A Delay Model for Router Micro-architectures

Li-Shiuan Peh
lspeh@cs.stanford.edu

William J. Dally
billd@csl.stanford.edu

Computer Systems Laboratory
Stanford University
Stanford, CA94305

Abstract. *Current router models [2, 3, 5, 6] assume that clock cycle time depends solely on router latency. However, in practice, routers are heavily pipelined, making cycle time largely independent of router latency. In this paper, we describe a router delay model that accurately accounts for pipelining based on technology-independent delay estimates derived through detailed gate-level analysis. Simulations of realistic router pipelines show significant performance differences compared with the commonly-assumed unit-latency model. Using realistic pipeline models, we compared wormhole and virtual-channel flow control. Our results show that virtual channels incur a modest additional cycle of per-hop router latency which is more than offset by the 25-40% throughput improvement over a wormhole router.*

1. Introduction

Most current literature in interconnection networks reports comparisons of different flow control and routing techniques without considering implementation complexity and the impact on router delay, simply assuming unit router delay. This can lead to inaccurate and skewed comparisons. A router delay model which enables designers and researchers to factor in implementation-specific delay estimates will thus be invaluable.

Chien [2, 3] proposed a router model for wormhole and virtual-channel routers[1] to address this need. In his model, he presented a canonical router architecture as depicted in Figure 1, which can be applied to all routers, regardless of the flow control or routing technique governing the router. The canonical router architecture consists of the following functions:- address decoding (AD), flow control (FC), routing header selection (SEL), crossbar arbitration (ARB), crossbar traversal (CB) and virtual channel controllers (VC), and the model defines per-hop router latency as the total delay of the functions on the critical path, i.e. router latency = $T_{AD} + T_{SEL} + T_{ARB} + T_{CB} + T_{VC}$. Through detailed gate-level design and analysis, the delay of these functions are expressed in parameterized equations which are then grounded in a 0.8 micron CMOS process. By substituting the parameters of a router into the parametric equations, a designer can obtain estimates of router latency easily, and use that as a basis of comparison.

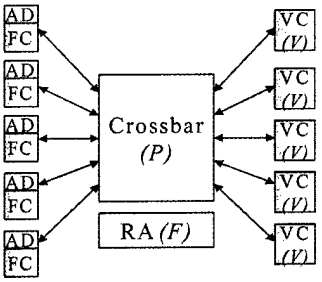

Figure 1. Canonical router architecture proposed in Chien's model. The parameters of the delay model are *P*, the number of ports on the crossbar; *F*, the number of output route choices; and *V*, the number of virtual channels per physical channel.

However, a major oversight in Chien's model is the omission of pipelining, which is present in most practical routers. Duato [5] hence proposed an extension of the model to pipelined routers. His model groups the functions in the original Chien's model into 3 pipeline stages :- T_R, the routing stage which encompasses T_{AD}, T_{SEL} and T_{ARB}; T_S, the switching stage which includes T_{FC}, T_{CB} and the delay incurred in latching a flit at the output port; T_C, the channel stage which includes T_{VC} and the internode delay. The clock cycle is then prescribed as the maximum delay of these 3 stages and per-hop router latency is thrice the clock cycle time. Both these models are based upon the premise that clock cycle time depends solely on router latency though, which is usually not the case in practical router design. Typically, router designers have to work within the limits of a clock cycle which is determined by factors beyond the router, such as the fundamental limits of chip-to-chip signalling [1], or the processor clock cycle. Hence, setting hard pipeline boundaries and fitting

1. Miller and Najjar extended Chien's model for virtual cut-through routers, modifying the parameterized delay equation for T_{FC} to include the parameter B, the number of buffers in that input queue [6].

the clock cycle around them will not work. A delay model has to advocate a good pipeline design for a router, given a particular clock cycle time.

Besides, these models abstract routers to a single canonical architecture, which result in a micro-architecture which is not optimal for certain types of routers. For instance, in the proposed canonical router architecture, passage through the crossbar is arbitrated on a per-packet basis and held throughout the duration of a packet, thus prompting the need for a huge crossbar in a virtual-channel router, with the number of ports equal to the total number of virtual channels. This contributes unnecessary delay in the crossbar arbitration and traversal functions. Buffering flits at virtual channel controllers whose arbitration delay increases with the number of virtual channels also adds needless cost to a virtual-channel router. Thus, there is a need for canonical router architectures which are tailored optimally for different flow control techniques.

2. Proposed model

In this paper, we propose a delay model which attempts to address the issues raised. It comprises a general router model which outlines a design methodology for the pipelining of a router given a clock cycle time, using the delay estimates derived by a specific router model to prescribe a pipeline design.

2.1 Canonical router architectures

First, the model proposes canonical router architectures which are tailored to each flow control technique. Figure 2 illustrates the canonical wormhole router architecture[2], where head flits enter the input controller at each port, proceed through *routing*[3], before sending their requests for desired output ports to the switch arbiter. A state machine at each input channel (*inpc_state*) dictates the state in which each packet is in: routing, switch arbitration or crossbar traversal. Upon receiving requests from all input controllers, the global *switch arbiter* consults the status of output ports as stored in the state machine for each output port (*outpc_state)*, resolves conflicts and grants free ports to requestors. Passage through the crossbar is then held for the entire duration of the packet, till the tail flit leaves and releases this hold. As shown in the figure, the parameters affecting the delay of the various modules of a wormhole

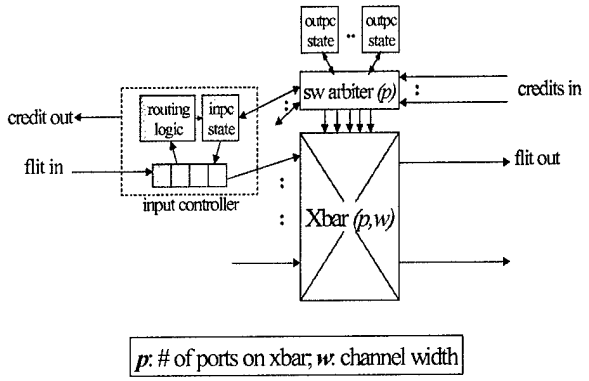

Figure 2. Canonical wormhole router architecture.

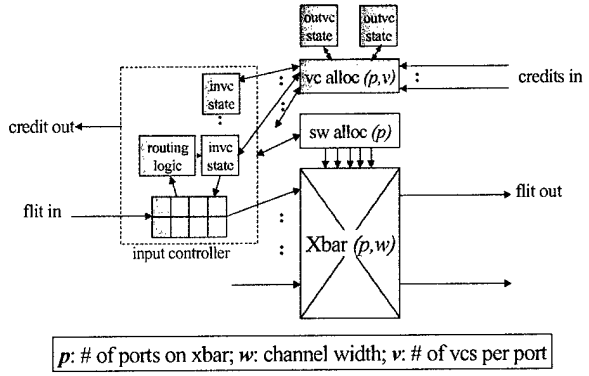

Figure 3. Canonical virtual-channel router architecture.

router are p, the number of ports on the crossbar, and w, the channel width or phit size.

Figure 3 shows the canonical router architecture proposed for virtual-channel flow control. Here, after selecting an output port through the *routing logic*, a head flit submits requests for desired virtual channels to the global *virtual channel allocator*, which consults the status of virtual channels in *outvc_state* and grants free virtual channels to requestors. Flits of a packet that has secured an output virtual channel then arbitrate for passage through the crossbar switch each cycle. Instead of reserving passage through the crossbar for the entire duration of a packet, the crossbar is apportioned to flits of different packets on a cycle-by-cycle basis. In this router architecture, the function of multiplexing virtual channels onto a physical channel rests upon the *switch allocator*, instead of virtual channel controllers. The additional parameter for virtual-channel flow control is v, the number of virtual channels per physical channel, which affects the delay of the virtual channel allocator.

2. This is similar to the canonical router architecture proposed in Chien's model.
3. Throughout the paper, our emphasis is on a comparison of flow control techniques. Hence, we will be viewing routing as a black box, and assuming decoding and routing takes a typical clock cycle of $20\tau_4$.

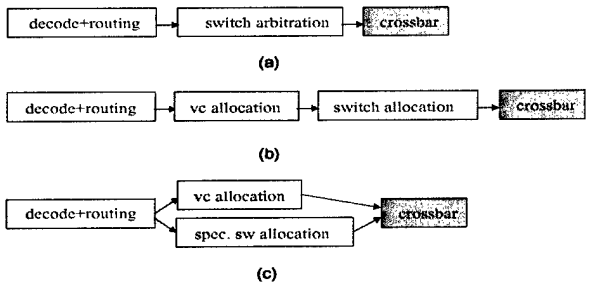

Figure 4. Atomic phases and dependences of (a) a wormhole router; (b) a virtual-channel router; (c) a speculative virtual-channel router.

2.2 Atomic modules and dependencies

The canonical router architectures prescribe the organization of the different functions of a router. Realization of these functions in hardware will uncover modules which are not amenable to pipelining. These modules are termed "*atomic modules*" in our model, and are best kept intact within a single pipeline stage. An example of an atomic module is the virtual-channel allocator in a virtual-channel router. If this module straddles multiple pipeline stages, it can result in grants not being reflected correctly before the next allocation. Besides, with a separable allocator, there are lots of wires connecting the input and output ports, which will require excessive latching should the allocator be binned in multiple pipeline stages. Figure 4 shows the various atomic modules of wormhole and virtual channel routers in our model.

The inputs of these atomic modules may depend on the outputs of another, in which case, a dependency exists. These dependencies determine the critical path of a router. Figure 4(a) and (b) shows the basic dependencies of a wormhole and virtual-channel router respectively.

Sometimes, these dependencies can be averted with speculation. For instance, the switch allocator in a virtual-channel router can speculatively assume that the packet will manage to obtain a free output virtual channel from the virtual channel allocator, and thus, proceed to request for the desired output port before it has secured an output virtual channel. Should the speculation be incorrect, the crossbar passage reserved will just be wasted. With speculation, a virtual-channel router removes the dependency between the virtual channel allocation and switch allocation phases, and cuts down on its critical path delay, as shown in Figure 4(c). However, the speculation may result in lower throughput as the crossbar switch may be allocated to packets which are unable to use it due to unavailable virtual channels. While this paper reports comparisons of just basic wormhole and virtual-channel routers, work is currently under-going to apply our model to speculative virtual-channel routers, which can potentially reduce virtual-channel router latency to that of a wormhole router.

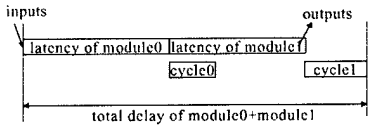

Figure 5. Latency and cycle time estimates derived by the specific router model.

2.3 Pipeline design

The delay of each atomic module is then modelled by the parametric equations derived by the specific router model (Section 2.4 on page 3) which generates two delay estimates: *latency* (t_i) and *cycle* (c_i) delay. Latency delay spans from when inputs are presented to the module, to when the outputs which are needed by the next module are stable. Cycle delay refers to the delay expended by additional circuitry required before the next set of inputs can be presented to the module. Figure 5 shows these delay components. In a switch arbiter, for example, latency spans from when requests for crossbar ports are presented to when grant signals are stable, while cycle delay refers to the delay in the circuits for updating the status of output ports, so that subsequent arbitrations do not grant output ports which have already been reserved.

Armed with t_i and c_i of each atomic phase on the critical path, the model prescribes the pipelining of the router as follows :-

Given a is the first atomic module in a pipeline stage and b is the last atomic module,

$$\sum_{i=a}^{b} t_i + c_b \leq clk \quad \text{and}$$

$$\sum_{i=a}^{b+1} t_i + c_{b+1} > clk \quad \text{and} \quad \sum_{i=a-1}^{b} t_i + c_b > clk \qquad \text{(EQ 1)}$$

2.4 Specific router model

The specific router model is responsible for the derivation of the parameterized delay equations for atomic modules. We ground our specific delay model on the theory of logical effort [7, 8] which was proposed by Sutherland, Sproull and Harris for estimating circuit delay and guiding the design of circuits with minimum delay.

The theory of logical effort prescribes that circuit delay along a path is the sum of the delay effort and the parasitic delay of that path, and delay effort of each stage is the product of logical effort and electrical effort (EQ 2). Logical effort refers to the number of times worse the gate is at

TABLE 1. Parameterized delay equations (in τ) for wormhole and virtual-channel routers.

		p=5; w=32; v=2; clk=20τ_4	
Module	Parameterized delay equations (in τ)	Model (τ_4)	Synopsys Timing Analyzer (τ_4)
Wormhole router			
Switch arbiter (swarb)	$t_{swarb}(p) = 21\frac{1}{2}\log_4 p + 14\frac{1}{12}$ $c_{swarb}(p) = 9$	9.6	9.9
Crossbar traversal (xbar)	$t_{xbar}(p,w) = 9\log_8(w\lfloor\frac{p}{2}\rfloor) + 6\lceil\log_2 p\rceil + 6$ $c_{xbar}(p,w) = 0$	8.4	10.5
Virtual-channel router			
Virtual-channel allocator (vcalloc)	$t_{vcalloc}(p,v) = 33\log_4 pv + 20\frac{5}{6}$ $c_{vcalloc}(p,v) = 9$	16.9	15.3
Switch allocator (swalloc)	$t_{swalloc}(p) = 28\log_4 p + 17\frac{1}{2}$ $c_{swalloc}(p) = 9$	11.8	11.9
Crossbar traversal (xbar)	$t_{xbar}(p,w) = 9\log_8(w\lfloor\frac{p}{2}\rfloor) + 6\lceil\log_2 p\rceil + 6$ $c_{xbar}(p,w) = 0$	8.4	10.5

delivering output current than would be an inverter with identical input capacitance, while electrical effort refers to the ratio of output capacitance to input capacitance, or fanout.

$$Delay(in\ \tau_4) = \{Delay\ Effort + Parasitic\ Delay\ (in\ \tau)\} + 5$$
$$= \{\sum g_i h_i + \sum p_i\} + 5$$

where g_i = *logical effort per stage;*
h_i = *electrical effort per stage;*
p_i = *parasitic delay;*

(EQ 2)

Through detailed gate-level design and analysis of the circuits of each atomic module[4] and the application of equation 2, *technology-independent* parameterized delay equations for each atomic module are derived and listed in Table 1. Projections of the model are validated against Synopsys timing analyzer in a 0.18 micron technology and found to be close.

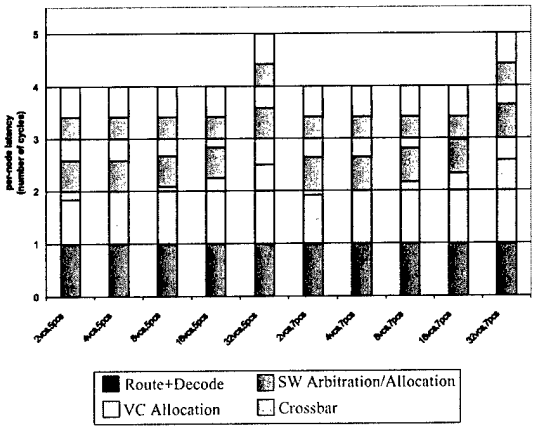

Figure 6. Effect of *p*, the number of physical channels (pcs) and *v*, the number of virtual channels (vcs) on per-node latency of virtual-channel routers. A typical clock cycle of 20 τ_4 was assumed. Each bar illustrates the router pipeline, with the shaded regions corresponding to the fraction of clock cycle time used by each atomic module.

3. Insights from model

Figure 6 details the effect of differing numbers of physical and virtual channels on the per-node latency of a virtual-channel router, with a typical clock cycle of 20 τ_4[5]. For a 2-dimensional virtual-channel router with 5 physical channels, 4 pipeline stages are sufficient for up to 16 virtual channels per physical channel. Thus, overall per-node latency does not increase as the number of virtual channels increases from 2 to 16. With a 3-dimensional network, a virtual-channel router with 7 physical channels will incur the same per-node latency of 4 cycles (due to its 4 pipeline stages) for up to 16 virtual channels per physical channel too.

These model projections indicate that a virtual-channel router typically requires just one more pipeline stage than a wormhole router[6]. Also, with most practical numbers of virtual channels used to date, the delay of a virtual-channel router remains unchanged.

4. Simulation results

Based on the pipelined designs prescribed by the model, detailed Verilog code for wormhole and virtual channel routers were crafted and simulations carried out to determine their latency-throughput characteristics. A typical

4. Due to space constraint, detailed gate-level designs of the routers cannot be presented. In brief, separable allocators are used, and the arbiter adopted is the matrix arbiter.

5. τ_4 refers to the delay of an inverter driving 4 other inverters [4].

6. If the virtual-channel router is speculative, the per-hop latency can potentially be the same as that of the wormhole router.

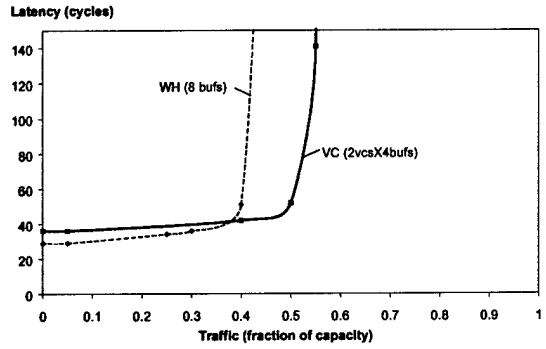

Figure 7. Latency-throughput curves of wormhole and virtual-channel routers with 8 buffers per input port. The routers adhere to the proposed pipelined router model.

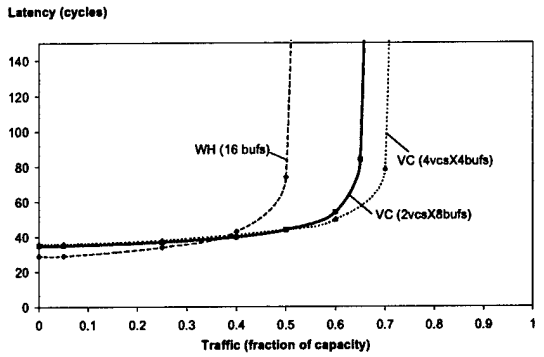

Figure 8. Latency-throughput curves of wormhole and virtual-channel routers with 16 buffers per input port. The routers are pipelined as prescribed by the proposed pipelined router model.

clock cycle of 20 τ_4 was assumed. The simulator generates uniformly distributed traffic across a 8-by-8 mesh network to random destinations. Each simulation is run for a warm-up phase of 10,000 cycles. Thereafter, 100,000 packets are injected and the simulation is run till these packets in the sample space have all been received. A constant rate source injects 5-flit packets at a percentage of the capacity of the network and the average latency of the packets is calculated. Latency spans the instant when the first flit of the packet is created, to the time when its last flit is ejected at the destination node, including source queuing time and assuming immediate ejection. Each router uses credit-based flow control to regulate the use of buffers, and propagation delay across the channel is assumed to take a single cycle. Since the purpose of our simulations is to explore the performance of flow control strategies, we chose simple dimension-ordered routing.

4.1 Performance of wormhole vs. virtual-channel routers

Figure 7 shows the latency-throughput curves of a wormhole router and a virtual-channel router with 8 buffers per input port. The base latency of the wormhole router (29 cycles) is lower than that of the basic virtual-channel router (36 cycles), since the wormhole router uses a 3-stage pipeline while the virtual-channel router is pipelined over 4 stages at each hop. However, virtual-channel flow control with 2 virtual channels extends the throughput achieved by wormhole flow control by 25% from 40% capacity to 50% capacity.

With 16 buffers per input port, similar effects were observed, as shown in Figure 8. Again, a basic virtual-channel router has a higher base latency (35 cycles) as compared to that of a wormhole router, due to the additional pipeline stage per hop. However, the advancement in throughput is large, as a virtual-channel router with 2 virtual channels enjoys a throughput of 65% capacity, a 30% improvement over that achieved by the wormhole router (50% capacity). With 4 virtual channels, throughput is further extended to 70% capacity, a 40% improvement over wormhole flow control.

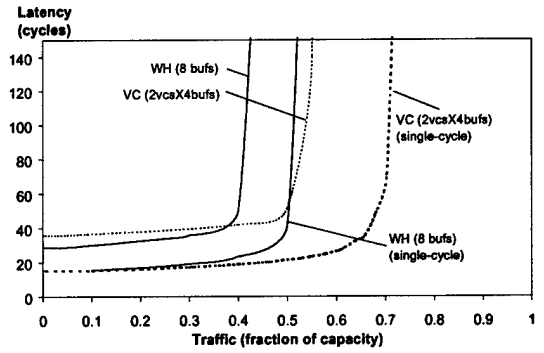

Figure 9. Performance of wormhole and virtual channel routers, as modelled by the proposed pipelined delay model, and as modelled assuming a single-cycle router delay. (8 buffers per input port)

4.2 Effect of assuming single-cycle router latency vs. multiple-cycle pipelined design

Most published research compares the performance of different router designs assuming a single-cycle router latency, without taking into account implementation complexity and cost. To quantify this effect, we ran simulations with a cycle-accurate C simulator which assumes single-

cycle router latency for both wormhole and virtual-channel flow control. All other experimental parameters are identical to that of the Verilog simulator.

As shown in Figure 9, assuming single-cycle router latency results in both wormhole and virtual-channel routers incurring the same low base latency of 16 cycles, whereas simulations which adhere to our proposed multiple-cycle pipeline design highlight the higher base latency incurred by virtual-channel flow control due to its longer pipeline.

It is also apparent that assuming that a single-cycle router latency results in inflated throughput figures. This is because throughput in wormhole and virtual-channel flow control is strongly influenced by buffer utilization, which depends on how quickly credits can be sent and received, prompting the re-use of buffers. In the simulations assuming a single-cycle routing latency, a credit can be sent and received in 2 cycles, while in our pipelined model, a wormhole router needs 4 cycles to turnaround credits, and a virtual-channel router needs 5 cycles. Thus, throughput is lower in the pipelined model than in one which ignores implementation delay.

5. Conclusions

We have presented a router delay model which accurately accounts for pipelining and models routers with canonical micro-architectures which are tailored to its flow control technique. Verilog simulations based on the model show modest additional router latency with virtual-channel flow control which is more than offset by its improvements in throughput over wormhole flow control. Work is currently on-going to investigate speculative virtual-channel flow control, whch can potentially reduce the router latency experienced by a virtual-channel router to that of a wormhole router.

Also, when compared with simulations ignoring pipeline delay and assuming unit router latency, significant discrepancies in base latency and throughput were observed, supporting the importance of considering implementation costs when simulating router performance.

References

[1] Kevin Bolding et. al., "The Chaos Router Chip: Design and Implementation of an Adaptive Router", In *Proceedings of IFIP Conference on VLSI*, September 1993.

[2] Andrew A. Chien, "A Cost and Speed Model for k-ary n-cube Wormhole Routers", In *Proceedings of Hot Interconnects*, Palo Alto, August 1993.

[3] Andrew A. Chien, "A Cost and Speed Model for k-ary n-cube Wormhole Routers", *IEEE Transactions of Parallel and Distributed Systems*, vol. 9, no. 2, February 1998.

[4] William J. Dally and J. W. Poulton, "Digital Systems Engineering", *Cambridge University Press*, 1998.

[5] Jose Duato and Pedro Lopez, "Performance Evaluation of Adaptive Routing Algorithms for k-ary n-cubes", In *Proceedings of Parallel Computer Routing and Communication Workshop*, pp. 45-59, May 1994.

[6] D. R. Miller and W. A. Najjar, "Empirical Evaluation of Deterministic and Adaptive Routing with Constant-Area Routers", In *Proceedings of International Conference on Parallel Architectures and Compilation Techniques*, San Francisco, November 1997.

[7] R. F. Sproull and I. E. Sutherland, "Logical Effort: Designing for Speed on the Back of an Envelope", *IEEE Advanced Research in VLSI*, C. Sequin (editor), MIT Press, 1991.

[8] Ivan Sutherland, Bob Sproull and David Harris, "Logical Effort: Designing Fast CMOS Circuits", *Morgan Kaufman Publishers*, 1999.