DESA: DISTRIBUTED ELASTIC SWITCH ARCHITECTURE FOR EFFICIENT NETWORKS-ON-FPGAS

Antoni Roca, José Flich

Parallel Architectures Lab (GAP)
Universitat Politècnica de València, Spain

Giorgos Dimitrakopoulos

Electrical and Computer Engineering Dept.
Democritus University of Thrace, Greece

ABSTRACT

Networks-on-FPGA consist of a network of switches connected with point-to-point links and can cover sufficiently the communication needs of complex systems implemented on FPGA platforms. The efficient implementation of such networks requires the appropriate tuning of their components to the characteristics of the FPGA's logic and memory resources. In this paper, we present a distributed switch architecture that exploits in the best way the structure of the FPGA and achieves significant area/delay savings when compared to baseline switch architectures; more than 50% increase in operating frequency is achieved for similar area. The proposed switch operates as an elastic pipeline and can be spread throughout the FPGA chip irrespective the topology of the network and without limiting the placement options of the corresponding EDA tools.

1. INTRODUCTION

Platform FPGAs integrate an increasing number of components including processors, application specific accelerators, memories and IO controllers [1, 2]. Simple communication media such as buses or ad-hoc point-to-point connections do not suffice for keeping the system's performance to acceptable levels. The system-wide communication needs to be efficiently organized both physically, structured wire connections, and logically, efficient communication protocols, using a soft on-chip interconnection network [3]. Scalable interconnection networks that use a network of switches connected with point-to-point links can parallelize the communication between these modules and improve performance significantly. Such on-chip interconnection networks are already a mainstream technology for ASICs, while they are becoming a need and critical in FPGA-based systems [4].

The first on-chip interconnection networks mimicked the designs that were architected for large, high performance multiprocessors. However, as interconnects migrate to the on-chip environment, constraints and tradeoffs shift and they should be appropriately adapted to the implementation fabric [5, 6, 7]. Similarly, the first Network-on-FPGA mimicked generic Network-on-Chip (NoC) designs without considering the characteristics of the FPGA [8]. ASIC-oriented NoCs, even if appropriately customized for the on-chip environment, still lead to sub-optimal designs when transferred directly to FPGAs. This deficit is attributed to the inefficient mapping of the network's switches and their components to the configurable logic array of the FPGA [5, 9].

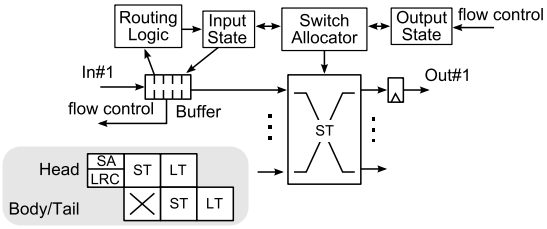


Fig. 1. The baseline organization of a wormhole switch.

The switches are the backbone of the NoC. A baseline wormhole (WH) switch organization is shown in Fig. 1. Routing logic unwraps incoming packet headers and determines their output destination. Such decoding and routing computation can be prepared in the previous switch and used in the current one. This optimization is called lookahead routing (LRC) and allows routing information to be performed in parallel with the rest of tasks. At the same time, the packet header competes for the selected output port, since the rest of input queues may have a request for the same output port at the same time. If it wins this stage, called switch allocation (SA), it will traverse the crossbar (ST - switch traversal) at next cycle, and, one cycle later, it will pass the output link (LT - link traversal) towards the next switch. In WH switches, the SA stage is constructed using a single arbiter per output of the switch that decides independently which input to serve.

Both LRC and SA stages are performed only for the head flit of each packet. The remaining body and tail flits follow the same route that has already been reserved by the head flit. Therefore, in a WH router, if a packet at the head of a queue blocks, either because it loses access to the output (SA) or because the downstream buffer is full, all packets behind it also stall.

The tasks implemented in each switch can be performed all on the same cycle. This approach reduces the switch latency measured in cycles but increases significantly the logic and wiring delay inside each switch. For this reason, many proposals have split the operation of the switch in multiple pipeline stages [9], as shown in Fig. 1.

1.1. Networks on FPGAs

Based on the implementation constraints set by the FPGA platform and the application environment, the NoC designer should determine the radix of the switches, their possible pipelined organization, and the width of the datapath, as well as the amount of buffering required to sustain an acceptable level of performance. A possible mapping of the baseline switch to the FPGA is shown in Fig. 2. The ST, SA and LRC stages are mapped to the LUTs and the registers of the FPGA, while the input buffers can be mapped either to the LUT-based distributed RAMs or to to the multi-port reconfigurable SRAM macros [5].

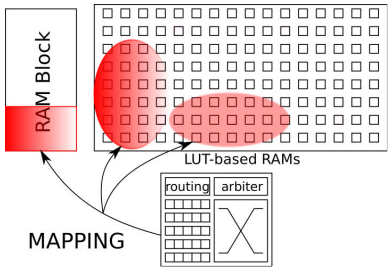


Fig. 2. Inefficient mapping of the baseline switch.

The buffers of the switch have a capacity much bigger than LUT-based RAMs but much smaller than the larger SRAM macros. Therefore, building such buffers on FPGAs will either over-utilize the available logic-based RAMs and limit the logic available for other functions, or under-utilize the scarce SRAM macros. On the contrary, in FPGAs, the wiring substrate available to the designer is abundant. Most of the time it is overprovisioned in order to cover sufficiently the mapping of a wide range of possible circuits. Thus, for the networks-on-FPGAs the wires can be considered for free compared to other resources such as logic and memory.

A set of guidelines for the implementation constraints of networks-on-FPGAs can be summarized as follows:

Guideline 1: Utilize wide datapaths taking full advantage of the abundant wiring without violating the area constraints. This approach benefits also packet latency by decreasing its serialization part. It takes less time to transfer a 1Kbit packet through 128b wide links relative to a 32b baseline.

Guideline 2: Use the appropriate radix for the switches that does not limit the maximum clock frequency of the network. As long as the clock frequency constraint is not violated high radix switches can be built further reducing network latency.

Guideline 3: Use cautiously the storage resources for buffering without negatively affecting network throughput.

Guideline 4: Allow any form of flexibility in the implementation that would not stress too much the mapping, placement, and routing of the corresponding FPGA-CAD tools.

In fact, recently, new optimized switch implementations have appeared in open literature [5, 9] that try to satisfy most of the above guidelines.

2. THE PROPOSED APPROACH

In this paper we go one step beyond and redefine the switch. We propose a fine-grained decomposition of the switch that allows us to match better its components to the FPGA resources. The basic element of our proposal is a small switching module, named AC, that handles independently link traversal, arbitration and multiplexing. Packets are buffered, and routed at the same time they are crossing the links of the network. The AC module suits perfectly to the logic blocks of the FPGA, thus optimizing the use of FPGA resources. Fig. 3 depicts graphically the main target of our proposal. The switch is decomposed to an equal number of AC basic blocks. Each block utilizes only the resources of the logic array (LUTs and registers) leaving any of the scarse RAM resource free to other modules of the system.

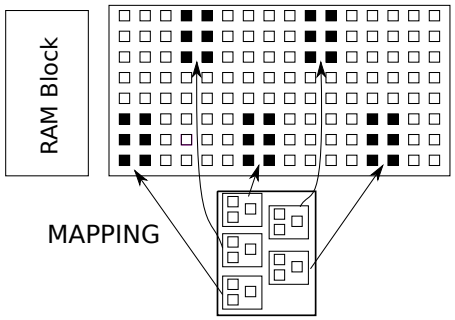


Fig. 3. The mapping of AC-based switches on an FPGA.

The AC modules operate as global elastic pipeline [10] and they are physically spread throughout the chip irrespective the network topology and without putting any constraint on the placement tools that accompanies the FPGA. In this way, both the area and the operating frequency of the NoC is optimized, allowing the AC modules to achieve a plethora of optimized configurations on an FPGA environment.

Evaluation results demonstrate that the proposed *Distributed Elastic Switch Architecture* (DESA) reduces the area

of the switch, while still offering high-speed implementations. The implementation-related savings achieved by DESA do not compromise network-level performance. Simulation results with synthetic traffic patterns prove that DESA can achieve better or at least the same throughput as the baseline design, under equal buffering resources, while still offering more than 50% better clock frequency and less implementation area. Under minimum buffering for both switches under comparison DESA improves both latency and throughput.

3. DESA: STEP-BY-STEP CONSTRUCTION

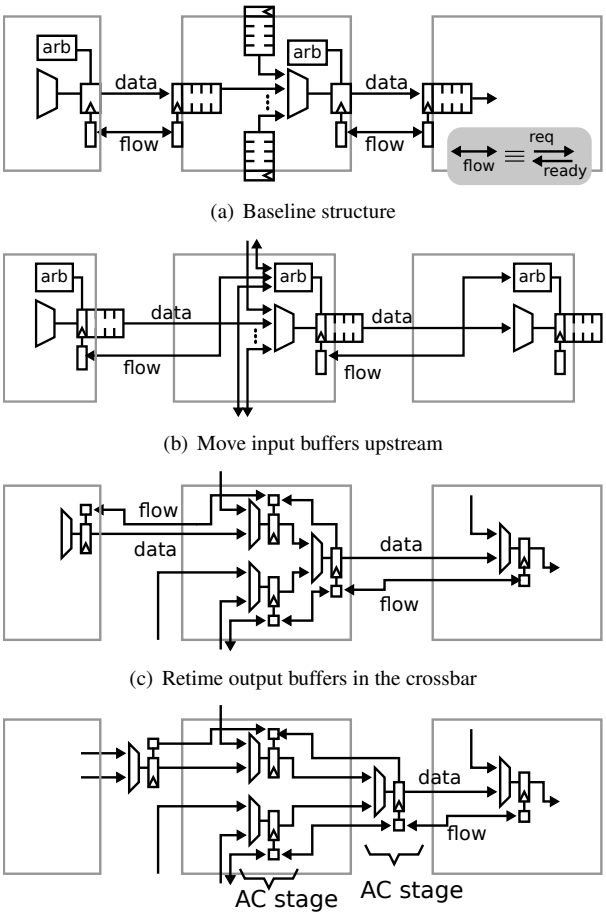
The proposed switch is an elastic pipelined buffered wormhole switch, which is constructed as a collection of independent switching modules called AC modules. Each AC module is able to store, arbitrate and forward its incoming flits. In the following, we will describe in detail the design of the distributed switch in a step-by-step manner beginning from the baseline wormhole architecture.

Fig. 4(a) depicts the organization of each output of a WH switch following the generic organization shown in Fig. 1. Each output consists of a multiplexer and a register. The data inputs of the multiplexer are connected to the input buffers of the switch and the select signals are driven by the arbiter that resolves contention in the case of many competing requests. The output link connects directly to the input buffer of the next node.

Data transfer on the link is guided by flow control information exchanged by the two switches. When the receiver can accept a new flit it asserts the *ready* signal. When the transmitter has a new flit available it asserts the request (*req*) signal. Data is transmitted when they both see *req* and *ready* signals as true. In this case, they independently know the transfer has occurred, without negotiation or acknowledgement.

If we move the input buffers of each node close to the output port of the upstream switch, the basic switch organization is transformed to the one shown in Fig. 4(b). The register at each output can be merged with the rest of the buffers and appear as an unified output buffer. Although the buffers are drawn at the output of each switch, the switch itself remains an input-buffered switch, since only one flit can arrive at each output on each cycle. Now, the flow control signal produced at the output of each switch needs to travel all the way back to the outputs of the corresponding upstream nodes. The flow control information of the output buffer is combined with the local arbitration decisions of the current switch and it is forked to all upstream switches. In this way, any upstream node receives its own ready signal and knows whether the output buffer is full or in the opposite case if it was granted access to the output.

This organization does not add any benefit relative to the classical input-buffered organization. The benefits of this



(a) Baseline structure

(b) Move input buffers upstream

(c) Retime output buffers in the crossbar

(d) Balance wire lenght per stage

Fig. 4. The step-by-step construction of DESA.

approach start to show up when we retime the buffer slots of the output buffer along with the flow control information inside the crossbar multiplexer[1]. In this case, shown in Figure 4(c), the centralized buffer per output has been transformed to an elastic pipeline. Each stage of the pipeline involves arbitration, multiplexing and the corresponding flow control. Actually, the input buffers of the baseline organization shown in Fig. 4(a) have disappeared and all the buffering of the switch is performed inside the per-output multiplexers.

By distributing the buffers in the crossbar, we completely remove the need for a centralized input-buffer and rely only on the registers available in each logic block. This feature simplifies significantly the design of the switches on the FPGA, since it leaves the scarce memory resources of the FPGA, either in the form of LUT-based distributed RAM or in the form of a larger centralized RAM macro, available to other modules of the system.

[1]We only use as any many buffers as the nodes of the multiplexer's tree and the rest can be discarded.

However, even after the retiming of the buffers, the elastic pipeline is not properly balanced since the first stage always involves a long link. Due to the elastic nature of the distributed switch, this limitation can be immediately removed by shifting the output stage of the switch to the middle between the two nodes (see Fig. 4(d)). The link length is reduced by half and actually buffering and multiplexing are performed along the link. Now, each stage of the elastic pipeline starts with an equal-length link and ends with a 2-to-1 multiplexer.

The complete design of an AC module is shown in Fig. 5. The arbiter receives the local requests from the incoming data flits and using a round-robin policy selects the one that will be transferred to the output of the AC module. When the next AC module is not available the arbiter does not grant any inputs. So the arbiters decisions are first masked with incoming flow control and then are distributed (forked) as ready signals to all connected upstream nodes.

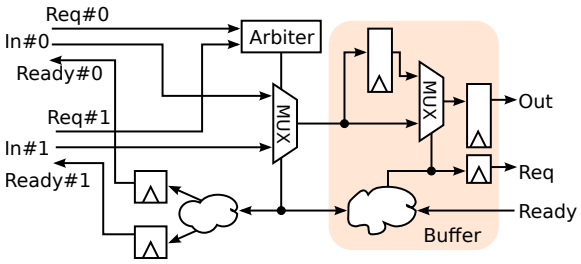
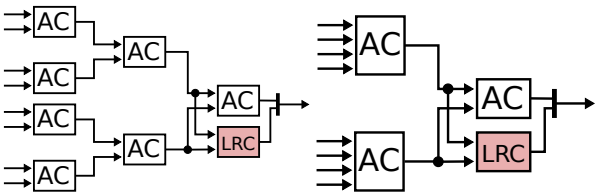


Fig. 5. The implementation of the AC module.

The elastic buffer at the output of the AC module replaces a simple edge-triggered register [11]. When the output of an elastic register chain stalls downstream, the stall can only propagate back one stage per cycle, since the outgoing ready signals are first registered at the current AC node [10]. This approach guarantees the maximum scalability in terms of clock frequency. Now, the output register of the AC module needs to hold two words. One for the stalled output, and one for storing the flit that is in flight until the stall signal propagates to the upstream AC modules. In this way, no data is lost due to a downstream stall. Even if deeper elastic buffers are possible, as we will show in Section 3, our analysis with the 2-slot configuration gives sufficiently good resutls.

At this point, the only missing item to implement a switch is the LRC module. This module is designed in isolation and placed at the last AC module at every output port of the switch. Figure 6 shows the placement of the LRC unit in different implementations of an 8-port switch.

For a radix-2 AC module a single flit spends $\log_2 N$ cycles in each switch due to the pipelined nature of DESA. This latency can be reduced by multiplexing more inputs per stage, i.e., by increasing the radix of the AC modules, as shown in Fig. 6. Without altering the basic design of the distributed switch, the radix of the AC modules can be chosen arbitrarily. A radix-M AC module consists of an M-input arbiter and multiplexer and one elastic buffer at each output. Now the stages to pass an N-input switch equals to $L = \log_M N$ while the total amount of buffering per output is reduced from $2(N-1)$ in the baseline radix-2 case to $2\frac{M^{L+1}-1}{M-1}$ in the radix-M case.



(a) With radix 2 AC modules (b) With radix 4 AC modules

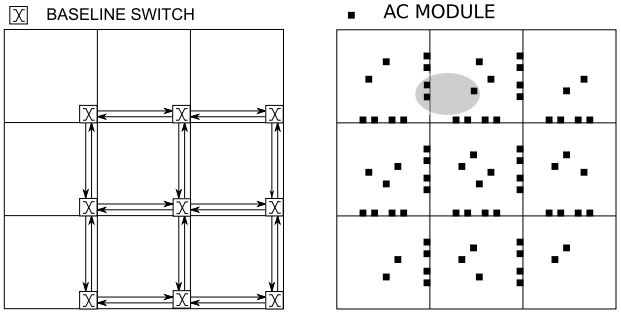
Fig. 6. The implementation of the output port of an 8-input switch using AC modules of different radix.

Although using high-radix AC modules reduces the number of cycles per switch, the maximum clock frequency of the network is negatively affected. High-radix AC modules add more logic per stage, while the wires that connect them are inevitably longer. In this case, the switching points per output (AC modules) are less but more concentrated, since the output links of more nodes are gathered together. In the rare case of extremely high-radix networks, where the link delay is the limiting factor, the links can be split by elastic buffers without altering the functionality of the rest of the network. The *req*/*ready* signals guarantee safe data transfer through the pipeline.

From the above description, it can be easily verified that DESA follows well guidelines 2, 3 and 4 imposed by the structure of modern FPGAs as described in the introduction. Satisfying guideline 4 is of particular importance since it provides the maximum freedom to the CAD tools and at the same time allows the designer to choose the topology that best fits the application domain. For example, a placement of an otherwise structured 2D mesh network using radix-2 AC modules is shown in Figure 7. Regarding guideline 1, it is the easiest to follow. In the next Section we will show that the delay of each AC stage is only slightly affected by the increased the data width.

4. EXPERIMENTAL RESULTS

In this section, we compare the proposed designs against the baseline switch organization shown in Fig 1. For attaining our comparison data, we first generated the equivalent Verilog descriptions of all designs under comparison. After extensive simulations that verified the correctness of each description, each design was synthesized and mapped to a Virtex 5 FPGA chip. For the synthesis, mapping, and placement & routing of the designs we used the ISE 12.2 toolset



(a) Floorplan with standard switches.

(b) Floorplan with AC-based switches.

Fig. 7. The unconstrained placement of an AC-based 2D mesh network.

of Xilinx. Please note that the reported results involve only the optimizations performed by the CAD tools alone, without any manual intervention that would further optimize the circuits under comparison. In this way, the presented results can be re-produced by the general designer by just following the same automated design flow.

At first, we compared the implemenation of the baseline switch and that of DESA in terms of area and delay after varying the width of the datapath and the number of buffers in each case. The switches correspond to a 2D mesh, with 5 inputs and 5 output ports that implement dimension-ordered routing. The latency of the baseline switch is set to a single cycle. This configuration offers the least implementation area for the baseline switch and it allows us to measure the delay complexity of the operations involved in the operation of the switch. Also, the input buffers are always mapped to distributed RAMs for better area efficiency after appropriately constraining the synthesis/mapping tool. For the case of AC-based switches, we used radix-2 AC modules. The obtained results are shown in Table 1.

Flit=16bits	Baseline		DESA	
	Area	Delay	Area	Delay
2 buf/input	321	6.7	272	2.7
4 buf/input	343	6.9		

Flit=32bits	Baseline		DESA	
	Area	Delay	Area	Delay
2 buf/input	406	8.5	427	2.9
4 buf/input	475	8.8		

Table 1. The area in slices and the delay in nanoseconds of the 5×5 switches under comparison.

For small datapaths of 16bits, the proposed design requires the least amount of area, while in the case of 32bits it is in the middle between the area of the baseline design with 2 and 4 buffers per input, respectively. Due to the small amout of logic in each elastic pipeline stage, the maximum clock frequency of the AC-based switch is more than 50% larger than the frequency of the baseline switch. The baseline switch needs to be pipelined in 2 or 3 stages in order to achieve the speed of the AC-based design, increasing even more its already aggravated area. Please notice that if we pipeline the baseline switch by 3 or more stages, then this increases the round-trip time between two switches and thus the minimum amount of buffering required to achieve full throughput. These extra buffers will severely affect the area of the baseline switch. In our case, due to the elastic nature of the switch and the need of only one cycle stall propagation, round-trip time remains the same irrespective the radix of the switch and the radix of the AC modules. In addition, the speed of DESA is only slightly affected by the increased flit size allowing fast operation even under wide datapaths.

As shown in Fig. 4(d), the critical path of an AC-based switch starts from the output buffer of an AC module and ends at the input of another AC module after crossing a link of arbitrary length. In order to verify how the delay of AC-based switches scales after increasing the link length, we performed an additional set of experiments. We implemented the same AC-based switch but we constrained the floorplanner to place the last (closer to the output) AC modules far away from the rest, i.e., the ones closer to the inputs. The delay measurements we obtained for a 16 and 32bit datapath after varying the distance between the AC modules is shown in Fig. 8. The delay increases almost linearly to the link's length and even when the AC modules are placed 70 logic slices apart their delay is still significantly less than the delay of a single-cycle baseline switch.

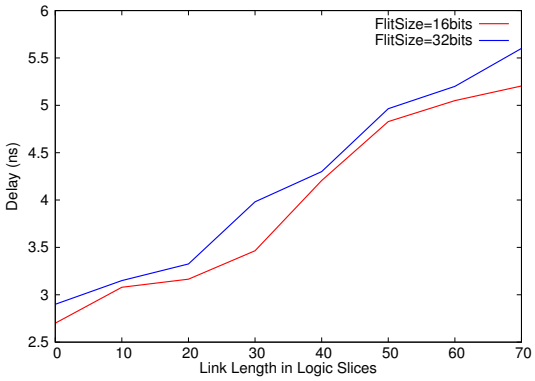
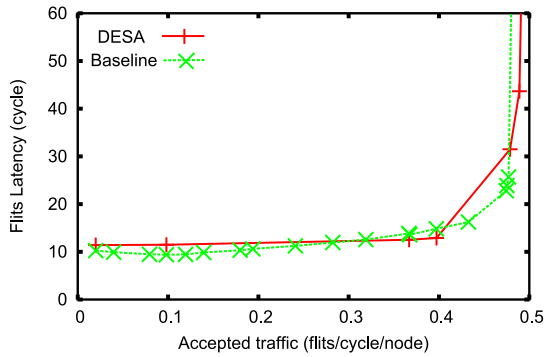
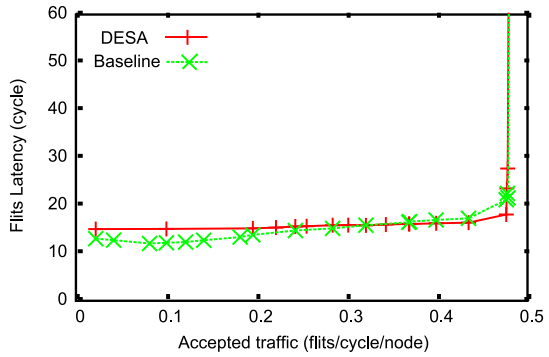


Fig. 8. The delay of AC-based switches after increasing the distance between the AC modules.

Next, we evaluate the proposed, AC-based switch microarchitecture, against the baseline switch organization using a 16 node, 4×4 2D mesh network. We evaluate the proposed architecture using a cycle-accurate interconnection network simulator. To evaluate the latency-throughput, the simulator is warmed up under load without taking mea-



(a) Uniform-random traffic



(b) Bit-complement traffic

Fig. 9. Network throughput vs average message latency for 4×4 mesh network.

surements until steady-state is reached. Synthetic traffic pattern results from uniform random and bit-complement traffic are presented when injecting 5-flit packets.

The latency(in cycles)-throughput of the single-cycle baseline switch and the proposed DESA is shown in Fig. 9, assuming for both switches an equal number of total buffering resources. As we can observe, latency and throughput are almost similar for both switches. The baseline switch has marginally lower latency (measured in cycles) than the AC-based switch at low loads. This is due to the 2 cycle latency imposed by the AC-based elastic design. However, please keep in mind that the clock frequency of the AC-based design is more than 50% higher compared to the baseline switch, and, hence, the absolute-time latency of DESA is always the smallest. Finally, since both design saturate at the same load and DESA can run at much higher clock frequencies than the baseline switch, even at wide datapaths, we can clearly identify the significant throughput benefit of the proposed architecture.

5. CONCLUSIONS

In this paper we have proposed a novel switch design, tailored for FPGAs implementation. By careful design of every block, mimicking the resources available in basic FPGA cells, we are able to optimize the automatic mapping of networks on FPGAs, avoiding the use of scarce memory blocks and reducing the effects of long links between network components. The basic AC module, when used to implement a NoC in an FPGA, is able to reduce significantly the per-switch latency while keeping network throughput un-affected. Area savings are also noticeable. If the baseline switch wants to compete in speed with the AC-based switch design, extra registers and buffering resources will be needed, that will significantly increase the area needs.

As future work we plan to deeply analyze the effect of different radix AC designs and how they affect frequency and performance of the switches. In this context the adoption of the merged arbiter multiplexer approach [12] is expected to increase significantly the area/delay savings.

6. REFERENCES

[1] G. Brebner and D. Levi, "Networking on chip with platform FPGAs," in *Proc. of the IEEE International Conference on Field-Programmable Technology*, Dec. 2003, pp. 13–20.

[2] Altera, "Applying the benefits of Network on a Chip architecture to FPGA system design," Tech. Rep., January 2011.

[3] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proceedings of the 38th annual Design Automation Conference*, ser. DAC '01, 2001, pp. 684–689.

[4] N. Kapre, N. Mehta, M. deLorimier, R. Rubin, H. Barnor, M. J. Wilson, M. Wrighton, and A. DeHon, "Packet-switched vs. time-multiplexed FPGA overlay networks," in *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines*, April 2006, pp. 205–216.

[5] M. Papamichael and J. C. Hoe, "CONNECT: Re-examining conventional wisdom for designing NoCs in the context of FPGAs," in *Proceedings of the ACM/SIGDA Intern. Symp. on Field Programmable Gate Arrays (FPGA)*, 2012, pp. 37–46.

[6] M. Saldaña, L. Shannon, J. S. Yue, S. Bian, J. Craig, and P. Chow, "Routability of network topologies in fpgas," *IEEE Trans. VLSI Syst.*, vol. 15, no. 8, pp. 948–951, 2007.

[7] J. Lee and L. Shannon, "Predicting the performance of application-specific nocs implemented on fpgas," in *Proceedings of the ACM/SIGDA Intern. Symp. on Field Programmable Gate Arrays (FPGA)*, 2010, pp. 23–32.

[8] C. Hilton and B. Nelson, "Pnoc: a flexible circuit-switched noc for fpga-based systems," *IEE Proceedings Computers and Digital Techniques*, vol. 153, no. 3, pp. 181 – 188, may 2006.

[9] Y. Lu, , J. V. McCanny, and S. Sezer, "Generic low-latency NoC router architecture for FPGA computing systems," in *Proceedings of the Intern. Conf. on Field Programmable Logic and Applications (FPL)*, 2011, pp. 82–89.

[10] H. Jacobson, P. Kudva, P. Bose, P. Cook, S. Schuster, E. Mercer, and C. Myers, "Synchronous interlocked pipelines," in *In Proceedings of the Intern. Conf. on Asynchronous Circuits and Systems (ASYNC)*, april 2002, pp. 3 – 12.

[11] J. Cortadella, M. Kishinevsky, and B. Grundmann, "Synthesis of synchronous elastic architectures," in *Proc. ACM/IEEE Design Automation Conference*, July 2006, pp. 657–662.

[12] G. Dimitrakopoulos, C. Kachris, and E. Kalligeros, "Scalable arbiters and multiplexers for On-FGPA interconnection networks," in *Proceedings of the Intern. Conf. on Field Programmable Logic and Applications (FPL)*, 2011, pp. 90–96.