within the bridge, reconfiguration signals are sent to the reconfigurable clock unit. The reconfigurable clock unit reconfigures the clocks of the two processors based on the reconfiguration signals issued by the bridge. The power and energy consumption of this dual-processor system is compared against the evaluation results of the uniprocessor system shown in Figure 4. This uniprocessor system uses also the virtual-IO component to receive the input data required for the applications. The bridge and the reconfigurable clock unit have been removed.

The following subsections explain the new components and their features more in detail.

4.1. Virtual-IO. The virtual-IO component [12] was designed to communicate with the host PC via the PCI bus. It provides an input and an output port to the PCI bus and one input and one output port for each MicroBlaze processor. It consists of two FIFOs, one for the incoming and one for the outgoing data of the PCI bus. Each FIFO is controlled via a finite state machine (FSM), as it is shown in Figure 5.

The virtual-IO is a wrapper around 6 different modules. The modules, their symbols, and functionalities are summarized in Table 1. The first module is virtual-IO 1, which sends data first to μB0 and then to μB1. It then receives the calculated results in the same order. The second module is virtual-IO 2, which sends data only to μB0. Results are only

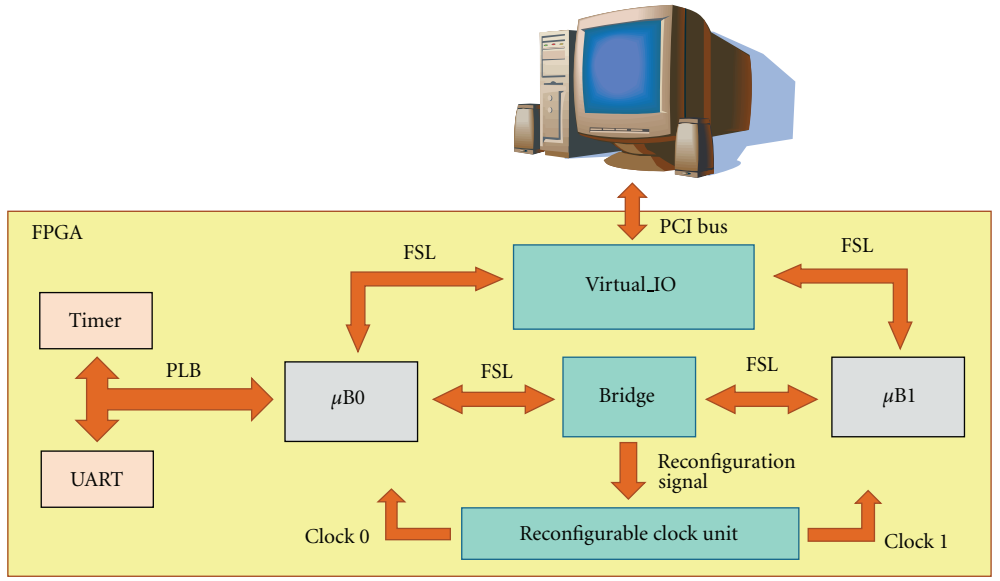
Table 1: Different modules of the virtual-IO component together with their symbol and basic functionality.

Virtual-IO module	Symbol	Function
Virtual-IO 1		Sends data first to μB0 then to μB1. Receives data in the same order
Virtual-IO 2		Sends data only to μB0. Receives data only from μB1
Virtual-IO 3		Sends data first to μB0, then to both and finally only to μB1. Receives data first from μB0, then from μB1
Virtual-IO 4		Sends data only to μB0. Receives data only from μB0. Used for uniprocessor designs
Virtual-IO 5		Sends same data to both processors. Receives results only from μB0
Virtual-IO 6		Sends same data to both processors. Receives results only from μB1

received over μB1. Therefore, μB0 sends its results to μB1, which then sends the results of μB0 together with its own results back to the virtual-IO 2. The third module is virtual-IO 3, which sends first data to μB0. Afterwards, it sends in parallel to both processors μB0 and μB1 the same data. Finally, it sends some data only to μB1. After the execution of the processors, first μB0 and then μB1 send their results back to the virtual-IO 3. The fourth module is virtual-IO 4, which is only connected to one of the processors, for example, μB0. Due to this, this module is used in all uniprocessor designs. For a dual-processor design it sends data to μB0, which then forwards parts of the data to μB1. After execution, μB1 sends its results back to μB0, which forwards the results of the execution of the two processors to the virtual-IO 4. The fifth module is virtual-IO 5, which sends the same data to both processors in parallel, but receives the results only via μB0. The sixth module is virtual-IO 6. It is very similar to virtual-IO 5. The only difference is that it receives the calculation results from μB1 instead of μB0.

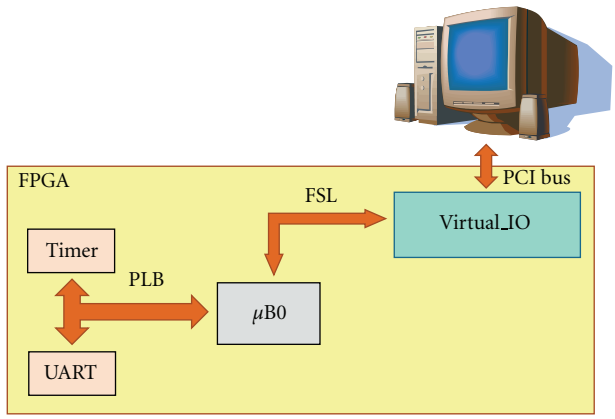
The modules can be selected in the XPS GUI via the parameters of the virtual-IO component. Other parameters that can be set by the user are the number of input and output words for each processor separately, the number of common input words, and the size of the image (only for image processing applications).

4.2. Bridge. The bridge module [12] is used for the interprocessor communication. It consists of two asynchronous FIFOs controlled by FSMs, to support a communication via the two different clock domains of the processors, as shown in Figure 6. This bridge component controls the fill level of



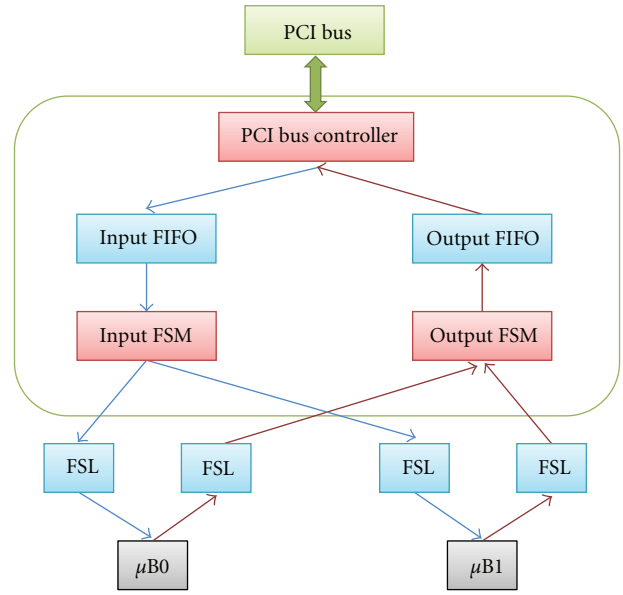
FSL: fast simplex link
PLB: processor local bus
UART: universal asynchronous receiver transmitter
μB: microblaze
PCI: peripheral component interconnect

Figure 3: Dual-processor design with three new components: virtual-IO, bridge, and reconfigurable clock unit.



FSL: fast simplex link
PLB: processor local bus
μB: microblaze
PCI: peripheral component interconnect
UART: universal asynchronous receiver transmitter

Figure 4: Uniprocessor system with the new virtual-IO component to enable a fair comparison with the dual-processor system.



FSL: fast simplex link
FSM: finite state machine
μB: microblaze

Figure 5: Virtual-IO component.

the two FIFOs. If one FIFO tends to be utilized with 75%, it is assumed that the processor, which reads from this FIFO, is too slow. As a result, a reconfiguration signal to increase the clock rate of this processor is sent to the reconfiguration clock unit.

4.3. Reconfigurable Clock Unit. Two different designs for the reconfigurable clock unit have been implemented and will be presented in the following subsections. The first implementation uses hardware reconfiguration to modify the frequency of clock signals. The advantage is that a variety of different clock signals can be provided at runtime. The disadvantage is that the reconfiguration requires a time period of 200 ms, which can be too long, depending on the application.

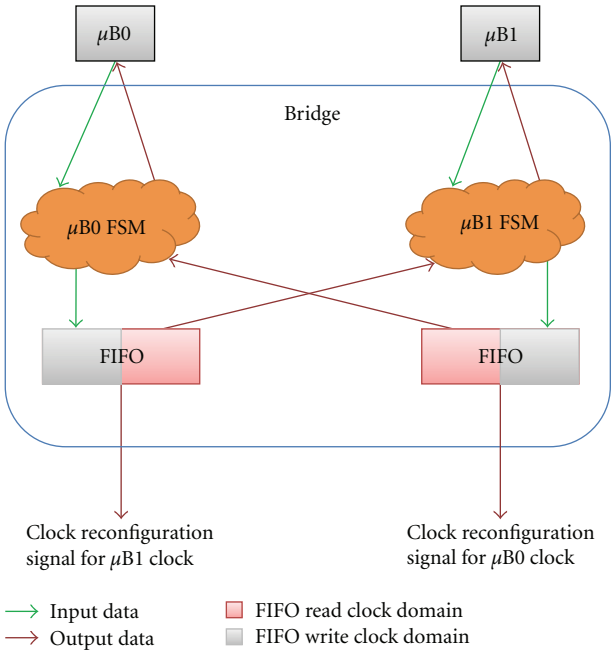


Figure 6: Internal structure of the bridge.

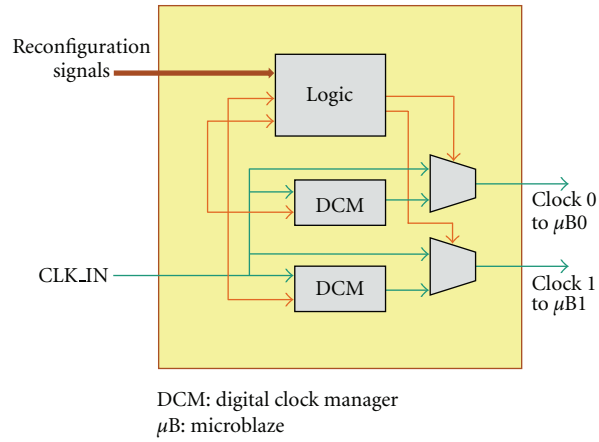


Figure 7: Internal structure of the reconfigurable clock unit.

The second implementation exploits the use of the multiple ports provided by the digital clock manager (DCM) [13] component of the Xilinx FPGAs. Here, clock buffer multiplexer primitives (BUFGMUXes) [14] are used to allow a faster switch between different clocks. The advantage is a faster switching time of few clock cycles between different frequencies. The drawback is that not as many different clocks are possible, as when dynamic reconfiguration is used. The number of the different possible clocks depends on the number of available DCMs and BUFGMUXes on the chosen FPGA device.

4.3.1. Reconfigurable Clock Unit Using Reconfiguration [12]. The internal structure of the reconfigurable clock unit is shown in Figure 7. It consists of two DCMs, two BUFGMUXes, and the logic component, which controls the reconfiguration of the DCMs.

The logic component shown in Figure 7 receives the reconfiguration signals from the bridge component. It then starts the reconfiguration of the DCM primitive for the slower processor. For the reconfiguration purposes, the specific ports provided by Xilinx for dynamic reconfiguration of the Virtex-4 DCM primitive are used. During the reconfiguration process, the DCM has to be kept in a reset state for a minimum of 200 ms. During this time interval, the outputs of this DCM are not stable and cannot be used. Instead of stalling the corresponding processor, the BUFGMUX primitive is used to provide CLK_IN, the original input clock of the two DCM, to the processor, whose DCM is under reconfiguration. The BUFGMUX is a special clock multiplexer primitive, which assures that no glitches occur when switching to a different clock. After the configuration of the DCM is finished, the BUFGMUX is used to switch back to the DCM clock. An alternative would be to stall the processor, while its clock is being reconfigured. Because 200 ms are quite a long time, especially for image processing applications where each 40 ms a new input frame is received from a camera, this would result in a loss of input data.

To prevent an oscillation caused by the frequency scaling mechanism, the controller logic will stop increasing the clock frequency, if 125 MHz for this MicroBlaze have been reached, which is the maximum frequency supported by the MicroBlaze and its peripherals, or if its clock frequency has been increased for three consecutive times. If the reconfiguration signal is furthermore asserted, meaning the processor is still too slow, then the DCM of the faster processor is reconfigured to provide a slower clock to the faster processor.

The internal structure of the logic block is shown in Figure 8. Its function is to supply the reset signal for the correct time span (at least 200 ms) required for correct dynamic reconfiguration (reset controller block) and to also provide the partial reconfiguration data to the corresponding DCM (reconfiguration monitor). The reconfiguration monitor block also monitors the LOCKED signals of the DCMs to know when the partial reconfiguration of a DCM is complete. The clock switcher block has the duty of providing the switching signal to the clock buffer multiplexer at the right time. Besides other minor functions, the logic component also implements the prevention of any race conditions that could result by both bridge component FIFOs having their respective clock reconfiguration signals active at the same time (it would not make sense to reconfigure both DCMs at the same time). For this purpose, the Logic component uses the XOR block on both reconfiguration signals coming from the bridge component, such that only one DCM can be partially reconfigured at one time. For its own synchronous activities, the logic component uses the input clock to the reconfigurable clock unit, which is also the input clock to the DCMs. The logic component also makes sure that the maximum clock frequency supported by the system (125 MHz) is not exceeded (reconfiguration counter block).

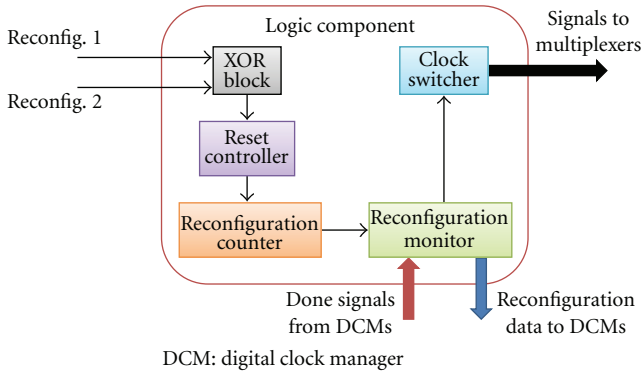


Figure 8: Internal structure of the logic block of the reconfigurable clock unit.

Therefore, the reconfiguration counter block keeps account of the number of times a DCM has been reconfigured such that no DCM can be configured more than 4 times. After 3 consecutive reconfiguration signals from the same bridge FIFO, the logic component actually slows down the originally faster processor by reconfiguring its clock to a lower value.

4.3.2. Reconfigurable Clock Unit Using Clock Buffer Multiplexers. Alternatively, instead of dynamically reconfiguring the DCM, different output ports of a DCM could be used to generate different clocks as shown in Figure 9. Using several BUFGMUXes, the different clocks could be selected.

Figure 10 shows the internal structure of the DCM_wrapper, consisting of two DCMs and several BUFGMUXes to allow fast switching between different clocks.

5. Application Scenarios

Four different applications scenarios were selected to explore the impact of the processor configurations, the task distribution, and the dynamic clock frequency scaling on the power consumption of FPGA-based processor systems. The four different algorithms are described in detail in the next subsections. The first algorithm is the well-known sorting algorithm called Quicksort [15]. It includes a number of branches and comparisons. The second algorithm is an image processing algorithm called normalized squared correlation (NCC), which consists of many arithmetic operations, for example, multiply and divide. The third algorithm is a variation of a bioinformatics algorithm called DIALIGN [16], which contains many comparisons and additions and subtractions. The fourth application is a character recognition algorithm using artificial neural networks (ANNs) [17], which consists of many floating point arithmetic operations. These algorithms with their different algorithm requirements, for example, branches, comparators, multiply and divide, add and subtract, and floating point, were used to provide a user guideline of designing a system with a good performance per power tradeoff for a specific application. By comparing the algorithm requirements of new applications with the four example algorithms, the system configurations of the most similar example algorithm is chosen as a starting system. Such a guideline to limit the design space is very important to save time and achieve a higher time-to-market, because the simulation and the power estimation with XPower are very time consuming. Also, the bitstream generation to measure the performance of the application on the target hardware architecture is time consuming. These long design times can be shortened by starting with an appropriate design, for example, the right processor configurations, a good task distribution, and a well-selected execution frequency.

5.1. Sorting Algorithm: Quicksort. Quicksort [15] is a well-known sorting algorithm with a divide and conquer strategy. It sorts a list by recursively partitioning the list around a pivot and sorting the resulting sublists. It has an average complexity of $O(n \log n)$.

5.2. Image Processing Algorithm: Normalized Squared Correlation. 2-D squared normalized correlation (NCC) is often used to identify an object within an image. The evaluated expression is shown:

$$C(p) = \frac{\left(\sum_{i=0}^{n}\sum_{j=0}^{m}\left(A_p(i,j)-\overline{A_p}\right)\times\left(T(i,j)-\overline{T}\right)\right)^2}{\left(\sum_{i=0}^{n}\sum_{j=0}^{m}\left(A_p(i,j)-\overline{A_p}\right)^2\right)\times\left(\sum_{i=0}^{n}\sum_{j=0}^{m}\left(T(i,j)-\overline{T}\right)^2\right)}, \quad (1)$$

where T: template image with n rows and m columns, A_p: subwindow of the search region with n rows and m columns, $\overline{T}$: mean of T, $\overline{A_p}$: mean of A_p.

This algorithm uses a template T of the object to be searched for and moves this template over the search region A of the image. A_p, the subwindow of the search region at point p with the same size as T, is then correlated with T. The result of this expression is stored at point p in the result image C. The more similar A_p and T are, the higher is the result of the correlation. If they are equal, the result is 1. The object is then detected at the location with the highest value.

5.3. Bioinformatic Algorithm: DIALIGN. DIALIGN [16] is an algorithm from bioinformatics domain, which is used for comparison of the alignment of two genomic sequences. It produces the alignment with the highest number of similar elements and, therefore, the highest score as shown in Figure 11.

5.4. Character Recognition Using Artificial Neural Networks (ANN). The character recognition algorithm was implemented as a multilayer perceptron (MLP) artificial neural network (ANN) to detect the numbers 0 to 9 and the characters "<" and ">". This example was extracted from a project that searches for connected pixels in images, preprocesses the connected pixels and performs the character recognition. This implementation was done according to [17]. For this work, only the preprocessed character set was employed. The network was trained using the preprocessed

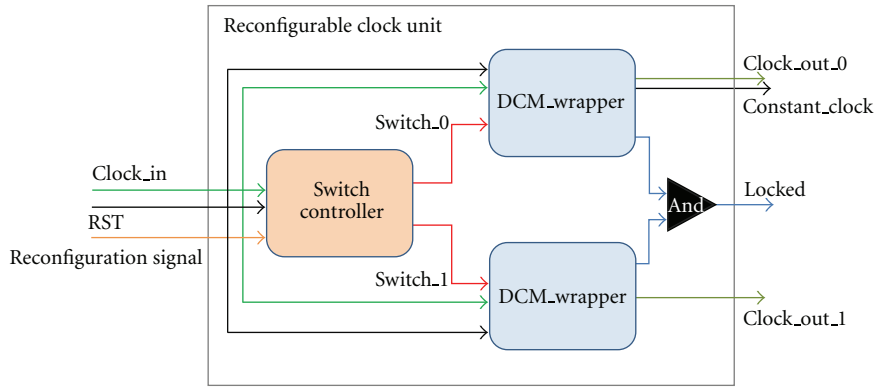


Figure 9: Alternative internal structure of the reconfigurable clock unit.

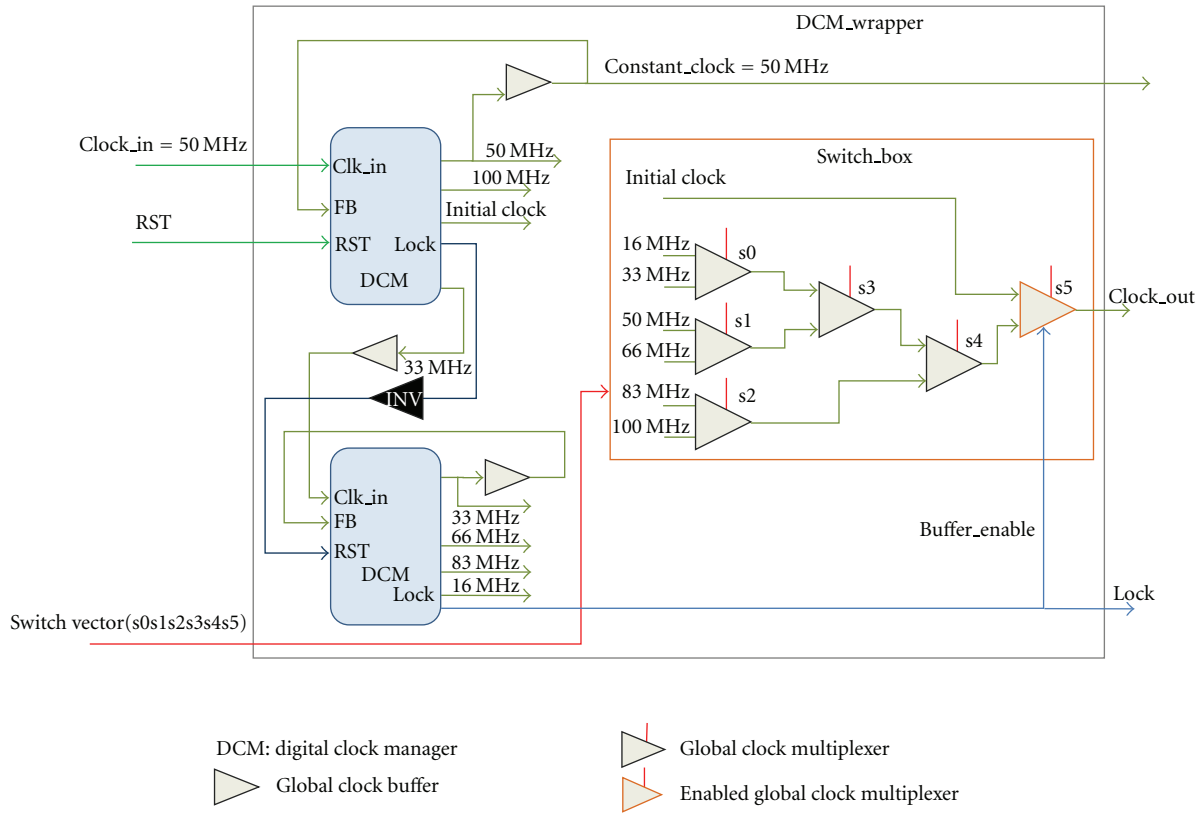


Figure 10: Internal structure of the DCM_wrapper component.

characters depicted in Figure 12. Two-thirds of the images were used for training and one-third for verification.

The trained ANN had 45 neurons in the hidden layer and 12 neurons in the output layer. The input layer contains 91 input elements, so each neuron in the hidden layer has 91 inputs. The output of these hidden neurons is given by (2), where i is the neuron index, j is the input number of the neuron, w_{ij} is the weight applied to the input j in that neuron (i), and b_i is a bias constant value for the neuron i

$$y_i = \frac{1}{1 + e^{(-b_i - \sum_{j=1}^{91} w_{ij} x_{ij})}}. \quad (2)$$

The output of each of the 12 neurons in the output layer is given by (3), where b_k is the bias constant for the output neuron k, y_i is the output of the ith neuron from the previous layer, and w_{ki} is the weight applied to that value. The recognized character is determined by the wining neuron, that is, the neuron for which the output is the nearest to 1;

$$y_k = -b_k - \sum_{i=1}^{12} w_{ki} y_i. \quad (3)$$

The characters were stored in MicroBlaze's main memory as bytes. During execution, they were expanded to arrays

Figure 11: Alignment of two sequences a and b with DIALIGN.

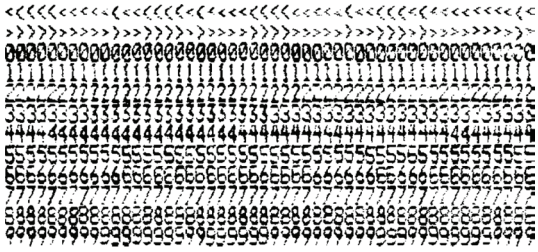


Figure 12: Character set used for training the network.

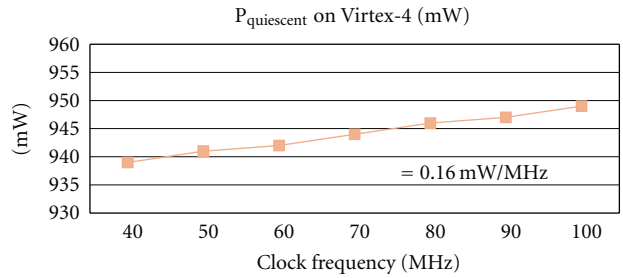


Figure 13: Impact of the clock frequency to the static power consumption of a uniprocessor design on a V4FX100.

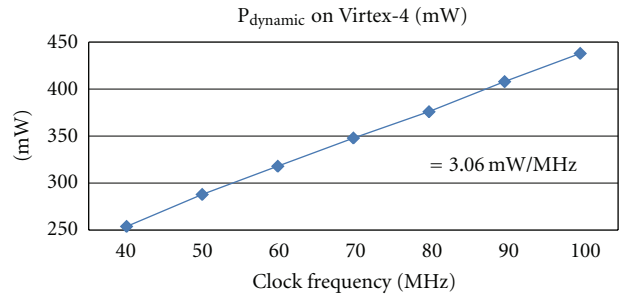


Figure 14: Impact of the clock frequency to the dynamic power consumption of a uniprocessor design on a V4FX100.

where each position contained the value of one pixel of the character image (1 or 0). A total number of 100 characters were presented during the experiments.

6. Integration and Results

Power consumption estimation and performance measurement were done for a Xilinx Virtex-4 (V4FX100) FPGA and a Xilinx Virtex-5 (V5LX110T) FPGA. The performance for the Virtex-4 FPGA was measured on the corresponding FPGA board from Alpha-Data [18] and the performance for the Virtex-5 FPGA was measured on the corresponding Xilinx XUP board. As measuring the exact power consumption of the FPGA on both boards is not possible, it was estimated at design time using the XPower tool flow as described in Section 3. The impact of the clock frequency, the configuration of the processor, and the task distribution onto the power consumption and the performance of the system has been explored, and the results are presented in the following subsections. For each exploration, some parameters had to be kept fixed to assure a fair comparison as shown in Table 2.

For the exploration of the impact of the clock frequency, the algorithm (NCC) and the processor configuration (default: 5-stage pipeline, no arithmetic unit (AU), no floating point unit (FPU)) have been kept fixed. For the exploration of the impact of the processor parameters, (default, AU, reduced pipeline (RP), AU + RP) ± FPU, the clock frequency was kept fixed at 100 MHz.

Finally, for the exploration of the task distribution, the processor configuration and the performance were kept fixed to lower the overall system power consumption, while maintaining the performance similar to the performance achieved with a reference uniprocessor design running at 100 MHz, which is a standard frequency for both Virtex-4- and Virtex-5-based MicroBlaze systems.

6.1. Impact of the Clock Frequency. In the following two subsections, the impact of the variation of the clock frequency to the power consumption of a Virtex-4 and a Virtex-5 FPGA was explored using a uniprocessor system, which executes the NCC algorithm on one MicroBlaze. The MicroBlaze was configured to have a 5-stage pipeline and no arithmetic unit. No AU means that the MicroBlaze has no integer multiplier/divider, no pattern comparator, and no barrel shifter. For the power consumption estimation, the results for quiescent, dynamic, and overall power consumption are given. The quiescent power consumption is also called static power consumption in the following, because it represents the power consumption of the user-configured FPGA without any switching activity. Furthermore, the execution time and the resulting energy consumption are presented. For each selected clock frequency, the processor system has been recompiled with the appropriate clock constraints.

6.1.1. Virtex-4 FX 100 FPGA. The results for the dynamic, the quiescent, and the total power consumption for the Virtex-4 FPGA are presented in Table 3 together with the execution time and the overall energy consumption.

The impact of the clock frequency onto the static and the dynamic power consumption for the Virtex-4 FPGA is presented in Figures 13 and 14, respectively. As can be seen, the static power consumption increases by around 0.16 mW/MHz while the dynamic power consumption increases by around 3.06 mW/MHz.

Table 2: Fixed parameters for the exploration of the impact of the clock frequency, the processor configuration, and the task distribution on the performance, power dissipation, and energy consumption.

Impact of	Fixed parameters	Variable parameters
Clock frequency	(i) Algorithm: NCC (ii) Processor configuration: default (5-stage pipeline, no AU, no FPU) (iii) # of processors: 1	(i) Clock frequency: 40–100 MHz (ii) FPGA: Virtex-4, Virtex-5
Processor configuration	(i) Clock frequency: 100 MHz (ii) No. of processors: 1	(i) Processor configurations: (default, AU, RP, RP + AU) ± FPU (ii) Algorithm: NCC, Quicksort, DIALIGN, ANN (iii) FPGA: Virtex-4, Virtex-5
Task distribution	(i) Execution time = execution time of a uniprocessor design at 100 MHz (ii) Processor configuration: 5-stage pipeline, integer multiplier, pattern comparator (iii) No. of processors: 2 (iv) FPGA: Virtex-4	(i) Application partitioning (ii) Algorithm: NCC, Quicksort, DIALIGN

Table 3: Impact of the variation of the clock frequency to the power consumption for V4FX100.

Clk Freq. (MHz)	$P_{Dynamic}$ (mW)	$P_{Quiescent}$ (mW)	P_{Total} (mW)	P_{Total} (%)	T_{exe} (ms)	Energy (mJ)
40	254	939	1192	−14,1	791,11	943,00
50	288	941	1229	−11,4	632,89	777,82
60	318	942	1260	−9,2	527,41	664,53
70	348	944	1292	−6,8	452,06	584,06
80	376	946	1322	−4,7	395,55	522,92
90	408	947	1355	−2,3	351,60	476,42
100	438	949	1387	NA	316,44	438,91

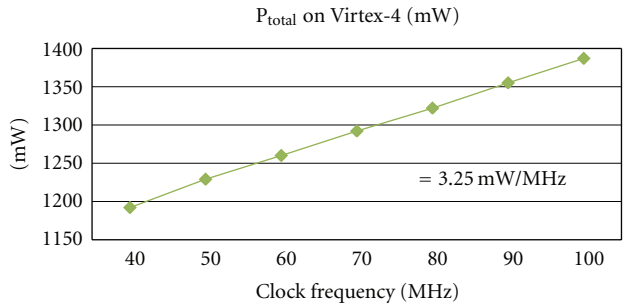


Figure 15: Impact of the clock frequency to the total power consumption of a uniprocessor on a V4FX100.

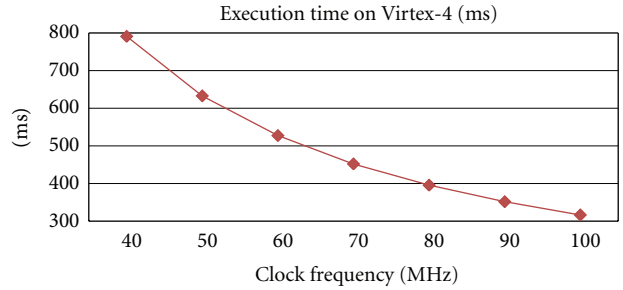


Figure 16: Impact of the clock frequency to the execution time of a uniprocessor design on a V4FX100.

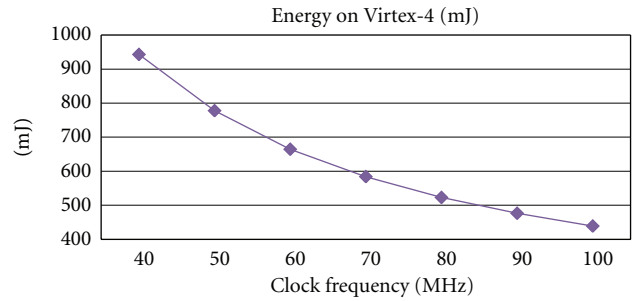


Figure 17: Impact of the clock frequency to the energy consumption of a uniprocessor design on a V4FX100.

Out of this results the impact onto the total power consumption, which is around 3.25 mW/MHz. The impact on the total power consumption is shown in Figure 15.

The impact on the execution time is shown in Figure 16.

The impact onto the overall energy consumption, which is around 8.42 mJ/MHz is shown in Figure 17.

6.1.2. Virtex-5 LX110T FPGA. The results for the dynamic, the quiescent, and the total power consumption together with the execution time and the overall energy consumption for the Virtex-5 FPGA are given in Table 4. As can be seen, the static power consumption is higher while the dynamic power consumption is lower compared to the Virtex-4 FPGA. This can be derived from the different CMOS processes: Virtex-5 has a 65 nm process while Virtex-4 has a 90 nm process.

The impact of the clock frequency onto the static and the dynamic power consumption for the Virtex-5 FPGA is presented in Figures 18 and 19, respectively. Here, a different behavior, compared to the Virtex-4 FPGA, can be seen. Due to the new Virtex-5 architecture, the static power

Table 4: Impact of the variation of the clock frequency onto the power consumption for V5LX110T.

Clk Freq. (MHz)	$P_{Dynamic}$ (mW)	$P_{Quiescent}$ (mW)	P_{Total} (mW)	P_{Total} (%)	T_{exe} (ms)	Energy (mJ)
40	215	1230	1444	−5,87	791,11	1142,36
50	229	1230	1459	−4,89	632,89	923,38
60	223	1230	1453	−5,28	527,41	766,32
70	258	1231	1489	−2,93	452,06	673,12
80	254	1231	1485	−3,19	395,55	587,40
90	238	1230	1468	−4,30	351,60	516,16
100	302	1232	1534	NA	316,44	485,42

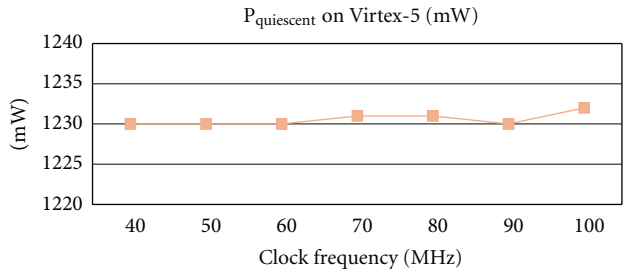


Figure 18: Impact of the clock frequency to the static power consumption of a uniprocessor design on a V5LX110T.

consumption is nearly constant for all clock frequencies. Minor variations occur to different placements of the design. The dynamic power consumption shows an unsteady behavior, when varying the clock frequency. A reason for this unsteady behavior could lie in the PAR process for Virtex-5 FPGAs, where different resources have been chosen for the different clock frequencies of the designs.

The total power consumption results from adding the dynamic and the static power consumption. As the static power consumption is nearly constant, the total power consumption shows the same behaviour over different clock frequencies than the dynamic power consumption. In total, the power consumption of the Virtex-5LX110T is around 200 mW higher than the one of the Virtex-4FX100 FPGA. The impact on the total power consumption is shown in Figure 20.

The impact on the execution time is equal for both FPGAs and is shown in Figure 21.

The overall energy consumption for the different clock frequencies is shown in Figure 22.

6.2. Impact of the Processor Configurations. For exploration purposes, a uniprocessor design consisting of a single MicroBlaze running at 100 MHz was used. The results were compared against a reference configuration, which was a MicroBlaze with a 5-stage pipeline and no arithmetic unit, which means no integer multiplier, no integer divider, no barrel shifter, and no pattern comparator. The following

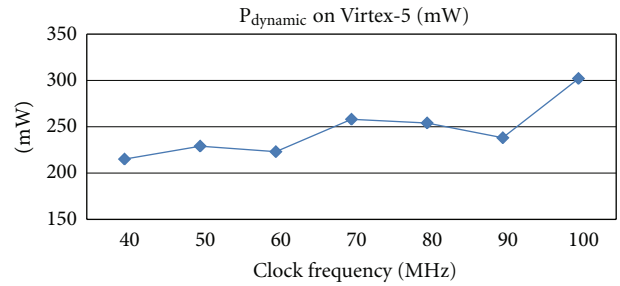


Figure 19: Impact of the clock frequency to the dynamic power consumption of a uniprocessor design on a V5LX110T.

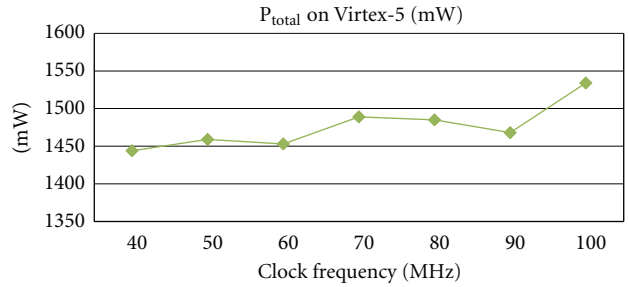


Figure 20: Impact of the clock frequency to the total power consumption of a uniprocessor design on a V5LX110T.

configurations were explored for the NCC, Quicksort and DIALIGN:

(i) default,
(ii) adding an arithmetic unit (AU),
(iii) reduction of the pipeline to 3 stages (RP),
(iv) combination of (i) and (iii) (AU + RP).

For the ANN, the configurations differ, as an additional parameter, the FPU, was added:

(i) default + FPU,
(ii) adding an arithmetic unit (AU + FPU),
(iii) reduction of the pipeline to 3 stages (RP + FPU),
(iv) combination of (i) and (iii) (AU + RP + FPU),
(v) arithmetic unit without FPU (AU).

The impact to the power consumption, the performance, and the energy consumption was explored for all four algorithms and is presented in the following subsections for Virtex-4 and Virtex-5, respectively. The impact is very different between the selected applications, due to the different algorithm requirements, as mentioned in Section 5 and its subsections.

6.2.1. Virtex-4 FX 100 FPGA. Figure 23 and Table 5 show the impact of the different configurations for the Quicksort algorithm. The combination of a reduction of the pipeline stages and the addition of the arithmetic unit provides the best solution in terms of performance, power and energy

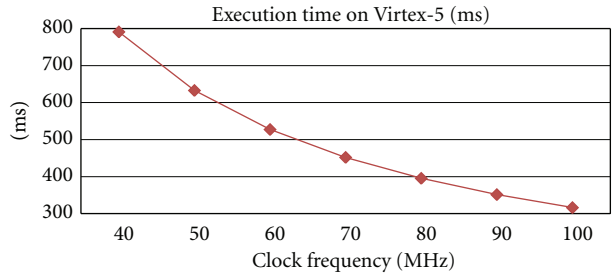


Figure 21: Impact of the clock frequency to the execution time of a uniprocessor design executing the NCC algorithm on a V5LX110T.

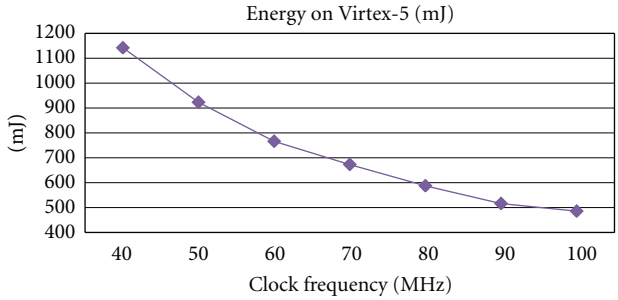


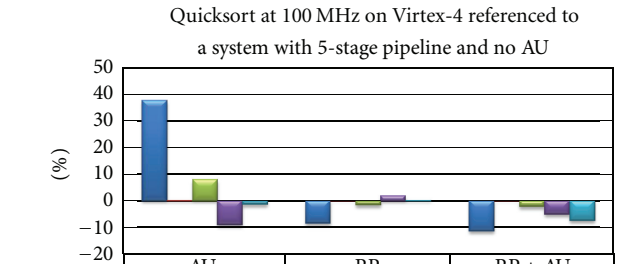
Figure 22: Impact of the clock frequency onto the overall energy consumption of a uniprocessor design executing the NCC algorithm on a V5LX110T.

Table 5: Impact of the MicroBlaze configurations (μB param) for the Quicksort algorithm at 100 MHz for V4FX100.

μB param	$P_{Dynamic}$ (mW)	$P_{Quiescent}$ (mW)	P_{Total} (mW)	P_{Total} (%)	T_{exe} (ms)	Energy (mJ)
Default	243	938	1181	NA	3,02	3,57
AU	336	943	1280	8,38	2,74	3,51
RP	222	937	1159	−1,86	3,09	3,58
RP + AU	215	937	1152	−2,46	2,86	3,29

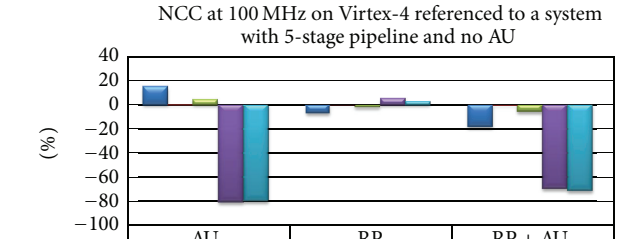
consumption. The reasons for this are, on the one hand, the multiple branches in the algorithm, which benefit from a reduction of the pipeline stages and, on the other hand, the multiple comparators, which benefit from the pattern comparator within the arithmetic unit. Therefore, the RP + AU system would be chosen. Using these systems, further performance and power evaluations could be done by adding or removing the different internal configurations of the arithmetic unit, as probably some, for example the integer divider, are not needed by the Quicksort algorithm, and, therefore, consume power but do not improve the overall performance.

Figure 24 and Table 6 show the impact of the different configurations for the NCC algorithm. As this algorithm requires many arithmetic operations, the addition of an AU improves the overall execution time very strongly (over 80%) while the reduction of the pipeline stages results in a slight degradation. This degradation is due to the reason that the execution of arithmetic operations takes more clock cycles, if the pipeline is reduced. Therefore, for this and similar



	AU	RP	RP + AU
Pdynamic	38.27	−8.64	−11.52
Pquiescent	0.53	−0.11	−0.11
Ptotal	8.38	−1.86	−2.46
Execution time	−9.27	2.32	−5.30
Energy	−1.67	0.41	−7.62

Figure 23: Impact of the MicroBlaze configurations for the Quicksort algorithm.



	AU	RP	RP + AU
Pdynamic	15.53	−7.53	−19.18
Pquiescent	0.42	−0.21	−0.53
Ptotal	5.12	−2.45	−6.42
Execution time	−82.55	6.04	−71.08
Energy	−81.66	3.44	−72.94

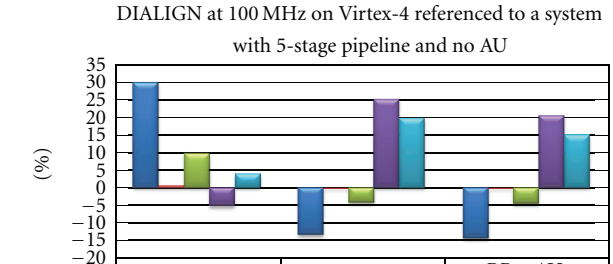
Figure 24: Impact of the MicroBlaze configurations for the NCC algorithm.

Table 6: Impact of the MicroBlaze configurations for the NCC algorithm at 100 MHz for V4FX100.

μB param	$P_{Dynamic}$ (mW)	$P_{Quiescent}$ (mW)	P_{Total} (mW)	P_{Total} (%)	T_{exe} (ms)	Energy (mJ)
Default	438	949	1387	NA	316,4	438,85
AU	506	953	1458	5,12	55,2	80,48
RP	405	947	1353	−2,45	335,5	453,93
RP + AU	354	944	1298	−6,42	91,5	118,77

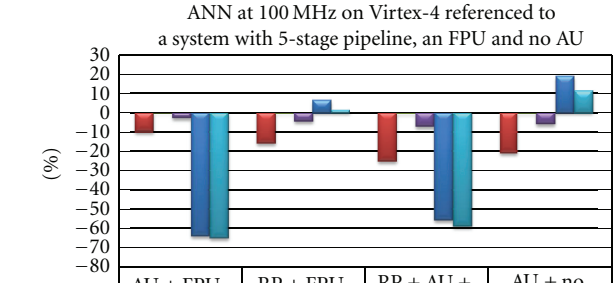
algorithms, a system with an AU and a 5-stage pipeline would be optimal from a performance and energy consumption perspective. If the power consumption needs to be reduced and some performance degradation is acceptable, then the AU + RP system would be a good choice.

In Figure 25 and Table 7, the impact to the performance and power consumption of the three different processor configurations compared to the reference system are presented for the DIALIGN algorithm. Adding an AU improves the execution time by 5% while increasing the overall power



	AU	RP	RP + AU
Pdynamic	30.37	−13.32	−14.25
Pquiescent	0.84	−0.32	−0.32
Ptotal	10.03	−4.36	−4.65
Execution time	−5.28	25.62	20.91
Energy	4.22	20.14	15.28

Figure 25: Impact of the MicroBlaze configurations for the DIALIGN algorithm.



	AU + FPU	RP + FPU	RP + AU + FPU	AU + no FPU
Pdynamic	−10.37	−16.38	−25.82	−21.41
Pquiescent	−0.23	−0.36	−0.56	−0.47
Ptotal	−3.06	−4.83	−7.61	−6.31
Execution time	−64.95	7.02	−56.64	19.21
Energy	−66.02	1.85	−59.94	11.69

Figure 26: Impact of the MicroBlaze configurations for the ANN algorithm.

Table 7: Impact of the MicroBlaze configurations for the DIALIGN algorithm at 100 MHz for V4FX100.

μB param	$P_{Dynamic}$ (mW)	$P_{Quiescent}$ (mW)	P_{Total} (mW)	P_{Total} (%)	T_{exe} (ms)	Energy (mJ)
Default	428	948	1376	NA	829,8	1141,80
AU	558	956	1514	10,03	786	1190,00
RP	371	945	1316	−4,36	1042,4	1371,80
RP + AU	367	945	1312	−4,65	1003,3	1316,33

Table 8: Impact of the MicroBlaze configurations for the ANN algorithm at 100 MHz for V4FX100.

μB param	$P_{Dynamic}$ (mW)	$P_{Quiescent}$ (mW)	P_{Total} (mW)	P_{Total} (%)	T_{exe} (ms)	Energy (mJ)
Default + FPU	365,85	944,95	1311	NA	2328	3052
AU + FPU	327,9	942,81	1271	−3,06	816	1037
RP + FPU	305,94	941,58	1248	−4,83	2491	3108
RP + AU + FPU	271,37	939,65	1211	−7,61	1009	1222
AU + no FPU	287,52	940,55	1228	−6,31	2775	3408

and energy consumption compared to the reference design by 10% and 4%. The reduction of the pipeline to 3 stages improves the total power consumption by 4%, but worsening the execution time by 25% and the energy consumption by 20%. The combination of AU + RP shows nearly the same impact as the RP system. Therefore, the reference system is the best choice.

In Figure 26 and Table 8, the impact to the performance and power consumption of the four different processor configurations compared to the reference system are presented for the ANN algorithm.

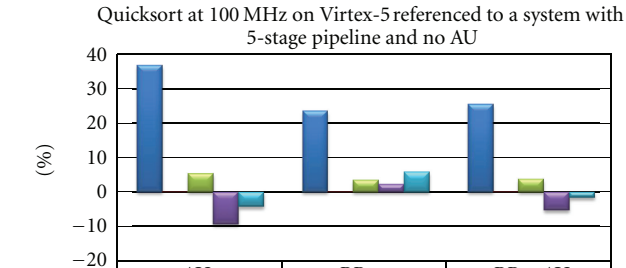
Adding an AU improves the execution time by 65% and reduces the overall power and energy consumption compared to the reference design by 3% and 66%. The reduction of the pipeline to 3 stages slightly improves the total power consumption by 4%, but worsening the execution time by 7%. The combination of AU + RP shows nearly the same impact as the AU system. Finally, the system with AU and without an FPU shows a slight improvement of the power consumption by 6% while the execution time is increased by 19%, and, also, the energy consumption is increased by over 11%. The reason for the longer execution time and, therefore, the higher energy consumption for the non-FPU system is due to the fact that the ANN uses floating point operations, which require much more clock cycles, if no FPU is provided. Therefore, the AU + FPU system is the best choice in terms of performance and energy consumption. If the power consumption is more important, then the RP + AU + FPU system would be the best choice for this algorithm.

Table 9: Impact of the MicroBlaze configurations for the Quicksort algorithm at 100 MHz for V5LX110T.

μB param	$P_{Dynamic}$ (mW)	$P_{Quiescent}$ (mW)	P_{Total} (mW)	P_{Total} (%)	T_{exe} (ms)	Energy (mJ)
Default	211	1230	1441	NA	3,02	4,35
AU	289	1231	1520	5,48	2,74	4,16
RP	261	1231	1492	3,54	3,09	4,61
RP + AU	265	1231	1496	3,82	2,86	4,28

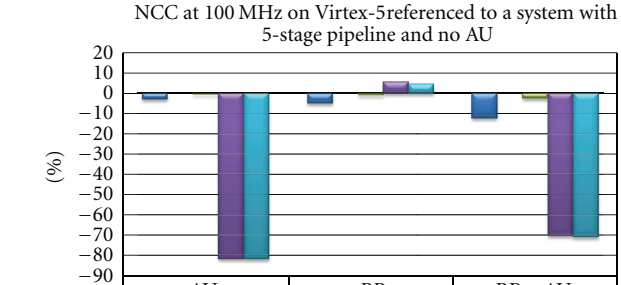
6.2.2. Virtex-5 LX110T FPGA. Figure 27 and Table 9 show the impact of the different configurations for the Quicksort algorithm. Due to the multiple comparators and the multiple branches in the algorithm, the combination of an arithmetic unit and a reduction of the pipeline stages is very beneficial in terms of execution time and energy consumption, but results in increased power consumption. If the power consumption is the critical factor, then the default system would be the best choice.

Figure 28 and Table 10 show the impact of the different configurations for the NCC algorithm. As this algorithm requires many arithmetic operations, the addition of an AU improves the overall execution time and the energy



	AU	RP	RP + AU
Pdynamic	36.97	23.7	25.59
Pquiescent	0.08	0.08	0.08
Ptotal	5.48	3.54	3.82
Execution time	−9.27	2.32	−5.3
Energy	−4.3	5.94	−1.68

Figure 27: Impact of the MicroBlaze configurations for the Quicksort algorithm.



	AU	RP	RP + AU
Pdynamic	−2.98	−4.97	−12.25
Pquiescent	0	−0.08	−0.08
Ptotal	−0.59	−0.98	−2.48
Execution time	−82.55	6.04	−71.08
Energy	−82.66	5	−71.8

Figure 28: Impact of the MicroBlaze configurations for the NCC algorithm.

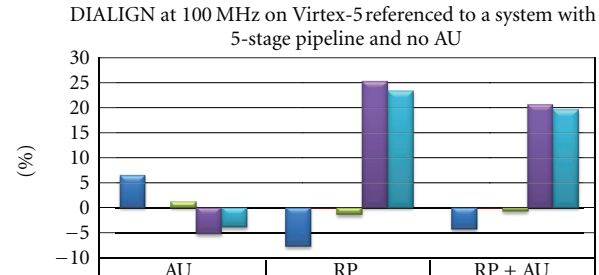
consumption by over 80% while the reduction of the pipeline stages results in a slight degradation. This degradation is due to the reason that the execution of arithmetic operations takes more clock cycles, if the pipeline is reduced. Therefore, for this and similar algorithms, a system with an AU and a 5-stage pipeline would be optimal from a performance and energy perspective. If the power consumption needs to be reduced and some performance degradation is acceptable, then the AU + RP system would be a good choice.

In Figure 29 and Table 11, the impact onto the performance and power consumption of the three different processor configurations compared to the reference system is presented for the DIALIGN algorithm.

Adding an AU improves the execution time and the energy consumption a little bit while slightly increasing the overall power consumption compared to the reference design. The reduction of the pipeline to 3 stages improves the total power consumption slightly, but worsening

Table 10: Impact of the MicroBlaze configurations for the NCC algorithm at 100 MHz for V5LX110T.

μB param	$P_{Dynamic}$ (mW)	$P_{Quiescent}$ (mW)	P_{Total} (mW)	P_{Total} (%)	T_{exe} (ms)	Energy (mJ)
Default	302	1232	1534	NA	316,4	485,36
AU	293	1232	1525	−0,59	55,2	84,18
RP	287	1231	1519	−0,98	335,5	509,62
RP + AU	265	1231	1496	−2,48	91,5	136,884



	AU	RP	RP + AU
Pdynamic	6.69	−7.69	−4.35
Pquiescent	0	−0.08	−0.08
Ptotal	1.37	−1.5	−0.85
Execution time	−5.28	25.62	20.91
Energy	−3.98	23.73	19.88

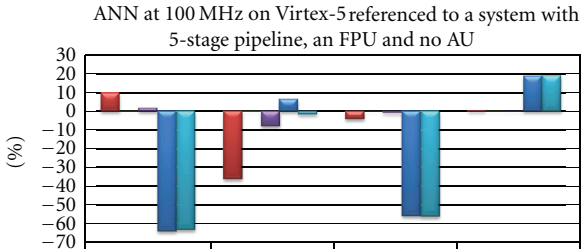
Figure 29: Impact of the MicroBlaze configurations for the DIALIGN algorithm.

Table 11: Impact of the MicroBlaze configurations for the DIALIGN algorithm at 100 MHz for V5LX110T.

μB param	$P_{Dynamic}$ (mW)	$P_{Quiescent}$ (mW)	P_{Total} (mW)	P_{Total} (%)	T_{exe} (ms)	Energy (mJ)
Default	299	1232	1530	NA	829,8	1270
AU	319	1232	1551	1,37	786	1219
RP	276	1231	1507	−1,50	1042,4	1571
RP + AU	286	1231	1517	−0,85	1003,3	1522

the execution time by 25% and, therefore, the energy consumption by over 23%. The combination of AU + RP shows nearly the same impact as the RP system. Therefore, the AU system is the best choice for these kinds of algorithms.

Figure 30 and Table 12 show the impact onto the performance and power consumption of the four different processor configurations compared to the reference system are presented for the ANN algorithm. Adding an AU improves the execution time by 65% and reduces the overall energy consumption compared to the reference design by 64% while the power consumption is increased slightly by 2%. The reduction of the pipeline to 3 stages improves the total power consumption by 8% and the energy consumption by 1.69%, but worsening the execution time by 7%. The combination of AU + RP shows nearly the same impact as the AU system. Finally, the system with AU and without an FPU shows a slight increase of the power consumption by 0.23% while



	AU + FPU	RP + FPU	RP + AU + FPU	AU + no FPU
Pdynamic	10.37	−36.71	−4.25	1.05
Pquiescent	0.06	−0.21	−0.02	0.01
Ptotal	2.3	−8.13	−0.94	0.23
Execution time	−64.95	7.02	−56.64	19.21
Energy	−64.14	−1.69	−57.05	19.49

Figure 30: Impact of the MicroBlaze configurations for the ANN algorithm.

Table 12: Impact of the MicroBlaze configurations for the ANN algorithm at 100 MHz for V5LX110T.

μB param	$P_{Dynamic}$ (mW)	$P_{Quiescent}$ (mW)	P_{Total} (mW)	P_{Total} (%)	T_{exe} (ms)	Energy (mJ)
Default + FPU	341,75	1232,59	1574,34	NA	2328	3665
AU + FPU	377,18	1233,33	1610,51	2,30	816	1314
RP + FPU	216,3	1229,98	1446,28	−8,13	2491	3603
RP + AU + FPU	327,21	1232,29	1559,49	−0,94	1009	1574
AU + no FPU	345,34	1232,67	1578,01	0,23	2775	4379

Table 13: Quicksort power and energy consumption.

	Uniprocessor (100 MHz)	Dual_2 (80/50 MHz)	Dual_5 (95 MHz)
Execution time-ms	18,42	18,80	19,27
Total power-mW	1576,93	1475,56	1570,79
Total power %	NA	−6,43	−0,39
Total energy-mJ	29,05	27,74	30,27
Total energy %	NA	−4,51	+4,20

the execution time and the energy consumption are both increased by 19%. The reason for the longer execution time and the higher energy consumption for the non-FPU system is that the ANN uses floating point operations, which require much more clock cycles, if no FPU is provided. Therefore, the AU + FPU system is the best choice in terms of performance and energy consumption. If the power consumption is more important, then the RP + AU + FPU system would be the best choice for this algorithm.

6.3. Impact of the Task Distribution and the Frequency Scaling. To measure the impact onto the power and energy consumption, the algorithms were partitioned onto two MicroBlaze processors on the Virtex-4 FPGA. The frequency for the two processors was chosen in such a way that the execution time of the dual-processor design was as similar as possible to the reference system consisting of a single MicroBlaze running at 100 MHz. For all systems, the configurations of the processors were fixed to a 5-stage pipeline, an integer multiplier, and the pattern comparator. For the reconfigurable clock, the first version using reconfiguration was chosen.

Table 14: NCC power and energy consumption.

	Uniprocessor (100 MHz)	Dual_3 (54 MHz)	Dual_2 (87.5/50 MHz)
Execution time-ms	67,74	67,28	67,62
Total power-mW	1472,78	1477,20	1504,02
Total power %	NA	+0,30	+2,12
Total energy-mJ	99,77	99,39	101,70
Total energy %	NA	−0,38	+1,93

Table 15: DIALIGN power and energy consumption.

	Uniprocessor (100 MHz)	Dual_5 (50 MHz)	Dual_6 (50 MHz)
Execution time-ms	30,21	30,16	30,16
Total power-mW	1569,69	1631,62	1536,44
Total power %	NA	+3,95	−2,12
Total energy-mJ	47,42	49,21	46,34
Total energy %	NA	+3,77	−2,28

Table 13 shows the results for distributing the Quicksort algorithm on two processors instead of one. Two partitions were done. The first one is called Dual_2 (80/50 MHz), which means that the virtual-IO 2 was used, and μB0 was running at 80 MHz while μB1 was running at 50 MHz. The algorithm was so partitioned that μB0 receives the whole data to be sorted. It then divides the data into two parts and sends the second part to μB1. Both then sort their partition. μB0 forwards its sorted part of the list to μB1, which sends the final combined sorted list via the virtual-IO 2 to the host PC. With this partition, the overall power consumption could be reduced by 6.43% and the energy consumption by 4.51% compared to the single processor reference system.

The second partition, called Dual_5 (95 MHz), uses the virtual-IO 5 to send incoming data to both processors running at 95 MHz. μB0 searches the list for elements smaller than and μB1 searches the list for elements bigger than, the pivot. When one has found an element the position of this element is sent to the other processor. Both processor then update their lists by swapping the own-found element with the one the other processor has found. At the end, both processors have, as a result, a searched list. μB0 then sends its resulting list back to the host PC via the virtual-IO 5. The power consumption of this version is nearly the same as the reference system while the total execution time and, therefore, the energy consumption increase.

Table 14 shows the result for the partitioning of the NCC algorithm onto two processors. The first partitioning uses the virtual-IO 3 to partition the incoming image into two

Table 16: First approach for a developers guideline.

Algorithm classifiers	Example algorithm	Processor configuration	Task distribution
(i) Comparators (ii) Branch and bound	Quicksort	RP + AU	Dual_2 (80/50 MHz)
(i) Complex arithmetic	NCC	AU	Uniprocessor (100 MHz)
(i) Comparators (ii) Basic arithmetic	DIALIGN	Default (V4), AU (V5)	Dual_6 (50 MHz)
(i) Floating point (ii) Complex arithmetic	ANN	AU + FPU	Needs to be explored

overlapping tiles, one for each processor. The overlapping part is sent to both processors simultaneously. As the NCC is a window-based image processing algorithm, the boarder pixels between the two tiles are needed by both processors. Each of the processors runs at 54 MHz, which results in a similar execution time, and also in a similar total power and energy consumption as the reference design.

The second partition, called Dual_2 (87,5/50 MHz), uses virtual-IO 2 to send the whole image to μB0. μB0 runs at 87.5 MHz and calculates the complete numerator and the denominator. Then, it forwards both to μB1, which does the division and sends the results back to the virtual-IO 2. μB1 runs at 50 MHz. While the execution time is nearly the same, the overall power consumption is increased slightly by 2.12% and the energy consumption by 1.93%.

Table 15 shows the result for executing the DIALIGN algorithm with two processors. Two partitions were done. The first one is called Dual_5 (50 MHz) and uses virtual-IO 5 to send the incoming sequences to both processors running at 50 MHz. Each processor calculates half of the resulting score matrix. μB0 calculates on a row-based fashion all values above the main diagonal. μB1 calculates in a column-based fashion all values below the main diagonal. The scores on the main diagonal are calculated by both processors. After μB0 has finished calculating one row and μB1 one column respectively, they exchange the first score nearest to the main diagonal, as this score is needed by both processors for calculating the next row/column, respectively. While the execution time is nearly the same, the overall power consumption is increased by 3.95% and the energy consumption by 3.77%.

The second partition is called Dual_6 (50 MHz). It uses the virtual-IO 6 to send the sequences to the processors, which run both at 50 MHz. Here, a systolic array approach is used for executing the DIALIGN algorithm. μB1 then sends the final alignment and the score back to the host PC. With this partition, the overall power consumption could be reduced by 2.12% and the energy consumption by 2.28% compared to the single processor reference system.

6.4. First Approach for a Developers Guideline. Table 16 is a first approach for a developer's guideline based on the exploration results done so far.

This guideline will be extended, by exploring more types of algorithms, more different FPGA families, and more detailed explorations for the processor configurations.

7. Conclusions and Outlook

This paper reports the research and evaluation of different microprocessor parameterization, application, and data partitioning on FPGA-based processor systems. Two different FPGA families are explored: Xilinx Virtex-4 and Virtex-5 FPGAs. The results of the experiments show the impact of the different parameterization on the power dissipation and energy consumption as well as performance in relation to a set of selected applications. Depending on the application type, it can be seen that different parameter configurations, for example configuration of the processors and their frequencies, but also a good application partitioning, are essential for achieving an efficient tradeoff between performance and power constraints. The results can be used to guide developers which parameter set suits to a certain application scenario, as was shown in Table 16. One important aspect studied in this work is the energy consumption of the different designs. In the experiments performed, it is noticed that the correct choice of the microprocessor configuration can lead to an economy of up to 90% of the energy consumption. This is significant especially regarding embedded applications, which normally depend on batteries to the power supply. Furthermore, the results show that for the selected FPGAs, DFS without any scheme to reduce the voltage is poorly interesting in terms of energy consumption. Under this condition for reducing the energy consumption, the policy should be compute as fast as possible and with the appropriate processor configuration and then to shut the power down.

The vision is that more application scenarios will be analyzed in order to provide a broad overview of the parameter impact. It is envisioned to extend existing hardware benchmarks from different application domains in terms of a parameterization guideline also for further FPGA series from Xilinx.

In addition, the paper provides a tutorial for the estimation of the power consumption on a high level of abstraction, but with a high accuracy through postplace and route simulation. Therefore, other research in this area can be done and exchanged in the community.

Acknowledgments

The authors would like to thank Professor Alba Cristina M. A. De Melo and Jan Mendonca Correa for providing

them with their C code implementation of the DIALIGN algorithm.

References

[1] "Xilinx MicroBlaze Reference Guide," UG081 (v7.0), September 2006, http://www.xilinx.com/.

[2] D. Meintanis and I. Papaefstathiou, "Power consumption estimations vs measurements for FPGA-based security cores," in *Proceedings of the International Conference on Reconfigurable Computing and FPGAs (ReConFig '08)*, pp. 433–437, Cancun, Mexico, December 2008.

[3] J. Becker, M. Huebner, and M. Ullmann, "Power estimation and power measurement of Xilinx virtex FPGAs: trade-offs and limitations," in *Proceedings of the 16th Symposium on Integrated Circuits and Systems Design (SBCCI '03)*, Sao Paulo, Brazil, September 2003.

[4] K. Poon, A. Yan, and S. J. E. Wilton, "A flexible power model for FPGAs," in *Proceedings of the 12th International Conference on Field-Programmable Logic and Applications (FPL '02)*, September 2002.

[5] F. N. Najm, "Transition density: a new measure of activity in digital circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 2, pp. 310–323, 1993.

[6] K. Weiss, C. Oetker, I. Katchan, T. Steckstor, and W. Rosenstiel, "Power estimation approach for SRAM-based FPGAs," in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA '00)*, pp. 195–202, Monterey, Calif, USA, February 2000.

[7] V. Degalahal and T. Tuan, "Methodology for high level estimation of FPGA power consumption," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC '05)*, Shanghai, China, January 2005.

[8] "Xilinx Power Estimator User Guide," UG440 (v3.0), June 2009, http://www.xilinx.com/.

[9] "Development System Reference Guide," v9.2i, Chapter 10 XPower, http://www.xilinx.com/.

[10] "Embedded System Tools Reference Manual," Embedded Development Kit, EDK 9.2i, UG111 (v9.2i), Chapter 3, September 2007, http://www.xilinx.com/.

[11] "Fast Simplex Link (FSL) Bus (v2.00a)," DS449 December 2005, http://www.xilinx.com/.

[12] D. Göhringer, J. Obie, M. Hübner, and J. Becker, "Impact of task distribution, processor configurations and dynamic clock frequency scaling on the power consumption of FPGA-based multiprocessors," in *Proceedings of the 5th International Workshop on Reconfigurable Communication Centric Systems-on-Chip (ReCoSoC '10)*, Karlsruhe, Germany, May 2010.

[13] "Virtex-4 FPGA Configuration User Guide," UG071 (v1.11), June 2009, http://www.xilinx.com/.

[14] "Virtex-4 FPGA User Guide," UG070 (v2.6), December 2008, http://www.xilinx.com/.

[15] C. A. R. Hoare, "Quicksort," *Computer Journal*, vol. 5, no. 1, pp. 10–15, 1962.

[16] A. Boukerche, J. M. Correa, A. C. M. Melo, and R. P. Jacobi, "A hardware accelerator for the fast retrieval of DIALIGN biological sequence alignments in linear space," *IEEE Transactions on Computers*, vol. 59, no. 6, pp. 808–821, 2010.

[17] R. Palacios and A. Gupta, "A system for processing handwritten bank checks automatically," *Image and Vision Computing*, vol. 26, no. 10, pp. 1297–1313, 2008.

[18] "Alpha-Data," http://www.alpha-data.com/.