2010 International Conference on Field Programmable Logic and Applications

Improving QoS of Multi-Layer Networks-on-Chip with Partial and Dynamic Reconfiguration of Routers

Leandro Möller[1], Peter Fischer[1], Fernando Moraes[2], Leandro Soares Indrusiak[3], Manfred Glesner[1]
[1] Darmstadt University of Technology - Institute of Microelectronic Systems - Darmstadt, Germany
[2] Faculty of Informatics - Catholic University of Rio Grande do Sul - Porto Alegre, Brazil
[3] Department of Computer Science - University of York - York, United Kingdom
Email: moller@mes.tu-darmstadt.de

Abstract— **Networks-on-Chip (NoC) allow several data transfers to occur in parallel and are indeed the communication infra-structure of future hundred-cores Systems-on-Chip (SoCs). However, if specialized modules are sending data at full speed to the NoC, Quality of Service (QoS) can be no longer guaranteed. This work presents a multi-layer mesh NoC approach to improve the QoS of such communication hungry SoCs. While one mesh layer is fixed in the system for control purposes, other data layers can be configured at runtime to provide the desired data throughput required by the application. This is accomplished by partially and dynamically reconfiguring the data layer routers. Arbitration algorithms, routing algorithms and huge crossbars are removed from the data layer routers, because all data routers in the path a configured accordingly before its utilization. A SoC following this idea was prototyped in a Virtex-4 FPGA and the Early Access Partial Flow was used to partially and dynamically reconfigure the NoC. We show that 120 (5!) different configurations are needed for each reconfigurable router with 5 bidirectional ports. Each configuration requires 33KB of memory and occupies 32 CLBs of area.**

Keywords— Quality-of-Service, Multiprocessor Systems-on-Chip, Networks-on-Chip, Partial and Dynamic Reconfiguration.

I. INTRODUCTION

FPGAs play a big role in today's development of microelectronic systems. These programmable hardware devices are not only used as first hardware design tests before the deployment into native hardware, but are also growingly used as the central parts of the final system. They offer a great flexibility as their hardware function is described by a configuration memory, which can be easily changed, reconfiguring the device. Partial and dynamic reconfiguration allows the exchange of a part of the configuration memory, changing its hardware function, while the rest of the FPGA keeps running.

However, FPGAs pay the price of flexibility with high power consumption, area and slow clock frequencies when compared to an equivalent logic implemented on ASIC. A trade-off between FPGA and ASIC is the embedded FPGA (eFPGA) [1], which implements the core functions of a system in ASIC and the parts that need flexibility in FPGA. Such architecture, if composed by several blocks of ASICs and FPGAs, would provide a highly massive parallel computing system. Parallel computations would exist between ASIC cores and FPGA cores. And parallel computations would exist on every FPGA core, allowing different pipelines to be configured to speed-up a specific task of the system.

This eFPGA approach helps to increase the computing performance of future Systems-on-Chip (SoCs) based on hundreds of cores, but impose a challenging task for the communication infrastructure. Clearly Networks-on-Chip (NoCs) with optimized schemes will be employed to provide Quality of Service (QoS) among the communicating cores. Extensive research has been done to improve the QoS in NoCs, covering many aspects like circuit switching [2], virtual channels [3] and priority schemes [4].

The main concern of this work is to provide QoS by providing a special link between source and target of communication. This is accomplished by partially and dynamically reconfiguring routers when long-term communications are required. While one mesh layer is fixed in the system for control purposes, other data layers can be configured at runtime to provide the desired data throughput required by the application. Crossbar, Arbitration and routing algorithms are extracted from the data layer routers, because source and target cores will be known in advance and the routers in the path will be configured accordingly before its utilization. The result is much smaller data routers when compared to standard routers, allowing the creation of several data layer networks for applications to use.

This work is divided as follows. Section II presents related works in the field of QoS for NoCs. The idea of the multi-layer NoC approach is explained in section III. Section IV presents the implementation of the multi-layer NoC. A detailed analysis of the system is presented in Section V. Section VI concludes this work.

II. RELATED WORKS

Multimedia applications involving voice, audio and video are known to transfer large amounts of data, tolerate low levels of jitter and to require high throughput [5]. In contradiction, the control of such applications over an MPSoC requires small urgent packets to be frequently transferred among IP cores. The coexistence of both types of traffic sharing a single communication infrastructure is the major concern of several works that address QoS for Systems-on-Chip.

A suitable alternative to provide hard real-time QoS is to use Circuit Switched (CS) NoCs. CS NoCs enforce latency, throughput and jitter constraints by reserving all network resources before starting a communication. CS NoCs also require only one position buffer, because after the circuit is established, flits are transferred ahead in a pipeline fashion. The main problem of a standard CS NoC is that currently reserved but not used resources cannot be used by other communications. One example of NoC that employ CS NoCs is [2].

That problem can be minimized by using Virtual Channels (VC). VCs can use the concept of time division multiplexing (TDM) to share one physical channel into several VCs, where each VC contains its own individual buffer. One examples of NoC that employ VCs with TDM is [3].

One problem of TDM is that it provides a deterministic behavior of choosing which packets can be forwarded, considering all packets with an equal importance. Priority schemes can substitute the deterministic behavior of TDM and be combined with VCs by setting each VC a different priority. This way all data traffics must fit in a priority scale before entering the NoC and with this information intermediate routers can preempt traffic with lower priority, passing more urgent packets ahead before less urgent ones [4].

Several works also observe that CS NoCs are indeed a good alternative to provide hard real-time QoS for multimedia applications, but fail to provide a low latency communication infrastructure required for control purposes in a single NoC. This problem has lead to the implementation of a second NoC following the Packet Switching (PS) technique to deal with control packets [6][7]. PS NoCs do not need an establishment of a circuit before sending the required information to the NoC, thus resulting a lower latency and the resource allocation is limited to the time that the packet is being transferred.

This work also employs one PS NoC for control purposes and one CS NoC for data purposes. The novelty of this work is that the CS routers are hard-wired and can have their connection between input and output ports changed only by means of partial and dynamic reconfiguration. IP cores use a static PS NoC to inform a configuration controller that they wish to establish a new long-term data communication with any other IP core. The main advantage of such solution is that CS routers only contain wires, thus crossbar, arbitration and routing algorithms are suppressed. The result is less area usage and a scalable infrastructure, allowing future reconfigurable MPSoCs to have several small CS NoC layers to be controlled by one PS NoC.

III. Proposed Architecture

Figure 1 illustrates the multi-layer NoC proposed by this work, as well as the IP cores that will be used for testing purposes and explained further in Section IV. As illustrated, all IP cores connect to one PS router and one CS router, following a mesh topology. A Network Interface (NI) is responsible to select which NoC will be used in each situation.

A. Packet Switching NoC

The packet switching NoC used in this work is the HERMES NoC [8], which is a low area overhead NoC available for free on the internet. The Hermes NoC follows a regular mesh topology, with input buffers and a simple but efficient XY-routing algorithm. The switching method used is the wormhole and the arbitration is performed in a round robin fashion. All data transfers can take place asynchronously and the flow control uses a standard handshake algorithm.

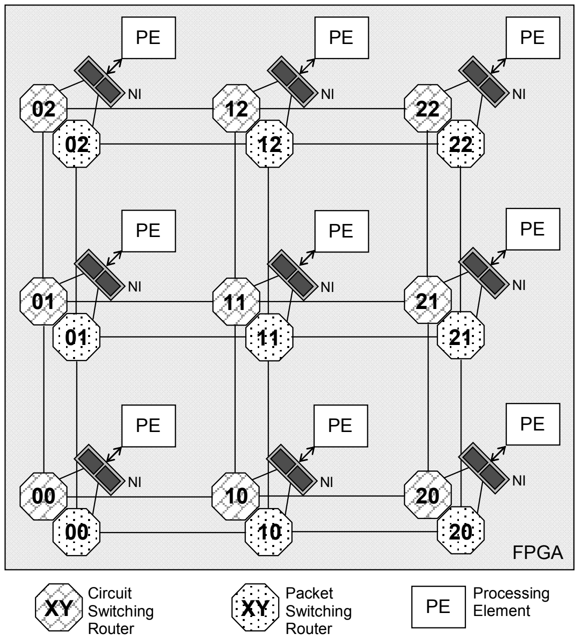

Figure 1. Conceptual view of a multi-layer NoC.

B. Circuit Switching NoC

The CS routers are pre-switched and hard-wired routers, as illustrated in Figure 2. In fact they only consist of simple wires, connecting inputs to outputs. No logic that was previously needed for the PS routers is needed for them, because the configuration controller is responsible to compute which inputs should be connected to which outputs and triggers the respective partial bitstream that configures the router accordingly.

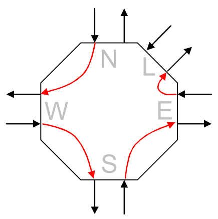

Figure 2. Graphical illustration of one hard-wired CS router.

The hard-wired CS router illustrated in Figure 2 is only one of 120 partially reconfigurable CS routers possibilities to connect five input ports to five output ports with no one connected twice. These 120 possibilities can be calculated by the factorial of five (5!).

C. The Communication Flow

If one IP core wants to send small messages or control information to another IP core, a control packet is assembled by the NI of the source IP core and sent normally through the PS NoC. If one IP core wants to start a long data communication with another IP core, the following sequence of events occur: 1) IP core source informs its local NI that it would like to establish a route to IP core target; 2) The NI sends a route establishment request packet through the PS NoC to the configuration controller (CC), which is mapped to one IP core of the NoC; 3) The CC is aware about the routing protocol, therefore it calculates the route and also which inputs should be connected to which output of each CS router in the path between source and target; 4) The CC search among the 120 partial bitstreams, which can satisfy the right connection between inputs and outputs calculated in the previous step (this is repeated for each router in the path between source and target); 5) If all CS routers between source and target are free, the CC triggers the partial reconfiguration of all CS routers; 6) The CC sends a confirmation packet through the PS NoC back to the source NI; 7) The source NI informs the source IP core that the data can now be sent; 8) The Source IP core starts sending data to the its NI, which directly bypass to the CS NoC; 9) After the complete data communication has been sent from IP core source to IP core target, the IP core source informs its local NI that the route can be canceled; 10) The source NI sends a disconnect packet through the PS NoC to the CC; 11) The CC removes the route from its internal table, which frees up the resources for later requests.

IV. IMPLEMENTATION

In order to test the architecture proposed in Section III, nine IP cores were connected to a 3x3 multi-layer mesh NoC similarly as illustrated in Figure 1. However, two RS-232 serial interfaces are used and connected to router addresses 00 and 22. The other seven PE cores contain a Compute-And-Forward (CAF) core.

Data can be sent and received to/from the system by the serial interface core connected to router address 22. The serial interface core connected to router address 00 is used as a configuration controller (CC), which its functionality is implemented in software by the PC. This software is responsible to verify if a new partial reconfiguration is required and if so to find the appropriate partial bitstream among the 120 possibilities and trigger the partial reconfiguration by executing the Impact tool from Xilinx.

The CAF cores are dedicated hardware blocks, each one composed by a different finite state machine. Each CAF core receives a packet, compute depending on the commands and data received in the packet, and finally forward the packet to another CAF or to the output of the system.

The system was prototyped in a ML403 board from Xilinx containing a Virtex-4 FPGA "XC4VFX12-FF668-10". As this board offers only one serial interface, a second one is attached in form of a simple level-shifter with a MAX3232 chip. This is then connected to the general-purpose I/Os of the FPGA. The two RS-232 serial interfaces are converted to USB and finally accessed as regular COM-ports by the PC.

The latest available software compatible with partial reconfiguration is the ISE version 9.2.4 with the Early Access Partial Reconfiguration (EAPR) software overlay version 12. In addition to this, PlanAhead version 10.1 is used to graphically support the EAPR flow and batch script the implementation of the different partial bitstreams.

The EAPR flow adds strong restrictions to the design, especially with respect to hierarchy. The top-level of the hierarchy is used to black-box instantiate the static and partially reconfigurable modules, as well as global logic as DCMs, clock drivers and I/Os. No other logic must be described at the top-level. Instead, it must be branched out and black-box instantiated as well. In addition to this Bus-Macros (BM) are required by the EAPR flow to connect partially reconfigurable modules with the static logic. Figure 3 presents how the BMs communicate with the partially reconfigurable CS router as well as it presents the boundary between reconfigurable and static logic.

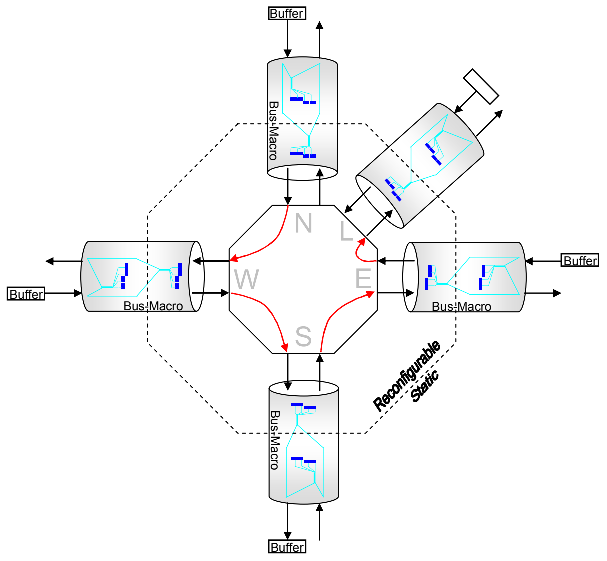

Figure 3. Graphical illustration of one reconfigurable hard-wired CS router.

These difficulties added to the fact that 120 partial bitstreams must be generated for each CS router and that Xilinx do not provide a tool to relocate partial bitstreams, restricted this work to test the partial and dynamic reconfiguration of only one CS router instead of the 9 that compose the 3x3 mesh of the case study. Therefore, only the CS router with address 11 presented in Figure 1 is partially and dynamically reconfigured.

Another restriction enforced by the EAPR flow is that signals cannot pass directly from inputs to outputs, i.e. a partially reconfigurable module cannot be composed by wires only. A countermeasure applied was to add inverters to every wire that connected inputs to outputs of the reconfigurable CS router. The signals are then inverted again outside the reconfigurable CS router, resulting in the original signals again.

Figure 3 also presents buffers kept outside the partially reconfigurable region. One flit input buffers are used associated to each CS router to comply with the "pipeline" scheme from NoCs, which avoid long wires and add 1 clock cycle latency on each hop.

V. Results

Figure 4 shows the placement of the complete SoC on the FPGA that satisfied the restriction of the EAPR flow. Due to the large Power-PC block at the bottom left of the FPGA (black rectangle) the CLB structure becomes a bit irregular. In addition to this, the IP cores as well as the routers have different sizes, resulting in a slightly irregular structure of placement. The prototyped system used CS routers with one flit buffer per input port and 8 bits per flit, while the PS routers are configured with 4 bits per flit and 8 flit position buffers per input port.

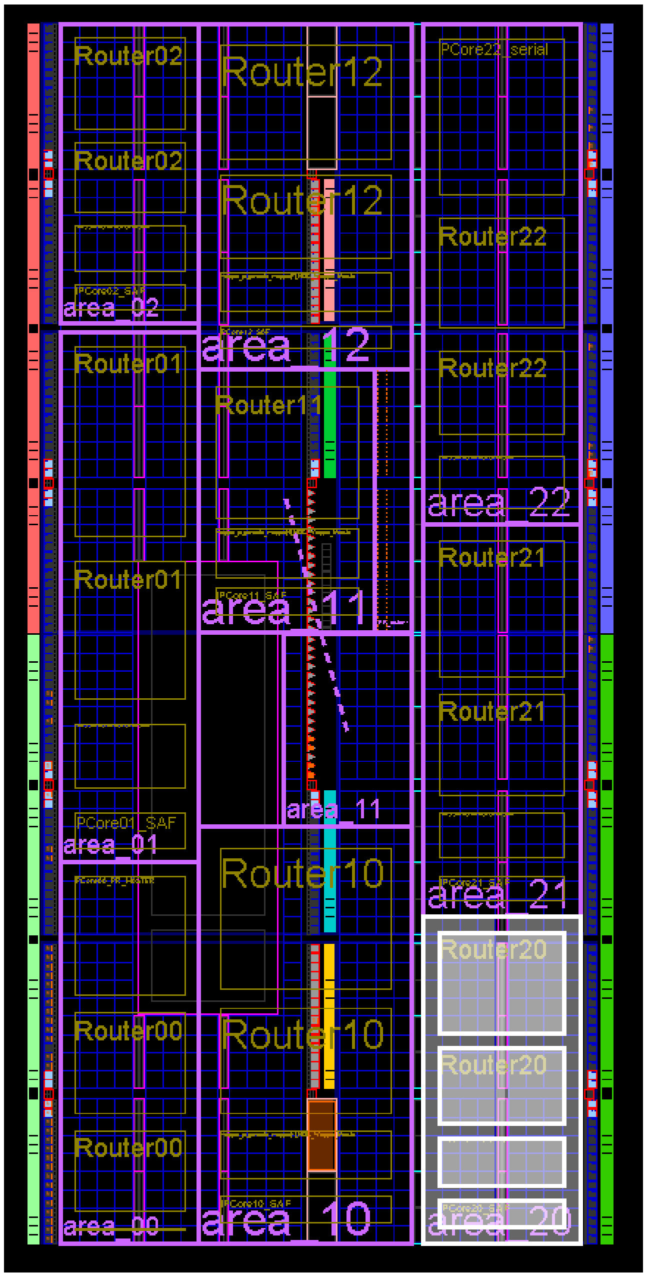

Figure 4. Floorplan of the case study.

The partially reconfigurable CS router, located on the right side of the upper area_11 rectangle of Figure 3, is of special interest due to its optimizations for reconfigurability. The partially reconfigurable CS router is placed in a narrow rectangle area of 2 CLBs. Such a narrow rectangle minimizes the size of the partial bitstream file and speeds up the partial reconfiguration process, since the minimum partial reconfiguration region allowed by the EAPR flow is one CLB column by 16 CLB rows for Virtex-4 devices. Our reconfigurable CS router actually uses 28 of the 32 available in the partially reconfigurable region, which would mean 224 LUTs and 224 FFs, but as 2 columns of 16 CLBs are reserved for the partial bitstream, the CS router accounts 256 LUTs (2% of XC4VFX12) and 256 FFs (2% of XC4VFX12).

ISE reported a total area usage for the static project of 8291 LUTs (75% of XC4VFX12) and 3019 FFs (27% of XC4VFX12). Adding the reconfigurable region of the CS router to the static project the total area usage is 8547 LUTs (78% of XC4VFX12) and 3275 FFs (30% of XC4VFX12).

Debugging and monitoring partially reconfigurable systems is another difficult issue when the EAPR flow is being used. ChipScope Pro Analyzer software from Xilinx is an easy to use tool to visualize signals in waveforms from a system executing on FPGA. However, ChipScope fails when the EAPR software overlay is installed and therefore it cannot be used in this work. With the help of a logic analyzer the system was debugged and extra timing results were obtained.

At time zero the CAF IP core connected to router 10 requests to its NI the establishment of a data communication with CAF IP core connected to router 12. Two clock cycles later, the NI starts sending a packet through the PS NoC to the CC connected to router 00. Then the header of the packet takes 6 clock cycles for routing and switching on router 10 and more 6 clocks cycles on router 00. Two clock cycles later, the NI forwards the packet to the RS-232 serial interface core. After two clock cycles again the serial interface core starts forwarding the packet to the physical serial interface available on the prototyping board. After that, timing cannot be measured so accurately on the PC, but the whole process of searching an appropriate CS router configuration that satisfies the required routing, triggering the partial reconfiguration with the Impact software from Xilinx, and sending an acknowledgement back to the RS-232 serial interface cores takes around 54ms, being 17.3ms for the actual loading of the partial bitstream measured by probing the JTAG interface with the logic analyzer. In the next step the acknowledgement packet travels from the RS-232 serial interface core back to the NI (2 clock cycles), router 00 (6 clock cycles), router 10 (6 clock cycles), NI (2 clock cycles), and arrives back (2 clock cycles later) to the CAF IP core source of the communication, totalizing 18 clock cycles. Finally the data packet can take its way from the source to the destination, using the data-NoC and taking 1 clock cycle latency per flit per hop.

Using the Internal Configuration Access Port (ICAP) the time needed for reconfiguration can be greatly lowered. The ICAP interface supports a clock frequency of 100MHz with a width of 32 bit and allows continuous data transfer. With this the loading of a partial bitstream of 33KB as it is used here is expected to take only about 20µs. This speeds up the configuration process by three orders of magnitude. An extra speed up of the ICAP may be achieved by overclocking the ICAP up to 300MHz [9].

VI. Conclusions

This work described a multi-layer NoC, where a packet switching NoC is used for control and one or more circuit switching NoCs can be used for data communication. The main novelty is to partially reconfigure the CS routers in the path of communication before sending data. The disadvantage is the initial latency to partially reconfigure the system. One advantage is that the crossbar, arbitration and routing algorithms can be excluded from the CS router, thus allowing the implementation of more CS NoC layers in the same area of a regular CS NoC. Another advantage is that after the path is set the latency is one clock cycle per hop and QoS is guaranteed.

Results have shown that each partially reconfigurable CS router bitstream needs 33KB of memory to be stored and that 120 possibilities to connect five input ports to five output ports with no one connected twice exist. The whole process of searching an appropriate CS router configuration that satisfies the required routing, triggering the partial reconfiguration, and sending an acknowledgement back to the RS-232 serial interface core takes around 54ms, being 17.3ms for the actual loading of the partial bitstream. However this time can be reduced to 20µs if the ICAP is used. Further partial reconfiguration time reduction can be achieved by overclocking the ICAP to up to 300MHz [9]. The drawback of this initial delay can be tolerated in return to the gain of lower latency for the data transport, especially in systems where long-living connections exist.

Even though Xilinx FPGAs are not a favorable architecture to prototype the work herein described, they were used as first step because they allow partial reconfiguration and they are large enough to map one 3x3 PS NoC, one 3x3 CS NoC and a few IP cores to test the described idea. One limitation is that only the switching matrix of the CLB was supposed to be partially reconfigured, but the switching matrix coexists with slices, thus wasting precious logic resources of the FPGA. Also, the EAPR flow limits the debugging options, waste resources for fixing logic at a specific place with bus-macros, and does not allow a reconfigurable module composed by wires only.

Suitable hardware architectures to prototype the described multi-layer NoCs are ASICs composed of several large embedded switching matrix blocks. These switching matrix blocks should allow hundreds of input wires to be routed to any of the five possible output ports. Another requirement is that they shall be fast and individually reconfigurable. For each input wire a one bit register should be associated to keep up the "pipeline" scheme of NoCs.

References

[1] S. Ahmed, J. Eydoux, L. Rougé, J. Cuelle, G. Sassatelli, and L. Torres, "Exploration of power reduction and performance enhancement in LEON3 processor with ESL reprogrammable eFPGA in processor pipeline and as a co-processor," In Design, Automation & Test in Europe Conference & Exhibition, 2009, pp. 184-189.

[2] A. Banerjee, P. Wolkotte, R. Mullins, S. Moore, and G. Smit, "An Energy and Performance Exploration of Network-on-Chip Architectures," IEEE Transactions on Very Large Scale Integration Systems, vol. 17 (3), pp. 319-329, 2009.

[3] M. Al Faruque and J. Henkel, "Minimizing Virtual Channel Buffer for Routers in On-chip Communication Architectures," In Design, Automation and Test in Europe, 2008, pp. 1238-1243.

[4] B. Vermeulen and K. Goossens, "A Network-on-Chip monitoring infrastructure for communication-centric debug of embedded multi-processor SoCs," In International Symposium on VLSI Design, Automation and Test, 2009, pp. 183-186.

[5] G. Varatkar and R. Marculescu, "On-Chip Traffic Modeling and Synthesis for MPEG-2 Video Applications," IEEE Transactions on Very Large Scale Integration Systems, vol. 12 (1), pp. 108-119, 2004.

[6] D. Göhringer, B. Liu, M. Hübner, and J. Becker, "Star-Wheels Network-on-Chip Featuring a Self-Adaptive Mixed Topology and a Synergy of a Circuit - and a Packet-Switching Communication Protocol," In International Conference on Field Programmable Logic and Applications, 2009, pp. 320-325.

[7] M.B. Stensgaard and J. Sparsø, "Renoc: A network-on-chip architecture with reconfigurable topology," In Proceedings of the Second ACM/IEEE International Symposium on Networks-on-Chip, 2008, pp. 55-64.

[8] F. Moraes, N. Calazans, A. Mello, L. Möller, and L. Ost, "HERMES: An Infrastructure for Low Area Overhead Packet-Switching Networks on Chip," Integration, the VLSI Journal, vol. 38 (1). 2004, pp. 69-93.

[9] C. Claus, F. Altenried, W. Stechele, "Dynamic Partial Reconfiguration of Xilinx FPGAs Lets Systems Adapt on the Fly", Xcell journal, pp 18-23, first quarter 2010, 2010.