DESIGN OF EFFICIENT ROUTING ALGORITHM FOR CONGESTION CONTROL IN NOC

[1]Pawar Ruchira Pradeep
M. E, E&TC Signal Processing, Dr. D Y Patil School of engineering, Ambi, Pune
Email:[1]ruchira4391@gmail.com

Abstract— Congestion is an important issue in networks and significantly affects network performance. The Scheduler acts as the central switch arbiter. The fundamental component in systems which contain shared resources are arbiters and a centralized arbiter is a tightly integrated design for its input requests. In this study, we propose a new centralized arbiter, which may be used in arbitration of a crossbar switch in NoC routers. We design Islip arbiter using Islip scheduling algorithm with mesh router for NoC More integration of system component into single die are allowed increasingly by smaller feature sizes as fabrication technology continues to improve. The limiting factor for performance can be made by communication between these components unless embodying the correct scheduling algorithm.

Keywords— Network-on-Chip, System-on-Chip, On-chip routing switch, Scheduler, Islip, Synthesis

I. INTRODUCTION

In traditional System-on-Chip (SoC) design, shared buses are used for data transfer among various subsystems. As SoC design involves a larger number of subsystems, so that it becomes more complex and traditional bus-based architecture gives rise to new paradigm for on-chip communication. This paradigm is called Network-on-Chip (NoC). The most constraining aspects in the design of embedded system is the complexity, following the rapid technological evolution. The issues of cost and timing add the difficulties in realization of network-on-chip, NoC, applications, where many IPs (Intellectual Property) such as processor cores, memories, DSP processors and peripheral devices are placed together, on a single die. Most often, these modules communicate by means of a shared resource, the on-chip network. The increasing demand for higher bandwidth on the network lines, the increasing complexity of the individual devices and an operating frequency hitting new limits with almost every new design, place the communication and/or computation resources arbitration being the performance bottleneck of the NoC system. To avoid large latencies between the cores on the chip the arbitration is desired to be completed within one clock cycle (e.g., between a processing element (PE) and a memory block). The overall delay introduced by the arbitration should be low so that it will not impact the overall system clock frequency to achieve the arbitration in
one clock cycle, which introduces new challenges for the design of the arbiters. The key research problems in the design of NoC include but are not limited to topology, channel width, buffer size, floor plan, routing, switching, scheduling, and IP mapping [8]. The components of the NoC include the network adapter, the routing node, and the network links [13]. The routing node in turn consists of four major components: the input ports, the scheduler, the crossbar switch, and the output ports. Scheduling algorithms have been developed by a number of researchers [5] [6][7][8], for research prototypes [2][4], and commercial products [3].

What makes a good crossbar scheduling algorithm for the backplane of a router?

We desire algorithms with the following properties:

• High Throughput — An algorithm that keeps the backlog low in the VOQs(virtual output queuing). Ideally, the algorithm will sustain an offered load up to 100% on each input and output.

• Starvation Free — The algorithm should not allow any VOQ to be unserved indefinitely.

• Fast — To achieve the highest bandwidth switch, it is important that the scheduling algorithm does not become the performance bottleneck. The algorithm should therefore select a crossbar configuration as quickly as possible.

• Simple to implement — If the algorithm is to be fast in practice, it must be implemented in special-purpose hardware; preferably within a single chip.

Satisfying the above criterion here we present Islip arbiter using Islip scheduling algorithm with on-chip mesh router. In this study, we explore the adaptation of network techniques and methodology for addressing two particular issues in next-generation buffer less NoC design: congestion management and scalability. Buffer less NoCs have recently gained serious consideration in the architecture community due to chip area and power constraints. While the buffer less NoC has been shown to operate efficiently under moderate workloads and limited network sizes.

II. BACKGROUND

As more complex SoCs begin to emerge, the task of communication between the subsystems of an SoC cannot be adequately handled by bus based communication architectures. NoCs are proposed to be a viable alternative to bus based architectures for communication within SoCs.

A generic 2D mesh NoC architecture is shown in figure 1.

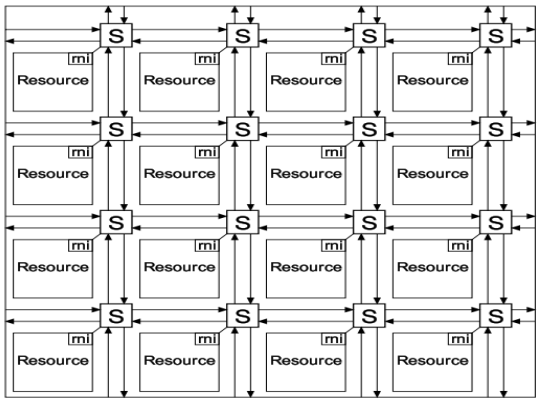

Fig. 1: SoC based on NoC

The SoC consists of a number of resources. These resources communicate with each other using an NoC. The NoC consists of the switches and point-to-point links. The internal structure of the NoC switch is given in figure 2.

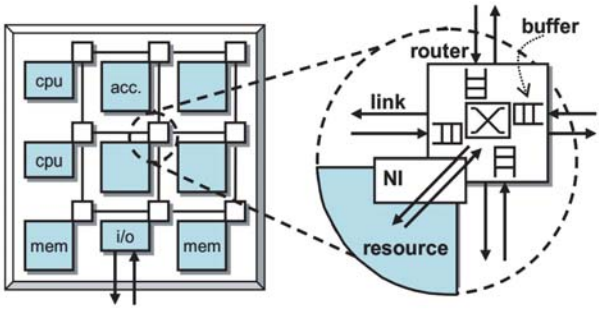

Fig. 2: NoC Switch Schematic

Each switch consists of four distinct components. A set of input blocks are connected to incoming packet lines. These input blocks contain buffers to queue the incoming packets so that they can be stored until they are ready to be transferred over the shared crossbar matrix. A set of output blocks are connected to the outgoing packet lines. No buffers are required in the output blocks as there is no possibility of conflict due to the absence of any shared resources at the outputs. The central crossbar matrix provides a direct link between each pair of input and output blocks. Finally, a scheduler is required to perform arbitration in to enable fair access to the common crossbar fabric for all incoming packets.

A. NoCs in Multi-Core Architectures:

In a chip multiprocessor (CMP) architecture, the NoC generally connects the processor nodes and their private caches with the shared cache banks and memory controllers. A NoC might also carry other control traffic, such as interrupt

requests, but it primarily exists to service cache miss requests. In this architecture, a high-speed router exists at each node, which connects the core to its neighbors by links.

B. Buffer less NoCs and Routing:

Specifically, recent work has shown that it is possible to completely eliminate buffers from the routers of on-chip networks routers. In such buffer less NoCs, application performance degrades minimally for low-to-moderate network intensity workloads, while some work shows that power consumption decreases by 20-40%, router area on die is reduced by 75%, and implementation complexity also decreases [16].

III. A GENERIC ROUND ROBIN ARBITER (RRA)

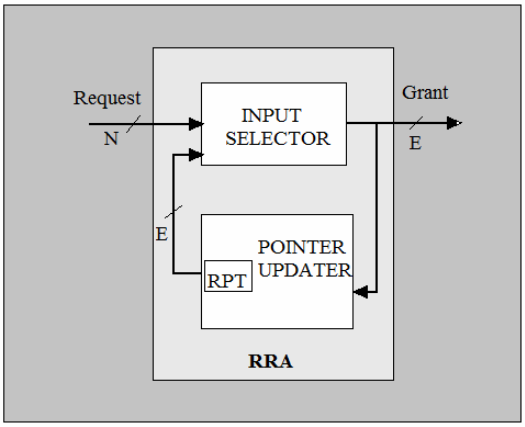

Fig. 1: Generic Round Robin Arbiter architecture

The operation of the RRA is as follows. At the beginning of every clock cycle, each input with a request sets the corresponding bit in the Request to high. Input Selector decides the input that would be granted next, based on the Request and the current value of RPT. If there is a request from the input pointed by RPT (i.e., Request [RPT] = high), this input will be granted by setting the Grant output value to the index of this input, g. If there is no request from the input pointed by RPT, but there are other requests, the Input Selector grants the first input with a request following the pointer in a circular manner (i.e., if there is any request from inputs i > RPT, the smallest indexed input, g > RPT with a request will be granted. If there are no requests from inputs i > RPT, but there are requests from inputs, i < RPT, the smallest indexed input, g 6= RPT with a request, will be granted.). At the end of the clock cycle, the Pointer Updater sets the pointer value to the input next to the granted input in a circular manner (i.e., RPT = (g + 1) mod N). If there is no request from any inputs, the RPT will not change. An optional no request (NoReq) output port can be added to the generic RRA, which will be high when no input has a request.

I. Limitations on the Scalability of RRA:

The most time-consuming operation of the generic RRA is the granting of the requests by the Input selector, which also dominates the critical path delay. The complexity of the Input Selector is due to two main issues: (1) The issue of changing priority: The priority of the inputs changes as inputs are granted and the RPT value changes. This requires the Input Selector circuit to consider all possible priority settings. (2) The issue of circular priority order: The priority order is circular, which makes the priority processing even harder. This leads to two separate conditions for the grant decisions: Grant decisions for requests at or below the Request [RPT] and grant decisions for requests above Request [RPT]. Any grant produced by the Former decision has a higher priority over any grant produced by the latter decision. This two parted decision deepens the critical path. These two issues are even more pronounced as the RRA size gets larger (i.e., an RRA with more inputs). Also RRA has its throughput near about 63%.

IV. THE ISLIP SCHEDULING ALGORITHM

The Islip algorithm is designed to meet our goals. Islip is an iterative algorithm — during each time slot, multiple iterations are performed to select a crossbar configuration, matching inputs to outputs. The Islip algorithm uses rotating priority (“round-robin”) arbitration to schedule each active input and output in turn. The main characteristic of Islip is its simplicity; it is readily implemented in hardware and can operate at high speed. Islip attempts to quickly converge on a conflict-free match in multiple iterations, where each iteration consists of three

steps. All inputs and outputs are initially unmatched and only those inputs and outputs not matched at the end of one iteration are eligible for matching in the next.

The steps for each iteration are as follows:

Step 1.: Request- Each input sends a request to every output for which it has a queued cell.

Step 2: Grant- If an output receives any requests, it chooses the one that appears next in a fixed, round-robin schedule starting from the highest priority element. The output notifies each input whether or not its request was granted.

Step 3: Accept.- If an input receives a grant, it accepts the one that appears next in a fixed, round-robin schedule starting from the highest priority element. The pointer to the highest priority element of the round-robin schedule is incremented (modulo N) to one location beyond the accepted output. The pointer to the highest priority element at the corresponding output is incremented (modulo N) to one location beyond the granted input. The pointers are only updated if and only if the grant is accepted after the first iteration. By considering only unmatched inputs and outputs, each iteration matches inputs and outputs that were not matched during earlier iterations.

V. SCHEDULAR ARCHITECTURE

The function of the scheduler is to arbitrate between requests from input blocks to output blocks. The arbitration scheme is based on a maximal size approach proposed in reference [14]. It is a variant of round robin matching, which is shown to prevent starvation under uniform traffic [14]. A scheduling decision is arrived at in three steps: (a) Requests are sent from the input blocks to the output grant generation arbiters. (b) Grant signals are generated by the output grant arbiters and sent to input accept arbiters. (c) Accept signals are generated by the input accept arbiters and these signals represent the final scheduling decision. This decision is sent back to the input blocks as well as the crossbar switch to enable transfer of packets from the input blocks to the output blocks. The top level diagram of the scheduler is shown in figure 3.

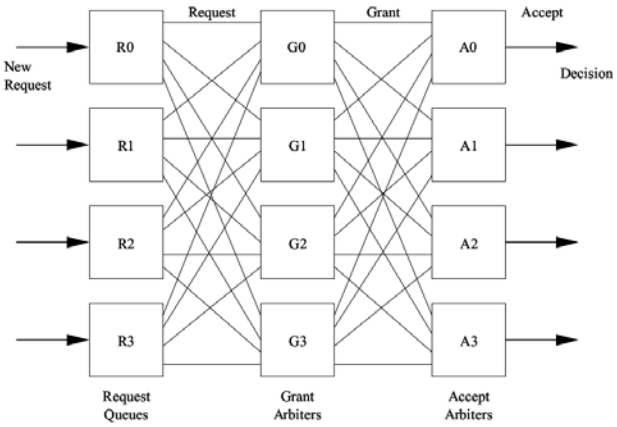

Fig 3: Scheduler Block Diagram

Each arbiter consists of a programmable priority encoder and a pointer to hold the value of the previously granted and accepted request. Figure 4 depicts the arbiter schematic.

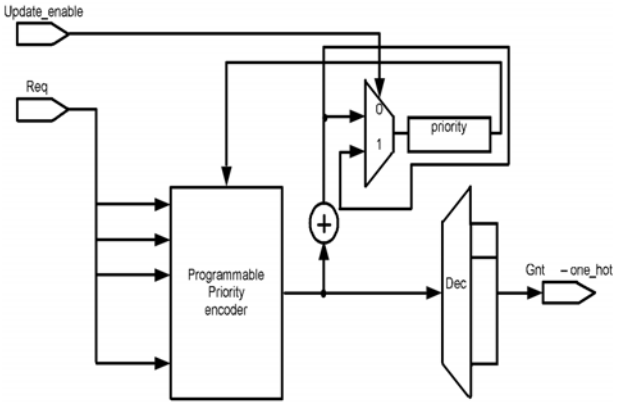

Fig 4: Arbiter Schematic

The programmable priority encoder generates a grant signal in response to a request signal based on a simple round robin scheme. Figure 5 depicts the programmable priority encoder schematic. The programmable priority encoder is realized using a hybrid design composed of two simple priority encoders. The programmable priority encoder design is based on thermometer encoding [15]. This design occupies less area compared to more classic programmable priority encoder designs. The most timing-critical component of the scheduler design is the Programmable Priority Encoder (PPE) utilized by each Arbiter. The speed of the programmable priority encoder will determine how fast the arbitration logic can run, and is likely to be the critical paths of the overall interconnect design.

VI. IP CORE FPGA DESIGN IMPLEMENTATION METHODOLOGY

Goal: To develop a synthesizable IP core in HDL, for Islip scheduler block of the routing logic.

A. HDL:

I. Design Description using Verilog-HDL

B. Verification Scheme:

I. Functional Simulation using Xilinx ISE Simulator

II. Test bench for top level design using Verilog-HDL

III. Stimulus for testing IP Core

IV. Simulation results / Timing diagram Analysis

C. FPGA Implementation:

I. Design to be synthesized using Xilinx ISE Software.

II. Generate a design bit stream for FPGA device.

III. Demo the download of design bit stream on FPGA device.

IV. No further verification on hardware / FPGA Kit

V. FPGA kit will NOT be provided

VI. Target Technology: Xilinx Spartan / Virtex Device.

VII. CONCLUSION

An iterative, round-robin algorithm, iSLIP can achieve 100% throughput for uniform traffic, yet is simple to implement in hardware. Prototype and commercial implementations of iSLIP exist in systems with aggregate bandwidths ranging from 50 to 500 Gb/s. When the traffic is nonuniform, iSLIP quickly adapts to a fair scheduling policy that is guaranteed never to starve an input queue. This new arbiter is fair for any input combinations and faster & it is designed using Xilinx ISE for simulation of Verilog code and Xilinx ISE for synthesis and FPGA implementation.

REFERENCES

[1] McKeown, N. W. 1995 Scheduling Algorithms for Input-Queued Cell Switches.Doctoral Thesis. UMI Order Number: UMI Order No. GAX96-02658., University of California at Berkeley.

[2] C. Partridge; P. Carvey et al. “A Fifty-Gigabit per Second IP Router,” Submitted to IEEE/ACM Transactions on Networking, 1996.

[3] T. Anderson et al. “High Speed Switch Scheduling for Local Area Networks,” ACM Trans. on Computer Systems, Nov 1993, pp. 319-352.

[4] N. McKeown et al. “The Tiny Tera: A small high-bandwidth packet switch core,” IEEE Micro, Jan-Feb 1997.

[5] LaMaire, R.O.; Serpanos, D.N.; “Two-dimensional round-robin schedulers for packet switches with multiple input queues,” IEEE/ACM Transactions on Networking, Oct. 1994, vol.2, (no.5):471-82.

[6] Lund, C.; Phillips, S.; Reingold, N.; “Fair prioritized scheduling in an input-buffered switch,” Proceedings of the IFIP-IEEE Conf. on Broadband Communications ‘96, Montreal, April 1996. p. 358-69.

[7] Hui, J.; Arthurs, E. “A broadband packet switch for integrated transport,” IEEE J. Selected Areas Communications, 5, 8, Oct 1987, pp 1264-1273.

[8] Ali, M.; Nguyen, H. “A neural network implementation of an input access scheme in a high-speed packet switch,” Proc. of GLOBECOM 1989, pp.1192-1196.

[9] N. McKeown, “Scheduling Cells in an input-queued switch,” PhD Thesis, University of California at Berkeley, May 1995.

[10] N. McKeown, “iSLIP: A Scheduling Algorithm for Input-Queued Switches,” Submitted to IEEE Transactions on Networking.

[13] T. Bjerregaard, and S. Mahadevan, “A Survey of Research and Practices of Network-on-Chip,” ACM Computing Surveys, Vol. 38, March 2006.

[14] N. McKeown, “Scheduling Algorithms for Input Queued Switches”, Ph.D. Thesis, Department of Electrical Engineering and Computer Science, University of California at Berkeley, 1995.

[15] P. Gupta and N. McKeown, “Designing and Implementing a Fast Crossbar Scheduler”, Micro, IEEE, Vol. 19, No. 1, pp. 20-28, Jan/Feb 1999

[16] T. Moscibroda and O. Mutlu. A case for bufferless routing in on-chip networks. ISCA-36, 2009.