Design and Implementation of a Packet Switched Dynamic Buffer Resize Router on FPGA

Vivek Raj .K[1] Prasad Kumar[2] Shashi Raj .K[3]
[2, 3]Assistant Professor
[1, 2, 3]Electronic and Communication Engineering Department
[1, 2]SJCIT, Chickballapur-562101, Karnataka, India. [3]DSCE, Bangalore-560078, Karnataka, India.

Abstract---NoC designs are based on a compromise of latency, power dissipation or energy, usually defined at design time. However, setting all parameters at design time can cause either excessive power dissipation (originated by router underutilization), or a higher latency. The situation worsens whenever the application changes its communication pattern. In this paper we propose an FPGA based, single cycle, low latency router design that can be reconfigured to 1-D and 2-D network on chip architectures with dynamic buffer resize technique. The results show that the technique improves the area and the performance of the architecture by greatly reducing power consumption and latency between the routers.

Keywords: Network on chip, wormhole router, flow control, interconnect, FSM, FPGA, LUT, GALS, Adaptive buffers, FIFO, Packets.

I. INTRODUCTION

Moore's Law continues to scale down device feature size and increase system performance. As a result, we observe a trend of increasingly complicated architectures, enhanced clock rates and on chip logic density. One emerging architecture is Network-On-Chip (NoC). This new design paradigm is quickly replacing older design strategies from the bus architectures of microelectronic chips. NoCs have been instrumental in achieving high bandwidth; low latency and scalable multi core inter communication architectures in order to keep up with Moore's law [1].

However, NoCs still face some limitations. First of all, in NoC architectures, several cores share the workload of a task. Data transfer from one core to another is very frequent and therefore, much emphasis needs to be placed on this area. Power consumption limits how many cores can be placed on a single chip and be utilized efficiently at the same time. Thus, reducing energy consumed per logic operation is becoming more and more important to keep power dissipation within limit. Secondly, since all cores are connected via an interconnect fabric, be it bus, ring or mesh, the interconnect topology plays a major role in the performance of the network. Henceforth, it is clear that router energy consumption and interconnect fabric topology are the two leading factors that limit the performance of NoCs.

NoCs provide much higher bandwidth than buses but have higher area and delay. Routers need buffers, routing tables, a switching circuit and arbiters. So, they occupy more area than a bus based network. Also, direct bus connections are faster than pipelined connections through one or more routers since these introduce a delay due to packing, routing, switching and buffering. Since Several researches has been made in the field of router design , In this paper, we analyze the proposed reconfigurable router for NOCs applications and some proposed dynamic buffer resize techniques when used to design NoCs for FPGA and then propose and analyze a more efficient buffer resize technique for reconfigurable router. In section II, the related work is described. In section III, we analyze the influence of buffer resize over routers implemented in FPGA and propose a dynamic buffer resize technique for FPGA. Section IV shows the results and finally, section V concludes the paper.

II. RELATED WORK

Adaptability in NoCs was proposed in several works considering several aspects of the NoC, including topology, routing, switching, buffering and crossbar. A few dynamically reconfigurable topologies for NoCs have been proposed recently [2] - [3].

The paper [2] provides ReNoC architecture that enables the network topology to be reconfigured using energy-efficient topology switches. The architecture was evaluated by mapping an application to a static 2D mesh topology as well as ReNoC architecture in two different topology configurations. The power consumption was decreased by 56% when configuring an application specific topology, compared to the static 2D mesh topology. The topology switches increased the area of the NoC architecture with 10%, and only contributed with 5% of the power consumption in the application-specific topology. The evaluation shows that the ReNoC architecture enables application-specific topologies to be configured with little overhead and indicates that the architecture has great potential for future SoC platforms.

In paper [3] and [7] presents, the architecture's main attractions of packet-switched NoC systems, while addressing the problem of hop-by-hop propagation latency. Each pipeline stage is optimized as such that the zero-load packet propagation latency of the proposed NoC is only two cycles per hop including the router pipeline and link traversal. This we believe represents a significant enhancement over state of the art FPGA designs. Key contributions include (a) the definition of a highly scalable router architecture capable of supporting various network topologies on FPGA (1D, 2D and 3D) (b) the architectural optimization of the router such that two cycles per hop can be achieved, (c) a detailed analysis of the proposed architecture in terms of scalability, hardware cost (area), operation speed (critical path) and power dissipation and (d) demonstrating the feasibility of the proposed router in an real-world on-board design environment.

Adaptive buffer resize have also been proposed for NoC adaptability. In [4], the architecture uses a runtime

agent to observe the presence of faults and remap tasks, reroute packets or reallocate buffers. In [5] a router can resize a buffer at run-time by moving slots buffers from one port to another. Buffer adaptation is based on a flow sensor that watches the traffic at run time, then a control equation adapts itself to change the buffer size of each port to achieve better performance. In [6], the authors use a shared buffer scheme to improve the input-buffering technique. The solution statically groups pairs of inputs to each of the two buffers. In [8], a centralized buffer structure is proposed, which dynamically allocates buffer resources using linked lists.

III. FPGA BASED ROUTER DESIGN

Figure 1 shows the packet structure used for the proposed router. The packet length for this design is extremely flexible and each packet transferred from one router to another is divided into a header flit, a series of body flits concluded by a tail flit. The header flit contains all the information a router needs to configure its crossbar in order to send the flit to the desired downstream router and set up the channel for following body flits. When a tail flit passes through a router, it indicates the end of the packet and de allocates all resources for the current router. Figure 2 illustrates the architectural details of the router. The sub blocks of the design are discussed as follows:

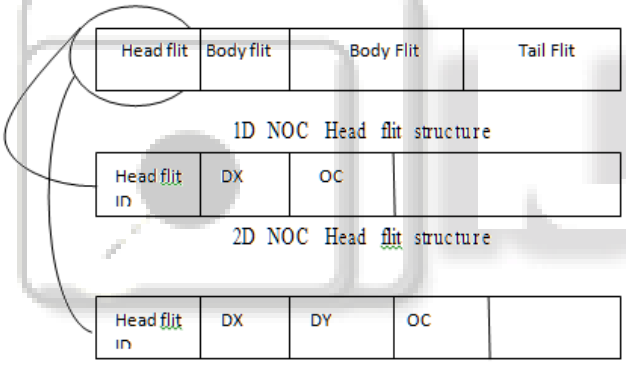

Fig .1: Packet structure

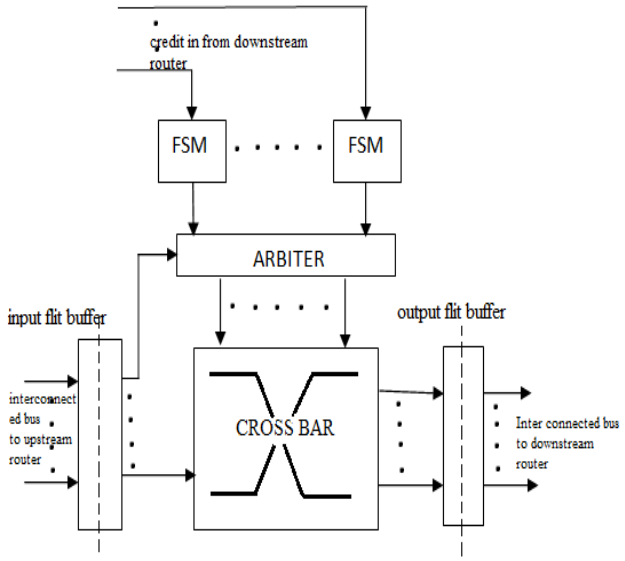

Fig. 2: The detailed architecture of the router

A. Arbiter

This block works in conjunction with a round robin priority encoder and the credit based flow control FSM. Due to this design choice, a very efficient way of congestion control and avoiding resource starvation was deployed. The arbiter has four allocators, one for each input-output port pair. The arbiter may receive resource allocation request from all four input ports simultaneously, and hence the priority logic is essentially processing multiple requests and once an output channel is matched to an input port, the selection logic holds the allocator active till a tail flit is processed through the crossbar.

B. Crossbar Switch

The switch is implemented using four 4x1 multiplexers (MUX). The output of each MUX is registered and goes straight to its corresponding output port. The crossbar is controlled by one of the select signals from the arbiter block. Figure 3 illustrates a node to node data traversal example. In this scenario, data packets 1 & 2 arrive at N4 from N1 and N3 respectively and need to be routed to N5. In this example, N4 determines that packet 1 gets priority and hence allocates the corresponding port resources to its traversal. Once packet 1 is routed, packet 2 gets the resource allocation. A flit traverses only after a credit is received back from the downstream node. Since the router immediately registers inbound and outbound data, the timing restriction of routers are self-contained. As a result, the router is compatible with globally asynchronous and locally synchronous (GALS) NoC topology as well.

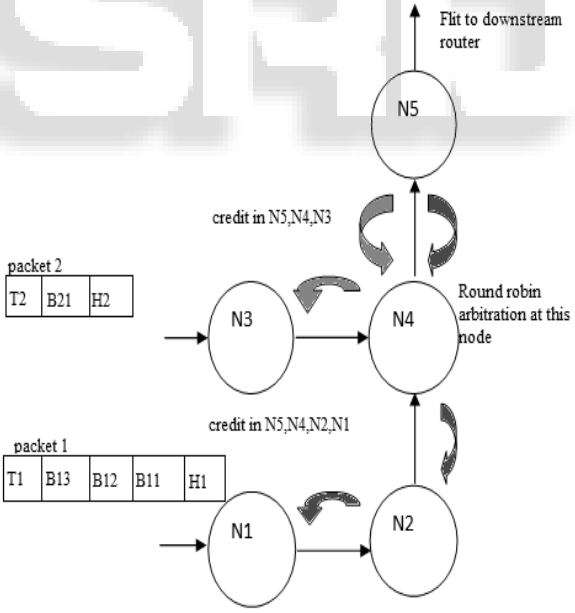

Fig. 3: Node to node data transmission

C. Input & output ports

The input port has look-ahead logic to determine the forwarding path of data for the downstream router. Upon receiving a header flit, the logic quickly checks if the destination address matches the current router's address, if so then the data is sent to the core port and it never requests an output port from the arbiter. Otherwise, dimensional order routing is used to compute the forwarding path. This logic computes the new output channel (oc) value and

replaces the current oc value in the header flit for the downstream router. The incoming header flit has the 2 bit encoded oc information which is the enumerated specific output channel to be used by the current router in flit transfer. The Input and output ports of router contains a buffers which uses technique known as dynamic buffer resize technique. Buffer resizing can be used to improve latency or minimize the area of a router.

1) Buffer Resize Technique For FPGA

Buffers with depth 4 for 8-bit words implemented with registers requires about 40 LUTs. So, unless we use very small buffers (with a depth lower than 4), it is better to implement FIFOs with SRL primitives. In our proposal, a buffer unit will consist of FIFOs of depth 16, 32 or 64, whose implementation uses 39, 47 or 64 LUTs for 8-bit data implemented with LUTs configured as SRL (see table I).

Other buffer size configurations can be also easily implemented. For the dynamic distribution of buffering resources, we propose the use of floating buffers that can be assigned to any output port to increase their buffering size. With extra buffers it is possible to reduce the size of the fixed buffers to a minimum (e.g., 16 words) and then dynamically compensate for the lack of buffering by assigning the floating buffers to the output ports most congested. The router will have a structure similar to the static router; except that the adaptive router includes a few extra modules to control the floating buffers and to dynamically put them in the data path of the router (see figure4).

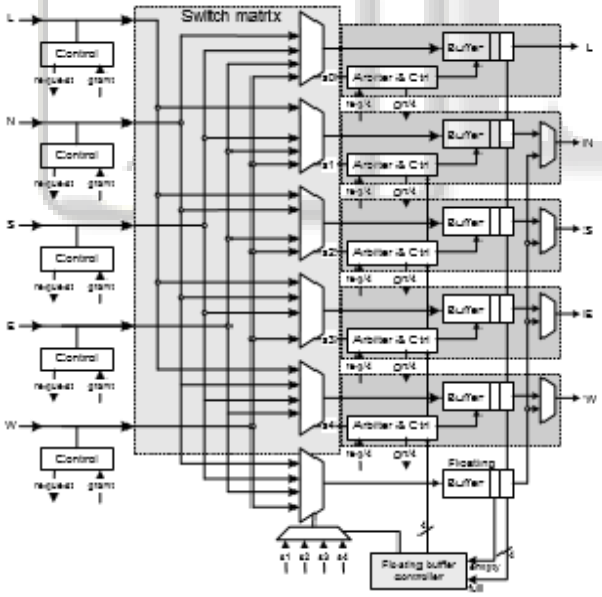

Fig. 4: architecture of a packet switched router with buffer resize.

The architecture show ninth figure has a single floating buffer that can be associated with any output port, except with the local port. This port has not been considered since we assume the processing element connected to the local port is unable to collect data simultaneously from two inputs. The arbiter associated with an output port receives the requests from the input ports and grants access to its buffer. Case the static buffer is full and the floating buffer is assigned to it then it grants access to the floating FIFO. If the floating FIFO gets full then it returns to the static FIFO the assignment of the floating buffer is made by the floating buffer controller. Several policies can be followed to assign the floating buffer. In this work, the floating buffer can be reassigned if it is empty and the controller assigns it to a port having a full fixed buffer for at least five cycles. For a fair assignment, all eligible ports for assignment (those with full static buffers) are chosen in a round-robin manner. More aggressive policies could be followed. For example, a port with both static and floating buffers could allow simultaneous writes on both buffers. This would allow two input ports to forward their fits simultaneously to the same output port. Also, instead of forwarding the fits alternately from each buffer, it could forward any buffer based on the arrival order and the congestion of the next router. This would potentially improve the performance of the network but at the cost of more control logic. This policy was not considered in this work.

2) ANALYSIS OF ADAPTIVE BUFFER SIZES

For heavy loaded networks increasing the buffer size will decrease blocking of packets since buffers can be emptied faster because the following buffers in the path have higher probability of not being full.

To illustrate this behavior, we tested a 6×6 NoC with uniform traffic and XY routing using buffers with different sizes moving packets with 32 fits (see figure5). The average latency grows faster with increasing injection rate for NoCs with smaller buffers .For example, with 35% network loading and buffers with depth of 16 the average latency is about 128 cycles. Increasing the buffers to 32 words will improve the latency by about 33 % and increasing it to 64 words will improve the latency by about 40 %. The main disadvantage of this approach is the area overhead associated with the buffers. Therefore, in the customization process of the router, the size of the FIFOs must be carefully chosen to avoid using buffers deeper than what is needed to achieve the system requirements while optimizing area utilization. A simple experiment can be followed to show this tradeoff. Using a NoC with the same configuration, we injected a burst of 50 packets from each processing element.

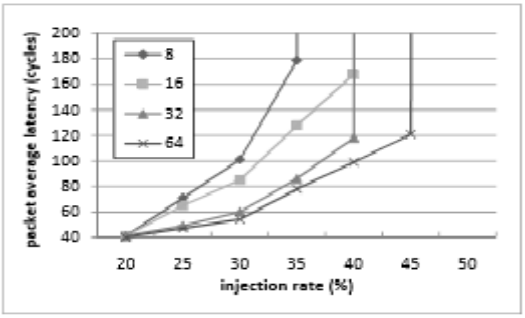

Fig. 5:Average latency for different buffer sizes and injection rate for uniform traffic.

(with an injection rate of 35%), considering 4-hotspot traffic and all buffers with a depth of 16. Then, the size of each buffer with an average utilization higher than a given threshold value was doubled. For each configuration, the area and average latency were determined through simulation. The results are as expected (see figure 6). As can be observed from the figure, the latency is reduced from 185 cycles down to133 cycles (almost30%improvement) by doubling the size of the buffers. The latency decreases rapidly when the depth of most utilized buffers are doubled.

For example, if buffers with an utilization higher than 40% are increased, the latency is reduced about 20%. The techniques already proposed to dynamically resize buffers achieve performance and /or power improvements at the cost of some area overhead. However, most of them target ASIC technology where area utilized by the buffers is proportional to their size. This is not the case with FPGA technology, where buffers as FIFOs can be efficiently implemented using the SRL primitives of LUTs (see table I) the areas of the FIFOs are practically the same for a set of sizes, as shown in table I. This is because each 4-input LUT implements a 16-bits shift register. So, the efficiency of the proposed techniques for buffer resizing has to be re analyzed considering these figures.

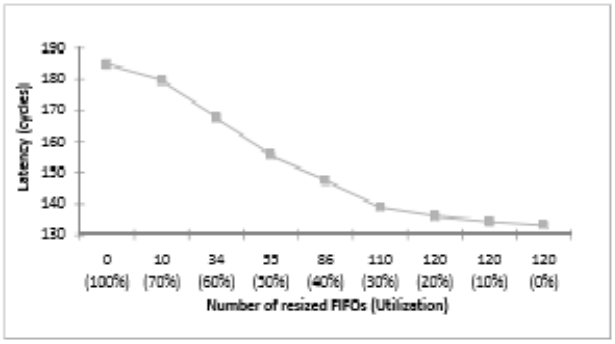

Fig. 6: Average latency for different buffer sizes and injection rate for uniform traffic.

Depth	8bit	16bit	32bit
<=16	[0,39]	[0,65]	[0,85]
[16,32]	[44,47]	[84,87]	[130,133]
[32,64]	[59,64]	[130,135]	[224,229]

Table. 1: Logic area (virtex-4 LUTs) utilized by the buffers implemented as FIFOs

IV. RESULTS

This section discusses the results obtained by running several tests mentioned in the previous section. A set of experiments were conducted to estimate the area, utilization and maximum clock rates for each network topology. The data was obtained post synthesis of the design and also providing the user defined constraints for clock, reset and other external user controlled inputs. Table II shows the achievable clock rates of the three NoC architectures under consideration. Also shown in the table are the setup and hold margin for the designs. Since the Virtex 6 is on a 28nm process, it can be argued that these margins are fairly healthy as the underlying gates should have fairly small latency and therefore there should be no timing violations during active switching of the design.

Table 3. tabulates the FPGA target device utilization of all three network topologies. In order to do so, the three a fore mentioned network architectures were synthesized and emulated on the target device. In order for the Xilinx synthesis engine to not remove the underlying blocks as a part of its optimization process, a set of outputs were brought out for each router. Since the target device is pin limited and the design may have an unusually high number of output ports, the outputs were tailored meet the maximum package pin out and synthesis would be successful whilst not impact its accuracy. As shown in table III, as the design grows in the node count, the utilization grows as well. The last column of the table lists the fully utilized LUT-FF pair and the small design of the 1-D ring was better packed in the FPGA clusters when compared to the larger designs.

Design	Timing		
	Setup margin (nsec)	hold margin (nsec)	max clock rate (MHz)
1D 8-node ring	2.976	0.659	323.9
4x4 mesh	3	0.659	330
8x8 mesh	2.87	0.659	325.44

Table. 2 Static timing analysis of noc topologies

size	design	FPGA utilization		
		LUTs(%)	Registers	LUT-FF(%)
Small	1D 8-node ring	4	1	40
Medium	4x4 mesh	11	3	38
Large	8x8 mesh	48	13	38

Table. 3. FPGA utilization of NOC topologies

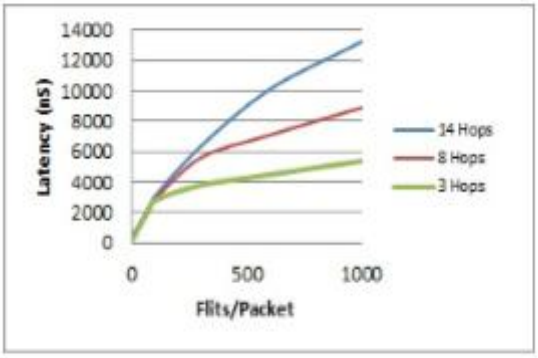

Fig. 7: Results from latency vs. packet size

Figure 7 demonstrates the latency associated with data transactions based upon hop count. As expected, latency increases from a 3 hop transaction to 8 hop transaction to a 14 hop transaction. Figure 8 provides the pie chart distribution between the three major categories of the 8x8 mesh network on chip; these are the clock buffers, router logic and signal (routing). The most power hungry component of the design is the clock buffers. This can be attributed to the fact that a global clock is used to synchronize the network. As a result the buffers have become quite large in size and quantity to make sure the clock skew and transition is within desired limits. At some point though, with increased network size and traffic, it would be expected the signal power consumption surpassed the clock buffer power consumption.

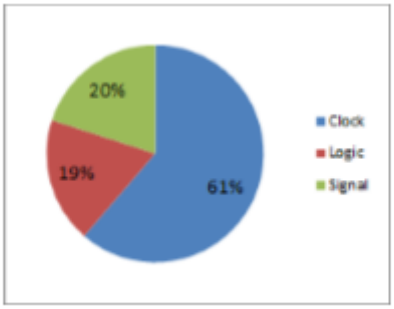

Fig. 8: Power distribution within router

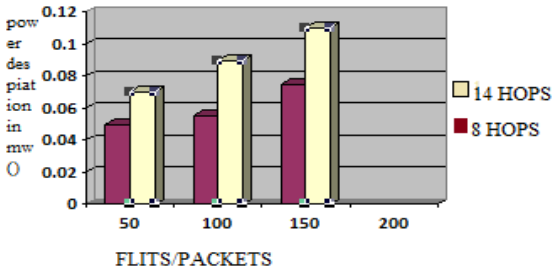

Fig. 9: Power dissipation based upon hop count

Figure 9 depicts the dynamic power dissipation between the two different hop traversal tests. As expected, the 14 hop traversal test consumes more power than the 8 hop traversal test. The data presented in fig. 8 & 9 is measured by conducting experiments post place and route of the design hence these results factor in the power consumption by the FPGA routing channels, switch boxes and connection boxes. Therefore, from the plot below, as we know the worst case traversal is 14 hops and a 200 flit packet where each flit is 18 bits wide, we can compute the power per bit transfer per hop to be:

Power consumption per bit per hop = Total power consumed/(hop count)/(packet size x number of packets) = 0.105/14/200*18 = 2.08 µW

To summarize, the details of the design of a packet switched dynamic buffer resize router where the flow control is a credit based flow control are presented in this paper. Using the router as the underlying architecture, several different size NoC topologies were implemented and validated. As expected, not only the utilization of the FPGA increases with the size of the router, the dynamic power dissipation and latency increases with the distance the data has to travel or the packet size.

V. Conclusion

The paper addressed the design of packet switched routers on FPGA using buffer resize. From the experiments presented, we conclude that packet switched routers are promising alternatives to static routers and must be considered in the design process. An analysis of the FPGA utilization of the implemented NoC topologies is also presented. The proposed router and NoC architectures will be more cost effective then its counterpart routers and mesh NoCs and also have better performance in terms of data latency, power consumption and area. The power consumed by an 8x8 mesh NoC is 2.08 µW per bit per hop and the maximum achievable clock rate is 325 MHz on the target FPGA device. The single cycle clock latency is a major improvement over the contemporary NoC routers. As a result, we are able to improve the throughput of the network. Another major contribution of this design is its ability to be implemented in various scalable network topologies and its compatibility with the GALS networks.

References

[1] T. Bjerregard and S. Mahadevan, "A survey of research and practices on network-on-chip," ACM Computing Surveys (CSUR), vol. 38, no. 1, pp. 1-51, 2006.

[2] Mikkel B. Stensgaard and Jens Spars, "ReNoC: A Network-on-Chip Architecture with Reconfigurable Topology", SLIP, 2010.

[3] Ye Lu, J. McCanny and S. Sezer, "Generic Low-Latency NoC Router Architecture for FPGA Computing Systems," Field Programmable Logic and Applications (FPL), pp.82-89, 2011.

[4] AlFaruque, M.A., Ebi. T., Henkel J., "ROAdNoC: Runtime Observability for an Adaptive Network on Chip Architecture". In IEEE/ACM International Conference on Computer-Aided Design, ICCAD 2008, 543-548.

[5] Concatto,C.,Matos,D.,Carro,L.,Kastensmidt,F.,Susin,A. , and Kreutz, M., "NoC Power Optimization Using a Reconfig- urable Router". In the IEEE Symposium on VLSI, 2009.

[6] Soteriu,V.,Ramanujam,R.,Lin,B.,and Peh,LS.," AHigh-Throughput Distributed Shared-Buffer NoCRouter"IEEE Computer Architecture Letters, vol.8 ,n1,January-June2009.

[7] W. J. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," in Proceedings of the Design Automation Conference (DAC), 2001.

[8] Wang, L., Zhang, J., Yang, X., and Wen, D., "Router with Centralized Buffer for Network-on-Chip". GLSVLSI, 2009.

[9] A. Janarthanan, V. Swaminathan, and K. Tomko, "MoCReS: an Area- Efficient Multi-Clock On-Chip Network for Reconfigurable Systems," IEEE Computer Society Annual Symposium on VLSI, pp. 0–1, 2007.

[10] W. Dally and B. Towles, "Principles and practices of interconnection networks," Morgan Kauffmann, 2004.

