# UPaRC—**U**ltra-Fast **P**ower-**a**ware **R**econfiguration **C**ontroller

Robin Bonamy, Hung-Manh Pham, Sébastien Pillement, and Daniel Chillet
University of Rennes 1 – CAIRN IRISA/INRIA,
6 rue de Kerampont
F-22300 Lannion
*firstname.lastname*@irisa.fr

***Abstract*—Dynamically reconfigurable architectures, which can offer high performance, are increasingly used in different domains. High-speed reconfiguration process can be carried out by operating at high frequency but can also augment the power consumption. Thus the effort on increasing performance by accelerating the reconfiguration should take into account power consumption constraints. In this paper, we present an ultra-fast power-aware reconfiguration controller (UPaRC) to boost the reconfiguration throughput up to 1.433 GB/s. UPaRC can not only enhance the system performance, but also auto-adapt to various performance and consumption conditions. This could enlarge the range of applications and optimize for each selected application during run-time. An investigation of reconfiguration bandwidths at different frequencies and with different bitstream sizes are experimentally quantified and presented. The power consumption measurements are also realized to emphasize energy-efficiency of UPaRC over state-of-the-art reconfiguration controllers—up to 45 times more efficient.**

***Index Terms*—dynamic partial reconfiguration, rapid reconfiguration speed, power consumption, ICAP**

## I. Introduction

Reconfigurable architectures like FPGAs are well known to be able to design flexible, high-performance systems with high scalability at low development cost. Partial reconfiguration, the feature to modify on-the-fly a part of the circuit without interrupting the rest, enhances yet the above advantages while offering a more energy-efficient solution thanks to hardware resources sharing [1]. The auto-adaptation using partial reconfiguration is usually handled by a reconfiguration controller which loads the partial bitstream to the reconfiguration port to modify the functionality of a part on the FPGA. That requires a controller to drive the reconfiguration process. This partial modification does not affect the rest of the circuit but the area to be modified is inactive during reconfiguration. A long inactive period of a part inside a system may be prohibited in certain applications especially in high-performance or fault-tolerant systems. Thus the reconfiguration controller should provide high reconfiguration speed to accelerate the hardware adaptation.

One possible solution is to increase the operation frequency of the reconfiguration controller, but power consumption increases along with the frequency which is contrary to actual efforts on power reduction. Moreover, power consumption is a key parameter in embedded systems and should be considered during the design stage. Disregarding this parameter can lead to problems in production such as over-heating, power supply overload or a significant autonomy reduction. These events may affect the whole system stability or reduce the device lifespan [2].

Furthermore, each application has an optimized condition of operation in terms of performance, power consumption, etc. depending on applications, the environment, and user constraints, so the need of an energy-efficient controller, which is able to handle the reconfiguration process at very high speed and auto-adapt to various power consumption constraints, is also becoming essential.

In this paper, an ultra-fast power-aware reconfiguration controller—UPaRC incorporated with a dynamic clock generator is presented. On the one hand, the reconfiguration controller can operate at ultimate frequency (up to 362.5 MHz) that provides an extreme high reconfiguration throughput. On the other hand, the dynamic clock generator varies the operation frequency of the reconfiguration controller to adapt to various performance and power consumption constraints.

The paper is organized as follows: next section introduces related works. Section III details the architecture of UPaRC, Section IV gives the implementations and the comparison with some other controllers. Section V emphasizes the advantages of proposed system with the measurement of power consumption. The last section concludes and gives some future works.

## II. Related Works

Hardware sharing by dynamic reconfiguration not only increases the flexibility of designs but also reduces power consumption [3], [4]. The reconfiguration takes place by sending the bitstream to the reconfiguration port. Figure 1 shows the common architecture of a reconfiguration controller driving Internal Configuration Access Port (ICAP) [5]. ICAP is the hardwired primitive inside the Xilinx

FPGA device by which the bitstream can be dynamically loaded into the configuration memory. Reconfiguration speed deadlock does not come from the ICAP side but in the bitstream loading from the reconfiguration controller to ICAP which is usually limited by the access to the bitstream memory. So if the reconfiguration controller is capable of loading bitstream at very high-speed, it can provide high reconfiguration bandwidth.

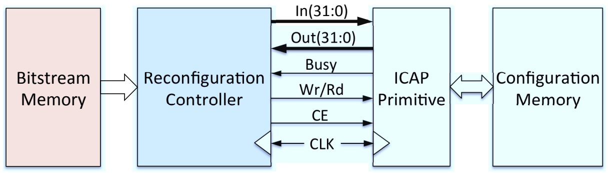

Fig. 1. Partial Reconfiguration with ICAP

Xilinx provides a reconfiguration controller xps_hwicap [6], which is processor-dependent (driven by a processor—either MicroBlaze [7] or PowerPC [8]). This reconfiguration controller only provides low reconfiguration throughput. The most important advantage of xps_hwicap is the high capacity for bitstream storage using external non-volatile memories like Compact Flash. To overcome the low reconfiguration speed, several controllers based on DMA access to the bitstream storage memory are proposed in the literature. The BRAM_HWICAP [9] applies DMA access to BRAM which can obtain the maximum theoretical throughput (400 MB/s) at 100 MHz. Nevertheless, this controller uses the same frequency as the other components in the system. So the frequency is limited to 120 MHz. Another disadvantage of BRAM_HWICAP is the low storage capacity due to limited BRAM amount. Also in [9], the authors proposed another controller—MST_ICAP which provides high storage capacity using DDR2 SDRAM memory. Nevertheless, the lower access speed of DDR2 SDRAM compared to BRAM leads to lower reconfiguration speed than BRAM_HWICAP.

FaRM [10] includes bitstream compression which allows to use less BRAM amount than the bitstream size. However, FaRM can operate up to 200 MHz which induces a maximum throughput of 800 MB/s. Additionally, FaRM is fixed to a certain frequency with a variable compression ratios depending on the regularity of the bitstreams, so the maximum throughput could vary. Moreover, compression algorithm—Run-Length Encoder (RLE) applied in FaRM does not provide an important gain for storage savings.

FlashCAP$_{(i)}$ [11] applies X-MatchPRO [12] which provides a more efficient compression ratio than RLE (74.2% against 63%). But the maximum frequency is limited to 120 MHz which can provide only a reconfiguration throughput of 385 MB/s.

We propose an ultra-fast power-aware reconfiguration controller which can achieve 1433 MB/s with operating frequency of ICAP at 362.5 MHz. This reconfiguration controller is processor-independent and requires only a "Start" signal which can be easily driven by any kind of module. This controller is customized to reach a significantly higher speed than the state-of-the-art reconfiguration speed and offers dynamic frequency scaling to satisfy different power constraints.

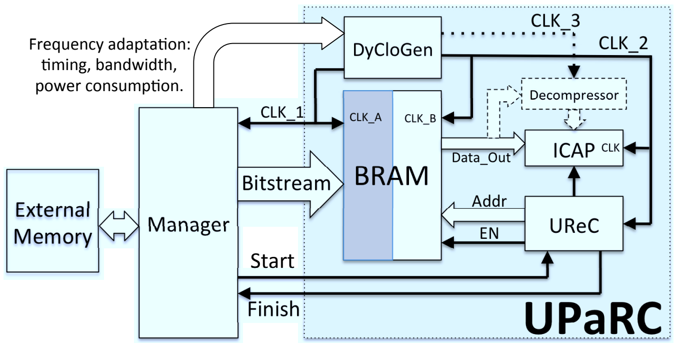

Fig. 2. Detailed structure of ultra-fast power-aware reconfiguration controller

## III. Ultra-Fast Power-Aware Reconfiguration Controller—UPaRC

Figure 2 details the internal structure of the proposed UPaRC consisting of an ultra-fast reconfiguration controller—UReC, a dynamic clock generator—DyCloGen, and a decompressor. The Manager, which connects to an external memory, drives UPaRC for rapid reconfiguration process.

### A. Manager

In the system, the manager carries out three tasks: bitstream preloading, reconfiguration control and frequency adaptation—driving DyCloGen to vary the frequency to adapt to various constraints. We use a MicroBlaze to perform these three tasks, but they can be handled by three different smaller hardware modules to save energy.

*1) Bitstream Preloading:* The Manager is in charge of reading the bitstream file in the external memory, parsing the preamble of the partial bitstream and then loading bitstream size followed by the configuration data into the BRAM. Figure 3 shows the data loaded by the Manager. The first 32-bit word contains the bitstream size and the operation mode determining the operation mode of the reconfiguration controller (with or without compression). Configuration data used to reconfigure the related module follow this word.

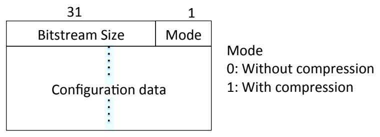

Fig. 3. Organization of BRAM contents

Scheduling may be able to predict the tasks to be executed on a reconfigurable module [13], thus the configuration data preloading can be done during idle time which does not affect the system computational performance and that could significantly improve the reconfiguration bandwidth.

Partial bitstream data contain a preamble which determines the attributes such as file name, FPGA device ID, bitstream size, etc. After this preamble, the bitstream contains data which configures the related module.

*2) Reconfiguration Control:* The manager controls the reconfiguration process by sending the "Start" signal to launch the reconfiguration and receiving the "Finish" signal when reconfiguration ends. Contrary to xps_hwicap in which the MicroBlaze is busy during the reconfiguration process, the MicroBlaze only launches UPaRC and is free during UPaRC performs the reconfiguration. And because the MicroBlaze consumes large amount of resources, the energy consumption during reconfiguration using UPaRC can be much less than using xps_hwicap. Moreover, the reconfiguration control can be handled by a small hardware module which can still save more energy.

*3) Frequency Adaptation:* The Manager analyzes different constraints (performance, power consumption, etc.) during runtime and chooses the appropriate frequency to meet these constraints by driving DyCloGen (see Section III-D).

## *B. Ultra-Fast Reconfiguration Controller—UReC*

The reconfiguration controller includes BRAM memory to store bitstreams which are used for reconfiguration. The dual-port BRAM memory is used. One port is interfaced with the bitstream manager to preload the bitstream whereas the other port is connected to UReC. The bitstream preloading (driven by the Bitstream Manager) does not affect the operation of the reconfigurable module, so the reconfiguration time of a module is the time to download bitstream data from BRAM to ICAP. The reconfiguration process includes DMA access to the second BRAM port and burst data transfer from BRAM to ICAP. That allows to reach the maximum reconfiguration throughput. This method is also applied by some other state-of-the-art reconfiguration controller [9]–[11]. However, these controllers re-use DMA module provided by Xilinx which is very large and does not permit to run at a higher frequency than 200 MHz. We have totally redesigned the BRAM interface so that configuration data can be transferred at each clock cycle in burst mode. Moreover, UReC performs only BRAM reading allowing to reduce maximum possible hardware resources. Doing so the UReC occupies very small area that can allow to accelerate the data transmission from BRAM to ICAP. This custom design allows to accelerate ICAP to very high frequency (up to 362.5 MHz), higher than the maximum BRAM operating frequency—300 MHz [14]. That is why UReC frequency is much higher than the fastest state-of-the-art reconfiguration controller—FaRM (200 MHz) [10].

Figure 4 explains the operation of UReC which controls the BRAM and ICAP to perform the reconfiguration.

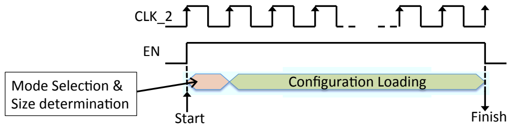

Fig. 4. Operation of UReC driving BRAM and ICAP

When UReC receives the "Start" signal, it immediately enables BRAM access and reads the first 32-bit word to decide the operation mode (without or with compression) and determine the size of the bitstream. Without compression, the configuration data can be loaded directly from the BRAM to ICAP, whereas in compression mode, the data should be fetched from BRAM to the decompressor before being sent to ICAP. The configuration data to ICAP is loaded using burst transfer without interrupt which permits to obtain the maximum reconfiguration bandwidth. A "Finish" signal indicates the end of the reconfiguration process and the EN signal deactivates the BRAM and ICAP access to save power consumption.

## *C. Lossless Bitstream Compression*

Because bitstreams contain configuration details, if compression is required, bitstreams must be compressed without loss. There exists a lot of lossless data compression RLE, LZ77, LZ78, Huffman, X-MatchPRO, Zip, 7-zip, etc. [15]. We compare in Table I different lossless compression algorithms, tested with different partial bitstream sizes and complexities. Because empty zones in the FPGA device can involve repetitive sequences in the configuration bitstreams, so in order not to exaggerate the compression effectiveness, we perform the compression only on partial bitstreams which have high resource utilization ratio to minimize number of blank frames in the reconfigurable partition.

TABLE I
Comparisons of Different Lossless Compression Algorithms

| Algorithm | Compression Ratio [%] |
|---|---|
| RLE | 63 |
| LZ77 | 71.4 |
| Huffman | 72.3 |
| **X-MatchPRO** | **74.2** |
| LZ78 | 75.6 |
| Zip | 81.2 |
| 7-zip | 81.9 |

At the moment X-MatchPRO, which is an open-source algorithm providing competitive compression effectiveness with Zip and 7-zip, is implemented in our system. The compression ratio recorded in our system is 74.2% in average which signifies that generally the compressed bitstream is about four times smaller than the original one.

The system operates in two modes depending on the partial bitstream size of the module to be reconfigured.

- Preloading without compression: If the original bitstream size is smaller than the BRAM size, the

Bitstream Manager is in charge of pre-loading the bitstream into the BRAM contents.

- Preloading with compression: In case that the original bitstream size is greater than the BRAM size, bitstream compression is required. By default the X-MatchPRO compression algorithm is implemented in our system. The over-sized bitstream is compressed offline using PC-running software and then put in the BRAM memory afterwards. The decompression part is performed by a hardware decompressor (Fig. 2). This decompressor is dynamically reconfigurable that allows to change compression/decompression algorithm by partial reconfiguration depending on the decompression speed, provided bandwidth, etc. Because each decompressor has its maximum frequency, after being reconfigured, its frequency (CLK_3 in Fig. 2) will be dynamically modified by DyClogen.

### D. Dynamic Clock Generator—DyCloGen

DyCloGen is a clock generator able to dynamically modify the frequency of its output clocks to meet the demand. The clock modification process is carried out by controlling the multiplication/division (M/D) factors defined in equation:

$$F_{out} = F_{in} \times \frac{M}{D}$$

Unlike partial reconfiguration where the reconfigured module is changed without affecting the others, the clock generator modifies the clock frequency while it is still operational. So DyCloGen controls Dynamic Reconfiguration Port (DRP) [5] of the specific digital clock manager (DCM) block found in Virtex-5 devices [16]. Doing so, the M and D values can be dynamically programmed without using partial reconfiguration via ICAP.

DyCloGen provides three different clock signals (CLK_1, CLK_2, and CLK_3) which are modifiable at run-time. CLK_1 is the clock source for preloading the BRAM contents, CLK_2 is the reconfiguration clock whereas CLK_3 determines the decompression frequency. Varying these clocks allows to change the power consumption to adapt to various conditions. Operating at high frequency can offer high-performance computations but can increase the power consumption, thus an appropriate frequency should be selected to meet both performance and consumption requirements.

## IV. Implementations and Comparison

We have implemented the system of Fig. 2 with two Xilinx FPGA platforms: ML506 [17] with Virtex-5 XC5VSX50T and ML605 [18] with Virtex-6 XC6VLX240T using Xilinx Design Suite v13.2. The resources required for blocks are shown in Table II.

The resources required for proposed modules (DyCloGen and UReC) are relatively small. The decompressor consumes a large amount of resources because it offers a

TABLE II
FPGA resources needed by basic blocks of UPaRC

| Module | Virtex-5 [slices] | Virtex-6 [slices] |
|---|---|---|
| DyCloGen | 24 | 18 |
| UReC | 26 | 26 |
| Decompressor | 1035 | 900 |

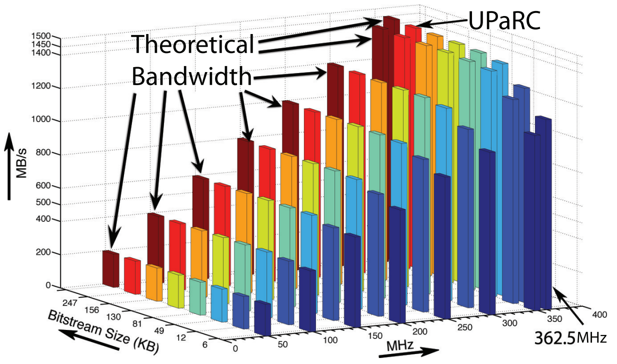

Fig. 5. Reconfiguration Bandwidths vs. Frequencies vs. Bitstream Sizes (measured with UPaRC$_i$ in preloading without compression mode using Virtex-5)

really fast decompression speed, inducing a high decompression bandwidth (more than 1 GB/s).

Table III shows the comparisons of performance (reconfiguration bandwidth, large bitstream handling capacity and maximum frequency) between different reconfiguration controllers. There are two instances of our reconfiguration controller corresponding to two modes: UPaRC$_i$—preloading without compression and UPaRC$_{ii}$—preloading with compression. With the ML506 platform, UPaRC$_i$ can reach 362.5 MHz (with $F_{in}$ = 100 MHz, $M$ = 29 and $D$ = 8 for DyCloGen configuration), offering 1433 MB/s reconfiguration bandwidth. UPaRC is tested on several Virtex-5 XC5VSX50T FPGAs and 362.5 MHz is a successful reconfiguration frequency in our working conditions (default core voltage 1 V, ambient temperature 20°C). Tests under the same conditions on a few Virtex-6 XC6VLX240T show that 362.5 MHz is not reliable, the maximum frequency seems to be few MHz lower. Experiments are underway on a larger number of samples to determine if the limitation is specific to the V6 family.

Thanks to large storing capacity of external non-volatile memory like the Compact Flash, xps_hwicap [6] can handle large bitstreams without the need of bitstream compression. However, it provides a very low reconfiguration throughput. With our experiments, the throughput recorded of this controller is about 180 KB/s. In [9], the author measures xps_hwicap performance using processor cache memory and it reaches 14.5 MB/s reconfiguration bandwidth.

Since BRAM_HWICAP [9] is implemented using Virtex-4 FPGA, it is not easy to compare the hardware overhead.

This controller is limited in bitstream sizes and limited in frequency (approximate theoretical throughput at this frequency). Without compression, our controller offers ultimate reconfiguration throughput up to 1.433 GB/s, which is 1.8 times higher than the fastest controller found in the literature (FaRM [10]—800 MB/s). MST_ICAP [9] which can handle large bitstreams is limited in frequency hence offers lower reconfiguration throughput. FlashCAP$_i$ [11], which is comparable to our UPaRC$_{ii}$ thanks to the same compression method, is also limited in frequency. Figure 5 shows the reconfiguration bandwidth measured with different sizes of bitstreams and at different frequencies. Because, the time overhead due to the control and the measurement using MicroBlaze is constant, the larger bitstream is, the less the control affects the reconfiguration time and the real bandwidth is more approximate to the theoretical bandwidth. As shown on Fig. 5, at 362.5 MHz and with the bitstream size of 6.5 KB, the effective bandwidth is 1.14 GB/s which is 78.8% of the theoretical bandwidth at this frequency (1.45 GB/s). With a bitstream size of 247 KB, the effective bandwidth is 1.44 GB/s, which is more approximate the theoretical bandwidth (99%).

TABLE III
Comparisons of Different Reconfiguration Controllers

| Reconfiguration Controller | Bandwidth [MB/s] | Large Bitstream | Max Freq. [MHz] |
|---|---|---|---|
| xps_hwicap [6] | 14.5 | +++ | 120 |
| MST_ICAP [9] | 235 | +++ | 120 |
| FlashCAP$_i$ [11] | 358 | ++ | 120 |
| BRAM_HWICAP [9] | 371 | - | 120 |
| FaRM [10] | 800 | ++ | 200 |
| **UPaRC$_{ii}$** | **1008** | ++ | **255** |
| **UPaRC$_i$** | **1433** | - | **362.5** |

The size of BRAM memory for bitstream preloading is 256 KBytes. With compression, this memory amount allows for storing the maximum bitstream of 992 KBytes, which is more than 40% of the full bitstream of the selected Virtex-5 device (2444 KBytes). That signifies that bitstream preloading with compression can handle the largest module which occupies half of the device resources. In preloading with compression mode, the decompression block has a throughput of 2 words/cycle (64-bit data path), which operates at maximum 126 MHz and supplies a reconfiguration throughput of 1.008 GB/s. To note that the frequency of the decompression block is different from the frequency of the reconfiguration block. In preloading with compression mode, it is not necessary that UPaRC operates at maximum frequency. The highest frequency at compression mode is 255 MHz.

However, reconfiguration throughput and bitstream size limitation are not only the major concerns. Power consumption should be considered, especially in embedded designs with a power budget, to ensure proper operation of the system.

## V. Power Consumption Consideration

Classically, FPGA power consumption can be separated into two main contributions: static and dynamic power. The first one is due to transistor leakage and dependent on the supply voltage and the size of the device. The second, which is caused by transistor switching, is highly dependent on the supply voltage, the signals activities, the application complexity and the clock frequency. The power consumption related to the reconfiguration is mostly composed of dynamic power. We measure the power consumption on Xilinx ML605 platform which has a Virtex-6 FPGA. ML605 includes a shunt resistor to monitor current through FPGA core, so we can measure Virtex-6 power consumption which is not possible using ML506. However, the measured values may change between Virtex-5 and Virtex-6 due to the difference of the process technologies (65 and 40 nm). Figure 6 presents the experimental setup for power measurements. We use the Virtex's core shunt with a high-precision amplifier to handle current and power consumption measurements which are logged with a digital oscilloscope.

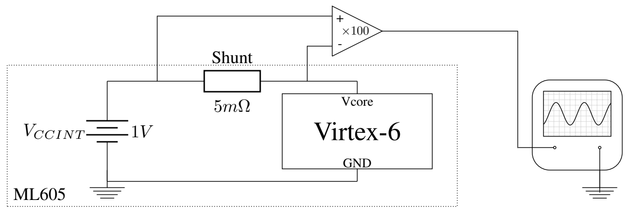

Fig. 6. Current measurement schematics on ML605 Board using a high-precision amplifier and an oscilloscope.

Net capacitance is a parameter of the dynamic power consumption, so to reduce dynamic power consumption reconfiguration controller must have short interconnections. Thanks to the lightweight of our reconfiguration controller, the power and energy consumptions are very low compared to state-of-the-art controllers. Since xps_hwicap proposed by Xilinx is the only available reconfiguration controller to public, we compare the energy consumption of UPaRC with xps_hwicap in the same conditions using a MicroBlaze running at 100 MHz with a 216.5 KB bitstream preloaded in 256 KB of 32-bit BRAM data bus. Without processor optimizations, we achieve a reconfiguration throughput of 1.5 MB/s of configuration data and the energy efficiency is 30 $\mu$J/KB of bitstream. In the same conditions, using a Microblaze as manager, UPaRC (without compression) consumes only 0.66 $\mu$J/KB which is 45 times more efficient than xps_hwicap.

Moreover, UPaRC can operate at a different clock frequency than the manager. This feature enables us to respect performance constraints with different power consumption characteristics during runtime. Figure 7 shows the power consumption during the reconfiguration of an uncompressed 216.5 KB bitstream file for different clock frequencies, from 50 MHz to 300 MHz which is the maximum guaranteed frequency for BRAMs. On these curves, the power peak before zero timestamp is caused by the activity of the manager to control UPaRC of which the duration and the power consumption are the same for each frequency since the frequency of the manager is not modified (MicroBlaze at 100 MHz). Then the reconfiguration procedure begins, all the configuration data are copied from BRAM to ICAP. This activity rises the power consumption immediately after the "Start" signal. Once the reconfiguration is completed, the power consumption decreases to the idle power consumption.

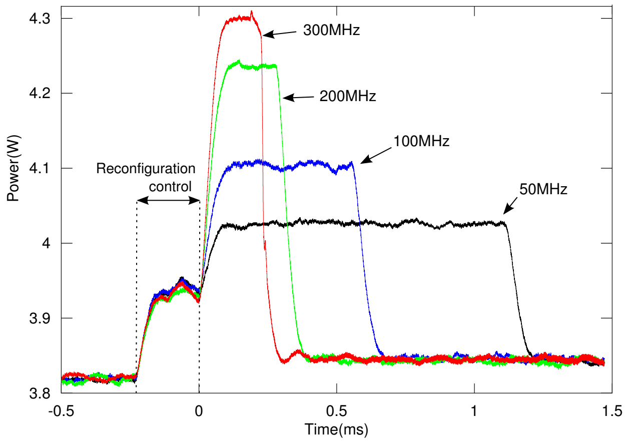

Fig. 7. FPGA core power consumption during dynamic partial reconfiguration using UPaRC with different frequencies on a Virtex-6. Only a MicroBlaze as the manager and UPaRC are implemented

The power consumption at 50 MHz is 183 mW, corresponding to a reconfiguration time of 1.1 ms. At 100 MHz, the power consumption is 259 mW during 550 $\mu$s. We notice that when the frequency is doubled, the reconfiguration time is halved, but the power is not doubled. This is caused by the manager which is implemented with an active wait for "Finish" signal and thus requires energy. At 200 MHz, the power consumption is 394 mW during 270 $\mu$s. Finally, reconfiguration at 300 MHz requires 453 mW during 180 $\mu$s.

The reconfiguration manager is right now not optimized for power consumption. The manager waits for the end of reconfiguration actively. This wastes some energy, that is why the energy decreases with the frequency, but in the case of a smaller manager or without actively waiting for reconfiguration to be done, the reconfiguration energy would be the same for each frequencies. The power consumption increases proportionally with the increase of reconfiguration frequency, so the power-aware solution is to use the lowest possible frequency which meets timing constraints for the current application. Frequency is dynamically adjusted, using DyCloGen, to meet application performance requirements and lower the power required when reconfiguring.

## VI. Conclusions and Future Work

An ultra-fast power-aware reconfiguration controller—UPaRC, which enhances the adaptivity to meet various demands during run-time, opens a higher degree of freedom in system design and behavior modification during run-time. This controller also offers ultimate reconfiguration bandwidth which can also meet the needs of high-performance applications. Moreover, the reconfiguration controller supports performance and power consumption on the fly adjustments. This feature enables UPaRC to be managed by a power optimization algorithm to reduce overall FPGA consumption. The energy efficiency of UPaRC is validated through the consumption measurement campaign.

We aim to further enhance the adaptivity by choosing different bitstream compression techniques at run-time using dynamic partial reconfiguration. Depending on the requirements of compression ratios, hardware resources, different frequency limits in compression modes, a wider range of application can be supported by UPaRC. We will focus our future work on the global power optimization of an application using high speed and energy efficient partial dynamic reconfiguration.

## References

[1] J. Becker, M. Hubner, G. Hettich, R. Constapel, J. Eisenmann, and J. Luka, "Dynamic and Partial FPGA Exploitation," *Proceedings of the IEEE*, vol. 95, no. 2, pp. 438–452, 2007.
[2] R. Viswanath, V. Wakharkar, A. Watwe, V. Lebonheur *et al.*, "Thermal performance challenges from silicon to systems," Intel Technology Journal, 2000.
[3] L. Sterpone, L. Carro, D. Matos, S. Wong, and F. Fakhar, "A New Reconfigurable Clock-Gating Technique for Low Power SRAM-based FPGAs," in *Design, Automation Test in Europe*, Grenoble, 2011, pp. 1–6.
[4] J. Becker, M. Huebner, and M. Ullmann, "Power Estimation and Power Measurement of Xilinx Virtex FPGAs: Trade-Offs and Limitations," in *Integrated Circuits and Systems Design*, New York, 2003.
[5] Xilinx, Inc, "Virtex-5 FPGA Config. User Guide UG191," 2009.
[6] ——, "LogiCORE IP XPS HWICAP DS586," 2010.
[7] ——, "MicroBlaze Processor Reference Guide UG081," 2011.
[8] ——, *PowerPC 405 Processor Block Reference Guide*, 2004.
[9] M. Liu, W. Kuehn, Z. Lu, and A. Jantsch, "Run-Time Partial Reconfiguration Speed Investigation and Architectural Design Space Exploration," in *Field Programmable Logic and Applications*, Prague, 2009, pp. 498–502.
[10] F. Duhem, F. Muller, and P. Lorenzini, "FaRM: Fast Reconfiguration Manager for Reducing Reconfiguration Time Overhead on FPGA," *Reconfigurable Computing: Architectures, Tools and Applications*, pp. 253–260, 2011.
[11] A. Nabina and J. Nuñez-Yañez, "Dynamic Reconfiguration Optimisation with Streaming Data Decompression," in *International Conference on Field Programmable Logic and Applications*, Milano, 2010, pp. 602–607.
[12] J. L. N. ez and S. Jones, "Gbits/s Lossless Data Compression Hardware," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, pp. 499–510, 2003.
[13] I. Belaid, F. Muller, and M. Benjemaa, "New Three-Level Resource Management Enhancing Quality of Offline Hardware Task Placement on FPGA," *International Journal of Reconfigurable Computing*, pp. 65–67, 2010.
[14] Xilinx, Inc, "LogiCORE IP Block Memory Generator v4.3."
[15] M. Nelson and J. Gailly, *The data compression book*, M. Books, Ed., 1996, vol. 619.
[16] Xilinx, Inc, *Virtex-5 FPGA User-Guide*, January 2009.
[17] ——, "ML505/ML506/ML507 Eval. Platform UG347," 2008.
[18] ——, "ML605 Hardware User Guide UG534," 2011.