Research Article

Reconfiguration Techniques for Self-X Power and Performance Management on Xilinx Virtex-II/Virtex-II-Pro FPGAs

Christian Schuck, Bastian Haetzer, and Jürgen Becker

Institut für Technik der Informationsverarbeitung (ITIV), Karlsruher Institut für Technologie (KIT), Vincenz-Priessnitz-Straße 1, 76131 Karlsruhe, Germany

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

Received 29 August 2010; Accepted 14 December 2010

Academic Editor: Michael Hübner

Copyright © 2011 Christian Schuck et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Xilinx Virtex-II family FPGAs support an advanced low-skew clock distribution network with numerous global clock nets to support high-speed mixed frequency designs. Digital Clock Managers in combination with Global Clock Buffers are already in place to generate the desired frequency and to drive the clock networks with different sources, respectively. Currently, almost all designs run at a fixed clock frequency determined statically during design time. Such systems cannot take the full advantage of partial and dynamic self-reconfiguration. Therefore, we introduce a new methodology that allows the implemented hardware to dynamically self-adopt the clock frequency during runtime by reconfiguring the Digital Clock Managers. We also present a method for online speed monitoring which is based on a two-dimensional online routing. The created speed maps of the FPGA area can be used as an input for the dynamic frequency scaling. Figures for reconfiguration performance and power savings are given. Further, the tradeoffs for reconfiguration effort using this method are evaluated. Results show the high potential and importance of the distributed dynamic frequency scaling method with little additional overhead.

1. Introduction

Xilinx Virtex FPGAs have been designed with high-performance applications in mind. They feature several dedicated Digital Clock Managers (DCMs) and Digital Clock Buffers for solving high-speed clock distribution problems. Multiple clock nets are supported to enable highly heterogeneous mixed frequency designs. Usually all clock frequencies for the single clock nets and the parameters for the DCMs are determined during design time through static timing analysis. Targeting maximum performance these parameters strongly depend on the longest combinatorial path (critical path) between two storage elements. For minimum power the required throughput of the design unit determines the lower boundary of the possible clock frequency. In both cases nonadjusted clock frequencies lead to waste of either processing power or energy [1, 2].

Considering the feature of partial and dynamic self-reconfiguration of Xilinx Virtex FPGAs, during runtime a high dynamic and flexibility arises. Static analysis methods

are no longer able to sufficiently determine an adjusted clock frequency during design time. At the same time a new partial module is reconfigured onto the FPGA grid, its critical path changes, and in turn the clock frequency has to be adjusted as well during runtime to fit the new critical path. On the other side the throughput requirement of the application or the environmental conditions may change over time making an adjustment of clock frequency necessary.

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 [3] the Digital on Demand Computing Organism (DodOrg) was proposed, which is derived from a biological organism. 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. The hardware architecture of the DodOrg system consists of many heterogeneous so-called Organic Processing Cells (OPCs) that communicate through the artNoC [4] router network as shown in Figure 1. In general, all OPCs are made of the same

blueprint. On the one side they contain a highly adaptive heterogeneous data path and on the other side several common units are responsible for the organic behaviour of the architecture. Among them a Power Management Unit is able to perform dynamic frequency scaling (DFS) on OPC level. It can control and adjust performance and power consumption of the cell according to the actual computational demands of the application and the critical path of the cell's data path. 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. Hence, the cell's clock domain is decoupled by the network interface and can operate independently from the artNoC and the other OPCs of the organic chip.

In [5], we presented a prototype implementation of the DodOrg architecture on a Virtex FPGA, where it is possible to dynamically change the cells data path through a 2-dimensional partial and dynamic reconfiguration. Therefore, a novel IP core, the Virtual-ICAP-Interface, was developed in order to perform a fast 2-dimensional self-reconfiguration and provide a virtual decentralisation of the internal FPGA configuration access port (ICAP). This paper enhances the methodology by enabling the partial and dynamic self-reconfiguration of the Virtex DCMs, which is inherently not given, through the Virtual-ICAP-Interface. As a result, the desired self-adoption with respect to a fine grained power management could be achieved. Further, an application for the 2-dimensional self-reconfiguration is given by presenting an online speed monitoring method in Section 5.

The rest of the paper is organized as follows. Section 2 reviews several other proposals for DFS on FPGAs while Section 3 summarizes important aspects of the Xilinx Virtex II FPGA clock net infrastructure. Section 4 describes the details of our approach to dynamically reconfigure the DCMs during runtime. Section 6 shows DCM reconfiguration performance and power saving figures. Finally, Section 7 concludes the work and gives an outlook to future work.

2. Related Work

Recently, several works have 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 [2, 6].

In [1], the authors show that even because of FPGA process variations and because of changing environmental conditions (hot, normal, and cold temperature) dynamically clocking designs can lead to a speed improvement of up to 86% compared to using a fixed, statically estimated clock during design time. The authors use an external programmable 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 [6], the authors proposed an online solution for clock gating. They propose 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

FIGURE 1: DodOrg organic hardware architecture overview.

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 [7] 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. All advantages of the high-speed clock distribution network could be maintained.

3. Xilinx Clock Architecture

This section gives a brief overview over the Xilinx Virtex-II clock architecture as our work makes extensive use of the provided features.

3.1. Clock Network Grid. Besides normal routing resources Xilinx Virtex-II FPGAs have a dedicated low-skew high-speed clock distribution network [8, 9]. They feature 16 global clock buffers (BUFGMUX, see Section 3.3) and support up to 16 global clock domains (Figure 2). The FPGA grid is partitioned into 4 quadrants (NW, SE, SW, and NE) with up to 8 clocks per quadrant. Eight clock buffers are in the middle of the top edge and eight are in the middle of the bottom edge. In principle any of these 16 clock buffer outputs can be used in any quadrant as long as opposite clock buffers on the top and on the bottom are not used in the same quadrant, that is, there is no conflict [9]. In addition, up to 12 DCMs are available. They can be used to drive clock buffers with different clock frequencies. In the following important features of the DCMs and clock buffers will be summarized.

3.2. Digital Clock Managers. Besides others, frequency synthesis is an important feature of the DCMs. Therefore, 2

FIGURE 2: Simplified view of the Xilinx Virtex II clock distribution network.

main different programmable outputs are available. CLKDV provides an output frequency that is a fraction ($\div 1.5, \div 2, \div 2.5 \dots \div 7, \div 7.5, \div 8, \div 9 \dots \div 16$) of the input frequency CLKIN.

CLKFX is able to produce an output frequency that is synthesised by combination of a specified integer multiplier $M \in \{2 \dots 32\}$ and a specified integer divisor $D \in \{1 \dots 32\}$ by calculating $\text{CLKFX} = M \div D * \text{CLKIN}$.

3.3. Global Clock Buffer. Global Clock Buffers have three different functionalities. In addition to pure clock distribution, they can also be configured as a global clock buffer with a clock enabler (BUFGCE). Then the clock can be stopped at any time at the clock buffer output.

Further, clock buffers can be configured to act as a “glitch-free” synchronous 2:1 multiplexer (BUFGMUX). These multiplexers are capable of switching between two clock sources at any time, by using the select input that can be driven by user logic. No particular phase relations between the two clocks are needed. For example, as shown in Figure 3 they can be configured to switch between two DCM CLKFX outputs. As we will see in the next section, our design makes use of this feature.

4. Organic System Architecture

Compared to μC ASIC solutions, SRAM-based FPGAs like Virtex-II consume a multiple of power. This is due to the fine-grained flexibility and adaptability and the involved overhead. By just using these features during design time to create a static design, most of the potential remains unused. Instead dynamic and partial online self-reconfiguration during runtime is a promising approach to exploit the full potential and even to close the energy gap. Therefore, in

FIGURE 3: Example BUFGMUX /DCM configuration.

[5], we proposed to implement the OPC-based organic computing organisms on a Virtex-II Pro FPGA as shown in Figure 4.

This paper focuses on the power-related issues of the cell-based DodOrg architecture on the FPGA prototype. Important aspects to reach the desired goal of a fine-grained, decentralized self-adaptive power management will be discussed in the subsequent subsections.

4.1. Clock Partitioning. Depending on the size of the device several OPCs are mapped onto a single FPGA (Figure 4). The clock net of the highly adaptive data path (DP) of every OPC is connected to a BUFGMUX that is driven by a pair of DCMs. There is a power management unit (PMU) inside every OPC, which is connected to the select input of the BUFGMUX. So it can quickly choose between the two DCM clock sources. The DP-clock is decoupled from the artNoC-clock by using a dual-ported dual-clock FIFO buffer. Further, the PMU is connected to the artNoC. Thus, it is able to exchange power-related information with the other PMUs. Beyond that, it has access to the Virtual-ICAP-Interface. Therefore, during runtime every PMU can dynamically adapt the DCM CLKFX output clock frequency through partial self-reconfiguration by using the features of the Virtual-ICAP-Interface.

4.2. Virtual-ICAP-Interface. The Virtual-ICAP-Interface is a small and lightweight IP, which on the one side acts as a wrapper around the native hardware ICAP and on the other side connects to the artNoC network. It provides a virtual decentralisation of the ICAP as well as an abstraction of the physical reconfiguration memory. Its main purpose is to perform the Readback-Modify-Writeback (RMW) method in hardware. Therefore, a fast and true 2-dimensional reconfiguration of all FPGA resources is possible, that is, reconfiguration is no longer restricted to columns. Due to its partitioning into two clock domains, one clock domain for the artNoC controller side and one clock domain for the ICAP side, maximal reconfiguration performance could be achieved [5].

Every bit within the reconfiguration memory can be reconfigured independently the configuration of the DCMs

FIGURE 4: DodOrg FPGA floor plan/clock architecture.

can be altered as well. However, a special procedure is necessary which is described in the next subsection.

4.3. DCM Reconfiguration Details. During reconfiguration of DCMs it is important that a glitchless switching from one clock frequency to another can be guaranteed. In general, after initial setup the CLKDV and CLKFX outputs are only enabled when a stable clock is established. After that, the DCM is locked to the configured frequency, as long as the jitters of the input clock CLKIN stay in a given tolerance range [9]. For our scenario we assume that the input clock is stable.

If we change the DCM configuration (D , M) in configuration memory to switch from one clock frequency to a different frequency while the DCM is locked, it loses the lock and no stable output, that is, no output can be guaranteed. Therefore, to ensure a consistent locking to the new frequency the following steps have to be performed:

- (1) stop the DCM by writing a zero configuration ($D = 0$, $M = 0$),
- (2) write the new configuration ($D = D_{\text{new}}$, $M = M_{\text{new}}$).

To simplify the handling of the DCM reconfiguration this two-step procedure is internally executed by the Virtual-ICAP-Interface. It therefore features a special DCM addressing mode, for an easy access to the DCM configuration bits. Figure 5 shows a plot of a DCM reconfiguration procedure performed by the Virtual-ICAP-Interface. The plots were recorded by a 4-channel digital oscilloscope with all important signals routed to FPGA pins. Figure 5(a) shows the ICAP enable signal that is asserted by the Virtual-ICAP-Interface during ICAP read and write operation. It is

an indicator for the overall duration of the reconfiguration procedure. It strongly depends on the device size or rather on the configuration frame length. In this case a Virtex-II Pro XC2VP30 device was used with a frame length of 824 Bytes. For reconfiguration of the DCM just a single configuration frame has to be processed. From the beginning of the ICAP enable low phase to the spike in Figure 5(a) the configuration frame is read back from the configuration memory.

Then, the ICAP is configured to write mode and the zero configuration to shut off the DCM is written followed by a dummy frame to flush the ICAP input register. As soon as the writing of the dummy frame is finished the DCM stops. Figure 5(c) shows a zoom of the DCM CLKFX output (Figure 5(b)) at this point in time. We see that the DCM CLKFX was running at 6.25 MHz and stops without any jitter or spikes. Immediately after the dummy frame, the read back frame which has been merged with the new DCM parameters is written back to the ICAP followed by a second dummy frame. As soon as the dummy frame is processed the DCM CLKFX output runs with the new frequency in this case 8.33 MHz. Figure 5(d) shows a zoom of this point in time. Again no glitches or spikes occur. The overall processing time for a complete DCM reconfiguration in this case is 60.7 μs . In general, the reconfiguration time for a different Virtex-II family device is given by

$$\begin{aligned}
 t_{\text{DCM}} &\sim \text{frame length [Byte]} * \frac{2}{67.7} [\text{Byte}/\mu\text{s}] \\
 &+ \text{frame_length [Byte]} * \frac{4}{90.5} [\text{Byte}/\mu\text{s}] \quad (1) \\
 &(@\text{ICAP_CLK} = 100 \text{ MHz}).
 \end{aligned}$$

FIGURE 5: DCM reconfiguration performance: (a) ICAP enable, (b) DCM CLKFX output, (c) zoom of DCM CLKFX output before reconfiguration, and (d) zoom of DCM CLKFX output after reconfiguration.

FIGURE 6: BUFGMUX clock switching: bufgmux_sel signal and bufgmux_output signal.

The two summands in the formula are resulting from the fact that ICAP has different throughputs for reading and writing reconfiguration data [5].

Therefore, this procedure presents a save method to dynamically reconfigure DCMs during runtime. However, even if self-adaptive decentralized DFS can be realized with the presented method two main drawbacks are obvious:

- (1) relatively long setup delay until the new frequency is valid (in this example: 60.7 μs),
- (2) interruption of clock frequency during reconfiguration (in this example: 18.2 μs).

This means that the method is appropriate for reaching long-term or intermediate-term power management goals, that is, a new data path is configured and the clock frequency is adapted to its critical path and then stays constant until a new data path 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 3, 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, as shown in Figure 6. Further the interruption of clock frequency during reconfiguration can be hidden. By a combination of both techniques a broad spectrum of different clock frequencies as well as an immediate uninterrupted switching is available.

5. Speed Monitoring

In the preceding section we presented the method to reconfigure the DCMs on Xilinx Virtex-II FPGAs during runtime. If the runtime optimization goal is to run the design at maximum clock speed, in order to achieve the maximum system performance, the question arises: “what is actually the maximum clock frequency that the reconfigurable datapath can run

- (1) in the specific situation (temperature, supply voltage)?
- (2) in the specific reconfigurable area?
- (3) on a specific FPGA?

TABLE 1: OSC macrofrequency (MHz).

	Macro 1	Macro 2	Macro 3
Real	105	91.2	78.9
STA 20C	88	76	65.3
STA 85C	82.7	71.5	61.1
Min gap	17	15.2	13.6
Min gap %	19.3	20	20.8

Static timing analysis during design time can only determine a worst case scenario, but it is not possible to consider factors like

- (1) device-specific local speed variations,
- (2) variable or instable device power supply,
- (3) different ambient temperatures.

In the following section we present a method of online speed monitoring which takes the device- and environment-specific factors into account. Hence, it is possible to determine the maximum clock frequency of a PRM depending on the current situation. The results can be used as an input for the DCM reconfiguration presented in Section 4.3.

5.1. Measurement Method. The measurement method is based on a ring oscillator monitoring module (OSC). Figure 7(a) shows the basic structure of the used ring oscillator. It is composed of an inverter and a delay line connected as a ring. This circuit oscillates with the frequency $1/(2 * t_{\text{delay}})$. To decouple the ring oscillator from the fan-out network a toggle flip flop was added as a second stage to keep the fan-out of the oscillator constant at one. For further calculations it has to be considered that the second stage divides oscillator frequency by two.

The actual layout of the oscillator macro 1 is shown in Figure 7(b). It was built with the Xilinx FPGA Editor, as a so-called hard macro. The goal was to build it very compact. For the delay line and the inverter we used 6 LUTs of the top 3 slices. The fourth slice is used to realize the decoupling toggle flip flop. The delay of the delay line is comprised of the propagation delay of the LUTs and CLB internal wire delays between the LUTs.

For our measurements we build 3 different oscillator macros (macro 1–3) with different delays of the delay line. This was achieved by altering the connection of the 6 delay LUTs and therefore altering the used routing resources.

5.2. Experiment 1: Static Timing Analysis. In our first measurement we performed a static timing analysis (STA) of each of the oscillator macros to determine the theoretical oscillator frequency and compared the results with real measurement on board. The results are shown in Table 1. The STA was performed with the following settings: temperature: 20°C/85°C and voltage: 1.4 V. The real measurements on the board were performed with a room temperature of ca. 20°C and a voltage of 1.497 V.

The results show that at least a speed gap of 19% exists. During the measurements we noticed that the voltage

FIGURE 7: Oscillator macro zoom: (a) schematic and (b) physical FPGA Editor layout.

parameter in the Xilinx Timing Analyzer V.9.103i has no effect on the STA results, even if it can be set. In later versions of the Xilinx Timing Analyzer the voltage parameter is taken into account but the Virtex-II series are no longer supported.

5.3. Experiment 2: Supply Voltage Speed Variation. Therefore, in our second experiment we did the same measurement with OSC-macro 2, but this time we altered the supply voltage and we did the measurement with 5 different but identical FPGA boards (board 1–5). The results of the oscillator frequency

TABLE 2: OSC macro 2 frequency (MHz).

Device id /voltage[V]	1.10	1.20	1.30	1.40	1.50	1.60
Board 1	66.304	72.064	75.904	78.08	78.592	77.696
Board 2	67.456	72.704	76.16	77.824	77.952	77.312
Board 3	65.536	71.04	74.752	76.928	77.824	77.44
Board 4	66.432	72.448	76.672	79.232	80.384	80
Board 5	66.688	72.192	76.416	78.464	79.104	78.208
Mean	66.4832	72.0896	75.9808	78.1056	78.7712	78.1312
Min	65.536	71.04	74.752	76.928	77.824	77.312
Max	67.456	72.704	76.672	79.232	80.384	80
Delta	1.92	1.664	1.92	2.304	2.56	2.688

are shown in Table 2. We altered the supply voltage in discrete steps of 0.1 V from the minimum voltage of 1.1 V to the maximum voltage of 1.6 V. Looking at a single board the biggest frequency delta across the different supply voltages was detected at board 3 with 13.95 MHz or 18.4%. The smallest delta was detected at board 5 with 10.49 MHz or 14% of the mean value.

Looking at different boards at the same supply voltage the biggest delta in frequency between two boards was detected at 1.6 V with a delta of 2.69 MHz or 3.44% of the mean value. The smallest delta was detected at 1.2 V with a delta of 1.66 MHz or 2.31% of the mean value. To summarize this experiment it can be said that

- (1) supply voltage has a great impact on the device speed,
- (2) even if we just measured 5 different devices this exemplary shows that there is potential for speed improvement if the clock frequency can be tailored device specific.

If we compare these results with the first experiment, we can derive that the Xilinx Timing Analyzer database assumes for the STA a worst case supply voltage of 1.3 V.

5.4. Experiment 3: Local Speed Variations. In Section 5.3 we have seen device-specific speed characteristics, probably based on fabrication variations. The question arises if there are even some regions on the device which are faster than other regions on the same device. In a third experiment we wanted to answer this question. Therefore, we constructed an automatic measurement system with the following main characteristics:

- (i) fast measurement of a large number of CLBs,
- (ii) on-Chip solution that can also be used for online self-monitoring in our organic system.

The key technique of the measurement system is based on a repeated 2-dimensional partial self-reconfiguration by using the Virtual-ICAP-Interface (see Section 4.2). In a first step the oscillator macro is placed at the CLB that should be monitored. In a second step an XY-online routing based on double lines is performed, in order to connect the frequency output of the oscillator macro with the measurement unit that determines the frequency. In the following, our method

FIGURE 8: Speed measurement system layout.

to characterize a complete device will be presented in detail followed by the discussion of the results.

Figure 8 shows the basic structure of the example measurement system on an XC2VP30 FPGA. It consists of the static measurement system on the bottom right corner. It contains a MicroBlaze (MB) soft-core processor (1) that controls the reconfiguration and measurement process. It has connections via FSL with the Virtual-ICAP-Interface (VICAP) (2) for reconfiguration, the frequency measurement unit (FMU) and an UART (3) to communicate with a host system (HS). The FMU is connected to the frequency output of the oscillator macro.

The frequency measurement of a single CLB consists of the following steps:

- (i) placement of the oscillator macro to target CLB (MB/VICAP),
- (ii) online routing to connect oscillator macro with frequency measurement unit (MB/VICAP),
- (iii) trigger the start of the measurement via FSL (MB),
- (iv) frequency measurement (FMU),
- (v) transmit the measurement result via FSL to MB (FMU),
- (vi) transmit the result via UART to host system (MB),
- (vii) store data into database (HS).

FIGURE 9: Zoom of online routing macros.

In the following, the basic technique of the used online routing strategy will be presented briefly. Figure 9 shows a zoom of a connection between the oscillator macro and the frequency measurement unit. The online routing is based on double lines only, in order to simplify the routing algorithm. Six different types of routing connections are used as marked in Figure 9:

- T1: connects the middle of a horizontal double line with the start of the next horizontal double line,
- T2: connects the end of a horizontal double line with the start of the next horizontal double line,
- T3: down corner: connects the end of a horizontal double line with the start of a vertical double line,
- T4: connects the end of a vertical double line with the start of the next vertical double line,
- T5: connects the middle of a vertical double line with the start of the next vertical double line,
- T6: right corner: connects the middle of a vertical double line with the start of a horizontal double line.

The output of the oscillator macro is routed to the middle connection of a horizontal double line. At the other end a so called “routing base” macro also connects to the middle connection of a horizontal double line, in order to ensure a fixed connection point for the routing algorithm. The output of the routing base macro is connected to the frequency measurement unit. The path between the output of the oscillator macro and the input of the “routing base” can be established from every CLB north-west of the routing base by reconfiguring the routing connections T1–T6 along the path. Similar routing connections could be created for other directions.

To reconfigure one routing connection just 2 FPGA configuration frames need to be processed by the Virtual-ICAP Interface. Therefore, without optimization (e.g., reconfiguring multiple vertical routing connections simultaneously)

each routing connection needs $80\mu\text{s}$ to be set. For the oscillator macro all 22 frames of a CLB are reconfigured, which takes $880\mu\text{s}$. For example, to place the oscillator macro and to establish the routing connection as shown in Figure 9 it takes

$$8 * 80\mu\text{s} + 880\mu\text{s} = 1520\mu\text{s}. \quad (2)$$

The configuration data for the routing types T1–T6 and the oscillator macro are stored as constants in the MicroBlaze program memory. All configuration data together require 302 bytes of memory. Because of its short reconfiguration times and little memory requirements the approach is suitable to be used for online speed monitoring of larger areas, for example, the reconfigurable area of an OPC.

In Figures 11, 12, and 13 the results for experiment 3 of boards 1–5 are shown. Boards 2 and 5 as well as boards 1, 3, and 4 show similar speed maps. At all boards the slowest CLBs are located along the edges of the device. Towards the middle of the device the speed gradually increases so that the fastest CLBs are located along the middle axes. The biggest frequency delta between maximum (106 MHz) and minimum frequency (101 MHz) shows board 2 with 5 MHz and a variance of 0.5 MHz (sample size 1440 CLBs). Figure 10 shows the distribution of the OSC speed for all boards. We repeated the measurements with macro 2 and macro 3 (different routing resources) for all boards (not shown here) and got very similar speed maps. That means fast CLBs remained fast and slow CLBs remained slow.

With respect to a real critical path which is mapped either to a region with fast CLBs or a region with slow CLBs the results become more relevant as faster the design runs. For example, on board 2 for a 100 MHz design the frequency delta is 4–5 MHz, whereas for a 200 MHz design the frequency delta grows proportional to ca. 8–10 MHz. This means in turn that for slower designs these effects can be more and more neglected.

To summarize the experiment, it can be said that local and device-specific speed variation could be measured. Their

FIGURE 10: Distribution of OSC speed board 1–5.

impact compared to speed variations caused by temperature and supply voltage variations is quite low but will become more important as the speed of the designs increases. Especially, if further technology scaling causes bigger process variations, which leads to an increase of local speed variations, the presented online monitoring method becomes more relevant. However, even for Virtex-II FPGAs it can be used for fine tuning the DCM clock speed according to the speed of the region the design should run.

6. Power and Resource Considerations

In Section 4.3 the results for DCM reconfiguration times and tradeoffs have already 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 DCM reconfiguration approach (PM, Virtual-ICAP-Interface, and DCM) is taken into account.

6.1. Test Setup Power Measurement. We calculated the power consumption by measuring the voltage drop over an external shunt resistor (0.4 Ohm) on the FPGA core voltage (FPGA VINT). As a test system 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 3, several distinct designs have been synthesised.

For DCM measurement an array of toggle flip flops at 100 MHz with and without a DCM in the clock tree have been recorded and the difference of both values has been taken. For extracting ICAP power consumption a system consisting of PM, Virtual-ICAP-Interface, and ICAP instance and a second identical system but without ICAP instance have been implemented. After activation the PM sends bursts

TABLE 3: Component power consumption.

	“Passive” Power [mW]	“Active” Power [mW]
static_offset	—	11
DCM	—	37
PM	<1	<1
artNoC-ICAP-IF	<1	9
ICAP	69	76

TABLE 4: Resource utilization Virtual-ICAP-Interface.

Resource	Number	Percentage
Slices	364	2%
Slice flip flops	177	0%
4 input LUTs	664	2%
BRAMs	1	0%
MULT18 × 18 s	2	1%

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 representative for a partial reconfiguration. Therefore, before PM 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 4.3 with two DCMs+BUFGMUX consume 196 mW when active, that is, 180 mW when passive. But it has to be considered that Virtual-ICAP-Interface as well as ICAP is also used for partial 2D reconfiguration.

6.2. Area and Resource Utilization. The resource requirement for the Virtual-ICAP-Interface with DCM reconfiguration mode is shown in Table 4.

6.3. Power Performance Evaluation. To put the previous power figures into a context we determined the power consumption of a MicroBlaze soft-core processor at different clock frequencies as shown in Figure 14. As we can see there is a high potential for power savings (e.g., the difference in power consumption in idle state between 100 MHz and 50 MHz is 170 mW).

The overhead (ICAP+VICAP) for DCM reconfiguration in a static design is in the range of an MB operating at 20 MHz. As expected, we see that there is a linear dependency between clock frequency and power consumption. Therefore, the energy consumed per clock cycle, $E = P/f_{clk}$; $f_{clk} = c * P$, is constant for all clock frequencies. This means, in terms of power savings for a static data path, there is no point for using reconfiguration of DCMs. A setup of DCM_{fmax} and BUFGCE to toggle between $f = f_{max}$ and $f = 0$ is most appropriate. In terms of performance, DCM reconfiguration can be used to evaluate maximum clock frequency during runtime.

FIGURE 11: OSC frequency map (MHz) board 1/board 2.

In turn, in a dynamic scenario, where the data path and therefore also the critical path change, DCM reconfiguration is necessary to achieve maximum module performance. It also comes without any additional overhead as ICAP+VICAP+DCM are already needed for reconfiguration. The capability of DCM reconfiguration together with BUFGMUX provides the basis for fine-grained short- or long-term power management strategies.

7. Summary and Future Work

In this paper we have presented a novel methodology to dynamically reconfigure Digital Clock Managers on Xilinx Virtex-II devices through ICAP. On one side optimal performance of partial modules and on the other side the goal of uniform power consumption can be achieved without external hardware. Our measurements show that power

FIGURE 12: OSC frequency map (MHz) board 3/board 4.

consumed by the components of the proposed hardware framework, especially the DCMs itself, is not negligible and has to be counterweighted. With DCM reconfiguration times in the range of $60\mu s$ long term power management goals can be reached. We also provide figures for reconfiguration times as well resource utilization.

Online on board speed measurements show the potential of device and situation dependent adjusted clock frequencies

during runtime. Compared to static timing analysis performed at design time at least 19% of speed improvement can be achieved. A speed analysis of five different boards at different supply voltages shows that a voltage variation of 0.5 V results in a speed variation of 18.4%.

In order to detect FPGA region-specific speed characteristics a new lightweight on-chip online routing method is presented which allows to scan complete FPGA areas. Results

FIGURE 13: OSC frequency map (MHz) board 5.

show that region-specific speed characteristics exist that can be used for fine tuning the design with respect to maximum performance.

Future work is targeting towards the integration of the speed monitoring into the PM and examination of the system level power saving effect resulting from distributed power management with multiple PM and multiple clock domains.

FIGURE 14: MicroBlaze power measurement.

References

- [1] 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, special issue on FPGAs.
- [2] I. Brynjolfson and Z. Zilic, "FPGA clock management for low power," in *Proceedings of International Symposium on Field-Programmable Gate Arrays (FPGA '00)*, 2000.
- [3] J. Becker, K. Brändle, U. Brinkschulte 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., 2006.
- [4] C. Schuck, S. Lamparth, and J. Becker, "artNoC—a novel multi-functional router architecture for organic computing," in *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL '07)*, pp. 371–376, August 2007.
- [5] C. Schuck, B. Haetzer, and J. Becker, "An interface for a decentralized 2D-reconfiguration on Xilinx Virtex-FPGAs for organic computing," in *Proceedings of Reconfigurable Communication-Centric SoCs (ReCoSoC '08)*, 2008.
- [6] Z. Yan, 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, September 2006.
- [7] I. Brynjolfson and Z. Zilic, "Dynamic clock management for low power applications in FPGAs," in *Proceedings of the 22nd Annual Custom Integrated Circuits Conference (CICC '00)*, pp. 139–142, May 2000.
- [8] "Xilinx Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet," DS083 (v4.7) November 2007.
- [9] "Xilinx Virtex-II Pro and Virtex-II Pro X FPGA User Guide," UG012 (V4.2) November 2007.