

Characterization of a Fault-tolerant NoC Router

Sumit Dharampal Mediratta, Jeffrey Draper
USC Information Sciences Institute
Marina del Rey, California, USA-90292
sumitm@isi.edu, draper@isi.edu

Abstract— With increasing reliability concerns for current and next generation VLSI technologies, fault-tolerance is fast becoming an integral part of system-on-chip (SoC) and multi-core architectures. Another concern for these architectures is increasing global wire lengths with associated issues leading to network-on-chips (NoC) becoming standard for on-chip global communication. We recognize these issues and present an on-chip generic fault-tolerant routing algorithm. The microarchitecture of a NoC router implementing the proposed routing algorithm for a k-ary 2-cube topology is provided. The proposed router works in two phases. In the first phase, the network is explored for an existing path between source-destination pairs after reset or during system reconfiguration after fault detection. Existing paths are cached and used in the second phase of data communication during normal system operation. The presented router architecture also proposes a concept of dynamic multiplexing of virtual channels on physical channels to efficiently utilize physical channel bandwidth. The above approaches complement each other and when combined together, result in an efficiently realizable high-performance NoC fault-tolerant router. An implementation characterization of this k-ary 2-cube torus router in terms of area, power and critical path delay in IBM Cu-08 technology is presented, along with bandwidth and latency characterization for relevant cases.

I. INTRODUCTION

With the introduction of multiple changes and shrinking feature sizes in VLSI technologies, it has become possible to fabricate billion-transistor chips. Emerging paradigms utilizing these vast on-chip resources are system-on-chip and multi-core architectures. But, widespread reliability challenges are expected [1][2] in near-term VLSI fabrication technologies because of the evolutionary changes in scaling current materials and devices and revolutionary changes associated with new materials and devices. The introduction of multiple materials, processes and structural changes in a short period of time will increase the difficulty of understanding and controlling failure modes. So, fault-tolerance should be considered a necessity, rather than as a feature.

Another concern for the above architectures is increasing global wire lengths with associated issues like long-delay and skin-effect causing significant problems at higher frequencies of operation. It is also noted that bus-based architectures have inherent scaling limitations due to physical implementation issues and performance

impact of a shared resource. Consequently, NoC is becoming standard for on-chip global communication [2][3].

Some approaches for achieving fault tolerance have been proposed for the NoC paradigm [4][5][6]. Some of them (e.g. probabilistic and directed flooding) are susceptible to failures during an initial phase of transmission where every intermediate node drops packets thus thwarting the transmission altogether. Also, the application domain is not very wide for these approaches as loss of information is possible because there is a non-zero probability that destinations are not reached. Another limitation of these approaches is that high packet injection rates and large message lengths are not desirable for [4][5][6] because of redundant communication during normal system operation.

In this paper, firstly, a novel fault-tolerant routing algorithm for on-chip networks has been reported¹. In contrast to previously proposed approaches, the proposed routing algorithm finds a path if it exists between a pair of nodes and removes indeterminism from the completion of communication because it explores paths beforehand. Path exploration is done during system reconfiguration, and it does not impact normal system operation, and thus it remains oblivious to injection rates and message lengths. Fault information does not need to be propagated globally with the proposed approach. This relieves the burden of handling faults on the interconnection network supposed to be used for fault propagation. The probability of completion in a given time is insensitive to the number of faults in the proposed approach. Parallel path exploration greatly decreases latency in the proposed approach as compared to the sequential backtracking mechanism in pipelined circuit switching (PCS) [8] and makes performance of the proposed approach less sensitive to the number of faults. The proposed routing algorithm can be used either in operand or memory networks [2], but emphasis is given on memory networks in this work. These features make the proposed routing algorithm suitable for a wide range of applications.

Secondly, a router microarchitecture for a k-ary 2-cube torus network, and hardware designs of critical router components are also presented. Finally, an implementation feasibility and static performance characterization of the router hardware targeting IBM Cu-08 technology is presented. The proposed approach of finding existing paths in case of faults using a path exploration algorithm is general. Although the presented router assumes a torus network, most of the framework can be used with other topologies also.

¹ This work [7] is submitted for publication for the first time.

Section II describes the router microarchitecture; section III reports the hardware implementation and static performance characterization. Finally, section IV concludes this paper.

II. ROUTER MICROARCHITECTURE

This section describes the router microarchitecture and various important components of the router.

A. Routing Algorithm

The proposed routing algorithm works in two phases. The two-phase approach prevents redundant *data* communication, in contrast to previous approaches. In the first phase, simple (loopless) paths are explored and path histories are recorded (using direction and turn information) by a path exploration (PE) packet. Paths are then cached in a chip multiprocessor [2] node's local memory (section II.B). PE is performed after reset or during system reconfiguration after fault detection [9]. The router works as follows in the first phase with the objective of finding existing simple paths (maximum length (N-1) hops, where $N=k \times k$ is total number of nodes) between all pairs of nodes: -

1. At the source node, send PE packet to all network output ports (Fig. 1) and record the direction of the output port (e.g. North, West, South or East) in the corresponding packet.
2. At all other nodes, deliver the packet to the current node for extracting path information and forward the packet to only unvisited neighbors. Record direction information in delivered packets and turn information in forwarded packets. Send PE packet as a return packet (Fig. 1) to the source node if it is the first packet that has arrived from a particular source. Save the path in the route cache if it contains any node for which the current node doesn't have a valid path, else discard the path.
3. When return packets arrive at a source node, then cache the path information for visited nodes in the path in a manner similar to that discussed in step 2.

The above algorithm can be generalized more to cache more than one path for a particular source-destination pair. Relevant theory of the above algorithm along with dynamic performance characterization can be found elsewhere [7][10]. This paper focuses on the router microarchitecture and implementation feasibility.

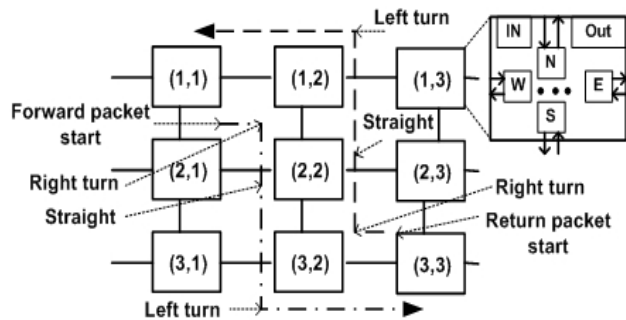


Figure 1. Relationship between forward and return paths.

In the second phase, normal communication happens by using cached paths, and the router simply behaves like a source wormhole router to route variable length packets, relying upon the paths provided by the source node. Topologies tend to become irregular after faults because of the random nature of faults, and table lookup based approaches have been suggested in the past for this purpose [8], instead of a finite state machine based approach.

This quality is well suited for the proposed two-phase algorithm of probing paths and sending data using cached paths.

B. Route-cache Organization

A route cache is implemented in a node's memory as a separate segment with a maximum limit on segment size dictating the size of the route cache. It has the benefit of the use of shared memory with the node processor and flexibility of increasing or decreasing the route cache for different applications for which dedicated route cache hardware can be insufficient or wasteful. The proposed approach achieves efficient utilization of hardware resources, resulting in an area efficiency essential to the NoC paradigm. The route cache is fully associative and will incur only cold misses and capacity misses. No conflict misses will be present because of the fully associative nature. The impact on the node's compute performance because of shared memory can be made minimal by careful memory organization. One option is the organization of memory in multiple banks (a standard technique in memory organization for having good aspect ratios and higher bandwidth) with the route cache segment located in some bank such that it interferes least with the node's memory demands. Another option is the use of memory with multiple read ports for allowing non-interfering access.

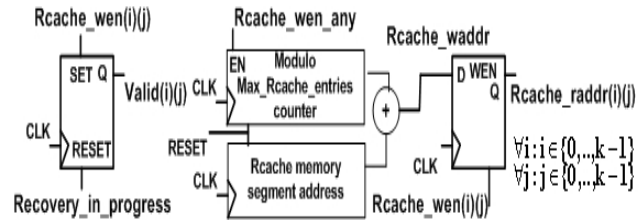


Figure 2. Route-cache access logic for k-ary 2-cube router.

A page-table like approach, with indirection from a destination address table (DAT) in the router with pointers to corresponding memory physical addresses, is used to access cached paths. The DAT approach avoids the searching overhead of fully associative caches for valid paths (Fig. 2). Putting route-accessing functionality in the router using a DAT frees up the node processor for performing other computation. Else, the processor could have directly provided paths using some software look-up table approach. If more paths need to be saved in the cache after the counter reaches the maximum segment size then either an entry can be invalidated or more memory can be requested. If the route cache segment size register is dynamically reduced to a smaller number, then the DAT should be truncated appropriately.

C. Deadlock and Livelock Handling

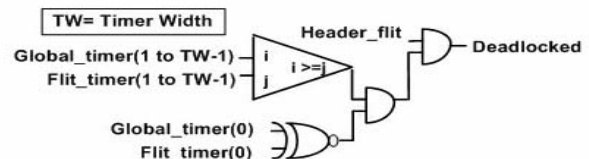


Figure 3. Deadlock determination logic.

Since no restrictions have been placed on the turns taken by the packets in the network to maximize adaptivity and find a path if it exists, deadlocks may be encountered in the proposed approach. Also, livelocks may happen during normal communication because of the variable length packet assumption and if the path overlaps largely with other simultaneous flows. The approach of providing

escape paths is known to not work in case of faults because faults on escape paths will not be tolerated [8]. Misrouting based approaches may introduce dependencies that may cause deadlock and require additional resources and restrictions to guarantee deadlock freedom [8]. Up*/down* routing with an acyclic spanning tree has lower performance and cannot provide all functional paths. The proposed approach uses a node's memory as deadlock buffers and re-injection of deadlocked packets in the network after breaking the cyclic dependency between packets. A deadlock condition is determined by comparing the current global time with a provided time-to-live value in the header flit. Special consideration is given to rollover in the value of the timers (Fig. 3). If all allocated space for deadlock buffers is used, then an exception will be generated at the node where the header flit is stalled. This approach gives visibility of severe network conditions to the programmer. As another power and performance inefficient but memory efficient option, backtracking of deadlocked packets to the source and re-injection after delay can be used during normal communication with possible large packets. PE packets must use deadlock buffers because of the absence of reserved paths.

D. Flow-control and Physical Transmission

To keep account of available space in the corresponding input port at the other end of a physical channel (PC) before transmitting data, some credit maintenance mechanism is required. This is achieved using virtual channel (VC)-specific credit bit inter-router communication. The credit network is V-bits wide instead of $\log_2(V)+1$ bits, because otherwise an additional requirement of the arbiter to choose candidate VCs for credit information communication in a fair manner results. Also, wires are less expensive with multiple metal layers than the arbiter's area and power.

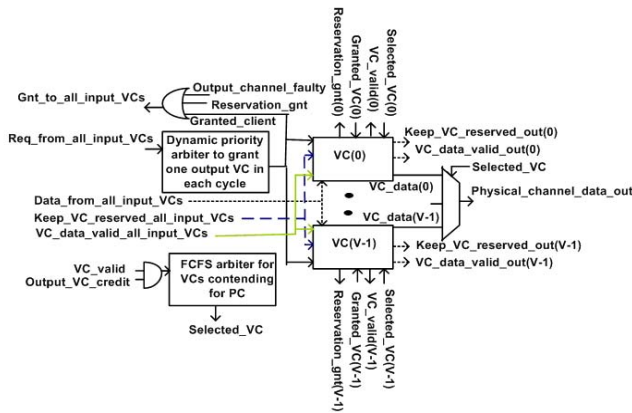


Figure 4. Output port VC handshaking and arbitration.

Any input VC can request any VC of the remaining (except U turn) output ports. Priority is given using a dynamic priority arbiter (Fig. 4) in the following order: return flits and re-injection of deadlocked packets, deadlocked forward packets, and forward non-deadlocked packets. This priority scheme is able to ensure in-order (in terms of injection time) delivery of packets to their destinations. However, this priority scheme doesn't introduce starvation because if forward packets are blocked for very long then they will get marked as deadlocked and their priority will get increased. Eventually when a forward PE packet does reach its destination then it either gets discarded or attains the highest priority level.

This router architecture presents a concept of dynamic multiplexing of VCs on a PC to efficiently utilize PC bandwidth. Instead of allocating fixed time slots for each VC, PCs are allocated using a

FCFS scheme to the output VCs. The proposed scheme will perform better than a fixed time slot scheme by using idle PC bandwidth for highly utilized VCs. With this approach, the VC index also needs to be communicated to the receiver, which is achieved using a V-bit wide bus given the on-chip paradigm.

E. Fault Information.

Fault information about the failed links in each direction is kept locally. Both input and output links for a particular direction are marked faulty, even if one of them is faulty, because of the assumption of functional return paths for corresponding forward paths. This may not be required if multiple phases of path exploration are used. An output link always gives a grant to the requests for the faulty output channel. Requests for a faulty channel can be made by PE packets (without any data) only. By giving them an unconditional grant they are prevented from holding network resources, and later are discarded. Data packets sent during system operation will not encounter any faults. Supporting a dynamic failure scenario when a fault interrupts a message in transit (faults encountered by the payload of a message when header has advanced) [8], is not necessary with the BIST approach [9] because it will be taken care of during a *test and recover* phase before committing of the correct results.

III. IMPLEMENTATION AND PERFORMANCE CHARACTERIZATION

In this section, an implementation characterization with respect to a variation in the number of VCs, number of network nodes supported and phit size in terms of area, power and critical path delay in IBM Cu-08 technology is presented. Bandwidth and latency characterizations for relevant cases are also given. The router design was specified in RTL-level behavioral VHDL code with generics used for relevant parameters. The route cache and destination address table are assumed to be big enough to hold entries for possible destination nodes. The default configuration is 4VC, 256-bit phit and 16 nodes, unless specified differently.

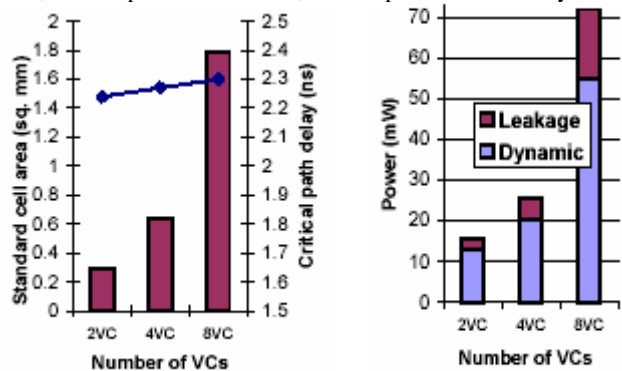


Figure 5. Characterization wrt number of VCs.

For small increments in the number of VCs, area and power (dynamic power measured with 20% toggle rate) grow almost proportionally to the number of VCs (Fig. 5). But for large increments, the overhead in area and power increments is quite high. This may be attributed to the critical path delay kept fixed to meet the target clock cycle. Also, the arbitration logic gets more complex with an increasing number of clients for one output VC (3xV) and number of VCs contending for a PC. To meet timing, the synthesis tool has generated resource-consuming hardware. More than 4VCs are not recommended unless higher performance

is more critical than low area. Either the same clock cycle and area-efficient hardware, or lower cycle time can be achieved with pipelining of the design at the cost of increased router latency.

Area, delay and power consumption of the router do not vary much over a wide range of number of nodes to be supported, N (Fig. 6). DAT entries grow linearly with N . Actual route cache entries (not included in the above characterization) may not need to grow linearly if overlapped paths are cached for multiple destinations. Also, route cache entries are ideally shared in a node's memory space, so they do not truly contribute to area and power overhead in an absolute sense. Aside from the issue of increasing route cache entries, the router design is scalable over large variations in N .

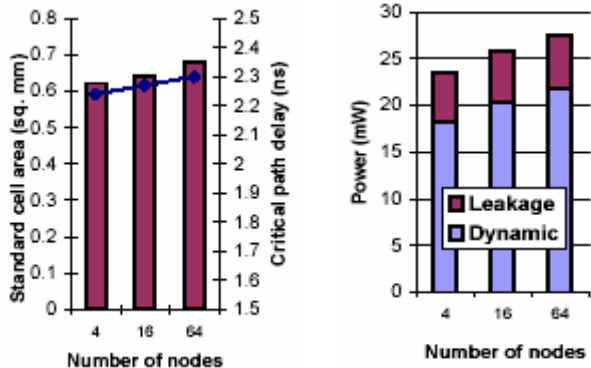


Figure 6. Characterization wrt number of nodes.

Area and power consumption (both dynamic and leakage) increased almost proportionally for small increments in phit size and increased with some overhead for large increments in phit size (Fig. 7). Arbitration logic is largely unaffected with increased sequential buffering. To develop an intuition, sequential register requirements are similar for 8VC, 256-bit phit and 4VC, 512-bit cases. Combinational logic increased proportionally with phit size. This is because of the affect of 14x1 phit-size wide multiplexers to latch data used individually in 16 output virtual channels. An increase in dynamic power is implied by a proportional increase in the number of switching elements. An increase in leakage power is also implied by a proportional increase in area. It is simply observed that small increments in the parameters give a more realistic picture of characterization, and dynamics of the synthesis tool become significant for large variations.

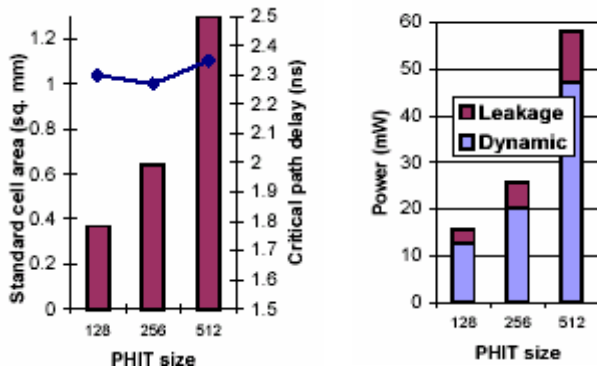


Figure 7. Characterization wrt phit size variation.

Bandwidth and best-case router latency have also been characterized with respect to variation in the phit size (Fig. 8). For other cases (change in V and N), this characterization is not very

significant because phit size is constant and the clock cycle is almost the same as that of the best-case router latency at the expense of increase in area and power. The effect of increasing V is more dynamic because of its interference with the blocking probability, and is reported in a separate paper [10]. Bandwidth increases linearly with increments in the phit size, and the best-case router latency remains almost the same. This is achieved at the cost of more resources with larger phit sizes.

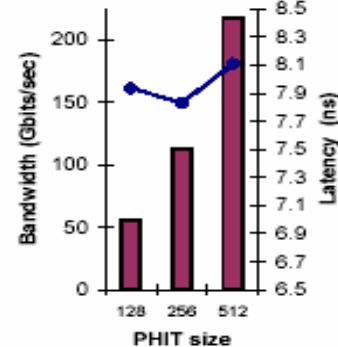


Figure 8. Router bandwidth and latency characterization.

IV. CONCLUSION

In this paper, a novel fault-tolerant routing algorithm for on-chip networks has been reported. The proposed routing algorithm finds a path if it exists between a pair of nodes and removes indeterminism from the completion of communication. Path exploration is done during system reconfiguration and it does not affect normal system operation, and thus it remains oblivious to injection rates and message lengths. The proposed routing algorithm can be used either in operand or memory networks and any topology. All of these features make the proposed routing algorithm suitable for a wide range of applications. An implementation feasibility and static performance characterization of the router targeting IBM Cu-08 technology are also presented.

REFERENCES

- [1] <http://public.itrs.net>
- [2] T.M. Pinkston, and J. Shin, "Trends toward on-chip networked microsystems.," Journal of HPCN, 2005
- [3] W.J. Dally, and B. Towles, "Route packets, not wires: on-chip interconnection networks," DAC, 2001
- [4] T. Dumitras, S. Kerner, and R. Marculescu, "Towards on-chip fault-tolerant communication," Asia and South Pacific DAC, 2003
- [5] R. Marculescu, "Networks-on-chip: the quest for on-chip fault-tolerant communication," Symposium on VLSI, 2003
- [6] M. Pirretti, G.M. Link, et al., "Fault tolerant algorithms for network-on-chip interconnect," Symposium on VLSI, 2004
- [7] S. Mediratta, Communication mechanisms for Processing-In-Memory systems, PhD dissertation, University of Southern California, 2006
- [8] J. Duato, S. Yalamanchili, and L. Ni, Interconnection Networks: An Engineering Approach, IEEE Computer Society Press, 1997
- [9] S. Mediratta, J. Draper. "On-chip Fault-tolerance Utilizing BIST Resources", MWSCAS, 2006
- [10] S. Mediratta, J. Draper. "Performance Evaluation of Probe-Send Fault-tolerant NoC Router", Unpublished