On the Management of Dynamic Partial Reconfiguration to Speed-up Intrinsic Evolvable Hardware Systems

S.Bhandari, F. Cancare, D. B. Bartolini, M. Carminati,
M. D. Santambrogio, and D. Sciuto

Politecnico di Milano - Dipartimento di Elettronica e Informazione
Via Ponzio 34/5 - 20133 Milano, Italy

Abstract. Turning away from traditional design techniques, which follow the various phases of design and testing, Evolvable Hardware circuits are created through evolutionary strategies aimed at improving the circuits behavior with respect to a given specification. These evolutionary strategies are stochastic search methods that mimic the process of natural biological evolution and are implemented as Evolutionary Algorithms (EAs). This approach permits the exploration of a large design search space, which can ideally enable Evolvable Hardware to find solutions that are more efficient than those found using traditional design methods.
This paper describes how an FPGA-based Evolvable Hardware system, using partial dynamic reconfiguration, has been improved introducing a new ICAP controller: the Speed Efficient Dynamic Partial Reconfiguration Controller (SEDPRC). Experimental results have shown that the novel controller brings benefits both to the overall running time (the reconfiguration time has been reduced by 37.2%) and to the resources required to implement the controller itself.

1 Introduction

Since their introduction in 1980, Field Programmable Gate Arrays (FPGAs) are gaining importance both in the commercial and in the research setting. The increased popularity of FPGAs results from significantly increased capability and performance. Contemporary FPGAs contain thousands of Look-Up Tables (LUTs), Flip-Flops and a large variety of other built-in digital components. An interesting feature that has been viable for most recent FPGAs is Dynamic Partial Reconfiguration (DPR). The possibilities offered by partial reconfiguration in the implementation of system adaptivity are enormous [1]. As a result of this and other advancements, FPGAs have evolved from being early prototyping and hardware emulation platforms to being devices widely used in both industrial and research applications. Since 1993, they have also been looked upon as promising devices for implementing Evolvable Hardware (EHW) systems [2, 3].

FPGA-based Evolvable Hardware systems have been successfully used in the past to create basic math components, controllers, image and audio filters and

to deal with fault recovery and system adaptation. Most of the experiments reported, however, are still nowadays based on outdated FPGA devices (e.g. Xilinx Virtex and Virtex-2 Pro) and do not benefit from the greater amount of available resources and capabilities of the most recent devices. In contrast with these evolvable hardware systems based on outdated FPGAs, the Hardware Evolution over Reconfigurable Architectures, from now on HERA, approach is based on the Xilinx Virtex-4 FPGA family and can as well scale to newer Xilinx device families. Recent papers [4, 5] describe the goals and the progresses of the HERA project. Briefly, the approach aims at having a self-evolvable hardware system able to exploit the advantages offered by the new FPGA generations, and, in particular, the possibility to make use of the two-dimensional reconfiguration mechanism. The approach is based on the direct manipulation of the configuration bitstreams but, at the same time, it allows a multi-grained evolution, enabling to mix the search capabilities offered by a fine-grained evolution and the exploitation of functional building blocks typical of functional level evolution. The advantages offered by this system over similar solutions [6] are the improved timing performance (thanks to the speed-up guaranteed by the internal reconfiguration), the solutions specificity (thanks to the fine-grained evolution obtained through the safe direct bitstream manipulation) and the overall flexibility (thanks to the hierarchical architecture and the functional level evolution). However, a major drawback of the HERA approach and of all those EHW systems based on bitstream manipulation and DPR is the time overhead caused by the reconfiguration process [7–9].

Starting from these considerations, the goal of this paper is to improve the HERA approach by managing the dynamic reconfiguration process in a more efficient way. To do so, a new reconfiguration controller, the Speed Efficient DPR Controller (SEDPRC), has been designed and implemented. This controller can partially reconfigure the target device faster than traditional controllers [10, 11] and does not require to involve the processor. The integration of the SEDPRC within the HERA system will allow to speed-up the overall evolutionary process and to tackle EHW problems more efficiently.

The remaining of this paper is organized as follows: Section 2 presents the HERA approach, describing its key components and functionalities. Section 3 presents the novel component used in order to speed-up the reconfiguration process and, in turn, the overall evolution time. Experimental results are provided in Section 4. Finally, in Section 5 concluding remarks are given along with future research directions.

2 The HERA approach

The HERA approach is based on a tight coupling between an evolutionary algorithm and a dynamic reconfigurable, FPGA-based, architecture. The evolutionary algorithm used is a variation of the canonic Genetic Algorithm (GA) modified taking into account some concepts typical of Cartesian Genetic Programming (CGP) [12]. The focus of this Section is, however, on the FPGA-based

architecture. For more details about the GA see [4, 5]. The innovative contribution of this paper consists in improving such architecture by managing the dynamic partial reconfiguration process in a more efficient way.

The used architecture is similar to the one proposed by Upegui and Sanchez in [13]; however, since the target device family is different (Virtex-4 instead of Virtex-II Pro), the evolvable region and the reconfiguration process have been completely redesigned. Figure 1 shows the high-level structure of the proposed architecture.

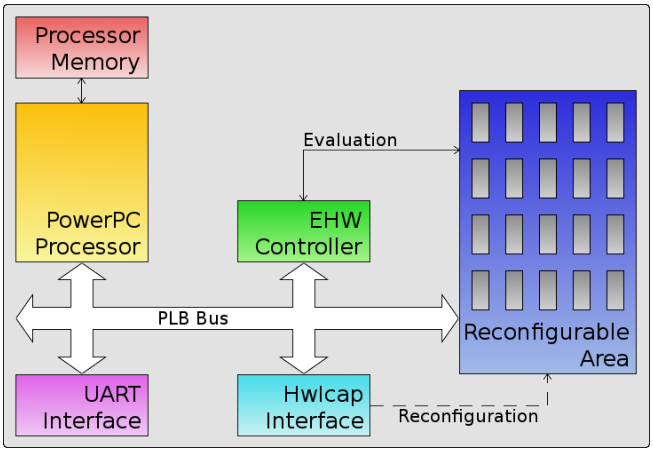

Fig. 1. High-level structure of the proposed architecture

The main architecture components are:

- the *Reconfigurable Area*, which contains the candidate solutions undergoing the evolutionary process;
- the *PowerPC Processor*, which executes the genetic algorithm and generates the information for the dynamic partial reconfiguration of the evolvable region;
- the *HwIcap Interface*, which performs the device internal reconfiguration process;
- the *EHW Controller*, which provides access to the evolvable region, thus allowing to retrieve and evaluate the outputs of the candidate solutions.

With two-dimensional reconfiguration it becomes possible to dynamically reconfigure device portions whose height is not constrained to be the device height. Exploiting this characteristic allows to speed up the reconfiguration process and to deploy more candidate solutions on the device at the same time. For what concerns the reconfiguration process, the Virtex-4 bitstream format presents some similarities with respect to the one of Virtex-II Pro. Since such format is not documented, an analysis was performed for understanding how to evolve candidate solutions without downloading illegal configurations onto the FPGA [4, 5].

A candidate solution is a $8 \times N$ two-dimensional array of cells, with an 8bit datapath. Every cell has four inputs and one output but, differently from traditional solutions, they are not connected to their four adjacent cells; instead, they are organized in columns; cells belonging to column i are connected to cells belonging to column $i+1$. Every column shares the same clock signal. The candidate solution column has been designed to span over a single FPGA frame (a frame is the smallest part of the FPGA that can be dynamically reconfigured).

A cell is equivalent to a 5-input Look Up Table (LUT). The first four LUT inputs are the external inputs, while the last one is represented by the internal flip-flop status. Such 5-input LUT is built using two 4-input LUTs and a multiplexer driven by the flip-flop output. All the needed resources are available in a single Virtex-4 slice (LUT-F, LUT-G and MUXF5).

Even if the reconfiguration time has been considerably shortened, it still remains a critical issue of the HERA EHW system. It accounts for more than the 50% of the overall execution time when evolving 8bit I/O combinatorial circuits composed of four modules of eight cells each. In the next Section a novel dynamic partial reconfiguration controller will be introduced. Such component will be plugged into the just described architecture with the goal of improving the reconfiguration process.

3 Dynamic partial reconfiguration process optimization

In the HERA approach, candidate solutions are evaluated once deployed to the architecture's' *reconfigurable area*. The process of changing the configuration of parts of the FPGA while the remaining ones are still working is called Dynamic Partial Reconfiguration (DPR). DPR can be driven either by an external device or by the FPGA logic itself . The HERA approach is based on this latter strategy; it makes use of the Xilinx internal interface for accessing the configuration memory, the ICAP (Internal Configuration Access Port).

3.1 Related ICAP controllers

Xilinx provides two ip-cores, namely XPS-HWICAP [11] and OPB-HWICAP [10], which can carry out the internal DPR of its FPGAs. Both these ip-cores enable an embedded microprocessor, such as the MicroBlaze or the PowerPC, to read and write the FPGA configuration memory through the ICAP at run-time. All the earlier architectures [4, 5] developed within the HERA project rely on the XPS-HWICAP ip-core to carry out DPR. That is because it is more stable and efficient than the outdated and deprecated OPB-HWICAP. Even the XPS-HWICAP, however, offer a reconfiguration throughput still far from the ICAP maximum.

Various researchers have concentrated their efforts in creating more efficient ICAP controllers for performing DPR. There is a wide literature (e.g., [14–16]) regarding reconfiguration techniques using Xilinx Virtex-II Pro devices; among them the one that emerged as the most promising one is presented in [15].

The ICAP controller for the Virtex-II Pro was directly connected to the PLB and equipped with Direct Memory Access (DMA) capabilities. The achieved throughput was 96% (95.77 MB/s) of theoretical throughput of the ICAP in Xilinx Virtex-II Pro devices. Nowadays, researchers moved from Xilinx Virtex-II-Pro devices to the newest Xilinx FPGAs, e.g., Virtex-4 and Virtex-5, to enhance DPR performance on them. In [17] authors perform partial reconfiguration using a PLB ICAP controller managed by a processor, obtaining a reconfiguration throughput of around 300 MB/s for Virtex-4 device. This is one of the early works improving the ICAP control logic, but it still requires a processor that has to drive the reconfiguration process. In [18] the authors present an evaluation of DPR for Software Defined Radio (SDR). In their system, an ICAP controller is implemented for carrying out fast DPR on Virtex-4 devices. This implementation achieves a throughput of 350 MB/s at 100MHz. This is close to the theoretical maximum of 400 MB/s at 100MHz on Virtex-4. Subsequently, a comprehensive survey of reconfiguration speeds by using different ICAP controllers is presented in [19]. The paper presents MST-HWICAP and BRAM-HWICAP structures and compares the obtained experimental results with those achieved using the cores by Xilinx. The maximum reconfiguration speed achieved was around 371 MB/s for Virtex-4 device. Both the approaches do not rely on the ICAP over-clocking, even if they improve the ICAP controllers state of art; however the control logic is still far from being efficient, and the latency of data availability forces frequent stalls in the process. Recently, in [20], authors have presented a modified ICAP controller for carrying out fast DPR in video based driver assistance systems to target Virtex-II Pro, Virtex-4 and Virtex-5 devices. With the proposed controller, the reconfiguration throughput achieved for Virtex-4 devices is 560 MB/s. Given the ICAP clock frequency, the control logic is able to perfectly sustain data throughput; however, there is still the possibility of providing a faster clock signal. Moreover, the controller has to be driven by a processor. In one of the most recent works dealing with reducing the reconfiguration time in DPR [21], authors have developed an ICAP Hard-Macro for Virtex-5 devices. The reported reconfiguration throughput is 2200 MB/s. Also in this work a processor is needed for driving the reconfiguration process. Another notable work targeting Virtex-5 is described in [22]; the throughput reached is 800 MB/s.

3.2 Reconfiguration Throughput and Time

The theoretical reconfiguration throughput of the ICAP can be calculated as:

$$Theoretical\ Throughput = \frac{ICAP\ input\ data\ width}{clock\ period}$$

However the ICAP is not always able to process the incoming data. When this happens it raises a *busy* signal and stalls the reconfiguration process. Moreover, the system could not be able to provide a datum per clock cycle, thus it could not use the ICAP at its theoretical throughput. In the light of these considerations,

the reconfiguration time can be computed as:

$$Rec.\ Time = \frac{Bitstream\ size}{Data\ throughput} + t_{busy}$$

Where t_{busy} is the time elapsed during the busy state of the ICAP and *Data throughput* is the rate at which data are sent to the ICAP. The overall reconfiguration time can be lowered in one or more of the following ways:

- Using the ICAP with the maximum possible input data width;
- Optimizing the data throughput, i.e., the rate of partial bitstream transfer to the ICAP;
- Keeping the bitstream size small;
- Performing reconfiguration at the maximum frequency.

While designing the SEDPRC all these aspects were taken into consideration. Based on the assumptions that the ICAP can process an incoming datum at every clock cycle (i.e. $t_{busy} = 0$), that the surrounding system is able to provide a datum to the ICAP per clock cycle, that the data width is set to 32bit and that the component works at 150MHz, the reconfiguration throughput is 600MB/s.

3.3 The Speed Efficient DPR Controller

The ICAP interface consists of two separate data ports for reading and writing, chip enable and clock input signals and a busy output signal. The ICAP data ports can be either 8bit or 32 bit wide in Virtex-4 FPGAs. In order to achieve a higher throughput, in the proposed controller the ICAP is used in 32bit mode. The SEDPRC controller manages the ICAP interface and contains a BRAM to store partial bitstreams in. This memory is kept large enough to hold the entire partial bitstreams generated by the EHW system.

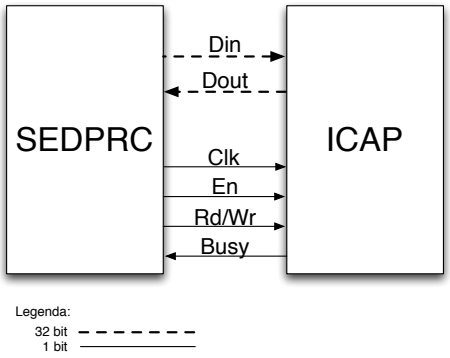

Fig. 2. Inteface between the ICAP and the Speed Efficient DPR Controller.

Figure 2 shows the interface between the ICAP and the SEDPRC component. During the reconfiguration process the SEDPRC drives the *En* and *Rd/Wr*

signals appropriately and transmits the partial bitstream to the ICAP from the BRAM. The ICAP indicates its readiness to accept the next data word through the handshaking signal *Busy*. Hence, before sending a data word to ICAP, it is necessary to check the status of the *Busy* signal. If the ICAP is busy, then the SEDPRC has to wait before sending the next data word. Further details about functioning of I/O ports, writing sequence and timing characteristics of the ICAP can be found in [23].

3.4 SEDPRC Flow

When the EHW system needs to evaluate a candidate solution, the processor writes the candidate solution bitstream to the SEDPRC memory and sends a *reconfiguration start* signal to the SEDPRC. Upon getting a command to start dynamic partial reconfiguration, the SEDPRC sets the *En* and *Rd/Wr* signals low for ICAP, accesses the stripped bitstream stored in the BRAM and sends it to the ICAP. The *Busy* signal is used for handshaking; data transfer is being stopped till ICAP gets ready to access data.

4 Experimental results

For gathering experimental results, The Speed Efficient Dynamic Partial Reconfiguration Controller (SEDPRC) has been designed using Xilinx ISE [24] and has been plugged into the HERA Evolvable Hardware system using Xilinx Platform Studio (XPS) [25]. The system has been implemented on a Xilinx Virtex-4 XC4VFX12-FF668 FPGA and validated with a ML403 board [26]. The experimentation has been carried out in two phases as follows.

4.1 SEDPRC vs Xilinx Standard Solutions

The cores are compared with respect to the maximum operating frequency and resource utilization in Table 1. The SEDPRC provides a maximum operating frequency of 220MHz. as against 212MHz and 121MHz being provided by OPB-HWICAP and XPS-HWICAP respectively. Though OPB-HWICAP can work at 212MHz, OPB (32-bit) can run at only 100MHz (Xilinx specification). Hence it cannot be operated beyond 100MHz, Similar is the case with XPS-HWICAP. Since SEDPRC architecture is processor/bus independent, we could successfully run SEDPRC and ICAP at 220MHz and achieve a maximum reconfiguration throughput of 838 MB/sec. To the best of our knowledge this is first attempt to use ICAP above 200MHz. Even at 100MHz, the throughput of SEDPRC is 400MB/sec and it is about 28 times more than that of XPS-HWICAP.

Looking at resource utilization in Table 1, the OPB-HWICAP along with the bridge utilize only one BRAM for write/read buffer and around 588 LUTs for logic. XPS-HWICAP utilizes almost ten times more LUTs compared to OPB-HWICAP but it does not need any BRAM and the write/read buffer is synthesized using LUTs. In SEDPRC, the LUTs consumption is inferior as compared

Table 1. Operating frequency and resource utilization comparison between SEDPRC and Xilinx IP-cores

Parameter	SEDPRC	OPB-HWICAP + Bridge	XPS-HWICAP
Max. operating frequency	220MHz	212MHz	121MHz
# 4-LUT	214	588	5580
# Slice Flip-Flop	224	372	443
# BRAM	33	1	0

to OPB-HWICAP and XPS-HWICAP, but BRAM resource utilization is considerably larger. In our case we have used 1 BRAM for header and 32 BRAMs for stripped partial bitstream files. The targeted FPGA has 192 BRAMs, hence the occupation is 17% of total available BRAMs. Based on the application and requirement it can be increased to hold more or bigger partial bitstreams. It should be noted that SEDPRC works independently from the processor, hence the processor can be be involved in other tasks while reconfiguration is in place.

4.2 Comparison with other solutions proposed in literature

Finally, we compared the developed core with other solutions that can be found in literature. For the sake of fairness, the comparison has been carried out against all the controllers developed for the Xilinx Virtex-4 FPGA family. The reconfiguration throughput of SEDPRC is compared with the reported throughput by researchers discussed in Section 3.1 [15,17–19,27,28]. The bar chart shown in Figure 3 indicates that SEDPRC provides the highest reconfiguration throughput.

5 Conclusions and future works

This paper describes how an Evolvable Hardware system based on dynamic reconfiguration has been improved introducing a new ICAP controller: the Speed Efficient Dynamic Partial Reconfiguration Controller (SEDPRC). Experimental results demonstrate that the novel controller brings benefits in both the overall running time and in the resource requirements. On one hand, the average time required by the reconfiguration time has been reduced by 37.2%. On the other hand, the number of LUTs needed is decreased by 81.3%; the controller requires the usage of one additional block RAM.

Ongoing works aim at speeding-up also the candidate solution evaluation process, as long as implementing the evolutionary algorithm operators as ip-cores. The overall goal is to make the HERA EHW system able to perform

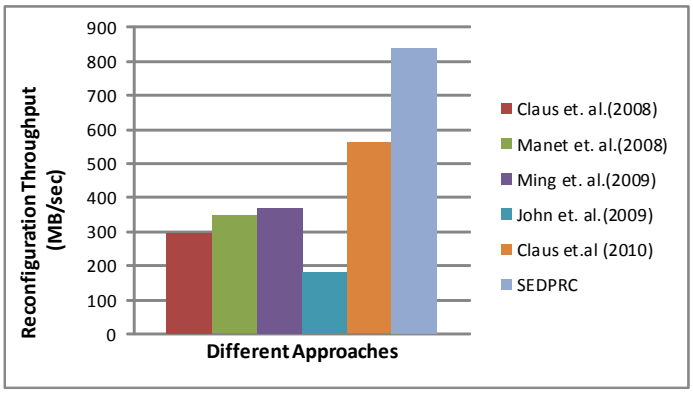

Fig. 3. Comparison with the reconfiguration throughput provided by the main Xilinx Virtex-4 ICAP controllers proposed in literature.

complete intrinsic evolution. Other ongoing works focus on porting the EHW system to Virtex-5 and Virtex-6 FPGAs, and on increasing the expressive power of candidate solutions. Finally, another interesting research topic is investigating whether alternative evolutionary algorithms can converge more efficiently to circuits characterized by the desired behavior.

References

1. P. Sedcole, B. Blodget, T. Becker, J. Anderson, and P. Lysaght, "Modular dynamic reconfiguration in Virtex FPGAs," *Computers and Digital Techniques, IEE Proceedings* -, vol. 153, no. 3, pp. 157 – 164, May 2006.
2. H. De Garis, "Evolvable Hardware Genetic Programming of a Darwin Machine," in *Artificial Neural Nets and Genetic Algorithms*, 1993, pp. 441–449.
3. T. Higuchi, T. Niwa, T. Tanaka, H. Iba, H. de Garis, and T. Furuya, "Evolving Hardware with Genetic Learning: a First Step Towards Building a Darwin Machine," in *Proc. of the 2nd Int. Conf. on From Animals to Animats*, 1993, pp. 417–424.
4. F. Cancare, M. Castagna, M. Renesto, and D. Sciuto, "A Highly Parallel FPGA-based Evolvable Hardware Architecture," in *Advances in Parallel Computing*, vol. 19, 2010, pp. 608–615.
5. F. Cancare, M. Santambrogio, and D. Sciuto, "A Direct Bitstream Manipulation Approach for Virtex4-based Evolvable Systems," in *Proceedings of thee IEEE International Symposium on Circuits and Systems*, 2010, pp. 853–856.
6. A. Upegui and E. Sanchez, "Evolving Hardware with Self-reconfigurable Connectivity in Xilinx FPGAs," in *Proceedings of the first NASA/ESA conference on Adaptive Hardware and Systems*, 2006, pp. 153–162.
7. L. Sekanina, "Evolutionary Functional Recovery in Virtual Reconfigurable Circuits," *J. Emerg. Technol. Comput. Syst.*, vol. 3, no. 2, July 2007.
8. K. Glette, J. Torresen, and M. Hovin, "Intermediate Level FPGA Reconfiguration for an Online EHW Pattern Recognition System," in *Adaptive Hardware and Systems, 2009. AHS 2009. NASA/ESA Conference on*, 2009, pp. 19 –26.

9. D. B. Vernekar, G. Malhotra, and V. Colaco, "Reconfigurable FPGA using Genetic Algorithm," in *Proc. of the Int. Conf. and Workshop on Emerging Trends in Technology*, 2010, pp. 493–497.
10. *OPB HWICAP (v1.00.b) User Guide*, Xilinx Inc., 2006.
11. *LogiCORE IP XPS HWICAP*, Xilinx Inc., March 2011.
12. J. F. Miller and S. L. Harding, "Cartesian Genetic Programming," in *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference*, 2009, pp. 3489–3512.
13. A. Upegui and E. Sanchez, "Evolving Hardware by Dynamically Reconfiguring Xilinx FPGAs," in *ICES*, ser. Lecture Notes in Computer Science, vol. 3637. Springer, 2005, pp. 56–65.
14. A. Cuoccio, P. Grassi, V. Rana, M. Santambrogio, and D. Sciuto, "A generation flow for self-reconfiguration controllers customization," in *Proc. of the 4th IEEE Int. Symp. on Electronic Design, Test and Applications*, 2008, pp. 279 –284.
15. C. Claus, F. Muller, J. Zeppenfeld, and W. Stechele, "A new framework to accelerate Virtex-II Pro dynamic partial self-reconfiguration," in *IEEE International Parallel and Distributed Processing Symposium*, 2007, pp. 1 –7.
16. Y. Krasteva, E. de la Torre, T. Riesgo, and D. Joly, "Virtex ii fpga bitstream manipulation: Application to reconfiguration control systems," in *Proc. of the Int. Conf. on Field Programmable Logic and Applications*, 2006, pp. 1 –4.
17. C. Claus, B. Zhang, W. Stechele, L. Braun, M. Hubner, and J. Becker, "A multi-platform controller allowing for maximum Dynamic Partial Reconfiguration Throughput," in *Proc. of the Int. Conf. on Field Programmable Logic and Applications*, 2008, pp. 535 –538.
18. P. Manet, D. Maufroid, L. Tosi, G. Gailliard, O. Mulertt, M. Di Ciano, J.-D. Legat, D. Aulagnier, C. Gamrat, R. Liberati, V. La Barba, P. Cuvelier, B. Rousseau, and P. Gelineau, "An Evaluation of Dynamic Partial Reconfiguration for Signal and Image Processing in Professional Electronics Applications," *EURASIP J. Embedded Syst.*, vol. 2008, pp. 1:1–1:11, January 2008.
19. M. Liu, W. Kuehn, Z. Lu, and A. Jantsch, "Run-time Partial Reconfiguration Speed Investigation and Architectural Design Space Exploration," in *International Conference on Field Programmable Logic and Applications*, 2009, pp. 498 –502.
20. C. Claus, R. Ahmed, F. Altenried, and W. Stechele, "Towards rapid dynamic partial reconfiguration in video-based driver assistance systems," in *Reconfigurable Computing: Architectures, Tools and Applications*. Springer Berlin / Heidelberg, 2010, vol. 5992, pp. 55–67.
21. D. K. Simen Gimle Hansen and J. Torresen, "High Speed partial Run Time Reconfiguration Using Enhanced ICAP Hard Macro," in *Proceedings of the 25th International Parallel and Distributed Processing Symposium*, 2011, pp. 169–175.
22. F. Duhem, F. Muller, and P. Lorenzini, "Farm: Fast reconfiguration manager for reducing reconfiguration time overhead on fpga," in *Reconfigurable Computing: Architectures, Tools and Applications*. Springer Berlin, 2011, pp. 253–260.
23. *Virtex-4 FPGA Configuration User Guide*, Xilinx Inc., June 2009.
24. *ISE Design Suite 13: Release Notes Guide*, Xilinx Inc., March 2011.
25. *Embedded System Tools Reference Manual*, Xilinx Inc., March 2011.
26. *ML401/ML402/ML403 Evaluation Platform User Guide*, Xilinx Inc., 2006.
27. J. C. Hofman, "High speed dynamic partial reconfiguration for fpga," Ph.D. dissertation, University of New Mexico, 2009.
28. C. Schuck, B. Haetzer, and Becker, "An interface for a decentralized 2d reconfiguration on xilinx virtex-fpgas for organic computing," *Int. J. Reconfig. Comput.*, vol. 2009, pp. 7:3–7:3, January 2009.