DPR in High Energy Physics

Wenxue Gao, Andreas Kugel, Reinhard Männer
Lehrstuhl Informatik V
B6,26
68131 Mannheim
wenxue.gao@ziti.uni-heidelberg.de

Norbert Abel, Nick Meier, Udo Kebschull
Kirchhoff Institute for Physics
Im Neuenheimer Feld 227
69120 Heidelberg
abel@kip.uni-heidelberg.de

Abstract

The Active Buffer project is part of the CBM (compressed baryonic matter) experiment and takes advantage of the DPR (dynamic partial reconfiguration) technology, in which a dynamic module can be reconfigured while the static part and other dynamic modules keep running untouched. Due to DPR, design flexibility and simplicity are achieved at the same time. The correctness and the performance have been verified by multiple tests.

1. Introduction

Over the past century, scientists have built up a deep understanding of the subatomic constituents of matter in the Universe and the fundamental forces binding them. More recently, they have developed compelling theories of how those building blocks came into being. Nevertheless, there are still significant gaps in our knowledge of the nature and evolution of matter on both a cosmic and a microscopic scale and there are many questions to explore. To answer these questions, a new international laboratory, the Facility for Antiproton and Ion Research, FAIR, is being constructed in Darmstadt, Germany. This is a highly sophisticated accelerator complex which will provide high-energy, precisely-tailored beams of antiprotons and many kinds of ions at unprecedented quality and intensities. These charged particle beams will then be accelerated and employed to create new, often highly exotic particles in a series of parallel experimental programms. The demanding experiments to be carried out at FAIR require considerable technological innovation: new superconducting magnet designs to steer highest possible intensities of particles, novel methods of accurately controlling the beam energies and large detectors to trace the variety of particles generated in experiments. Furthermore the extremely high data rate will require new hard- and software solutions such as DPR for processing, accessing and storing the results.[6]

Before processing or storing the data, it first has to reach the appropriate computer clusters. For this microchips like specialized ASICs and flexible FPGAs are usually used together. The data chain beginning at the detector and ending at the processing units is called DAQ (data acquisition). In the following we focus on one board in the DAQ chain of FAIR's CBM (Compressed Baryonic Matter) experiment, called ABB (active buffer board). In the CBM experiment, particle multiplicities and phase-space distributions will be determined. For example, the study of collective flow of charmonium and multi-strange hyperons will shed light on the production and propagation of these rare probes in dense baryonic matter.[3] The envisioned interaction rate of 10 MHz will produce a data rate of up to 1 TByte/s. The full data rate must be switched through a high speed network fabric into a computational network with configurable processing resources for event building and filtering.[5]
The ABB is the interface between the optically communicating hit data read out elements and the computer based DABC (Data Acquisition Backbone Core) [8]. It receives the hit data through an optical port and writes it to the PC-memory via PCI Express. At this point DPR technology is used. Today, typical applications of DPR are audio or video streaming [4], low power [9], automotive [7], networking [13], reconfigurable coprocessors [10] and fault tolerance [14]. We are using DPR to permit great flexibility (namely the possibility to dynamically set performance critical thresholds to optimal values) in combination with minimal design complexity – allowing higher clock rates and thus higher data rates, which are crucial for high energy physics experiments. In the following we want to take a closer look at the ABB and at DPR – and we want to clarify how these two things come together. Section 2 focuses on the ABB and the problems that occurred during the development. Furthermore it shows how DPR helped to solve these problems. Section 3 goes in detail with the implementation. In section 4 we present actual performance and resource consumption measurements and in section 5 we summarize our work.

978-3-9810801-5-5/DATE09 © 2009 EDAA

2. The Active Buffer Board

2.1 Base Functionality

Due to the high data rates (about 1TB/s) a high-speed network is used in the back-end processing unit design, where InfiniBand, Ethernet and other network technologies are utilized. [8] To realize the data transport from the Active Buffer to such a network, PCs are used to implement the complex network and event building algorithms. The Active Buffer Board (ABB), as briefly described in Figure 1, is plugged into the PCI Express slot of its host machine. On the ABB an FPGA (Xilinx XC4VFX60) provides the base functionality, that is receiving, buffering and forwarding the hit data. For receiving the data, two MGTs (Multi-Gigabit Transceivers) of the FPGA are used together with SFPs (Small Form-factor Pluggables). For the active buffering two DDR SDRAM modules are connected to the FPGA. And for the data forwarding four MGTs are connected to a PCI Express connector. We are using the Xilinx Virtex4 FX60, since this chip comes with the needed functionality (e.g. MGTs), it is big enough to contain all the needed logic and it can be reconfigured. The reconfigurability is used on the one hand for prototyping, on the other hand to provide scalability. In a high energy physics experiment, like CBM, the surrounding conditions can change due to measured values. Thus it is possible that the ABB functionality has to be changed after the installation of the ABB into the host PC. For this we need the FPGA's reconfigurability.

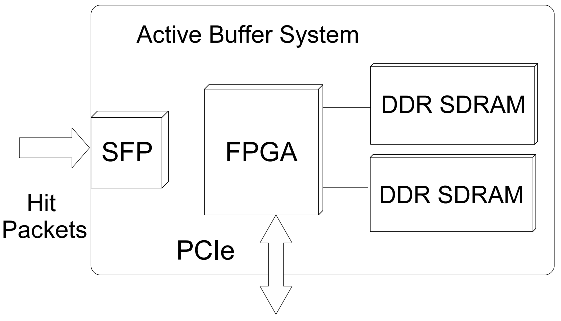

Figure 1. Active Buffer System Overview

The ABB provides the DMA (Direct Memory Access) functionalities *downstream* and *upstream*, targeting high-speed data transfer between the host memory and the event buffer (DDR SDRAM). Furthermore the ABB supports PCI Express PIO transactions to enable parameter control and status inspection over the board. In this paper, *downstream* DMA names the DMA transfer from the node host memory to the event buffer on the Active Buffer board while *upstream* DMA denotes the opposite direction. A PCI Express interface is much work if starting from the scratch. Thus we build it along with the Endpoint IP Core (V3.6) provided by Xilinx, which takes care of the physical and data link layer transactions of PCI Express. On the transaction layer, we build one PIO channel and two DMA channels, *downstream* and *upstream* respectively. The Endpoint IP core has a data width of 32 bit and 4 lanes, requiring 250 MHz operating frequency for the DMA logic. In PCI Express (Base Specification 1.1), the transaction is realized via packets (TLP, Transaction Layer Packet). For example, a write operation uses "posted" TLP, which needs no acknowledgment from the target. A read operation uses "non-posted" TLP, which involves a 2-step routine: first a read request is send out to the target, then the target responds with one or more packets, containing the requested data (CplD, completion with data payload) or returning unsuccessful information (Cpl, completion without data payload). Due to such features the PCI Express interface supports full-duplex mode and the bandwidth of both directions can be utilized independently. The overall throughput can be optimized using the according independent thresholds. The Base Specification has a restriction regarding the maximum (non-posted) read request size, namely MAX_READ_REQUEST_SIZE (MRS), which has one of the following values (in bytes): 128, 256, 512, 1024, 2048 or 4096. Also the maximum (posted) payload size, namely MAX_PAYLOAD_SIZE (MPS), is settable. Possible values are the same as those above. The *downstream* DMA logic has to obey the MAX_READ_REQUEST_SIZE when it is sending read requests to the host. The *upstream* DMA logic has to obey the MAX_PAYLOAD_SIZE when it is sending packets with payload to the host memory.

The DMA engines in the Active Buffer system have a proven performance close to the maximum speed of the DDR SDRAM chip, which is about 5 Gbps. The operation pattern of the DMA engines is scatter-gather DMA, which processes chain-like transactions. In figure 2, the block diagram of the DMA logic is depicted. The blocks "TB" denote the transaction buffers, which are realized as FIFOs.

2.2. Problems

Unfortunately, the special properties of PCI Express do not allow a reconfiguration of the whole FPGA due to link details. During the reconfiguration of the FPGA it's input and output pins are set to high-impedance. This causes the host PC to deactivate the PCI Express slot until the next hard reset. As a consequence, every reconfiguration of the PCI Express core area demands a host reboot. For prototyping this behavior is very disturbing. For the detector runtime it is absolutely inadmissible, since the host PC is part of a complex cluster network and can't be

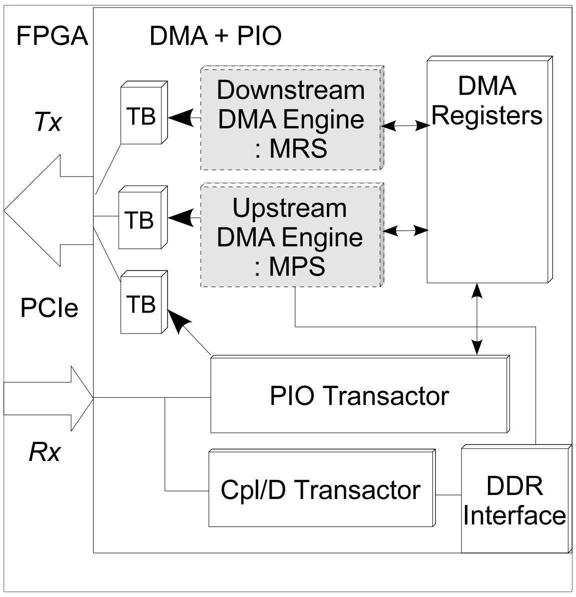

Figure 2. Design hierarchy

restarted at will.
A second problem comes with the DMA thresholds (MAX_READ_REQUEST_SIZE and MAX_PAYLOAD_SIZE). The safest way of implementation is to use the lowest values for both settings (that is 128 bytes). However, this has been proven to be unable to provide the best performance, which is crucial in our application. To get the best performance, the DMA engines have to be able to calculate the suitable request size and payload size according to the PCI Express configuration space and they have to be able to set these calculated values dynamically. The problem is, that such flexibility costs logic resources as well as verification effort. It also causes timing errors, due to the very tight timing constraints of 4 ns (250 MHz). So the situation is, that we need to be able to set the thresholds dynamically, but we do not have the logic speed to realize the necessary switch logic inside the FPGA.

2.3. Solutions

The PCI Express link problems are solvable in two ways. Firstly, one could use an external PCI Express chip, that realizes the PCI logic and will never be reconfigured – and thus never causes any link problems. Secondly, one could divide the FPGA using partial reconfiguration. Using this method, one part (the PCI Express logic) would be defined as *static* and would never be reconfigured, other parts would be defined as *dynamic* and could be reconfigured, while the *static* part continues uninterrupted. Both methods have drawbacks. A separated PCI Express interface chip brings additional costs and additionally occupied IO pins of the FPGA. The one-chip solution uses 8 pairs of differential pins of the FPGA, while the two-chip solution occupies more than 80 pins. The DPR solution needs more internal FPGA resources than the one with an external PCI Express chip and the DPR tool flow is more complex than the integrated one. Since in our case the internal resource consumption of the FPGA is not critical, but the pin consumption is very critical (besides the memory chips, there are a number of mezzanine connectors on the ABB that utilize most of the IO pins of the FPGA), the one-chip solution using DPR is the better choice.
Furthermore DPR also provides a very elegant solution for the threshold problem. It makes it possible to change the thresholds on demand without the need for extra logic inside the DMA controller. For this one can set up two dynamic modules for the two DMA engines. Each has 6 versions of dynamic bit files. Every version is correlated to a MAX_SIZE parameter value and loaded on demand. Hence, the flexibility is implemented, while the design complexity stays low and thus a timing closure becomes possible. It might also be possible to use only one dynamic module, containing both DMA engines, but this way the number of partial bit files would increase to $6 * 6 = 36$, instead of $6 + 6 = 12$. The PIO channel also has settable thresholds controlling its maximum buffer sizes. However, in our application the minimum MAX_SIZE setting (128 bytes) is never exceeded in the PIO channel and thus we do not have to change that parameter dynamically. Thus, the PIO channel logic is placed in the static part.
So, for a flexible structure and scalability, we applied DPR technology to the project. This way, the fixed part, including PCI Express core and DMA register space, is kept static, while the data transfer logic, mainly the DMA engines, is put into dynamic modules. That way we can reconfigure the DMA engines any time during the operation, without having to reboot the host machine. Furthermore DPR helps reducing the design complexity, since the several MAX_SIZE parameter values are realized via according partial bit files.

3. Implementation Details

In our project DPR helped solving many problems, but DPR does not only come with advantages. It also has its drawbacks. One of the biggest hindrances regarding DPR is the need to change the tool flow and the resulting need to change the design hierarchy. To realize the dynamical partial reconfiguration we used the Xilinx tools PlanAhead 10.1 and ISE 9.1 (with an additional DPR patch). It is possible but not recommended to do partial reconfiguration without PlanAhead.[12] PlanAhead makes the partitioning of the chip and the placement of the busmacros[11] much easier. Furthermore, PlanAhead introduces a new tool flow (figure 3), making it possible to generate the static bit file and all needed dynamic bit files with one tool. Using PlanAhead one has to change the design's hierarchy. The busmacros have to be placed in separate components which are instantiated at top level. The dynamic and the static parts have to be placed in separate components which are instantiated at top level, too. Of course, these components can contain subcomponents. Also all I/O pins should be placed at top level.[2]

The next step, after generating a suitable VHDL design, is the synthesis. First the static subcomponent and the dynamic subcomponents of the VHDL design have to be synthesized. These components are marked as top module and synthesized one after another. For PlanAhead every dynamic module must have the name given in the top level component. It is recommended that the XST does not add any input buffer or output buffer, as these components are internal. After every synthesis the resulting ncd file has to be backed up from the project directory, because the project must be cleaned afterwards. Finally the real top module has to be selected. Any other VHDL component has to be removed from the project. Thus, the top module component handles the dynamic and the static modules as black boxes. For the further steps with PlanAhead, the following files are recommended: the ngc files of the top module, of the static part and of every realization of the dynamic part, an ucf file for the used board and the nmc files of the used busmacros. Now PlanAhead is used to place the static part, the dynamic parts and the busmacros. The next step is to generate the bit files of the static and dynamic parts of the design using the patched *par* of ISE 9.1. After this, PlanAhead is able to merge the different bit files. That means it generates one static bit file with empty dynamic areas, one full bit file containing the static part and a default dynamic module for every dynamic area and partial bit files representing all the dynamic modules.

A dynamic module is placed inside a dynamic area. Using PlanAhead, dynamic areas have to be rectangles. In our project the PCI Express Endpoint IP core, the DMA register space and the PIO transactor reside in the static part. The two DMA engines make up two dynamic modules (figure 2). The interfaces between the static and the dynamic parts are the busmacros. A busmacro is a hard macro built on ordinary slices, usually containing an 8-bit wide bus. It can be customized with Xilinx tools such as the FPGA Editor and is also available from the busmacro library provided by Xilinx[1]. Properties like direction, synchronization, etc. have to be considered by creating or choosing a busmacro. Regarding the timing there are two versions of busmacros: synchronous and asynchronous. Synchronous busmacros have a better timing behavior, since they refer the data only at the rising edge of a connected clock and thus they separate the timing of the dynamic part from the timing of the static part. The disadvantage is the resulting additional pipeline step, which demands logic changes to keep the correctness of the logic. Using asynchronous busmacros, the logic does not have to be modified, but the timing is much harder to meet. In fact, for our DMA logic the combinatorial delay is a little bit too high to get timing closure. This is primarily caused by the necessary high clock rate of 250 MHz. Hence, the busmacros in our project are all synchronous. Furthermore, we use those with enable pins to prevent unpredictable behavior caused by unexpected signal toggling during the partial reconfiguration. The reconfiguration software holds the enable pins of the busmacros low until the corresponding dynamic reconfiguration is done.

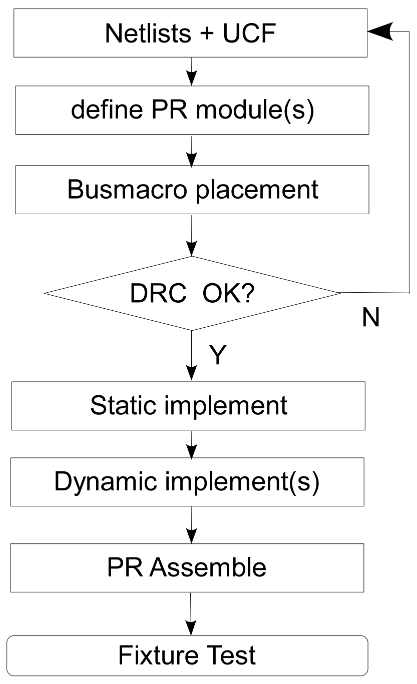

Figure 3. DPR procedure in PlanAhead

In the integrated version of DMA logic, the maximum size parameters are variables, while in the partial reconfigurable version, those two parameters are constants, defined via generics (VHDL) or parameters (Verilog HDL). In DPR mode, the FPGA is configured at power-up with the static bit stream, which supports basic PCI Express operations as well as basic PIO operations. After the host system (Linux 2.6) is up and the PCI Express driver is loaded, the software first checks the PCI Express configuration of the host system, especially the maximum value parameters, and accordingly configures the two DMA channels with proper partial bit streams. The dynamic reconfiguration process for every dynamic module is finished within a few milliseconds,

since every partial bit stream is small (less than 100 KBytes in size). Such dynamic reconfiguration is only necessary once at system startup, and hence, the reconfiguration overhead can be neglected.
Generally the flexibility and scalability of a design is increased by putting more logic into the dynamic parts. However, more dynamic logic will cause more boundaries between the static and the dynamic part. If there are too many busmacros in a project, not only the available resources are reduced for implementing logic, but also the timing constraints are more and more difficult to meet, no matter whether synchronous or asynchronous busmacros are used. To have a higher nominal frequency for a DPR project, synchronous busmacros are strongly suggested. However, as mentioned, the synchronous busmacros introduce an additional cycle of delay for the signals across it. Hence, to reduce resulting logic changes, it should be attempted to use the synchronous busmacros together with signals, that are not cycle-sensitive. In our project, it are the transaction buffers (”TB” in figure 2) that contain the busmacros. When introducing the busmacros we changed the FIFO controller a little bit, so that the writing FSM (finite state machine) contains the synchronous busmacros and thus contains one more pipeline step. Due to the already existing FIFO, the additional pipe line step did not change the general behavior of the design. From an external point of view, only the FIFO depth increased by one. For the FIFO controller it was much better to place the busmacros at the write port as to put them at the read port, since the read operation is realized via a complex handshaking and the delays are critical for the logic’s correctness. By contrast the FIFO write operation is not that critical. We just had to use the ALMOST-FULL signal for flow control instead of the FULL signal, as shown in figure 4.

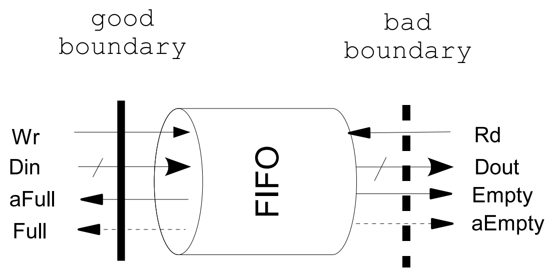

Figure 4. Module partitioning on the ABB

4. Test Analysis

A very important concern for a DPR project is that the design still meets its specifications and still meets timing after partitioning it. To reach this, one or more integrated designs, not using DPR but using busmacros, can be tried before implementing a partitioned one. Thus, introducing DPR, the DMA logic in the Active Buffer system stays unchanged. Our first tests referred to the performance. Some performance penalty was expected since the synchronous busmacros come with two additional clock cycles of delay for the DMA engines. However, in the tests, the performance difference between the integrated version and the dynamic version was as small as the measuring error range and thus can be neglected. Closer measurements showed, that this result is caused by the use of the FIFOs inside the transaction buffers (figure 2), which are almost never empty during the DMA. Table 1 shows the typical performance comparisons between integrated and partial designs. The tests were performed 100 times and the values listed are averaged.

	DMA size (kB)	Integrated (MB/s)	DPR (MB/s)
Downstream	64	673.9±0.32	673.9±0.78
	256	694.8±0.15	692.0±0.10
	1024	700.1±0.02	696.7±0.73
	4096	701.2±0.02	701.2±0.08
Upstream	64	430.8±1.75	432.9±0.18
	256	421.2±0.04	437.3±0.04
	1024	433.8±0.02	435.4±0.02
	4096	437.3±0.01	437.7±0.01

Table 1. DMA performance comparison

We also measured the resource consumption. For this the thresholds MRS=512 and MPS=128 were used. Table 2 shows the resulting resource consumption in detail. One can see, that the resource utilization for DPR is not greater than that of the integrated design. Since the integrated design has to implement all possible thresholds it needs more logic inside the DMA modules than the dynamic design. Using DPR only **one** threshold has to be implemented on the chip. The other thresholds are represented by partial bit files, that reside in the host system’s RAM and do not consume any space on the chip. However, using DPR one has to implement the bus macros, which consume LUTs (look up tables) and FFs (flip flops). Thus, in our project, the resource consumption of the integrated design and of the dynamic design is almost equal.

Altogether we use 87 (horizontal and vertical) 8-width busmacros, 41 for upstream module and 46 for downstream. These busmacros sum up 696 signals. 34 of these signals are open (meaning not used), since one busmacro always implements 8 signals, whether they are needed or not. A downstream bit file is 81738 bytes in size and an upstream bit file is 86848 bytes in size (as reference: the total configuration bit stream for XC4VFX60 is 2625438 bytes). The partial bit stream of the upstream module has a larger size

Resource type	FFs	LUTs
DPR Static	6901	8742
DPR module: upstream	460	559
DPR module: downstream	602	811
Busmacros	2088	1392
DPR total	10051	11504
Integrated total	10818	12051

Table 2. Resource consumption

than the one of the downstream module although the upstream module occupies fewer slices. The reason is, that some RAMB16s and FIFO16s increase the bit stream size. In the downstream area there are no such primitives included.

5. Summary

The ABB is part of the CBM high energy physics experiment and realizes the interface between read out elements and a computer cluster. It receives the hit data through an optical port and writes it to the PC's memory via PCI Express. Due to the high data rates in the CBM experiment, new hard- and software solutions such as DPR are needed to meet the demands. Regarding the ABB, DPR solves two major problems. First, no external PCI Express chip has to be used to enable updating reconfigurations. Thus, DPR helps saving costs while offering high scalability. Second, DPR permits a much higher flexibility regarding performance critical thresholds inside the DMA engines. It allows to change thresholds on demand without the need for extra logic inside the DMA controller. Hence, the design complexity stays low and thus a timing closure becomes possible.
Our tests show that DPR is feasible for large and complex designs. The dynamic design has been proven to work correctly. Furthermore, we successfully implemented the synchronous busmacros (coming with an additional pipeline step) without any performance lost. Due to the higher flexibility of the dynamic system, the performance critical thresholds can be set the best way – and thus the performance of the dynamic system is much better than the performance of an integrated design, which realizes only one value (remark: for resource comparison we implemented a flexible integrated design, but this design never met the timing constraints). The resource consumption of the flexible integrated design is almost the same as the one of the dynamic design. The reason is, that using DPR some elements (like busmacros) have to be added, but on the other side the flexibility is realized via partial bit files and not via resource-killing multifunctional hardware.
Many features are still being exploited. Our next step intends to have a pair of alternative DMA channels, in which the direction of a DMA channel can be reconfigured. Besides the current DMA channel-pair mode, we can have two upstream channels or two downstream channels at the same time, depending on the demand of the transportation situation around the Active Buffer. This structure is sure to give the system more powerful functions which cannot be done without DPR technology.

References

[1] *www.xilinx.com*. Xilinx Inc.

[2] *PlanAhead User Guide*. Xilinx, 2008.

[3] *www.gsi.de/fair/experiments/CBM*. GSI, Darmstadt, 2008.

[4] C. Claus and J. Zeppenfeld. *Using Partial-Run-Time Reconfigurable Hardware to accelerate Video Processing in Driver Assistance System*. DATE conference, Nice, France, 2007.

[5] H. Essel. *FutureDAQ for CBM: on-line event selection*. GSI, Darmstadt, 2006.

[6] M. Hinkelmann. *Facility for Antiproton and Ion Research*. GSI, Darmstadt, 2008.

[7] K. M.-G. R. C. J. L. J. E. J. Becker, M. Huebner. *Automotive Control Unit Optimisation Perspectives: Body Functions on-Demand by Dynamic Reconfiguration*. DATE conference, Munich, Germany, 2005.

[8] N. K.-S. L. Joern Adamczewski, Hans Essel. *Data Acquisition Backbone Core DABC*. GSI, Darmstadt, 2007.

[9] M. H. K. Paulsson and J. Becker. *Cost- and Power Optimized FPGA Based System Integration: Methodologies and Integration of a Low-Power Capacity-Based Measurement Application on Xilinx FPGAs*. DATE conference, Munich, Germany, 2008.

[10] M. S. Lars Bauer and J. Henkel. *A Computation- and Communication-Infrastructure for Modular Special Instructions in a Dynamically Reconfigurable Processor*. FPL conference, Heidelberg, Germany, 2008.

[11] P. Lysaght and B. Blodget. *Enhanced architectures, design methodologies and cad tools for dynamic reconfiguration of Xilinx FPGAs*. FPL conference, Munich, Germany, 2006.

[12] M. G. Nij Dorairaj, Eric Shiflet. *PlanAhead Software as a Platform for Partial Reconfiguration*. Xilinx, 2005.

[13] S. H. R. Sinnappan. *A reconfigurable approach to packet filtering*. FPL, Belfast, Northern Ireland, UK, 2001.

[14] N. M. W. Zheng and S. Chau. In-system partial run-time reconfiguration for fault recovery applications on spacecrafts. *Proc. of IEEE International Conference on Systems, Man, and Cybernetics*, 4:3952–3957, October 2005.