Design and Analysis of an NoC Architecture from Performance, Reliability and Energy Perspective [*]

Jongman Kim, Dongkook Park, Chrysostomos Nicopoulos, N. Vijaykrishnan, C. R. Das
Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802
{jmkim, dpark, nicopoul, vijay, das}@cse.psu.edu

ABSTRACT

Network-on-Chip (NoC) architectures employing packet-based communication are being increasingly adopted in System-on-Chip (SoC) designs. In addition to providing high performance, the fault-tolerance and reliability of these networks is becoming a critical issue due to several artifacts of deep sub-micron technologies. Consequently, it is important for a designer to have access to fast methods for evaluating the performance, reliability, and energy-efficiency of an on-chip network. Towards this end, first, we propose a novel path-sensitive router architecture for low-latency applications. Next, we present a queuing-theory-based model for evaluating the performance and energy behavior of on-chip networks. Then the model is used to demonstrate the effectiveness of our proposed router. The performance (average latency) and energy consumption results from the analytical model are validated with those obtained from a cycle-accurate simulator. Finally, we explore error detection and correction mechanisms that provide different energy-reliability- performance tradeoffs and extend our model to evaluate the on-chip network in the presence of these error protection schemes. Our reliability exploration culminates with the introduction of an array of transient fault protection techniques, both architectural and algorithmic, to tackle reliability issues within the router's individual hardware components. We propose a complete solution safeguarding against both the traditional link faults and internal router upsets, without incurring any significant latency, area and power overhead.

Categories and Subject Descriptors: B.4[I/O and Data Communications] Interconnections(Subsystems):B.8[Performance and Reliability] Performance Analysis and Design Aids
General Terms: Design, Performance.
Keywords: Networks-On-Chip, Adaptive Routing, Reliability.

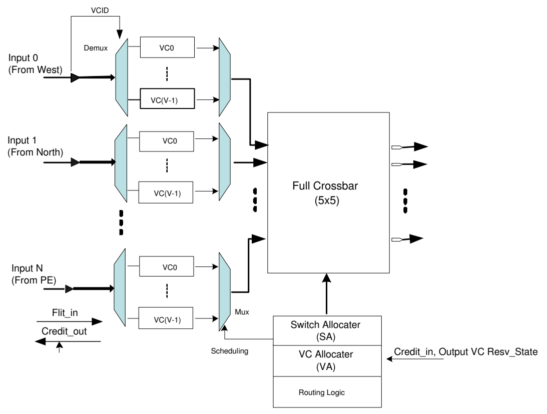
1. INTRODUCTION

On-chip interconnects, also known as network-on-chip (NoC) architectures, are expected to play a crucial role in designing complex system-on-chip (SoC) architectures because of the increasing wiring delay with reduced feature size in deep sub-micron technology [5, 8, 9, 15]. Unlike the traditional off-chip interconnects, NoC architectures pose complex design challenges to meet the performance, reliability and power constraints. This is primarily because of the area and power constraints in exploring the design space. Furthermore, the vulnerability of the NoCs to several types of error such as crosstalk, electromagnetic interference (EMI), and radiation induced soft errors [5, 26] makes reliable communication even more difficult. Many commercially available SoCs currently use a shared bus architecture for connecting the functional units [14, 31]. Since a shared bus architecture is not suitable from the scalability standpoint, recent NoC designs have proposed using switch-based networks such as 2-D mesh, and torus [1, 4, 5, 9, 13, 16, 22, 23, 34, 35]. However, NoC design is believed to be in its infancy since there are no conclusive answers to many design issues.

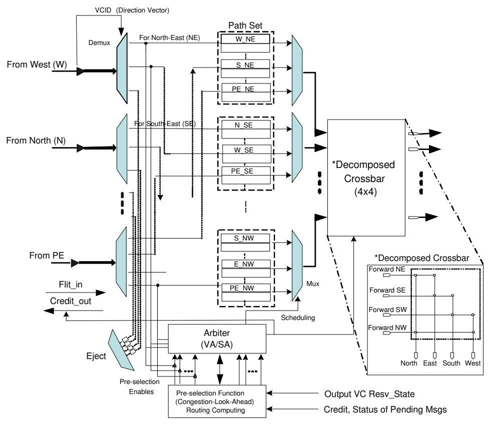
While researchers have examined the area-constraint design alternatives [17, 25], energy models [28] or fault-tolerance issues [3, 6, 11, 21, 24, 29] individually, a systematic design methodology encompassing the interplay of performance, fault-tolerance and energy constraints is yet to evolve. Such a design methodology is impeded in part due to the lack of efficient techniques to quickly explore alternatives from a large design space. Towards this end, we present a queuing-theory-based tool quantifying the performance and energy behavior of on-chip networks. While most prior on-chip interconnect analyses are based on time consuming simulation models, in order to provide fast performance estimates during the design cycle, we have developed a queuing-theory-based model for quantifying the performance and energy behavior of on-chip networks.

Although wormhole switched, traditional off-chip networks have been analyzed extensively in the literature [27, 19], analytical models for NoCs considering detailed architectural artifacts are almost nonexistent. To our knowledge, the first analytic techniques [12] appeared recently. Our model is different from [12] in that we compute the average delay due to path contention, virtual channel and crossbar switch arbitration using a queuing theory approach, which we believe, can capture the blocking phenomena of wormhole switching quite accurately. We first present the model for performance analysis for a generic wormhole switched router. The model is then used to estimate the power consumption by estimating the utilization of the router components and multiplying them with component level power profiles, obtained from actual circuit-level synthesis. Comparison with simulation results indicate that the proposed analytical model is quite accurate and can be used as an efficient design tool. An extension of the proposed analytical model is also shown to demonstrate the utility of the model for

[*]This research was supported in part by NSF grants CCR-0208734, CCF-0429631, CNS-0509251, EIA-0202007, a grant from DARPA/MARCO GSRC, and CAREER 0093085.



(a) A 2-Stage Virtual Channel Look-ahead Router



(b) Proposed Path-Sensitive Router

Figure 1: On-chip Router Architecture

fault-tolerance study.

We also demonstrate the applicability of the proposed model to evaluate our novel low latency on-chip router (originally proposed in our prior work [20] and augmented with fault resilience mechanisms in this work). Our two-stage router architecture, called *path-sensitive router*, utilizes look-ahead routing in selecting the next route. The router is called path-sensitive because based on the destination address, it selects one of the four possible quadrants (NE, NW, SE, and SW) and routes the packet to the corresponding VCs, assigned for that quadrant. Thus, unlike the prior router designs, the VCs are partitioned into four groups to facilitate efficient packet routing. Furthermore, based on this partitioned VCs, we use a decomposed crossbar that needs only half the size (connects) of a full crossbar, thereby reducing packet conflict probability in the crossbar. The router can support both deterministic and adaptive routing. In addition to the reliability augmentation, our router model in this work improves on our prior work [20] through additional performance analysis and the synthesis of the router using a 90nm TSMC cell library. The synthesis results are used to analyze its feasibility for on-chip interconnects and extract the timing parameters for different components for performance analysis. Evaluation of our architecture in a 2-D mesh using various traffic patterns shows that it can provide better performance (lower latency) and energy conservation compared to a traditional 2-stage lookahead router. Our transient fault resilience enhancement covers both link errors and logic (component) errors in a router. For the link errors, we analyze five possible retransmission mechanisms, and argue that a separate header error checking based retransmission is a better choice than the conventional End-to-End (E2E) and Hop-by-Hop (HBH) techniques because packet misrouting can be alleviated by identifying header bit errors. Then, we propose several architectural and algorithmic safeguards to our two-stage router architecture to protect against six types of intra-router soft faults without inducing prohibitive area, power and latency overheads. To the best of our knowledge, this is the first attempt to account for and protect against internal router soft faults in on-chip net works. The effectiveness of these measures is illustrated through a series of cycle-accurate simulations.

The paper is organized as follows. The path sensitive router architecture is described in Section 2. Section 3 presents our analytical model for latency and power consumption analysis. Section 4 describes the reliability issues resulting from link errors and transient faults within the router components and proposes several protection methods. The conclusions of this study are drawn in the last section.

2. ROUTER DESIGN

Typically, wormhole routing with virtual channel flow control is used for providing high performance, with minimum buffer requirement in a pipelined router. A general structure of a two-stage virtual channel router with its major components is depicted in Figure 1(a). Such routers have *N+1* input/output ports, including the connection with the local Processing Element (PE), and support V Virtual Channels (VC) at each input port. The Virtual Channel Allocator (VA) determines the VC assignment on a packet-by-packet basis, while the Switch Allocator (SA) advances on a flit-by-flit basis. Link contention at the VA and switch contention at the SA can account for a significant part of the overall packet latency. Thus, the contention arbitration policy directly affects the switch performance.

The proposed path-sensitive router employs look-ahead routing and speculative allocation [10]. Using traffic information from neighboring routers and the current switch state, the router preselects a direction for the next hop. This task is handled by a novel pre-selection unit within the first pipeline stage of the router, which works one cycle ahead of the flit arrival. Furthermore, look-ahead information allows a flit destined for the local PE to be ejected after the DEMUX instead of traversing the whole router logic. This early ejection saves two cycles at the destination node by avoiding the switch allocation and switch traversal stages.

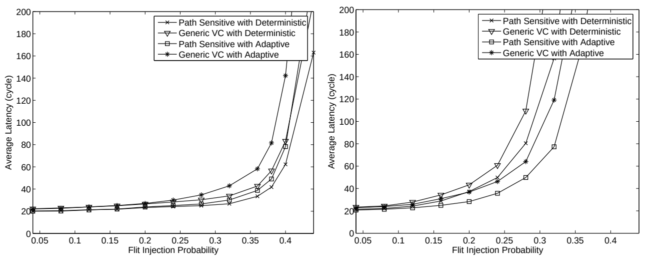
The detailed design of the path-sensitive router is shown in Figure 1(b). Path sensitivity ensures that a flit will traverse the NE quadrant only if it is coming from the west, south or the PE itself. Similarly, it will traverse the NW quadrant if the entry point is from the south, east or the local PE. Thus, based on this grouping, we have 3VCs per PC. It is possible to provide more VCs per group if there is adequate on-chip buffer.

This router is also novel in its use of a 4×4 decomposed crossbar with half the connections of a full crossbar. This architecture is facilitated by the use of topology-tailored routing. Usually a 5×5 crossbar is used for 2-D networks with one of the ports assigned to the local PE. In our architecture, a flit destined for the local PE does not traverse the crossbar, as explained before. This provides a significant advantage for nearest-neighbor traffic, and can take advantage of NoC mapping which places frequently communicating PEs close to each other [18]. Furthermore, because of the smaller number of connections in the decomposed crossbar, the output contention probability is reduced. Moreover, the decomposed crossbar offers two advantages for on-chip design. It occupies less area and it consumes less energy compared to a full crossbar.

A cycle-accurate simulator was used to analyze the performance under each aforementioned error model. For the simulation, we used an 8×8 MESH topology. The source and destination nodes were randomly chosen and each message consisted of 4 flits, each 128 bits long. 100,000 messages were injected, 20,000 of which were warm-up messages. For comparison purposes, we tested our proposed router scheme with both minimal adaptive and deterministic routing algorithms with a variety of traffic patterns. Our architecture clearly outperforms the generic one with both uniform and non-uniform traffic, as shown in Figure 2.

2.1 Hardware Implementation

The router architecture was implemented in structural Register-Transfer Level (RTL) Verilog and then synthesized in Synopsys Design Compiler using TSMC 90 nm cell library. The resulting design operates at supply voltage of 1 V and a clock speed of 200 MHz. Both dynamic and leakage power estimates were extracted from the synthesized router implementation, using a 50% switching activity. These power numbers were then imported into our cycle-accurate network simulator and used to accurately portray the power profile of the entire on-chip network. While some researchers have modeled power consumption based solely on hop traversals per flit/packet [24], we also account for individual router component utilizations to precisely estimate the dynamic and leakage envelope of each router throughout the simulation. This allows for a more realistic estimation when handling different flit types. For example, a header flit requires processing by the routing, preselection, and virtual channel allocation units in the router, while a data or tail flit do not. This attribute affects the different component utilizations during the simulation, and, hence, the power consumption, and is accurately reflected in our power estimation model.



(a) Uniform Traffic (b) Self-Similar Traffic

Figure 2: Performance Comparison

3. PERFORMANCE ANALYSIS OF NOC INTERCONNECTS

In this section, we present an analytical model for computing the average latency in a 2-D mesh network. The average network latency consists of two parts. The first part is the actual message transfer time. The second part is due to blocking time, which is mostly caused by the conflicts at the VA, contention at the SA and limited buffer size. The actual transmission time with a P-stage pipelined router is $(P-1+M)$ cycles for an M-flit packet. In order to compute the second part of the network latency, let us define B_j as the average blocking length (in number of flits) seen by each incoming flit at the input and arbitration stages of a router j. B_j captures flit blocking in a pipelined virtual-channel router. Then, the effective length of the packet becomes $(M + B_j)$ flits. Let W_j be the average waiting time in the router queue for the flit. Thus, the network latency for a single router (T_j) is

$$T_j = (M + B_j)W_j + P - 1. \tag{1}$$

Let T_{link} be the delay through the link connecting adjacent routers. We compute the the latency (T) for an M-flit packet to traverse through a P-stage pipelined router for a given $path$ as

$$T = \sum_{j \in path} ((B_j W_j + P) + T_{link\_j}) + ((B_{dest} + M)W_{dest} + P - 1). \tag{2}$$

The first term in Equation 2 represents the time spent at intermediate hops, and the second term denotes the time at the ejection node. Note that this does not include the queuing delay outside the router. For simplicity, using the average blocking length (B), the average waiting time (W) and the average distance (H hops) on a given $path$, and assuming T_{link} is single cycle, the total latency can be approximated to

$$T \approx (BW + P + 1)H + ((B_{dest} + M)W_{dest} + P - 1). \tag{3}$$

Now we will derive the blocking length (B and B_{dest}) and waiting time (W and W_{dest}) in terms of the contention probability (P_{con}) and the buffer-full probability (P_{block}), so that we can determine the total latency.

3.1 Contention Probability

The router model assumes an $(N + 1) \times (N + 1)$ router with V virtual channels and a deterministic routing algorithm. Our network model is characterized on a flit basis, which is appropriate for virtual channel router architectures. We assume that flit arrivals at the N inputs from neighbors and at the local input from the PE are governed by independent and identical Poisson processes. Let the probability of a flit arriving at an input virtual queue per cycle be P_c. Note that it is the normalized arrival rate [2]. Let the flit injection probability into each virtual channel queue in any given cycle from a PE be P_{pe}. The flits are assumed to travel H hops on the average. Let the incoming flits have equal probability $1/N$ of being addressed to any given output. Thus, we can compute the P_c from P_{pe} as follows:

$$P_c = \frac{P_{pe} H}{N} \tag{4}$$

The total contention probability (P_{con}) consists of the VA conflict probability (P_{con_va}) and the SA contention probability (P_{con_sa}). They are independent of each other in a speculative virtual channel router, and thus P_{con} is given by

$$P_{con} = 1 - (1 - P_{con\_va})(1 - P_{con\_sa}). \tag{5}$$

First, we analyze the VA conflict (P_{con_va}), which has two parts, P_{ready_busy} and P_{con_idle}. The first part is the probability (P_{ready_busy}) that the candidate output VCs are already used by other packets. P_{ready_busy} is the probability there is at least one header flit destined to a free output VC, multiplied by the effective packet length, since the VC will be reserved for the entire packet transmission. Thus, P_{ready_busy} in a given cycle is given by

$$P_{ready\_busy} = P'_h(M + B)W. \tag{6}$$

We derive P_{ready_busy} and P_{con_idle} as a function of the probability of a header flit arriving in an arbitrary time slot (P_h), which is calculated as P_c/M. When k header flits are destined to a particular output VC, one of the k header flits is granted an output VC, while the rest $k - 1$ headers are blocked at the respective input queues. These $k-1$ blocked headers can be regarded as independent header flit arrivals, and thus the average probability of an incoming header flit at each VC queue can be approximated to P'_h as follows:

$$P'_h = P_h(1 + P_{con\_va} + P^2_{con\_va} + ...) = \frac{P_h}{1 - P_{con\_va}}. \tag{7}$$

Next, we compute P_{con_idle}. In the steady state, $(1 - P_{ready\_busy})$ represents the probability that a particular output VC is idle and available. The probability that a header flit at the head of a queue can be routed to a particular VC is $\frac{P'_h(1-P_{ready\_busy})}{NV}$. Thus, we can compute the probability, $P'_h(j)$ such that j headers out of NV buffer are addressed to a free output VC, as

$$P'_h(j) = \begin{pmatrix} NV \\ j \end{pmatrix} \left(\frac{P'_h(1 - P_{ready\_busy})}{NV} \right)^j \cdot \left(1 - \frac{P'_h(1 - P_{ready\_busy})}{NV} \right)^{NV-j}. \qquad (8)$$

The contention probability below (P_{con_idle}) can be calculated using Equation 8. The first term in Equation 9 represents the probability that more than two headers ($j >= 2$) request a particular output when there is no header from the local PE toward that output VC. In this case, the probability that a header fails to be granted a VC is $(j-1)/j$, since $(j-1)$ headers among j headers should wait for the next arbitration. The second term denotes the probability that a header from the local PE is addressed to the same output VC. We consider the incoming effective header probability (P'_{peh}) at the injection link from the PE as $\frac{P_{pe}/M}{1-P_{con\_va}}$. Thus, the contention probability, P_{con_idle} can be estimated as

$$P_{con\_idle} = \sum_{j=2}^{NV} \frac{j-1}{j} \Big\{ P'_h(j) \left(1 - \frac{P'_{peh}(1 - P_{ready\_busy})}{NV} \right) + P'_h(j-1) \left(\frac{P'_{peh}(1 - P_{ready\_busy})}{NV} \right) \Big\}. \qquad (9)$$

Thus, we have P_{con_va} as a combination of P_{ready_busy} and P_{con_idle}, which are recursively correlated. In the steady state, the VA conflict probability can be evaluated as

$$P_{con\_va} = P_{ready\_busy} + P_{con\_idle}. \qquad (10)$$

Another contention exists at the SA module. The contention probability, P_{con_sa} consists of two parts. One is the input port contention probability ($P_{con\_sa\_in}$), and the other is the output port contention probability ($P_{con\_sa\_out}$). We can compute the SA contention probability as follows:

$$P_{con\_sa} = 1 - (1 - P_{con\_sa\_in})(1 - P_{con\_sa\_out}). \qquad (11)$$

$P_{con\_sa\_in}$ results from the sharing of an input port by V virtual channels at a V input arbitration, which is given by

$$P_{con\_sa\_in} = \sum_{i=2}^{V} \frac{i-1}{i} \begin{pmatrix} V \\ i \end{pmatrix} P_c'^{i} (1 - P'_c)^{V-i}. \qquad (12)$$

The output port contention probability is analyzed similar to P_{con_idle}, but it is different in that the allocation process is made on every flit including the data flits. While the switch is held throughout the entire duration of a packet in a conventional wormhole router, individual flits arbitrate for access to the crossbar on a cycle-by-cycle basis. In order to compute the output port contention probability, $P_{con\_sa\_out}$, let us define the incoming flit probability at the input of a physical channel, P_p, as $(1 - (1 - P'_c)^V)$, where $P'_c = \frac{P_c}{1-P_{con}}$. The effective physical channel utilization, P'_p, is also recursively computed as $P'_p = P_p/(1 - P_{con\_sa})$. Similar to Equation 8, the probability that k flits compete for a particular output port, assuming that each flit has an equal probability $1/N$ of being destined to any output, is

$$P'_p(k) = \begin{pmatrix} N \\ k \end{pmatrix} \left(\frac{P'_p}{N} \right)^k \left(1 - \frac{P'_p}{N} \right)^{N-k}. \qquad (13)$$

Similar to Equation 9, the first term in Equation 14 indicates the probability that more than 2 flits ($k >= 2$) are destined to a particular output with no flit coming from the local PE. The second part takes into account the probability of an arriving flit from the local PE, P'_{pe}. The $k - 1/k$ term indicates that only one of the competing flits is granted the output port, while the rest have to be included in the contention probability calculation. Thus, the output port contention probability is

$$P_{con\_sa\_out} = \sum_{k=2}^{N} \frac{k-1}{k} \Big\{ P'_p(k) \left(1 - \frac{P'_{pe}}{N} \right) + P'_p(k-1) \frac{P'_{pe}}{N} \Big\}. \qquad (14)$$

By integrating the virtual channel conflict probability (P_{con_va}) and the switch allocation contention probability (P_{con_sa}), we can determine the total contention probability, P_{con} as a function of P_c.

3.2 Modeling the Finite Size Buffer in NoCs

We should also consider the probability that a buffer becomes full. The buffer unavailability probability (P_{block}) can be estimated from the average queuing length (B) and the traffic intensity (ρ), which are estimated by using a discrete-time state transition diagram as shown in Figure 3. From Figure 3, we can also obtain the

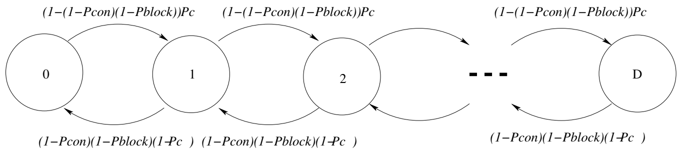


Figure 3: State Transition Diagram for a Finite Buffer

probability $P_{hold}(r)$ that r flits are waiting in a queue as

$$P_{hold}(r) = (1 - \rho)\rho^{r+1}, \qquad (15)$$

where the traffic intensity ρ can be simplified as

$$\rho = \frac{(1 - (1 - P_{con})(1 - P_{block}))P_c}{(1 - P_{con})(1 - P_{block})(1 - P_c)}. \qquad (16)$$

As shown in Figure 3, $P_{hold}(r)$ is related to P_{block} and is recursively obtained by computing P_{block}. For a D-flit buffer depth,

$$P_{block} = P_{hold}(r >= D) \qquad (17)$$

We now obtain the average number of flits in the queue, B, as

$$B = \sum_{r=1}^{D} r P_{hold}(r). \qquad (18)$$

The average waiting time can be estimated from the steady-state traffic intensity, ρ, under uniform traffic density as follows:

$$W = \frac{1}{(1 - P_{con})(1 - P_{block})(1 - P_c)} \frac{\rho}{1 - \rho}. \qquad (19)$$

In order to estimate the blocking length (B_{dest}) and the waiting time (W_{dest}) per flit at the input of the destination node, we can consider only the contention probability without considering the buffer full probability at the next node. Thus, Equations 16, 15, 18 and 19 can be changed to

$$\rho_{dest} = \frac{P_{con} P_c}{(1-P_{con})(1-P_c)} \qquad P_{hold\_dest}(r) = (1 - \rho_{dest})\rho_{dest}^{r+1}$$
$$B_{dest} = \textstyle\sum_{r=1}^{D} r P_{hold\_dest}(r) \qquad W_{dest} = \frac{1}{(1-P_{con})(1-P_c)} \frac{\rho_{dest}}{1-\rho_{dest}}$$

Now, we have all the parameters to estimate the average latency using Equation 3.

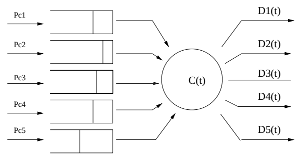
3.3 Modeling of the Path-Sensitive Router

We now extend the model to analyze our path-sensitive router. In a generic architecture, flits arriving with the probabilities P_{c1} through P_{c5} are enqueued at the respective input ports as shown in Figure 4(a), whereas in the path-sensitive router, the inputs are P_α, P_β, P_γ and P_δ (Figure 4(b)). Under the same traffic condition, the total incoming traffic per router ($\sum_{i=1}^{5} P_{ci}$) is larger than the traffic ($\sum_{j=\alpha}^{\delta} P_j$) in the path-sensitive model by flit ejection ($\cong$ injection) probability (P_{pe}). The lower arrival probability due to early ejection, influences both average contention probability and blocking length in higher workload. On the other hand, in lower workload, the average latency is dominated by the critical path in the pipeline stage. The flits traverse $H-1$ hops on an average in a path-sensitive model compared to a generic architecture with H hops. Thus, H in Equation 3 is replaced by $H-1$, and Equation 3 is changed to

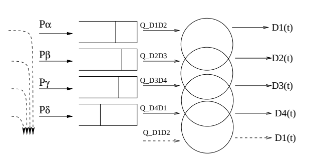
$$T_{ps} \approx (B_{ps}W_{ps}+P+1)(H-1)+1+((B_{ps\_dest}+M)W_{ps\_dest}+P-1) \quad (20)$$

The average blocking length (B_{ps}) and waiting time (W_{ps}) per flit are modified from the generic B and W, because of the changes in the VA contention probability (P_{con_idle}) and the SA output port contention probability ($P_{con\_sa\_out}$). The number of competing inputs for a particular output is reduced by half ($N+1$) to ($\lfloor (N+1)/2 \rfloor$), since our path-sensitive architecture sends incoming flits to a path candidate queue via the DEMUX according to the routing algorithm and direction vector ID as shown in Figure 4(b). Thus, we can rewrite the probability ($P'_h(j)$) in Equation 8 that j headers are destined to an unused output VC as follows:

$$P'_{ps\_h}(j) = \binom{\lfloor (N+1)/2 \rfloor V}{j} \left(\frac{P'_h(1-P_{ready\_busy})}{(N-1)V}\right)^{j} \left(1-\frac{P'_h(1-P_{ready\_busy})}{(N-1)V}\right)^{(\lfloor (N+1)/2 \rfloor V-j)} \quad (21)$$



(a) A Generic Router Queuing System



(b) The Path-Sensitive Queuing Model

Figure 4: Queuing Systems of the Generic and the Path-Sensitive Models

There is no separate injection input port to access the crossbar, and injected flits are evenly spread out across other input ports. Note that the amount of ejection is the same as that of injection on average. Thus, we can remove the effect of the incoming flits at the injection link from the PE in Equation 9. Thus, $P_{ps\_con\_idle}$ is given by

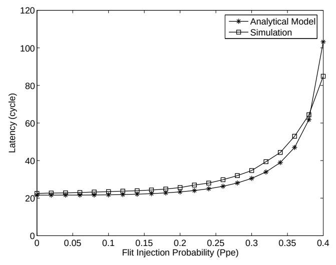
$$P_{ps\_con\_idle} = \sum_{j=2}^{\lfloor (N+1)/2 \rfloor V} \frac{j-1}{j} \Big\{ P'_{ps\_h}(j) \Big\}. \quad (22)$$

Similarly, Equations 13 and 14 are modified as follows, for the path-sensitive router.

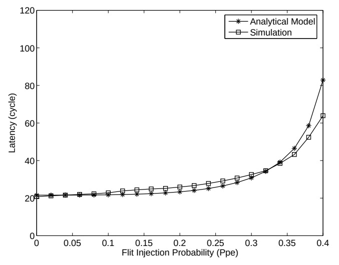
$$P'_{ps\_p}(k) = \binom{\lfloor (N+1)/2 \rfloor}{k} \left(\frac{P'_p}{(N-1)}\right)^{k} \left(1-\frac{P'_p}{(N-1)}\right)^{(\lfloor (N+1)/2 \rfloor -k)}. \quad (23)$$

$$P_{ps\_con\_sa\_out} = \sum_{k=2}^{\lfloor (N+1)/2 \rfloor} \left(\frac{k-1}{k}\right) P'_{ps\_p}(k). \quad (24)$$

The number of competing input queues for a particular output is reduced to $\lfloor (N+1)/2 \rfloor$, while the flit contention probability in each input queue for a particular output slightly increases by $\frac{N}{N-1}$ in the example of Figure 4(b). Thus, we have an overall lower contention probability, which results in lower average queuing length (B_{ps}) and waiting time (W_{ps}). Figure 5 shows that the path-sensitive router outperforms the generic virtual channel one. The result depicts the comparison between our analytical model and a cycle-accurate simulator in terms of average packet latency for a deterministic routing algorithm. As we described in section 3.2, the model is based on an 8x8 mesh network with 4-flit packets, 3VCs/Physical Channel and a 4-flit buffer depth. It is clear that the simulation and analytical values are in close agreement validating the correctness and accuracy of our mathematical model.



(a) Virtual Channel Router



(b) Path-Sensitive Router

Figure 5: Performance Comparison in Analytical Model and Simulation

3.4 Power Model

We now extend our analytical model to include power calculations at the resolution of individual router components. For this, we will use the buffering delay and contention probability parameters from sections 3.1 and 3.2. The power dissipated in an NoC can be decomposed as follows:

$$P_R = \sum_{r \in PE} \left(P_{r\_buf} + P_{r\_arbiter} + P_{r\_crossbar} + P_{r\_link}\right), \quad (25)$$

where P_{r_buf} is the average buffer power consumption including both dynamic and static power, $P_{r\_arbiter}$ is the average power consumption in the routing computation, VA and SA modules, $P_{r\_crossbar}$ is the average crossbar traversal power consumption, and P_{r_link} is the average link power consumption between neighboring routers. P_{r_buf} can be estimated using the average flit arrival probability (P_c) in Equation 4 at each VC and average queue length (B) in Equation 18 as

$$P_{r\_buf} = (P_c(P_{wrt\_flit}+P_{rd\_flit})+BP_{leak\_flit})(N+1)V, \quad (26)$$

where P_{wrt_flit}, P_{rd_flit} and P_{leak_flit} are the power consumptions for the read and write operations per flit, and the leakage power dissipation per flit. These power numbers are obtained from the synthesized design, as explained in section 2.2. To estimate the crossbar and link power consumption, we use the flit arrival probability (P_p) at a single physical link, which is computed as $(1-(1-P'_c)^V)$ in section 3.1. $P_{crossbar\_flit}$ is the power dissipation of a flit traversal through each output port, and P_{link_flit} is flit power consumption per link. In a 2D mesh, the router at each edge has $N-1$ links and the four routers at the corners have $N-2$ links. Otherwise, they have maximum N connections per router. Similarly, the number of crossbar output ports varies from $N-1$ to $N+1$ including the ejection channel. Thus, $P_{r\_crossbar}$ and P_{r_link} are given by

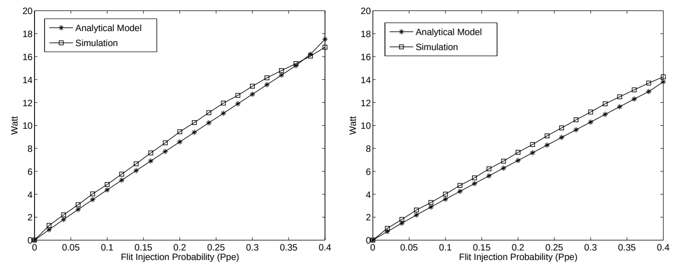
$$P_{r\_crossbar} = P_p P_{crossbar\_flit} Max(N+1, N, N-1), and \quad (27)$$

$$P_{r\_link} = P_p P_{link\_flit} Max(N, N-1, N-2). \quad (28)$$

For computing $P_{r\_arbiter}$, Equation 29, we consider the probability of three separate activities and the power consumption for each activity. The first term is the probability that there is at least one header flit arrival (P'_h) to one of the (N+1)V virtual channels and the corresponding power consumption $P_{r\_arb\_va}$. The second term represents the probability, that there is at least one data flit that needs SA operation (P'_p) and the corresponding power consumption, $P_{r\_arb\_sa}$. The last term is, the probability of a header arrival (P_h) to one of the VCs and the corresponding power consumption ($P_{r\_rt\_hflit}$). Thus,

$$P_{r\_arbiter} = (1-(1-P'_h)^{(N+1)V})P_{r\_arb\_va} + (1-(1-P'_p)^{(N+1)})P_{r\_arb\_sa} + P_h P_{r\_rt\_hflit}(N+1)V. \quad (29)$$

The three elements (P'_h, P'_p and P_h) are combined to compute the total power consumption. The three power consumption parameters ($P_{r\_arb\_va}$, $P_{r\_arb\_sa}$, and $P_{r\_rt\_hflit}$) are obtained from the synthesized design. The leakage power is assumed to be negligible in logic and link propagation. Figure 6 compares results from the analytical model and a cycle-accurate simulation. The simulation environment is the same as in section 3.3. The results from the proposed analytical model match closely the experimental ones.



(a) Virtual Channel Router (b) Path-Sensitive Router

Figure 6: Total Power Consumption in Analytical Model and Simulation (8x8 Mesh)

4. COMMUNICATION RELIABILITY

The possible soft faults that could afflict a network architecture can be grouped in two main categories: link errors that occur during the traversal of flits from router to router, and router errors that occur within the router hardware components. Link errors are caused mainly by channel disturbances such as cross-talk, coupling noise and transient faults [32]. They have so far been considered the dominant source of network infrastructure errors and error detecting and correcting codes are being used extensively in on-chip communication links as a form of protection against link errors [33, 36]. Our proposed architecture employs both error detecting (in the form of Cyclic Redundancy Check [CRC]) and error correcting (in the form of Single-Error-Correction-Double-Error-Detection [SECDED]) codes to combat this problem.

Existing models [7, 36], however, fail to capture the effects of transient errors (e.g. soft errors) occurring within a router. Transient faults are rapidly becoming a force to be reckoned with in the deep sub-micron era. In fact, the susceptibility of circuits to such errors increases exponentially with technology scaling [30]. It is imperative that modern designs effectively account for these events. Soft errors within the router would escape the error detecting/correcting blanket because they do not actually corrupt the data, but, instead, cause erroneous behavior in the functionality of the routing process. Our proposed router architecture and routing algorithm work in concert to protect against a multitude of such faults. To the best of our knowledge, this is the first attempt to account for and protect against router soft faults by utilizing both architectural and algorithmic traits.

4.1 Link Errors

To handle link errors, we simulated 5 different retransmission/error correction schemes as shown in Table 1. Three of them, End-to-End (E2E), Hop-by-Hop (HBH), and Forward Error Correction (FEC), are widely used techniques in traditional networks, while Header E2E (HE2E) and Header FEC (HFEC) are extensions of E2E and FEC schemes, respectively, adopting hop-by-hop head-flit error checking. In these two schemes, each intermediate router checks the header flits for possible errors and if an error is detected, the correct head flit is retransmitted from the previous node. These schemes are different from HBH in that the head error-checking logic checks only part of the head flit, not the whole flit, and, thus, we can use smaller and faster error-checking logic. The portion of the head flit that needs to be checked is the one which contains the source and destination addresses, thus ensuring that a packet is not misdirected to a wrong destination. Delivery to a wrong destination is a problem that both E2E and FEC schemes suffer from when the destination address is corrupted. In Table 1, we summarize the modules used for each scheme, how much data should be retransmitted when an error is detected, and the overhead in terms of both area and latency. Error checking and correction are assumed not to be in the critical path, causing no latency overhead. This is because error detection/correction can be performed in parallel with other router or Network Interface Controller (NIC) operations.

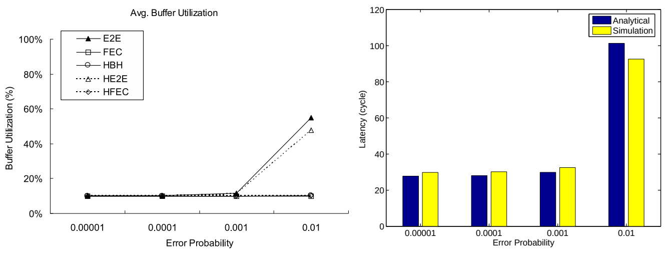
The CRC and SECDED units were implemented in 90nm technology as separate modules, analyzed in terms of dynamic and leakage power consumption and the numbers imported into our simulator. Depending on the retransmission scheme employed (i.e. End-To-End or Hop-By-Hop) the router architecture changes accordingly. The overhead of the retransmission control logic and buffer size and implementation are markedly different. In the ETE schemes, the number of flit buffers required is much larger than the HBH, because a router must keep a copy of each flit transmitted for as long as it takes to receive a possible NACK (Negative Acknowledgement) from the receiver which could be far away. The buffers, however, can be disabled when not in use to save leakage power. In the HBH schemes, the number of flit buffers required is much smaller, but the buffer is cyclic since the number of cycles between acknowledgements from neighboring routers is known precisely. This implies that the buffers cannot be disabled, since they rotate contents every clock cycle. These architectural differences are vital in correctly estimating power consumption in various retransmission schemes. Our hierarchical hardware implementation and analysis of all these components individually allows our model to account for these subtleties, thus substantially improving our projected results.

	Scope	E2E	HE2E	HBH	FEC	HFEC
Error Correction/ Detection Module	Dest. Node	CRC	CRC	None	SECDED	SECDED
	Every Node	None	CRC	CRC	None	CRC
Unit of Retransmission	Src to Dest	Packet	Packet	N/A	N/A	N/A
	Every Hop	N/A	Flit (Head Only)	Flit	N/A	Flit (Head Only)
Buffer Requirement Overhead	Source Node	Packet	Packet	None	None	None
	Every Node	None	Flit	Packet	None	Flit
Latency Overhead when an error is detected	Head Flit (Dest. Addr. Error)	▨ NACK (Dest to Src) + Packet Retrans (Src to Dest)	NACK(1hop) + Flit Retrans (1hop)	NACK (1hop) + Flit Retrans (1hop)	▨ NACK (Dest to Src) + Packet Retrans (Src to Dest)	NACK(1hop) + Flit Retrans (1hop)
	Head Flit (Dest Addr. Correct) or Middle Flit or Tail Flit	NACK (Dest to Src) + Packet Retrans (Src to Dest)	NACK (Dest to Src) + Packet Retrans (Src to Dest)		N/A	N/A

▨ : When a packet is delivered to a wrong destination because of head flit error.

Table 1: Retransmission Schemes

We also tracked buffer utilization under these schemes to identify the trends in buffer usage. The average buffer utilization at a fixed flit arrival probability of 0.35 was measured. In FEC, HFEC and HBH, only about 10% of a single-flit buffer is utilized on an average, as shown in Figure 7(a). But in E2E and HE2E, buffer utilization increases abruptly as the error probability increases, since the retransmissions from the source to the destination inject additional messages into the network, hence increasing the overall network traffic. The HE2E scheme, being able to fix some of the head flit errors using HBH head flit retransmission, shows less buffer utilization than E2E at higher error probabilities.



(a) Average buffer utilization (b) Performance Analysis

Figure 7: Buffer utilization & Performance Analysis under Error Detection and Retransmission

4.1.1 *Modeling End-to-End Retransmission*

Our proposed mathematical model can capture any reliability-related parameters to account for both latency and power overhead imposed by any of the five link error control schemes. We demonstrate the utility of our analytical model by analyzing the behavior of an end-to-end (E2E) error detection and retransmission scheme, as an example.

The end-to-end retransmission technique significantly alters the traffic load parameter in our analytical model. Retransmitted packets from error detection at the destination node increase the effective traffic load, which increases contention and reduces buffer availability. These changes can be reflected mathematically. As discussed in section 3.1, the zero-error latency (T) (Equation 3) is estimated as a function of the flit arrival probability (P_c). Assuming a switch-to-switch error probability, P_{error}, we can determine the end-to-end reliability latency as follows:

$$T_{ete}(P_c) = T(P_c'') + T_{ret}(P_c'')(1-(1-P_{error})^H), \quad (30)$$

where H is the average number of hops between a source and a destination, and P_c'' (the modified traffic load) is given by

$$P_c'' = \frac{P_c}{(1-(1-P_{error})^H)}. \quad (31)$$

T_{ret} includes the round-trip time of message retransmission and the re-send request control signal ($NACK$) delay, where the $NACK$ signal is assumed to be error free for simplicity. Thus, the total latency is given as

$$T_{ret}(P_c'') = T_{ete}(P_c) + T_{NACK}(P_c''). \quad (32)$$

The control signal delay (T_{NACK}) is the network latency for one single flit transmission. Using Equation(3) in Section 3, we can compute (T_{NACK}) as

$$\begin{aligned} T_{NACK}(P_c'') \quad &\approx \quad (B(P_c'')W(P_c'') + P + 1)H \\ &\quad +B_{dest}(P_c'')W_{dest}(P_c'') + P - 1. \quad (33) \end{aligned}$$

End-to-end error detection and retransmission schemes inflict a significant performance penalty to the network operation under bursty error conditions and high error probabilities. Figure 7(b) compares the results of our analytical model to those of a cycle-accurate simulator. Clearly, results from our mathematical model closely match the experimental results.

4.2 Router Logic Errors

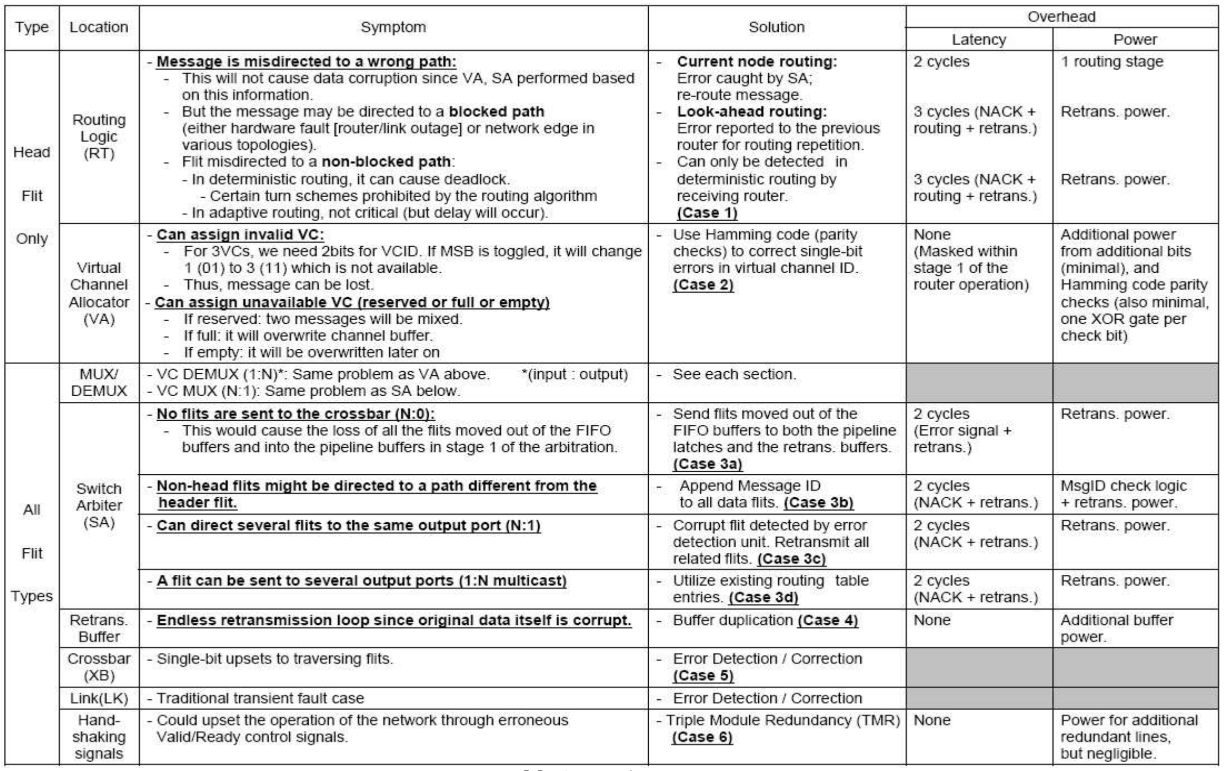
Our router implementation (shown in Figure 1(b)) consists of six major components, each susceptible to transient faults: the routing unit, the VA, the SA, the crossbar, the retransmission buffers (included in the CRC and SECDED units) and the valid/ready handshaking signals (used between neighboring routers). Following is a detailed description of possible faults and proposed solutions for all major router components. Additionally, the latency and power overhead of our proposed solutions/safeguards are also listed. All the results are summarized in Table 2.

Case 1 - Routing Unit Protection: A transient fault in the routing unit logic could cause a flit to be misdirected. This will not cause any data corruption, since the subsequent virtual channel allocation and switch arbitration would be performed based on the misdirection. The erroneous direction, however, may be blocked, either because of a link outage, or a network edge in various topologies (see Figure 8(a)). The proposed solution to such errors depends on the routing algorithm: in current-node routing schemes (i.e. the routing decision for the current router is taken at the current router), the misdirection will be caught by the switch arbiter which can inform the routing unit to repeat the routing procedure. This will incur a two-cycle delay. In look-ahead routing (i.e. the routing decision for the current router was taken at the previous router), the error will also be caught by the switch arbiter and reported to the previous router through a NACK, thus incurring a 3 cycle delay.

Misdirection to a functional path, however, will not be caught by the switch arbiter. It could potentially cause deadlock in deterministic routing algorithms. In such algorithms, however, the error can be detected in the router that receives the misdirected flit. A NACK to the sending router would then fix the problem within 3 clock cycles (NACK + re-routing + retransmission). In adaptive routing schemes the error cannot be detected. However, in such schemes a misdirection fault is not fatal; it would merely delay the flit traversal.

Figure 8(a) illustrates an example effect of a soft error in the routing unit. As explained above, such an error could direct a flit to a blocked path. For example, a router at the top edge of a mesh topology has a blocked output link to the north. A transient fault in the routing unit could direct a flit coming from the local processing element (or any input link) to be placed in the blocked path to the north. This error will be caught by the switch arbiter which knows that the path is blocked.

Case 2 - Virtual Channel Allocator (VA) Protection: A soft error in the VA unit could lead to an invalid virtual channel flit assignment. A bit flip in the virtual channel ID could give rise to either a non-existent virtual channel or a different existing one. This channel could be reserved or full. In the former case, two different messages would be mixed, and in the latter case an existing flit would be overwritten. Both problems would lead to flit/packet loss. If the erroneous virtual channel is unreserved and empty, then the flit will eventually be overwritten when a new, correct packet is assigned to that channel. The proposed solution involves the use of a Hamming code within the virtual channel ID to correct all single-bit errors. The overhead for such a code is minimal because of the small length of the virtual channel ID. In our proposed architecture, there are 3 virtual channels per physical path set, thereby requiring a 2-bit virtual channel ID. For a 2, 3, or 4-bit word, a single-error-correcting Hamming code requires 3 check bits. A 3-bit overhead is inconsequential compared to the protection and flexibility that such a code offers; with those 3 check bits, the virtual channels per physical path can be increased to any number up to 16 without any impact on the virtual channel ID protection. The area and power overhead is also minimal, since the code requires only one XOR gate per check bit. There is no latency incurred since the Hamming check is masked within stage 1 of the router operation (along with routing and virtual channel and switch arbitrations).



Type	Location	Symptom	Solution	Overhead	
				Latency	Power
Head Flit Only	Routing Logic (RT)	- **<u>Message is misdirected to a wrong path:</u>** - This will not cause data corruption since VA, SA performed based on this information. - But the message may be directed to a **blocked path** (either hardware fault [router/link outage] or network edge in various topologies). - Flit misdirected to a **non-blocked path**: - In deterministic routing, it can cause deadlock. - Certain turn schemes prohibited by the routing algorithm - In adaptive routing, not critical (but delay will occur).	- **Current node routing:** Error caught by SA; re-route message. - **Look-ahead routing:** Error reported to the previous router for routing repetition. - Can only be detected in deterministic routing by receiving router. **<u>(Case 1)</u>**	2 cycles 3 cycles (NACK + routing + retrans.) 3 cycles (NACK + routing + retrans.)	1 routing stage Retrans. power. Retrans. power.
	Virtual Channel Allocator (VA)	- **<u>Can assign invalid VC:</u>** - For 3VCs, we need 2bits for VCID. If MSB is toggled, it will change 1 (01) to 3 (11) which is not available. - Thus, message can be lost. - **<u>Can assign unavailable VC (reserved or full or empty)</u>** - If reserved: two messages will be mixed. - If full: it will overwrite channel buffer. - If empty: it will be overwritten later on	- Use Hamming code (parity checks) to correct single-bit errors in virtual channel ID. **<u>(Case 2)</u>**	None (Masked within stage 1 of the router operation)	Additional power from additional bits (minimal), and Hamming code parity checks (also minimal, one XOR gate per check bit)
All Flit Types	MUX/ DEMUX	- VC DEMUX (1:N)*: Same problem as VA above. *(input : output) - VC MUX (N:1): Same problem as SA below.	- See each section.	▨	▨
	Switch Arbiter (SA)	- **<u>No flits are sent to the crossbar (N:0):</u>** - This would cause the loss of all the flits moved out of the FIFO buffers and into the pipeline buffers in stage 1 of the arbitration.	- Send flits moved out of the FIFO buffers to both the pipeline latches and the retrans. buffers. **<u>(Case 3a)</u>**	2 cycles (Error signal + retrans.)	Retrans. power.
		- **<u>Non-head flits might be directed to a path different from the header flit.</u>**	- Append Message ID to all data flits. **<u>(Case 3b)</u>**	2 cycles (NACK + retrans.)	MsgID check logic + retrans. power.
		- **<u>Can direct several flits to the same output port (N:1)</u>**	- Corrupt flit detected by error detection unit. Retransmit all related flits. **<u>(Case 3c)</u>**	2 cycles (NACK + retrans.)	Retrans. power.
		- **<u>A flit can be sent to several output ports (1:N multicast)</u>**	- Utilize existing routing table entries. **<u>(Case 3d)</u>**	2 cycles (NACK + retrans.)	Retrans. power.
	Retrans. Buffer	- **<u>Endless retransmission loop since original data itself is corrupt.</u>**	- Buffer duplication **<u>(Case 4)</u>**	None	Additional buffer power.
	Crossbar (XB)	- Single-bit upsets to traversing flits.	- Error Detection / Correction **<u>(Case 5)</u>**	▨	▨
	Link(LK)	- Traditional transient fault case	- Error Detection / Correction	▨	▨
	Hand-shaking signals	- Could upset the operation of the network through erroneous Valid/Ready control signals.	- Triple Module Redundancy (TMR) **<u>(Case 6)</u>**	None	Power for additional redundant lines, but negligible.

Table 2: Logic Errors

In the block diagram of our proposed architecture shown in Figure 8(b), the VA unit is combined with the switch arbiter; however, their operations are distinct. Just like the routing process, the virtual channel allocation is done entirely in stage 1 of the router operation. Thus, a fault would immediately cause an erroneous result, i.e. an invalid virtual channel assignment within a path set, as indicated by the arrows (see Figure 8(b)).

Case 3 - Switch Arbiter (SA) Protection: A transient fault in the switch arbiter could give rise to more complex errors than the previous cases because the control signals span both stages of the router operation (see large arrows in Figure 8(c)). The control and scheduling signals for the crossbar traversal are generated during stage 1, but used in stage 2. Therefore, the switch arbiter instructs the FIFO virtual channel buffers to shift one position in stage 1, and the flits latched in the pipeline buffers are sent to the crossbar in stage 2. Thus, a soft error in the SA could give rise to several packet-loss problems:

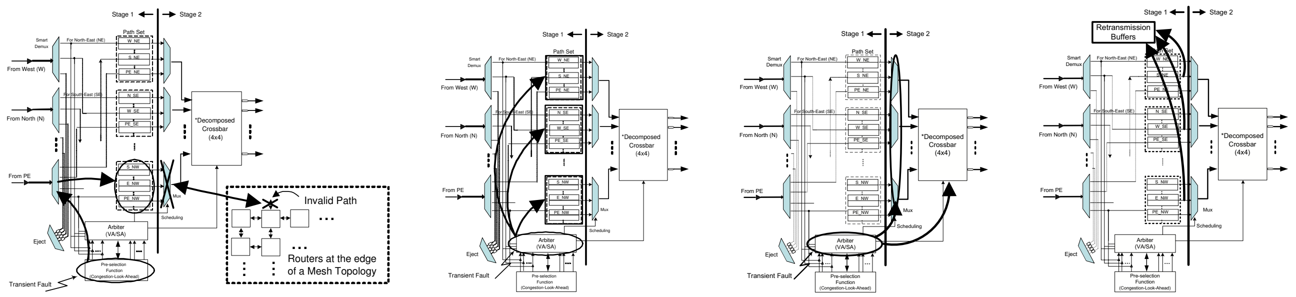
(a) A soft error in the control signals might prevent the flits from traversing the crossbar in stage 2. This would cause the loss of all the flits moved out of the FIFO buffers and into the pipeline buffers in stage 1 of the arbitration. To avoid this fatal situation, the retransmission buffer and an additional flag bit could be used for recovery. The flits moved out of the FIFO buffers should be sent to both the pipeline latches and the retransmission buffers (see Figure 8(d)). The retransmission buffers in our proposed architecture are large enough to keep four consecutive flits sent out on each output link. Additionally, whenever valid data is moved out of the FIFO buffers and into the pipeline latches a "Data Valid" bit is set. This bit is subsequently ANDed with the control signal of the switch arbiter in stage 2. If the control signal was erroneously reset to a zero state, i.e. no crossbar traversal for any flit, then the mistake will be caught. A signal is sent to the switch arbiter which can retransmit the flits from the retransmission buffer. The additional latency is two clock cycles (Error signal + retransmission).

(b) If a data flit is mistakenly sent to a direction different from the header flit, it would cause flit/packet loss. We propose using a very compact header in all data flits which would include a message ID. Additionally, each router would have a simple message ID checker unit. If the message ID check reveals that the incoming flit ID is not in the router's routing table, then a NACK is sent to the sending router which would retransmit the flit to the correct router from its retransmission buffer. The incurred overhead comes from the message ID logic which is very simple and compact in area. The header overhead in the data flits is also minimal since only the message ID is stored. The additional latency is two clock cycles (NACK + retransmission) on an error.

(c) The error could cause the arbiter to direct two flits to the same output. This will lead to a corrupt flit which will be detected by the error detection code in the next router. A NACK will be sent and the correct flits retransmitted from the retransmission buffer. This error recovery process will incur two cycles (NACK + retransmission) latency overhead.

(d) The error could cause the arbiter to send a flit to multiple outputs (multicasting). If the flit is a data flit, the error will be taken care of by case 3(b) above at the next router. If, however, the flit is a header flit then one needs to consider two possibilities:

(i) If there is no existing message in the erroneous path, then the wrong header flit will be considered as the beginning of a new packet at the next router. Thus, a virtual channel will mistakenly be reserved. The way to tackle this is to utilize the existing routing table entries. When a correct header comes in on the same path later on, the router will detect that the path is already reserved through its InputVC-OutputVC pair in its routing table. It can, therefore, request a retransmission of the last header flit. If it receives the same flit again, it will realize that the channel was erroneously reserved and release the path to the correct header flit. (ii) If a correct header flit has already used the current path, then the next router will request a retransmit, as in case 3(d)(i) above. The previous router



(a) Routing Unit Error (b) VA Error (c) SA Error (d) Retrans. Buffers

Figure 8: Routing Unit, VA and SA Errors

will not resend the wrong header flit to the same router again, so the receiving router can discard the erroneous header flit from its buffer without disrupting the correctly reserved path.

The incurred overhead in both cases is a single flit retransmission from the previous router, i.e. a two-cycle latency and minimal power overhead.

Case 4 - Retransmission Buffer Protection: A soft error in the retransmission buffer would yield an endless retransmission loop since the original data itself is now corrupt. The way to avoid this problem is to use duplicate retransmission buffers. This will double the buffer area overhead and power. However, the number of retransmission flit buffers in on-chip routers is very small (four per output link in the case of our proposed two-stage router), so doubling the area and power of this component will not be prohibitive.

Case 5 - Crossbar Protection: A transient fault within the crossbar would produce single-bit upsets, not entire flits being misdirected as in the switch arbiter case. Single-bit upsets are taken care of by the error detection and correction unit employed within each router, thus eliminating the problem.

Case 6 - Valid/Ready Handshaking Signal Protection: Every router has several valid/ready handshaking signal lines with neighboring routers to facilitate proper functionality and synchronization. For example, each output link has a ”Data Valid” output line and a ”Data Ready” input line to/from the adjacent router. Transient faults on these lines could severely hamper the operation of the network. Therefore, Triple Module Redundancy is proposed by which three lines and voting are used, instead of one, to ensure protection against soft errors. There is an area and power overhead increase, but the area occupied by these lines is negligible compared to the area of the other router components.

4.3 Simulation Results

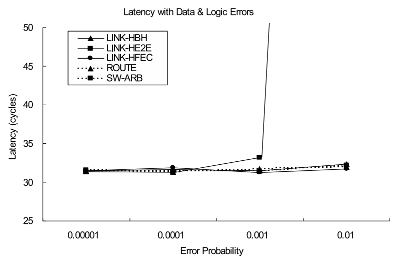
The experimental setup described in section 2 was used. Single-bit errors were uniformly injected in the network, both as link errors and as logic errors within the router architecture. The definition of error probability is slightly different depending on the error model. For link errors, it is defined as the probability of a flit error during link traversal. For routing and switch arbitration logic errors, it is defined as the probability that a flit is mishandled by the logic as a result of single-event upset.

Figure 9(a) shows the overall latency of each scheme for a wide range of error probabilities. In all schemes, except HE2E, the latency overhead incurred on the overall latency as a result of error detection/correction was minimal, because all the schemes typically require only 1-3 cycles, as shown in Tables 1 and 2. However, in the HE2E scheme, if a message has errors in the flit portion where error checking is not performed at intermediate routers, then the whole packet should be retransmitted, and these additional flits significantly increase the overall message injection rate of the network, and increase the average message latency. The proposed architectural improvements within the router components (i.e. routing logic (ROUTE) and switch arbiter logic (SW-ARB) curves) inflict no significant latency increase, as predicted. Figure 9(b) shows the number of detected and corrected errors under each error protection scheme. Without error detection/correction, the message will either be corrupted or lost, and, thus, no errors will be corrected at all. By using these schemes, errors can be corrected either through retransmission or error-correction. Since errors in the routing logic only affect head flits, our routing logic protection scheme has the least number of errors detected/corrected. Both switch arbiter errors and link errors can corrupt all flits. Since most flits go through the switch arbitration several times at each node as network congestion increases, while they only traverse the link once, errors in the switch arbiter are more frequently detected/corrected than link errors. This trend justifies the inclusion of our proposed router-error safeguards in on-chip network architectures, since, as mentioned before, soft error faults within the routers will continue to increase exponentially as technology scales down. As shown in Figure 9(b), as error probabilities increase, the number of errors detected and corrected by our switch arbiter reliability measures rises abruptly. All those errors would otherwise escape and cause severe packet/flit losses. Figure 9(c) shows the energy consumption per packet. In the case of HE2E, source to destination retransmission incurs significant energy consumption, while the other schemes have minimal energy overhead, since either they are not involved in retransmission, or are involved in hop-by-hop retransmission which has minimal power overhead. The proposed router-logic protection measures do not cause any significant increase in energy consumption.

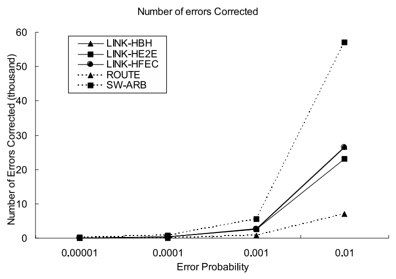
Even though our reliability measures provide a blanket of protection against both link and router logic errors, the protection is limited to single-bit errors. Temporal multi-bit errors (i.e. two independent transient faults affecting the same flit at different times), and spatial multi-bit errors (i.e. one fault causing two different errors at different locations) are not tackled. However, the probability of such events is still considered extremely low.

5. CONCLUSIONS

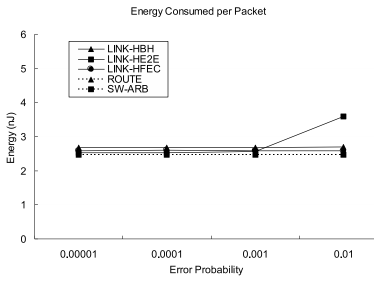
The rapidly increasing use of SoC architectures has accentuated the need for efficient on-chip communication infrastructures. Packet-based NoC architectures are currently the dominant choice to address the communication requirements of SoCs. The resource-constrained nature of such networks differentiates them from traditional off-chip networks, and thus, needs precise and efficient analysis of their performance, fault-tolerance and energy behaviors. To this end, we propose a queuing-theory-based analytical model for 2D mesh networks, which performs latency and power analysis at the granularity of individual router sub-modules for increased accuracy. Simulation results validate the correctness and accuracy of the model for a generic 2-stage router. The model is then used to



(a) Overall Latency



(b) Errors Corrected



(c) Energy Consumption

Figure 9: Simulation Results

demonstrate it's effectiveness in analyzing a novel path-sensitive router architecture that can minimize average packet latency by intelligent path selection and reduced switching activities. Furthermore, the analytic model is used in quantifying the overall power consumption by capturing the utilization of different components and their corresponding energy consumption.

The study then focuses on the fault-tolerance aspects of on-chip interconnects by analyzing two types of faults: link errors and router logic errors. We explore five types of retransmission techniques to combat link errors and conclude that by providing a separate error coding technique for the header flits, the packet misrouting probability is significantly reduced, thereby providing better fault-tolerance. We then extend our analytical model to evaluate the network under the End-to-End (E2E) protection scheme to demonstrate the utility of the model in analyzing all three parameters: performance, energy and fault-tolerance. We additionally propose a series of transient fault protection techniques to tackle soft errors within the router's individual components. Our safeguards are shown to incur no significant area, latency, or power overhead to the network.

6. REFERENCES

[1] A. Adriahantenaina, H. Charlery, A. Greiner, L. Moriiez, and C. A. Zeferino. SPIN: A Scalable, Packet Switched, On-chip Micro-network. In *Proceedings of the DATE*, 2003.

[2] A. Agarwal. Limits on interconnection network. *IEEE Transaction on Parallel and Distibuted Systems*, 2(4), 1991.

[3] L. Anghel and M. Nicolaidis. Cost Reduction and Evaluation of Temporary Faults Detecting Technique. In *DATE 2000*, pages 591–598, March 2000.

[4] J. M. Baker, S. B. Jr., M. Bucciero, B. Gold, and R. Mahajan. SCMP: A Single-Chip Message Passing Parallel Computer. In *Parallel and Distributed Processing Techniques and Applications (PDPTA'02)*, pages 1485–1491, 2002.

[5] L. Benini and G. D. Micheli. Networks on Chips: A New SoC Paradigm. *IEEE Computer*, 35(1):70–78, 2002.

[6] D. Bertozzi, L. Benini, and G. D. Micheli. Low Power Error Resilient Encoding for On-chip Data Buses. In *Proceedings of Design Automation and Test Conference in Europe*, March 2002.

[7] D. Bertozzi, L. Benini, and G. D. Micheli. Low power error resilient encoding for on-chip data buses. In *Proceedings of 2002 DATE*, 2002.

[8] Y. Cao, C. Hu, A. B. Kahng, S. Muddu, D. Stroobandt, and D. Sylvester. Effects of Global Interconnect Optimizations on Performance Estimation of Deep Submicron Designs. In *International Conference on Computer-Aided Design*, pages 56–61, 2000.

[9] W. J. Dally and B. Towles. Route Packets, Not Wires: On-Chip Interconnection Networks. In *Proceedings of the 38th DAC*, June 2001.

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

[11] T. Dumitras, S. Kerner, and R. Marculescu. Towards On-chip Fault-Tolerant Communication. In *Proc. Asia & South Pacific Design Automation Conf.(ASP-DAC)*, January 2003.

[12] N. Eisley and L.-S. Peh. High-level power analysis for on-chip networks. In *Proceedings of the 7th International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES)*, September 2004.

[13] A. H. et al. Network on a Chip: An Architecture for Billion Transistor Era. In *Proc. of the IEEE NorChip Conference*, November 2000.

[14] D. Flynn. AMBA: Enabling Reusable On-Chip Design. In *IEEE Micro*, pages 20–27, July 1997.

[15] R. Ho, K. W. Mai, and M. A. Horowitz. The Future of Wires. In *Proceedings of the IEEE*, pages 490–504, 2001.

[16] J. Hu and R. Marculescu. Exploiting the Routing Flexibility for Energy/Performance Aware Mapping of Regular NoC Architectures. In *Proc. Design, Automation and Test in Europe Conference*, 2003.

[17] J. Huh, S. W. Keckler, and D. Burger. Exploring the Design Space of Future CMPs. In *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques (PACT)*, 2001.

[18] R. M. Jingcao Hu. Energy-aware mapping for tile-based noc architectures under performance constraints. In *Proc. of ASPDAC*, January 2003.

[19] E. Jung, K. K. H. Yum, and C. R. Das. Calculation of deadline missing probability in a qos capable cluster interconnect. In *NCA '01: Proceedings of the IEEE International Symposium on Network Computing and Applications (NCA'01)*, page 36, Washington, DC, USA, 2001. IEEE Computer Society.

[20] J. Kim, D. Park, T. Theocharides, N. Vijaykrishnan, and C. R. Das. A low latency router supporting adaptivity for on-chip router. In *42nd DAC*, June 2005.

[21] A. Krstic, Y. M. Jiang, and K. T. Cheng. Pattern Generation for Delay Testing and Dynamic Timing Analysis Considering Power-Supply Noise Effects. *IEEE Transactions on CAD*, 20(3):416–425, March 2001.

[22] S. Kumar, A. Jantsch, J. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani. A Network on Chip Architecture and Design Methodology. In *Proc. IEEE Computer Society Annual Symposium on VLSI*, pages 105–112, 2002.

[23] J. Liang, S. Swaminathan, and R. Tessier. aSOC: A Scalable, Single-Chip Communications Architecture. In *the IEEE International Conference on Parallel Architectures and Compilation Techniques*, pages 524–529, October 2000.

[24] R. Marculescu. Networks-On-Chip: The Quest for On-Chip Fault-Tolerant Communication. In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI'03)*, 2003.

[25] C. A. Moritz, D. Yeung, and A. Agarwal. SimpleFit: A Framework for Analyzing Design Trade-Offs in Raw Architectures. *IEEE TPDS*, 12(7):730–742, 2001.

[26] V. Raghunathan, M. B. Srivastava, and R. K. Gupta. Energy-Aware System Design: A Survey of Techniques for Energy Efficient On-Chip Communication. In *Proceedings of the 40th DAC*, pages 900–905, 2003.

[27] H. Sarbazi-Azad, M. Ould-Khaoua, and L. M. Mackenzie. A performance model of adaptive wormhole routing in k-ary n-cubes in the presence of digit-reversal traffic. *J. Supercomput.*, 22(2):139–159, 2002.

[28] L. Shang, L.-S. Peh, and N. K. Jha. Dynamic Voltage Scaling with Links for Power Optimization of Interconnection Networks. In *Proc. HPCA*, February 2003.

[29] K. L. Shepard and V. Narayanan. Noise in Deep Submicron Digital Design. In *IEEE/ACM ICCAD-96*, pages 524–531, November 1996.

[30] P. Shivakumar, M. Kistler, S. Keckler, D. Burger, and L. Alvisi. Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic. In *Proceedings of the International Conference on Dependable Systems and Networks*, 2002.

[31] Sonics, Incorporated. http://www.sonicsinc.com.

[32] S. R. Sridhara and N. R. Shanbhag. Coding for system-on-chip networks: a unified framework. In *DAC*, pages 103–106, June 2004.

[33] P. Vellanki, N. Banerjee, and K. Chatha. Quality-of-Service and Error Control Techniques for Network-on-Chip Architectures. In *Proceedings of the Great Lakes Symposium on VLSI*, 2004.

[34] T. T. Ye, L. Benini, and G. D. Micheli. Packetized On-Chip Interconnect Communication Analysis for MPSoC. In *Proc. Design Automation and Test in Europe*, pages 344–349, 2003.

[35] H. Zhang, M. Wan, V. George, and J. Rabaey. Interconnect Architecture Exploration for Low-Energy Reconfigurable Single-Chip DSPs. In *Proceedings of the WVLSI*, April 1999.

[36] H. Zimmer and A. Jantsch. A Fault Model Notation and Error-Control Scheme for Switch-to-Switch Buses in a Network-on-Chip. In *Proceedings of the 1st IEEE/ACM/IFIP international conference on Hardware/software codesign & system synthesis*, pages 188–193, 2003.