Low-Overhead Hard Real-time Aware Interconnect Network Router

Michel A. Kinsy
Department of Computer and Information Science
University of Oregon
Email: mkinsy@cs.uoregon.edu

Srinivas Devadas
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Email: devadas@@mit.edu

***Abstract*—The increasing complexity of embedded systems is accelerating the use of multicore processors in these systems. This trend gives rise to new problems such as the sharing of on-chip network resources among hard real-time and normal best effort data traffic. We propose a network-on-chip router that provides predictable and deterministic communication latency for hard real-time data traffic while maintaining high concurrency and throughput for best-effort/general-purpose traffic with minimal hardware overhead. The proposed router requires less area than non-interfering networks, and provides better Quality of Service (QoS) in terms of predictability and determinism to hard real-time traffic than priority-based routers. We present a deadlock-free algorithm for decoupled routing of the two types of traffic. We compare the area and power estimates of three different router architectures with various QoS schemes using the IBM 45-nm SOI CMOS technology cell library. Performance evaluations are done using three realistic benchmark applications: a hybrid electric vehicle application, a utility grid connected photovoltaic converter system, and a variable speed induction motor drive application.**

I. Introduction

With multicore and many-core architectures becoming mainstream computing platforms, they are deployed in many computation environments that require concurrent execution of different tasks. These tasks may require hard real-time and/or normal computation support with certain inter-core communications guarantees. Some data communications may be highly latency-sensitive while others may not. The current trend in system-on-chip (SoC) design is system-level integration of heterogeneous technologies consisting of a large number of processing units such as programmable RISC cores, memory, DSPs, and accelerator function units/ASIC [1], [2]. In fact, many control applications today have both general-purpose and real-time requirements. These SoCs must guarantee: (1) real-time operation reactive to external events, like traditional embedded systems; and (2) high average computing throughput and programmability, like general purpose multicore systems. In new automotive applications, dynamic stability control (DSC) support and engine control systems are coupled with advanced active safety programs, e.g., collision avoidance algorithms, to help prevent the driver and vehicle from a possible accident. Collision avoidance functionality determines the safe approaching speed to the front car by detecting the relative speed and distance through one or several types of sensors, like video, laser, or ultrasound, etc., and converting in hard real-time these parameters into a collision avoidance strategy. These safety critical system components are integrated onto a single chip with multimedia and comfort applications, e.g., seat-adjusted braking, fuel utilization analysis, data streaming, and voice commands [3].

Dally et al [2] argue that an SoC, composed of a number of hardware components: processors, DSPs, memories, peripheral controllers, and custom logic, should be connected by a network that routes packets between them, instead of connecting these modules via dedicated wires. Network-on-chip (NoC) architectures constitute an effective data communication infrastructure, providing both flexible connectivity and scalability [4].

A. Typical Virtual Channel Router

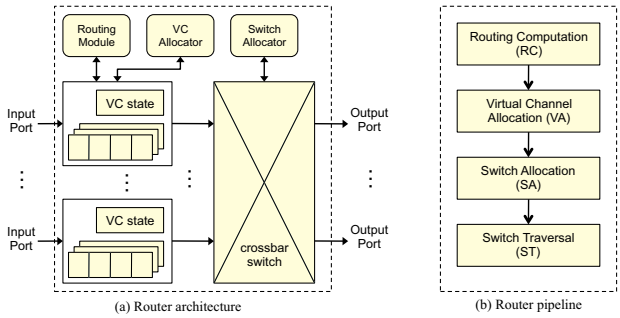


Fig. 1. Typical virtual-channel router architecture.

In conventional virtual-channel routers [5], the routing operation takes four steps or phases; namely, routing (RC), virtual-channel allocation (VA), switch allocation (SA), and switch traversal (ST), where each phase corresponds to a pipeline stage in the router. When a head flit (the first flit of a packet) arrives at an input channel, the router stores the flit in the buffer for the allocated virtual channel and determines the next hop for the packet (RC phase). Given the next hop, the router then allocates a virtual channel in the next hop (VA phase). Finally, the flit competes for a switch (SA phase); if the next hop can accept the flit, it moves to the output port (ST phase). Figure 1 illustrates such a virtual-channel router (VCR).

B. Predictability in virtual channel router

Unfortunately, virtual-channel based routing in multicore systems in general, and SoCs in particular, further weakens the notion of predictability and determinism so critical to hard real-time inter-core communications. The fact that various traffic must compete for physical link access leads to non-deterministic data transfer delays. Therefore the main drawback of conventional NoCs is their inadequacy in latency predictability. A few techniques, such as express virtual channels [6], dedicated virtual channels, priority-based NoC routing [7], QoS at the network level [8], and RTOS support

for NoC-based architectures [9], have been used to mitigate the lack of deterministic latency guarantees in NoC-based communications. Still, they have not been able to meet the hard real-time constraints required by many distributed real-time applications.

The two main design requirements for the on-chip communication layer in these SoCs are: (1) hard real-time processes have absolute deadlines that must be met, and this includes processing time at the cores and inter-core data transfer latencies; (2) the network-on-chip is a shared resource and it needs to be used by all processes (real-time and best-effort). In this work, we propose a novel network-on-chip router, called the Hard Real-time Support (HRES) router. It provides predictable and deterministic communication latency for real-time data traffic while maintaining high concurrency and throughput for normal/general-purpose traffic with minimal hardware overhead. Specific key contributions are as follows:

- Decoupling of hard real-time and best effort traffic by designing a two-datapath router;
- Routing algorithm to maximize link throughput, guarantee hard real-time timing constraints, and provide fairness of link utilization among the two classes of traffic;
- Acknowledgment-free, retransmission-free, lossless, bufferless, table-based routing;
- Deadlock and livelock freedom guarantees with no modification to the buffered datapath of the router, and low hardware overhead for supporting hard real-time communications.

II. Related Work

There are several research efforts with the goal of providing some amount of predictability at different architectural levels in multi-processor systems. At the interconnect network level for example, Grot et al [10] propose a new lightweight topology-aware QoS architecture that provides service guarantees for applications such as consolidated servers on CMPs and real-time SoCs. But to reduced hardware complexity, they adopt a scheme that restricts the areas on the chip where QoS is guaranteed. Our approach provides hard real-time support throughout the chip at low hardware complexity. Shi and Burns [7] also present a method for evaluating at design time the schedulability of a traffic-flow set with different quality of service (QoS) requirements in a real-time SoC/NoC communication platform. Their approach uses priority-based wormhole switching policy and off-line schedulability analysis of traffic-flows. In this paper, the router and routing algorithm are flexible to support run-time application scheduling. Æthereal [11] is a generalized NoC architecture that uses a centralized scheduler and time-division multiplexing to allocate link bandwidth to provide guaranteed throughput. Authors made the point that with higher average latency, time-division multiplexed access is not ideal for high-priority control traffic. Kakoee et al [12] propose ReliNoC, a network-on-chip architecture that can withstand failures, while maintaining not only basic connectivity, but also quality-of-service support based on packet priorities. This work primarily focuses on QoS in the presence failure. Das et al present in [13] an application-aware prioritization approach for On-Chip Networks. The main idea is to divide processor execution time into phases, rank applications within a phase based on stall-time criticality, and have all routers in the network prioritize packets based on their applications' ranks. To overcome the lower resource utilization associated with traditional QoS routers, Rijpkema et al in [14] present a prototype router implementation which combines guaranteed throughput and best-effort routers by sharing resources. The IBM Colony router [5] found in the ASCI White supercomputer uses a hybrid datapath packet routing architecture. The router has three crossbars, and one of those crossbars is used by packets to cut through the router when contention is low to reduce latency. This architecture informs some of the decisions made in designing our HRES router.

III. HRES Router Architecture

As described in Section I, a flit routing operation generally takes four steps. Resource sharing conflict may arise in three of those four stages: at the buffer read and route computation level, at the virtual-channel allocation level, and at the switch arbitration level. The alternative to the multi-stage routing scheme is the bufferless approach. Bufferless routing generally consists of arbitrating between two flits competing for the same physically link at each network router or node. The flit that wins access to the link continues through the network competing for link access at each node. Some type of acknowledgment mechanism is implemented to inform the source node of a successful transmission or a failure. Commonly, time-out or negative acknowledge schemes are used for the acknowledgment [15]. Although the process as just described generally has lower network traversal latency than the buffered approach, dropping of flits even with acknowledgment mechanism makes data communication through the bufferless datapath less predictable and deterministic, both key desirable characteristics for effective hard real-time communication.

The main challenge is to guarantee the hard deadline of certain packets while promoting a high degree of communication concurrency, optimal bandwidth utilization, as well as predictability, determinism, and low latency, with no significant area or power increase. The HRES-router uses a hybrid datapath with an interface identical to the conventional virtual-channel router shown in Figure 1. This simplifies the interface verification and allows for quick and seamless integration of the router into existing SoCs with little or no system-level change. Data communications are grouped into two categories: *guaranteed latency* for hard real-time traffic and *best effort latency* for general-purpose traffic. Note that traditional quality of service based on priority can still be used in the case of *best effort latency* traffic.

A. Router Input Port Micro-Architecture

Flit type at the network interface is expanded by a single bit to mark the flit as hard real-time or normal traffic. Flits coming in to the router have two datapaths, one for real-time traffic

or *guaranteed latency* and one for general-purpose (non real-time) traffic or *best effort latency*. Just as the virtual-channel (VC) bits are read from the incoming flit for VC allocation, the service bit is read from the flit. If the *guaranteed latency* bit is set to 1, the flit is not buffered, it is directly routed to the real-time crossbar. The datapath for general-purpose (non real-time) traffic consists of input buffers storing flits while they are waiting to be forwarded to the next hop. When a non real-time flit is ready to move, the switch connects an input buffer to an appropriate output channel via a series of actions: route computation, virtual-channel at the next hop allocation, and finally switch allocation. These stages are pipelined and stalled in case of resource contention.

B. Switching structures and Switch Allocations

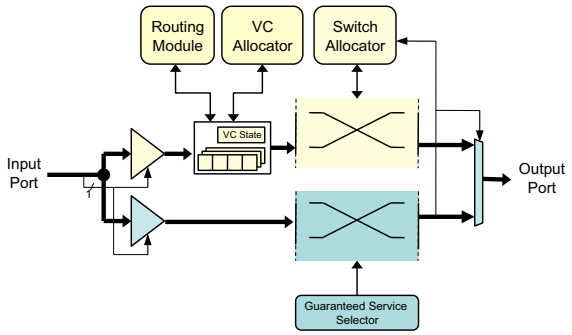
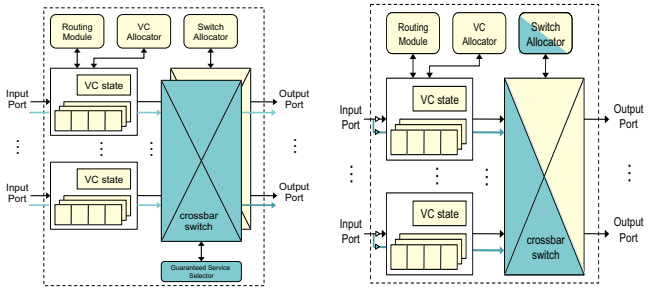


Fig. 2. Switching logic of the two types of traffic.

There are two crossbars at the router, one for the buffered general-purpose traffic and one for the bufferless real-time traffic. Figure 2 illustrates switching structures and their locations on the datapath. As previously mentioned, the buffered datapath is left unmodified. The only added logic is on the output port side. In addition for the switch allocator to grant requests, it now also checks that granted requests are not blocked by real-time flits during link traversal. If it is the case, the switch allocator proceeds as though the port request lost switch allocation. This modification is not on the critical path logic and can be done without introducing another cycle in the pipeline or affecting the timing characteristics of the datapath logic. In the Verilog code for the conventional virtual-channel router, a switch allocation stall is done on a VC at a port when *request* is 1 and *grant* is 0. In the HRES-router switch allocation stall is done on a VC at a port when *request* is 1 and *grant* is 0, or when *valid* bit for real-time flit assigned to the port is 1.

Hard real-time communications are more predictable in terms of source and destination processing elements participating in the transactions. This is due to the fact that these transactions are event-driven, and event handlers mapping is generally done offline. We use this property of real-time communication to derive the algorithm for determining routes. The details of the algorithm are presented in Section IV. In the HRES router, the switch for the bufferless traffic is supplemented with a programmable *guaranteed-service* selector table. This approach prevents flit dropping and removes the need for acknowledgment (and acknowledgment logic) while constraining route selection minimally. Once routes are determined, routers are pre-configured by setting the proper selector bits at each router. For a 5-port router, the real-time traffic switch consists of five 5-to-1 multiplexers set up offline and per-application. The *guaranteed-service* selector table allows multiple concurrent real-time traffic through a router as long as there is no sharing of physical links. Two types of real-time communications are supported in the HRES-router; one-to-one and one-to-many. Many-to-one will undoubtedly introduce potential conflicts. It is worth noting that in this router, one-to-many data communications are automatically supported with no additional logic. It has been shown that for low to medium network load, significant network power savings, with minimal performance loss, can be made with bufferless routing [15], [16]. Therefore, one can envision disabling the buffered datapath at the routers even in the absence of real-time traffic for power savings under low network traffic applications.

C. Other router architectures considered



(a) Two-network router. (b) Single shared crossbar router.

Fig. 3. Other alternative designs.

One approach to guarantee fixed latencies for the real-time traffic is to have a two-network routing scheme, where the real-time and the normal traffic share no physical link. Figure 3(a) shows such a router. The key disadvantages of this type of router are duplication of wires and logic that lead to more cell area, shown in Table I, and changes in the network interface compared to the conventional router. A third router architecture considered consists of a single large crossbar, where the switch arbitration logic is modified to give priority to the real-time traffic. Figure 3(b) depicts this router architecture. This approach increases the switch arbitration datapath and adds to the router critical path. The arbiter must serialize real-time and normal traffic requests. All three routers use the same table-based routing algorithm presented in Section IV.

	Number ports	Cell area
HRES router	392	47190.44
Two-Network router	721	51766.84
Single crossbar router	392	52812.94

TABLE I
AREA COMPARISON OF ROUTER ARCHITECTURES.

IV. Routing Algorithm

Algorithms used to compute routes in network-on-chip (NoC) architectures, generally fall under two categories: *obliv-*

ious and *dynamic* [17]. For the normal traffic using the buffered datapath, any traditional routing algorithm will still work and under the same assumptions. For bufferless, conflict-free, hard real-time traffic, the routing scheme is slightly more restricted. *Oblivious* routing with table-based support for both *minimal* and *non-minimal* routing is more appropriate. Considering application communication characteristics in terms of the real-time data constraints and normal data bandwidth requirements, the routing algorithm establishes the real-time traffic routes statically and offline. The key challenge is to find a fair and an effective tradeoff between load balancing of the network, to avoid premature congestion or lack of forward progress of normal traffic, and low data communication latency for both types of traffic.

A. Definitions and routing formulation

We first introduce standard definitions of flow networks.

Definition 1: A flow f_i is a data communication/traffic pattern from one processing element–source node s_i–to another processing element–destination node t_i.

The set of packets/flits part of flow f_i passing through the network link from node u to node v consuming bandwidth and/or buffer space during a certain time interval is represented with the real-valued function $f(u, v)$. We may have multiple flows with the same source and destination.

Definition 2: Given a multicore topology, we can derive a network flow $G(V, E)$, a directed graph, where an edge $(u, v) \in E$ represents a physical inter-router link and has capacity $c(u, v)$. The capacities $c(u, v)$ are the available bandwidths on edges. If $(u, v) \notin E$, then $c(u, v) = 0$.

Definition 3: For the purpose of differentiating between real-time flows from normal flows, we denote real-time flow i from node s_i to node t_i as $f_{i[r]}$. We also introduce $c_{[i]}(u, v)$ (the upper bound on flow demand $f_{i[r]}$), $w_{[i]}(u, v)$ (a real-valued weight function for the edge (u, v)), and k_i (the maximum number flow $f_{i[r]}$ can be divided for routing).

We use the following mixed-integer linear programming (MILP) formulation to compute the set of admissible or feasible routes for the real-time flows.

$$Minimize \quad \sum_i \sum_{(u,v)\in E} f_{i[r]}(u, v) \cdot w_{i[r]}(u, v) \tag{1}$$

subject to:

$$\forall i,\ \forall (u, v) \in E \quad f_{i[r]}(u, v) \in S_{[r]}(u, v), \quad \left|S_{[r]}(u, v)\right| \leq 1 \tag{2}$$

$$0 \leq \sum_i f_{i[r]}(u, v) \leq c_{[i]}(u, v) \leq c(u, v) \tag{3}$$

$$\forall (u, v) \in E \quad (\sum_i f_i(u, v) + \sum_i f_{i[r]}(u, v)) \leq \ c(u, v) \tag{4}$$

$$\forall i \quad f_{i[r]} \in K_{i[r]}, \quad \left|K_{i[r]}\right| \leq k_i \tag{5}$$

$$\forall i,\ \forall j \in \{1, ..., k_i\} \quad \frac{\sum_{(u,v)\in E} f_{(i,j)[r]}(u, v)}{f_{(i,j)[r]}} \leq deadline_i \tag{6}$$

The real-valued weights $w_{[i]}(u, v)$ are selected per application. A uniform weight of 1.0 will try to minimize the hop count. A good heuristic is a weight selection that is the inverse proportion of the number of adjacent nodes. This approach assigns higher weights to edge links, because flows using these edges have less path selection diversity. k_i allows a real-time flow $f_{i[r]}$ to be spread across k different paths if the application permits. This MILP formulation, which routes real-time traffic in a predictable and deterministic fashion, also makes the splitting of flows more manageable because it eliminates many of the problems encountered with buffered flow splitting, such as out-of-order packets delivery and destination buffering and rearrangements of packets. $deadline_i$ is derived from the application specification, and Equation 6 enforces the condition that each sub-flow $f_{(i,j)[r]}$ of flow $f_{i[r]}$ is routed through a path where the deadline can be met. Although the algorithm allows splitting of real-time flows to relax routing constraints, no guarantee is made on the optimality of splits, since the classical splittable flow and unsplittable flow problems are NP-hard [18]. Equation 2 simply constrains the link (u, v) to be used by at most one real-time flow. For the routing of normal flows and virtual-channel allocation, any traditional routing algorithm (oblivious or adaptive) and VC allocation scheme (static or dynamic) can be used.

B. Quality of service (QoS)

Quality of service is the method for guaranteeing that bandwidth and buffer utilization by the various flows in the network is done in a matter that promotes *throughput fairness* and *latency fairness*. QoS in this work needs to be ensured for each flow type and among flows of the same type. Real-time traffic is contention-free, therefore to enforce *throughput fairness* and avoid starvation of normal flows, the variable $c_{[i]}(u, v)$ is used to ensure a minimum bandwidth availability to normal flows. The flow splitting property of the routing algorithm allows real-time flows that violate the minimal bandwidth threshold to be split. Due to the deterministic nature of real-time flow paths, any adverse effect of the splitting, e.g., out-of-order packets/flits, can be resolved through sender-receiver synchronization. *Latency fairness* is partially controlled by the application. In some cases, deterministic latency, especially for real-time flows, may be more important than low latency. The real-valued weights $w_{[i]}(u, v)$ are used to control the *latency fairness* of real-time flows and load-balancing of the network at the routing level. For *throughput fairness* and *latency fairness* among buffered traffic, any traditional fairness mechanism can be applied.

C. Deadlock and livelock

The algorithm for routing real-time traffic is deadlock-free because it assumes bufferless paths and no sharing of a physical link. Similarly, it is livelock-free because there is no run-time misrouting of flow. Deadlock-freedom and livelock-freedom for normal traffic must be independently ensured by the algorithm used to route normal flows.

V. Evaluation

We use the *Heracles* RTL-based multicore design platform [19], [20] to evaluate the HRES-router and the proposed routing algorithm. The *Heracles* design environment uses synthesizable MIPS cores, supports shared-memory, and has a compiler for C/C++ applications.

A. Router Configurations

We construct five different router configurations:

1) a virtual-channel reservation-based QoS (v-QoS) router: VCs are statically partitioned into two sets, one set is used by the real-time flows and the other by the best effort flows. VC allocation per packet is done dynamically within a set;
2) a priority-based QoS (p-QoS) router: real-time flows are assigned the highest priorities, and packets can only be given VCs off the same priority. Physical link access is also prioritized;
3) a lossy bufferless (L-Bless) router with acknowledgment;
4) a lossless bufferless (Ls-Bless) router: to void packet loss and retransmission we use a routing table to statically configure routes using the algorithm designed for the HRES-router;
5) our proposed HRES-router.

For the virtual-channel based routers, we use 2 VCs, 4 VCs, and 8 VCs per port configurations.

B. Benchmarks

To test the efficiency the various routing techniques, we explore three benchmark applications: one from the automotive industry, hybrid electric vehicle (HEV), one smart grid system application, utility grid connected photovoltaic converter system (smart-grid), and a third one from the industrial motor control domain, variable speed induction motor drive (motor drive). These applications have fast switching electronic components and their control and monitoring require high-performance and hard real-time computation guarantees. We profile the applications and construct a directed task-graph for each application. We group tasks and inter-task communications under best-effort and hard real-time depending on the application functional requirements.

C. Results and Comparisons

1) Area and Power Estimates: Area and power estimates are done using Synopsys Design Compiler with an IBM 45-nm SOI CMOS technology cell library. Typical operating point is 1.0 V, 25 C, and worst-case is 0.9 V, 125 C. Figure 4 shows the area utilization per router with different virtual-channel configurations. Overall, VC-based router occupies significantly more cell area than bufferless router. The total cell area of the lossless bufferless router is less than 3% of the area recorded for the 2-virtual-channel VCR. Across different VC configurations, the HRES router takes less than 3% more cell area on average to build. This shows that the HRES router, which guarantees predictability, determinism, and low latency for hard real-time of packets while maintaining all the functionalities of the conventional VC router, has negligible area overhead. Figure 5 shows the power consumption per router. Power summary does not include the clock tree power. The power consumption per router is inline with the area estimates. HRES router has comparable power consumption to the conventional VC router. Dynamic power is lower than the cell leakage power because application-based switching activity is not taken into account.

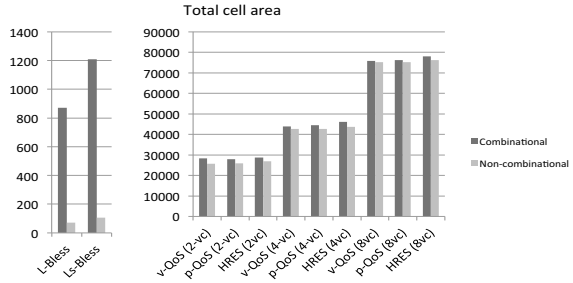


Fig. 4. Total cell area utilization per router.

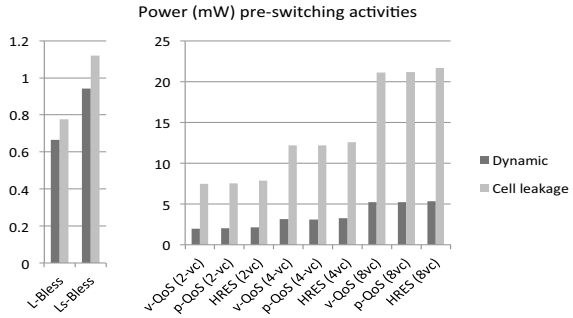


Fig. 5. Total power consumption per router.

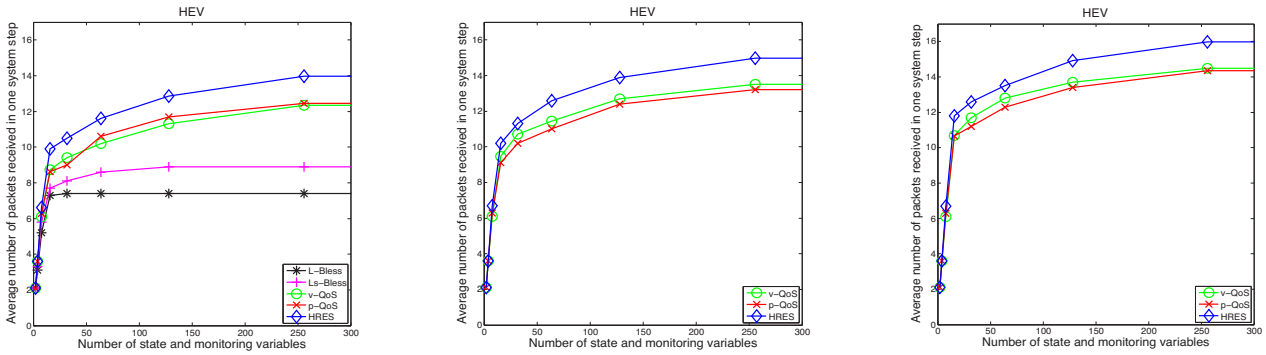
Table II shows the clocking speed of the various routers per number of virtual channels. The conventional VC router runs the fastest around $6.7GHz$, but it has 4 pipeline stages. On the other hand bufferless routers are single stage and run close to $3.6GHz$. The HRES router, which architecturally inherits from both, runs closer to the bufferless router speed.

	v-QoS	p-QoS	HRES	L-Bless	Ls-Bless
2-VC	6.67	5.88	3.45	3.70	3.57
4-VC	6.25	5.56	2.94	3.70	3.57
8-VC	5	5	2.78	3.70	3.57

TABLE II
Clocking speed of routers.

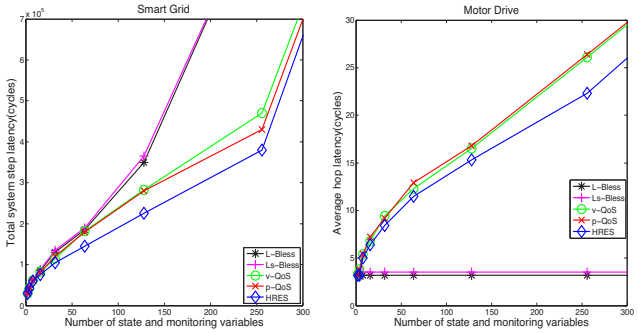
2) Throughput and Latency Results: We use the *Heracles* default setup: 2D-mesh, 46-bit flits, 8 VC depth, and variable packet length. We run our benchmark applications on a 16-core system synthesized on the Virtex6 XC6VL75T. We use XY-routing for all routers except Ls-Bless and HRES. Since our benchmarks are control applications we can increase the network traffic by increasing the number of state variables and monitoring variables in the application.

Figures 6(a), 6(b), and 6(c) show the throughput results for the HEV applications for 2 VCs, 4 VCs, and 8 VCs respectively. Other applications show similar trends. In Figure 7(a) we present the total system step latency per router for the smart grid application, and Figure 7(b) shows the average per hop latency using the motor drive application. Overall, the HRES-router outperforms other routers in terms of throughput and



(a) Using 2 Virtual channels per port. (b) Using 4 Virtual channels per port. (c) Using 8 Virtual channels per port.

Fig. 6. Throughput results for the HEV application.



(a) System step latency. (b) Average per hop latency.

Fig. 7. Latency results per router.

latency. Overall the proposed router requires less area than non-interfering networks, and provides better QoS in terms of predictability and determinism to hard real-time traffic than priority-based routers. It also outperforms other routers in terms of throughput and latency.

VI. Conclusions

In this paper, we propose a network-on-chip router that provides predictable and deterministic communication latency for real-time data traffic, while maintaining high concurrency and throughput for normal/general-purpose traffic with minimal hardware overhead. The proposed router requires less area than non-interfering networks, and provides better QoS in terms of predictability and determinism to hard real-time traffic than priority-based routers. With the HRES-router, we are able to realize these design goals: low latency, ability to easily integrate best-effort and hard real-time services, high effective bandwidth utilization, with low hardware overhead.

References

[1] W. Wolf, A. Jerraya, and G. Martin, "Multiprocessor system-on-chip (mpsoc) technology," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 27, pp. 1701 –1713, oct. 2008.

[2] W. J. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," in *Proc. of the 38th Design Automation Conference (DAC)*, June 2001.

[3] V. von Tils, "Trends and challenges in automotive electronics," in *Power Semiconductor Devices and IC's, 2006. ISPSD 2006. IEEE International Symposium on*, pp. 1 –3, june 2006.

[4] A. Ivanov and G. D. Micheli, "The Network-on-Chip Paradigm in Practice and Research," *Design & Test of Computers*, vol. 22, no. 5, pp. 399–403, 2005.

[5] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, 2003.

[6] A. Kumar, L.-S. Peh, P. Kundu, and N. K. Jha, "Express virtual channels: towards the ideal interconnection fabric," in *Proceedings of the 34th annual international symposium on Computer architecture*, ISCA '07, (New York, NY, USA), pp. 150–161, ACM, 2007.

[7] Z. Shi and A. Burns, "Real-time communication analysis for on-chip networks with wormhole switching," in *Networks-on-Chip, 2008. NoCS 2008. Second ACM/IEEE International Symposium on*, april 2008.

[8] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "Qnoc: Qos architecture and design process for network on chip," *J. Syst. Archit.*, vol. 50, pp. 105–128, Feb. 2004.

[9] J. Madsen, S. Mahadevan, K. Virk, and M. Gonzalez, "Network-on-chip modeling for system-level multiprocessor simulation," in *Real-Time Systems Symposium, 2003. RTSS 2003. 24th IEEE*, dec. 2003.

[10] B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu, "Kilo-noc: a heterogeneous network-on-chip architecture for scalability and service guarantees," in *Proceedings of the 38th annual international symposium on Computer architecture*, ISCA '11, (New York, NY, USA), pp. 401–412, ACM, 2011.

[11] K. Goossens, J. Dielissen, and A. Radulescu, "æthereal network on chip: Concepts, architectures, and implementations," *IEEE Des. Test*, vol. 22, pp. 414–421, Sept. 2005.

[12] M. Kakoee, V. Bertacco, and L. Benini, "Relinoc: A reliable network for priority-based on-chip communication," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, pp. 1 –6, march 2011.

[13] R. Das, O. Mutlu, T. Moscibroda, and C. R. Das, "Application-aware prioritization mechanisms for on-chip networks," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 42, (New York, NY, USA), pp. 280–291, ACM, 2009.

[14] E. Rijpkema, K. G. W. Goossens, A. Radulescu, J. Dielissen, J. V. Meerbergen, P. Wielage, and E. Waterlander, "Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip," pp. 350–355, 2003.

[15] T. Moscibroda and O. Mutlu, "A case for bufferless routing in on-chip networks," *SIGARCH Comput. Archit. News*, vol. 37, June 2009.

[16] G. Michelogiannakis, D. Sanchez, W. J. Dally, and C. Kozyrakis, "Evaluating bufferless flow control for on-chip networks," in *Proceedings of the 2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip*, NOCS '10, (Washington, DC, USA), pp. 9–16, IEEE Computer Society, 2010.

[17] L. M. Ni and P. K. McKinley, "A survey of wormhole routing techniques in direct networks," *Computer*, vol. 26, no. 2, pp. 62–76, 1993.

[18] J. M. Kleinberg, *Approximation algorithms for disjoint paths problems*. PhD thesis, Massachusetts Institute of Technology, 1996. Supervisor-Michel X. Goemans.

[19] M. Kinsy, M. Pellauer, and S. Devadas, "Heracles: Fully synthesizable parameterized mips-based multicore system," in *Field Programmable Logic and Applications (FPL), 2011 International Conference on*, pp. 356 –362, Sept. 2011.

[20] M. A. Kinsy, M. Pellauer, and S. Devadas, "Heracles: A tool for fast rtl-based design space exploration of multicore processors," in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '13, (New York, NY, USA), pp. 125–134, ACM, 2013.