Indian J.Sci.Res. 17(2): 114-117, 2018 ISSN: 0976-2876 (Print)
ISSN: 2250-0138(Online)

DYNAMIC BUFFER RESIZINGTECHNIQUE FOR NETWORKS ON CHIP: FPGA PERSPECTIVE

[1] Gunasekhar Reddy.P
[1] Software Engineer, Huawei Technologies, Bangalore

Abstract-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. Application specific systems can benefit from heterogeneous NoCs providing high bandwidth in a localized fashion where it is needed to eliminate bottlenecks and sized communication resources to reduce area utilization. Networks-on-chip have a relative area and delay overhead compared to buses. These can be improved in application specific systems where heterogeneous communication infrastructure provide high bandwidth in a localized fashion and reduce underutilized resources. However, for general purpose architectures, design time techniques are not efficient. One approach for improving area and/or performance of NoCs for general purpose systems is to consider dynamic adaptation of the resources at runtime. In this paper, we analyze the different buffer resize approaches and FPGA implementation.

Keywords - NoC, DSB, QoS, FIFO, VCB.

I. Introduction

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.

Application specific systems can benefit from heterogeneous NoCs providing high bandwidth in a localized fashion where it is needed to eliminate bottlenecks and sized communication resources to reduce area utilization.

For general-purpose computing, the communication architecture cannot be tailored for any specific application. So, in general, designers consider that regular NoC structures are the most adequate for general-purpose computing where processing and data communication are relatively equally distributed among all processing units and traffic characteristics cannot be predicted at design time. However, in general, resources are used differently when executing different applications on the same NoC. So, if some resources are necessary and sufficient for one application, they may be under or over utilized in the execution of another.

The efficiency of the interconnection network can be improved if runtime changes are considered. A system running a set of applications can benefit from the runtime reconfiguration of the topology and of the routers to improve performance, area and power consumption considering a particular data communication pattern.

Customizing the number of ports, the size of the buffers, the switching technique, the routing algorithm and the switch matrix are possible runtime changes that can be considered in an dynamically adaptive NoC.

In this paper, we analyze some proposed dynamic buffer resize techniques and more efficient buffer resizes technique for FPGA. In section II, the related work is described. In section III, we analyze the previous techniques for buffer resize and In section IV we propose a adaptive buffer resize technique for FPGA. 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 [1]. But most of the work on dynamic adaptability of NoCs has concentrated on the routers. Some works improve the performance of routing algorithms using adaptive techniques use a pseudo adaptive XY-routing that depends on traffic conditions. In [5] the best path is determined at runtime considering the distance to the destination and link bandwidth usage. While adaptive algorithms are able to achieve the best redistribution of traffic, they are harder to implement, need virtual channels to avoid deadlocks, by default do not guarantee ordered packet arrival and depends on the ability to keep real- time information about the network utilization, which is a complex task.

[1]Corresponding Author

Generally, these routing algorithms rely only on neighbor information, which restricts its efficacy. Runtime switching was considered in [6] with a hybrid switching mechanism (packet/circuit switching) to guarantee quality of service for real-time applications.

III. Analyses Of Previous Techniques For Buffer Resize

The efficiency of dynamic buffer resize depends on the cost of buffers in terms of occupied resources compared to that of other blocks of the router. This cost relation depends on the target technology, that is, buffers occupy more area relative to the other blocks when implemented in standard-cell technology than when implemented in FPGA. Since in this work we are targeting the FPGA technology, we have analyzed the efficiency of some state-of-the-art dynamic buffer resize techniques for standard- cell technology when implemented on FPGA.

In [3], a number of buffer blocks are dynamically reassigned on-demand. The area overhead of the adaptive router compared to that of the static router is 500 LUTs. With such number of LUTs it is possible to have 12 FIFOs of depth 16, which reduces considerably the efficiency of the solution.

In the buffer resize technique in [4] a channel may borrow some buffer units from its neighbor. The implementation uses extra multiplexers so that any buffer unit can be assigned to a neighbor. The tests show that the router with adaptive buffers of size 4 achieves the same performance of a static router with buffers of size 9, with a decrease of 6% in the area. If implemented in FPGA, the buffers have to be implemented as registers, which are very expensive in terms of resources given that an implementation of buffers with size 4 or 9 are best implemented with SRL16 primitives using the same number of LUTs. Also, the solution is not scalable since the number of extra logic increases more than linearly with the size of the buffers. So, the solution it worthless for FPGA. In [5] the same authors propose the same router that adapts itself to provide appropriate buffer depth for each channel to sustain the performance with minimum power dissipation. This time to achieve the same performance, the adaptive router consumes about 30% less power but uses 55% more resources than the static buffer.

In [7], a centralized buffer structure is proposed, which dynamically allocates buffer resources based on traffic requirements. Each buffer is divided into slots implemented as registers. Slots are than linked by one linked list. This centralized buffer management approach dynamically allocates buffer slots to different packets according to the traffic needs. The tested network uses a central buffer with 60 slots(an average of 12 slot per port). 60 slots with 8-bit each needs 480 bit registers. When implemented in an FPGA, it needs 480 LUTs. This number of LUTs is enough to implement seven 8-bit FIFOs of depth 16 or five 8-bit FIFOs of depth 64, more than the number of slots available for a single port (60 slots).

Therefore, using this approach, we can achieve a higher performance with static buffers when implemented in FPGA.

The problem with these solutions is that they are quite expensive in terms of resources when implemented in an FPGA since they cannot use the SRL primitives. Typically, implementing a FIFO with flip-flops uses 16× more LUTs in a Virtex-4 than when implemented with SRL primitives. Also, some solutions use many multiplexers, and control, increasing even more the cost of the solution in terms of area.

IV. Adaptive Buffer Resize

Buffer resizing can be used to improve latency or minimize the area of a router. 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.

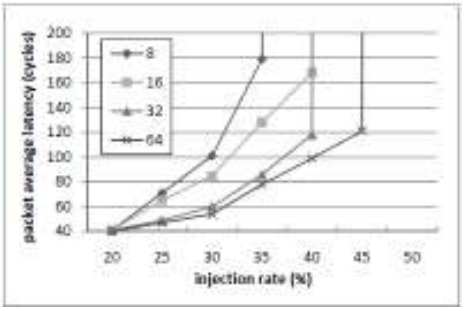

Figure 1. Average latency for different buffer sizes and injection rate for uniform traffic.

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 flits (see figure 1).

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 (with an injection rate of 35%), considering 4-hotspot traffic and all buffers with a depth of 16. In, 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 2).

As can be observed from the figure, the latency is reduced from 185 cycles down to 133 cycles (almost 30% 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%.

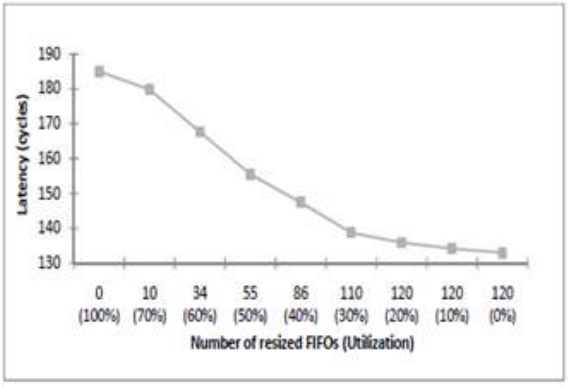

Figure 2. Average latency for different buffer sizes and injection rate for uniform traffic.

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.

The areas of the FIFOs are practically the same for a set of sizes, as shown in table 1. 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 reanalyzed considering these figure

V. Adaptive Buffer Resize Technique For FPGA

A buffer 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. 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 datapath of the router (see figure 3).

The architecture shown in the 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.

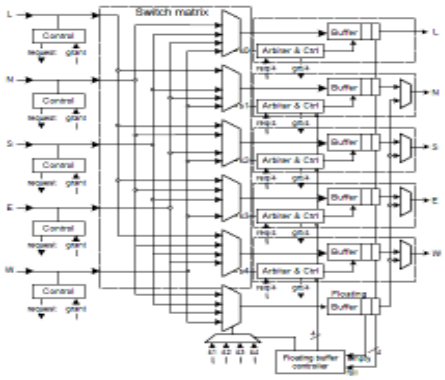

Figure 3. Architecture of an adaptive router with buffer resize.

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 flits simultaneously to the same output port. Also, instead of forwarding the flits 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.

VI. Conclusion

The paper addressed the various buffer resize approaches. From the experiments presented, we conclude that adaptive routers are promising alternatives to static routers and must be considered in the design process. This paper also addressed the design of adaptive routers on FPGA using buffer resize.

References

[1] Stensgaard, M. and Spars, J. ‖ReNoC: A Network-on-Chip Architecture with Reconfigurable Topology‖. In the 2nd ACM/IEEE International Symposium on Networks-on-Chip, 2008, pp. 55-64.

[2] Faruque, M., Ebi, T. and Henkel, J., ‖Run-time Adaptive onchip Communication Scheme‖. In Proceedings of ICCAD, 2007, pp. 26 -31.

[3] Al Faruque, 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.

[4] Concatto, C., Matos, D., Carro, L., Kastensmidt, F., Susin, A., and Kreutz, M., ‖NoC Power Optimization Usinga Reconfigurable Router‖. In the IEEE Symposium on VLSI, 2009.

[5] Matos, D., Concatto, C., Kologeski, A., Carro, L., Kastensmidt, F., Susin, A., and Kreutz, M., ‖ Adaptive router architecture based on traffic behavior observability‖. In Proceedings of the 2nd international Workshop on Network on Chip Architectures, 2009.

[6] Soteriu, V., Ramanujam, R., Lin, B., and Peh, L-S., ‖A High- Throughput Distributed Shared-Buffer NoC Router‖. IEEEComputer Architecture Letters, vol. 8, n 1, January-June 2009.

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

[8] Hu, J., and Marculescu, R., ‖Energy- and Performance- Aware Mapping for Regular NoC Architectures‖. In IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2005, vol. 24, n 4, pp. 551 -562.