RUN-TIME PARTIAL RECONFIGURATION SPEED INVESTIGATION AND ARCHITECTURAL DESIGN SPACE EXPLORATION

Ming Liu[†‡], *Wolfgang Kuehn*[‡], *Zhonghai Lu*[†], *Axel Jantsch*[†]

‡ II. Physics Institute
Justus-Liebig-University Giessen, Germany
{ming.liu, wolfgang.kuehn}@physik.uni-giessen.de

† Dept. of Electronic, Computer and Software Systems
Royal Institute of Technology, Sweden
{mingliu, zhonghai, axel}@kth.se

ABSTRACT

Run-time Partial Reconfiguration (PR) speed is significant in applications especially when fast IP core switching is required. In this paper, we propose to use Direct Memory Access (DMA), Master (MST) burst, and a dedicated Block RAM (BRAM) cache respectively to reduce the reconfiguration time. Based on the Xilinx PR technology and the Internal Configuration Access Port (ICAP) primitive in the FPGA fabric, we discuss multiple design architectures and thoroughly investigate their performance with measurements for different partial bitstream sizes. Compared to the reference OPB_HWICAP and XPS_HWICAP designs, experimental results show that DMA_HWICAP and MST_HWICAP reduce the reconfiguration time by one order of magnitude, with little resource consumption overhead. The BRAM_HWICAP design can even approach the reconfiguration speed limit of the ICAP primitive at the cost of large Block RAM utilization.

1. INTRODUCTION

Reconfigurability denotes the capability of programmable devices such as FPGAs, to change customized designs by loading different configware [1]. A more advanced reconfigurable technology, so-called Partial Reconfiguration (PR), enables the process of reconfiguring a particular section of an FPGA design while the remaining part is still operating. This vendor-dependent technology provides common benefits in adapting hardware algorithms during system runtime, sharing hardware resources to reduce device count and power consumption, shortening reconfiguration time, etc. [2] [3][4]. Typically partial reconfiguration is achieved by loading the partial bitstream of a new design into the FPGA configuration memory and overwriting the current one. Thus the reconfigurable portion will change its behavior according to the newly loaded configuration. In this procedure, the reconfiguration speed (or reconfiguration throughput) is a significant parameter, which determines the switching time of PR modules. This factor must be taken into account in many cases where performance-critical applications require fast switching of IP cores.

Nowadays, Xilinx is the main vendor whose silicon products and tools support the PR feature. The Virtex-II, Virtex-4 and Virtex-5 family FPGAs all provide a dedicated Internal Configuration Access Port (ICAP), which directly interfaces to the configuration memory and accesses it. Based on the Xilinx Virtex-4 partial reconfigurable platform, we investigate the run-time reconfiguration speed and explore the architectural design space to shrink the needed reconfiguration time. The remainder of the paper is organized as follows: In section 2, some related work on PR designs and reconfiguration speed studies will be addressed. In section 3, we introduce the ICAP interface used for dynamic reconfiguration. In section 4, multiple ICAP designs are described and we will explore the architectural design space to improve the run-time reconfiguration speed. Performance measurement results on the development board will be revealed in section 5, comparing the reconfiguration speed of different architectures. Correspondingly, the resource utilization is also analyzed for ICAP designs. Finally conclusions and a discussion of future work are presented in section 6.

2. RELATED WORK

The partial reconfiguration feature has been investigated in some applications such as [3][4][5][6]. As a consequence, the designer can obtain the benefits of dynamically adapting parameters (in [4]), reduced chip size/count (in [3]), or reduced power consumption (in [3][5][6]). In the Xilinx design support, two ICAP cores are typically used for run-time PR, specifically XPS_HWICAP to the PLB bus [7] and OPB_HWICAP to the OPB bus [8]. Concerning the reconfiguration speed when using ICAP, much work is still left to study this important parameter: In [3], the authors compare the speed of different configuration approaches, including serial configuration, JTAG, SelectMAP and ICAP. However the listed ICAP configuration speed of 1.64 μs/Frame (one frame contains 41 32-bit words) is only theoretically calculated from the interface bandwidth, and the dominating overhead when the bitstream data are supplied from the memory to ICAP has not been taken into account. In [9], the authors considered all factors and did practical measurements on OPB_HWICAP. Unfortunately the results are not

satisfactory. For partial bitstreams of about 60 to 70 KBytes, reconfiguration requires more than 100 milliseconds. In [10] and [11], the authors achieve good reconfiguration throughput by optimizing the PLB ICAP design. However the extra Device Control Register (DCR) bus is required to control the reconfiguration process, in addition to PLB for configuration data delivery. Moreover, detailed resource consumption overhead is not entirely presented and compared with other designs. Hence in this paper, we will do a thorough investigation on different ICAP architectures and optimize them to make the PR module switching faster.

3. RUN-TIME PR WITH ICAP

The ICAP primitive is the hardwired FPGA logic by which the bitstream can be dynamically loaded into the configuration memory. As shown in figure 1, ICAP interfaces to the configuration memory and furthermore provides parallel access ports to programmable resources. During system run-time, a master device (normally an embedded microprocessor) may transmit the partial reconfiguration bitstream from storage devices to ICAP to accomplish the reconfiguration process. The complete ICAP design, in which the ICAP primitive is instantiated, interfaces to the system interconnection fabric to communicate with the processor and memories. Detailed design architectures will be discussed in the next section.

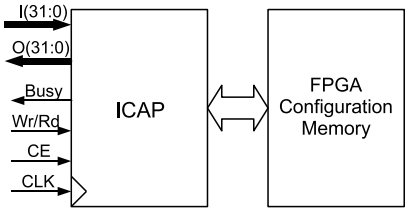


Fig. 1. The ICAP primitive on Xilinx FPGAs

4. ICAP DESIGN ARCHITECTURE

4.1. System Architecture

A sample embedded design based on the Xilinx Virtex-4 FX FPGA is depicted in Figure 2, in which a hardcore PowerPC 405 processor (or the softcore Microblaze), a Multi-Port Memory Controller (MPMC) and some other peripheral controllers are interconnected by the system PLB bus. To study the PR feature, a PR Region (PRR) is integrated in the system. Since communications exist between the PRR and the PLB or I/O buffers to external devices, two Bus Macro (BM) [12] interfaces (BM_interface) with output enable pins are utilized to lock the routing and isolate the unpredictable state in the PRR during the PR process. The BM output enable/disable signals are controlled by a General-Purpose I/O

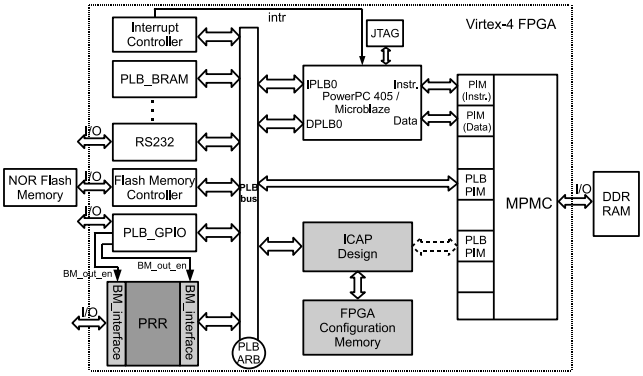


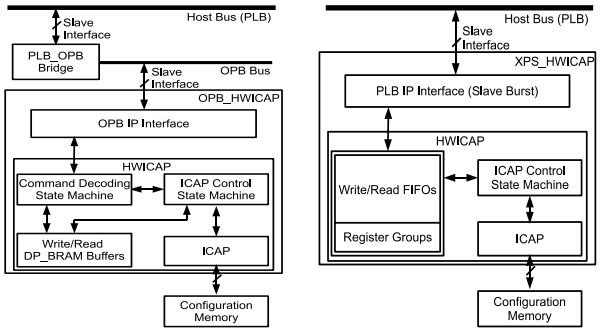
Fig. 2. A sample embedded design for the run-time PR study

(GPIO) core on the PLB. PRR outputs are to be switched off during reconfiguration, and switched on again after the newly loaded module starts functioning. To initiate the run-time PR, the processor fetches bitstream data from the memory devices, such as DDR or the non-volatile flash memory, and delivers them to the ICAP design. The ICAP design takes charge of writing to or reading from the FPGA configuration memory. In Figure 2 it is represented as a black box and will be elaborated in the following.

4.2. ICAP Design Evolution

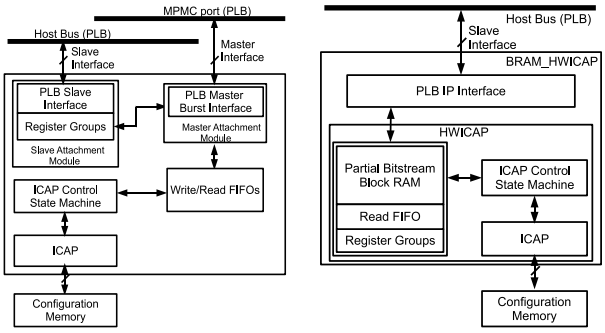
As Xilinx reference designs for PR, the structures of OPB_HWICAP [13] and XPS_HWICAP [14] are demonstrated respectively in Figure 3(a) and Figure 3(b). The OPB_HWICAP core was previously designed for the low-performance OPB bus. To interface the OPB core to the host PLB, a bridge is needed for protocol adaptation. Via the OPB slave interface, the partial reconfiguration data are buffered in the Dual-Port Block RAM (DP_BRAM), if the command is decoded as "write". A control state machine diagnoses the occupancy status of the buffer and continuously supplies configuration data to ICAP, by which the FPGA configuration memory is overwritten. XPS_HWICAP shares a similar structure except that Write/Read FIFOs and register groups take the place of DP_BRAM buffers and the command decoding state machine in the OPB_HWICAP design. Also to increase the data transport efficiency, the PLB interface supports the burst transfer mode.

Considering the inefficiency when the processor moves data to ICAP, we propose a design with DMA support as shown in Figure 4. We reuse the XPS_HWICAP structure and attach the DMA function on it. The DMA controller has two interfaces: The slave interface is used to receive commands, for example starting a DMA transaction by writing to the Source Address (SA), Destination Address (DA), and length registers. The master interface then initiates the configuration data movement from the memory to the ICAP



(a) OPB_HWICAP (b) XPS_HWICAP

Fig. 3. Structure of the Xilinx ICAP designs



(a) MST_HWICAP (b) BRAM_HWICAP

Fig. 5. Structure of MST_ICAP and BRAM_ICAP

design. With the burst transfer supported, the DMA module is expected to transport data more efficiently than the processor, releasing the precious CPU time to other processor-hungry programs.

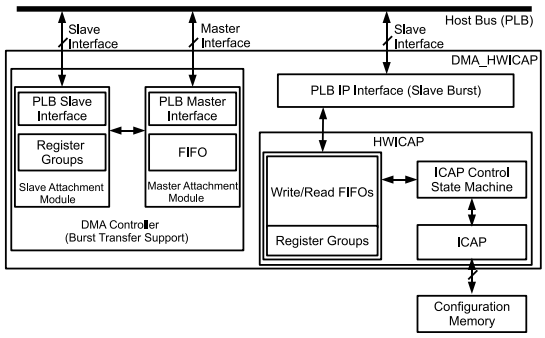


Fig. 4. Structure of the DMA_HWICAP design

To further reduce the communication overhead of the external DMA, we optimized the design using an integrated bus master (MST) with burst transmission support instead of DMA, as shown in Figure 5(a). In this design, bitstream data can be actively fetched by the master device from the memory, and hence the communication overhead between DMA and HWICAP is avoided. The master interface may be directly coupled to one port of MPMC using the PLB protocol (see the dashed arrow in Figure 2). The slave interface which receives control commands, is simply connected to the host PLB without using the DCR bus as in [11].

A particular design shown in Figure 5(b) is used to investigate the configuration capability of the ICAP primitive. Rather than the Write FIFO, a dedicated BRAM block is instantiated to store bitstream data. It must be large enough to hold the entire partial bitstream, because all the reconfiguration data for one PR module will be initialized there before each time reconfiguration. Partial bitstreams are alternatively loaded in BRAM through the PLB IP interface. The BRAM block works as a cache which removes the needed time to transfer data from the memory into the HWICAP module. Therefore ICAP always has the required data ready in the high-speed BRAM. This approach can be used to evaluate the ultimate reconfiguration speed of the ICAP primitive. The design architecture is well suited for the cases in which the PR region is small, and extremely fast reconfiguration speed is required. Its disadvantage is the high utilization of the BRAM resource on the FPGA, which will be quantified in sub-section 5.2.

5. PERFORMANCE INVESTIGATION AND SYNTHESIS RESULTS

5.1. Performance Measurements

Based on the Xilinx ML405 development board with a Virtex-4 FX20 FPGA, experiments have been done to investigate the performance of various design structures. We use the system architecture shown in Figure 2. The processor programs are executed in DDR. Partial bitstream files are initialized in DDR as well, and different ICAP designs are employed to reconfigure the PR region with the help from the processor, DMA or MST. The only exception is BRAM_HWICAP, in which bitstream data are initialized directly in BRAM via PLB. In the system design, both PLB (64-bit) and OPB (32-bit) run at 100 MHz, and so do all ICAP designs. The reconfiguration time is measured from the master device starting to feed data to ICAP, and ends until the partial bitstream is completely downloaded into the configuration memory. Measured with various sizes of partial bitstreams in test 1 to 4, the reconfiguration speed is calculated from the recorded time. Results are presented in Table 1 and Figure 6. To demonstrate the performance dependency on the processor, we did measurements using a 300 MHz PowerPC and a 100 MHz Microblaze processor together with OPB_HWICAP and XPS_HWICAP respectively. Shown in top rows of Table 1, we observe that enabling the seperate 16 KB ICache and DCache of the PowerPC processor enhances the program execution speed and

ICAP design	Test 1 (Bit. Size/Reconf. Time)	Test 2 (Bit. Size/Reconf. Time)	Test 3 (Bit. Size/Reconf. Time)	Test 4 (Bit. Size/Reconf. Time)	Avg. reconfig. speed	Max. reconfig. speed
OPB_HWICAP (PowerPC cache_disabled)	7.7 KB/12.1 ms	23.2 KB/36.5 ms	44.5 KB/75.6 ms	79.9 KB/135.6 ms	0.61 MB/s	0.64 MB/s
XPS_HWICAP (PowerPC cache_disabled)	7.7 KB/9.2 ms	23.2 KB/27.9 ms	46.5 KB/57.9 ms	80.0 KB/99.7 ms	0.82 MB/s	0.84 MB/s
OPB_HWICAP (PowerPC cache_enabled)	7.7 KB/694.8 μs	22.7 KB/2.3 ms	43.9 KB/4.5 ms	75.9 KB/7.8 ms	10.1 MB/s	11.1 MB/s
XPS_HWICAP (PowerPC cache_enabled)	7.7 KB/336.9 μs	23.2 KB/1.3 ms	44.5 KB/2.5 ms	74.6 KB/4.2 ms	19.1 MB/s	22.9 MB/s
OPB_HWICAP (Microblaze cache_enabled)	7.7 KB/1.3 ms	23.2 KB/3.9 ms	47.1 KB/7.9 ms	77.7 KB/13.0 ms	6.0 MB/s	6.0 MB/s
XPS_HWICAP (Microblaze cache_enabled)	7.7 KB/532.6 μs	23.2 KB/1.6 ms	47.2 KB/3.3 ms	79.1 KB/5.4 ms	14.5 MB/s	14.6 MB/s
DMA_HWICAP	7.7 KB/95.1 μs	23.2 KB/282.3 μs	46.8 KB/566.3 μs	81.9 KB/991.1 μs	82.1 MB/s	82.6 MB/s
MST_HWICAP	7.7 KB/33.0 μs	23.2 KB/98.9 μs	44.8 KB/190.7 μs	76.0 KB/323.1 μs	234.5 MB/s	235.2 MB/s
BRAM_HWICAP	7.7 KB/28.0 μs	23.2 KB/66.3 μs	45.2 KB/121.7 μs	none	332.1 MB/s	371.4 MB/s

Table 1. Reconfiguration speed measurements of ICAP designs for various sizes of partial bitstreams[1]

thus the reconfiguration speed by 16.6 and 23.3 times in average for two reference cores. Moreover, the more powerful hardcore PowerPC achieves better performance compared to the softcore Microblaze. The improved DMA_HWICAP, MST_HWICAP and BRAM_HWICAP designs are listed in the three bottom rows of the table. The integrated DMA controller or the master device replaces the processor to move data and obtains a speedup of one order of magnitude. Especially MST_HWICAP reaches an average speed of 234.5 MBytes/s after architectural optimization, which is almost 3 times faster than the original DMA design. We also find that the bottleneck in MST_HWICAP is the data delivery throughput of the DDR memory (32-bit, 100 MHz) and the MPMC controller (100 MHz). In designs with wider DDR bandwidth and higher clock frequency, the performance value is potentially improvable. In addition, the reconfiguration speed of DMA_HWICAP and MST_HWICAP is guaranteed to be processor independent, due to little processor participation in the reconfiguration work. The BRAM_HWICAP design arrives at the maximum speed of 371.4 MBytes/s, which approaches the physical limit of 400 MBytes/s of the ICAP primitive interface (32-bit, 100 MHz). In the experiments for BRAM_HWICAP, we didn't measure large bitstreams because of the constraint from the high BRAM utilization on the small Virtex-4 FX20 FPGA. The normalized speed of all designs is listed in Figure 7, both for the average reconfiguration speed and the maximum one.

5.2. Synthesis Results

All ICAP designs have been synthesized with Xilinx ISE v10.1 software. The resource utilization is summarized in Table 2. We observe that OPB_HWICAP plus the PLB-OPB bridge consumes the least resources. LUTs are mainly used as logic and one BRAM array constructs the Write/Read

[1]For different ICAP designs in each test, we fit the PR region in the same rectangular area on the FPGA. However the sizes of partial bitstreams vary a little due to the difference of design implementation.

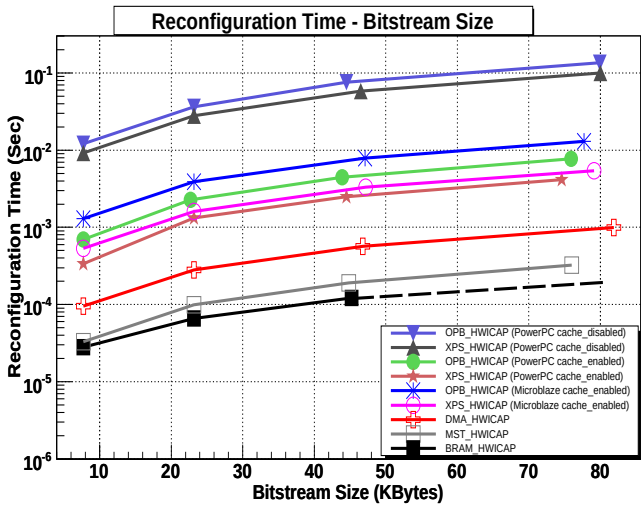


Fig. 6. Reconfiguration performance of ICAP designs

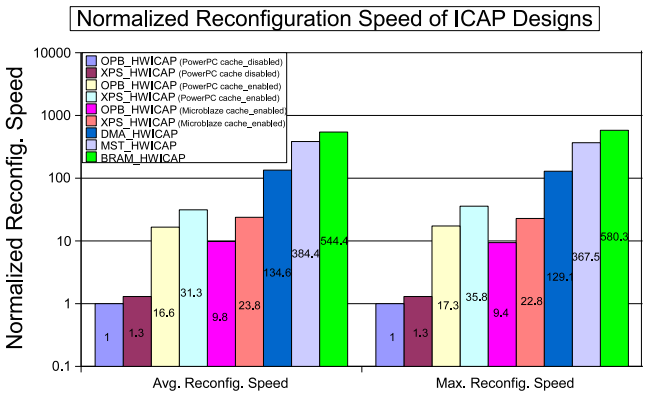


Fig. 7. Normalized average/maximum reconfiguration speed of ICAP designs

Resources	OPB_HWICAP+Bridge	XPS_HWICAP	DMA_HWICAP	MST_HWICAP	BRAM_HWICAP
4-LUT (total)	608 out of 17088 (3.6%)	3275 (19.2%)	4277 (25.0%)	1083 (6.3%)	963 (5.6%)
4-LUT used as logic	576 out of 17088 (3.4%)	907 (5.3%)	1843 (10.8%)	1083 (6.3%)	614 (3.6%)
4-LUT used as shift registers	32 out of 17088 (0.2%)	2368 (13.9%)	2434 (14.2%)	0	320 (1.9%)
Slice Flip-Flops	368 out of 17088 (2.2%)	417 (2.4%)	977 (5.7%)	918 (5.4%)	469 (2.7%)
Block RAM (BRAM)	1 out of 68 (1.5%)	0	0	2 (2.9%)	32 (47.1%)

Table 2. Resource utilization of ICAP designs on Virtex-4 FX20

buffers. XPS_HWICAP utilizes much more 4-LUTs than OPB_HWICAP but zero BRAM. The reason is that its buffer device (Write/Read FIFOs) in HWICAP is synthesized into LUTs as shift registers, rather than into BRAM. On the basis of XPS_HWICAP, DMA_HWICAP adds more consumption (totally 25% 4-LUTs) due to the direct integration of the DMA controller without architectural optimization. Comparatively, the optimized alternative (MST_HWICAP) uses much less 4-LUTs and migrates the FIFO devices into two BRAMs. So MST_HWICAP is more efficient to perform run-time PR in practical designs, taking into account its less resource consumption plus very large performance speedup over others including DMA_HWICAP. The last column in Table 2 lists the BRAM_HWICAP design. The LUT and Flip-Flop resource consumption is negligible but almost half of the BRAM resource provided by Virtex-4 FX20 is used to construct the dedicated buffer for partial bitstreams. In our tests, 32 BRAMs constitute 64 KBytes address space, which implies a maximum 64 KBytes bitstream initialization in the performance experiments. Nevertheless the large BRAM consumption can bring the ultimate reconfiguration speed of 371.4 MBytes/s.

The maximum clock frequencies of all designs for -10 speed grade Virtex-4 FPGAs are listed in Table 3. We observe that our improved designs based on XPS_HWICAP do not degrade the timing performance (MST_HWICAP even improved). In practical uses, normally we choose 100 MHz for all to match the clock frequency of PLB or OPB.

ICAP designs	Max. clock frequency
OPB_HWICAP/PLB-OPB bridge	248 MHz/182 MHz
XPS_HWICAP	121 MHz
MST_HWICAP	200 MHz
DMA_HWICAP	121 MHz
BRAM_HWICAP	121 MHz

Table 3. Timing performance of ICAP designs

6. CONCLUSION AND FUTURE WORK

We have explored the design space of ICAP architectures, and investigated the reconfiguration speed of five designs. Experimental results show that DMA_HWICAP and MST_HWICAP may achieve a processor independent reconfiguration speed of one order of magnitude faster than OPB_HWICAP and XPS_HWICAP. BRAM_HWICAP can even approach the reconfiguration speed limit of the ICAP primitive, at the cost of a large BRAM resource utilization on FPGAs. In practice, MST_HWICAP is more applicable for fast reconfiguration requirements, and BRAM_HWICAP is useful in ultimate scenarios especially when reconfigurable blocks are small but extremely fast IP switching is needed.

Proper ICAP cores will be utilized in our future PR designs for adaptive computing studies. A high reconfiguration speed guarantees the IP switching fast and largely reduces the time overhead when the system is partially changed.

Acknowledgment

This work was supported in part by BMBF under contract Nos. 06GI179 and 06GI180, and by FZ-Juelich under contract No. COSY-099 41821475.

7. REFERENCES

[1] C. Morra, "Configware Design Space Exploration Using Rewriting Logic", *In Proc. of the International Conference on Field Programmable Logic and Applications 2006*, pp. 1 - 2, Aug. 2006.

[2] C. Kao, "Benefits of Partial Reconfiguration", *Xcell Journal*, Fourth Quarter 2005, pp. 65 - 67.

[3] E. J. Mcdonald, "Runtime FPGA Partial Reconfiguration", *In Proc. of 2008 IEEE Aerospace Conference*, pp. 1 - 7, Mar. 2008.

[4] C. Choi and H. Lee, "An Reconfigurable FIR Filter Design on a Partial Reconfiguration Platform", *In Proc. of First International Conference on Communications and Electronics*, pp. 352 - 355, Oct. 2006.

[5] K. Paulsson, M. Huebner, S. Bayar, and J. Becker, "Exploitation of Run-Time Partial Reconfiguration for Dynamic Power Management in Xilinx Spartan III-based Systems", *In Proc. of 2006 Reconfigurable Communication-centric SOCs*, Jun. 2007.

[6] J. Noguera and I. O. Kennedy, "Power Reduction in Network Equipment Through Adaptive Partial Reconfiguration", *In Proc. of the 2007 Field Programmable Logic and Applications*, pp. 240 - 245, Aug. 2007.

[7] IBM Inc., "128-Bit Processor Local Bus Architecture Specifications Version 4.7", May. 2007.

[8] IBM Inc., "On-Chip Peripheral Bus Architecture Specifications Version 2.1", Apr. 2001.

[9] A. Lagger, A. Upegui, E. Sanchez, and I. Gonzalez, "Self-Reconfigurable Pervasive Platform for Cryptographic Application", *In Proc. of the International Conference on Field Programmable Logic and Applications 2006*, pp. 1 - 4, Aug. 2006.

[10] C. Claus, F. H. Mueller, J. Zeppenfeld, and W. Stechele, " A New Framework to Accelerate Virtex-II Pro Dynamic Partial Self-reconfiguration", *In Proc. of the Parallel and Distributed Processing Symposium 2007*, pp. 1 - 7, Mar. 2007.

[11] C. Claus, B. Zhang, W. Stechele, L. Braun, M. Huebner, and J. Becker, "A Multi-platform Controller Allowing for Maximum Dynamic Partial Reconfiguration Throughput", *In Proc. of the International Conference on Field Programmable Logic and Applications 2008*, pp. 535 - 538, Sep. 2008.

[12] M. Huebner, T. Becker, and J. Becker, "Real-time LUT-Based Network Topologies for Dynamic and Partial FPGA Self-Reconfiguration", *In Proc. of the 17th Symposium on Integrated Circuits and System Design*, pp. 28 - 32, Apr. 2004.

[13] Xilinx Inc., "OPB HWICAP (v1.00.b) Product Specification", DS280, Jul. 2006.

[14] Xilinx Inc., "XPS HWICAP (v1.00.a) Product Specification", DS586, Oct. 2007.