Tackling Permanent Faults in the Network-on-Chip Router Pipeline

Pavan Poluri
Department of Electrical and Computer Engineering
University of Arizona
Tucson, USA
Email: pavanp@email.arizona.edu

Ahmed Louri
Department of Electrical and Computer Engineering
University of Arizona
Tucson, USA
Email: louri@email.arizona.edu

***Abstract*—The proliferation of multi-core and many-core chips for performance scaling is making the Network-on-Chip (NoC) occupy a growing amount of silicon area spanning several metal layers. The NoC is neither immune to hard faults and transient faults nor unaffected by the adverse increase in hard faults caused by technology scaling. The ramifications for the NoC are immense: a single fault in the NoC may paralyze the working of the entire chip. To this end, we propose a Permanent Fault Tolerant Router (PFTR) that is capable of tolerating multiple permanent faults in the pipeline. PFTR is designed by making architectural modifications to individual pipeline stages of the baseline NoC router. These architectural modifications involve adding minimum extra circuitry and exploiting temporal parallelism to accomplish fault tolerance. Tolerance of multiple faults is achieved by striking a balance between three important design factors namely, area overhead, power overhead and reliability. We use Silicon Protection Factor [13] (SPF) as the reliability metric to assess the reliability improvement of the proposed architecture. SPF takes into account the number of faults required to cause failure and the area overhead of the additional circuitry to evaluate reliability. SPF calculation reveals that the proposed PFTR is 11 times more reliable than the baseline NoC router. Synthesis results using Cadence Encounter RTL Compiler at 45nm technology show that the additional circuitry adds an area overhead of 31% and power overhead of 30% with respect to the baseline NoC router. PFTR provides much better reliability with much less overhead as compared to other fault tolerant routers such as BulletProof [13], Vicis [14] and RoCo [15].**

***Keywords*-Network-on-Chip, Router Architecture, Reliability, Area, Power, Latency**

I. Introduction

As the feature size keeps scaling down, concerns regarding the reliability issues keep amplifying [1], [3], [4] affecting the functionality and lifetime of future devices and systems. Devices and circuits are predominantly vulnerable to two kinds of faults namely, permanent and transient faults. A permanent fault is a fault that continues to affect the operation of a circuit from the time of its inception. Common sources of permanent faults include electro-migration [5], hot carrier degradation [6], time dependent dielectric breakdown [7] etc. A transient fault is a fault that affects the operation of a circuit for a smaller period of time typically in the order of a single clock cycle. Common sources of transient faults include thermal radiation from cosmic rays [8], process variation [9], alpha particles from the packaging material [10] etc.

NoC [2], [11], [22], [23] is a packet based interconnection network that facilitates communication between many cores on a chip. NoC plays an effective role in decoupling the intra-core computation from the inter-core communication. Major components of a NoC include routers and links. Decrease in the feature size is making the NoC increasingly vulnerable to faults. As the number of cores on a chip increase, faults in the NoC could have a significant impact on performance as well as functionality of the entire chip. Hence, it is of utmost importance to tackle the increasing reliability concerns in the NoC.

II. Contributions

This work contributes to the ongoing efforts of designing fault-tolerant NoCs. In this paper, we deal with the router architecture and focus on the design of permanent fault tolerant router pipeline. We tackle transient faults in a different study. The pipeline of a NoC router is responsible for the smooth flow of packets from the time of their arrival at the router to the time of their departure from the router. Since the pipeline plays such a pivotal role in the functionality of the router, it is vital to tackle its reliability.

We propose Permanent Fault Tolerant Router (PFTR), a router architecture that is capable of tolerating multiple permanent faults in its pipeline. These architectural modifications involve adding minimum extra circuitry to individual stages and taking advantage of the temporal parallelism thereby enabling each pipeline stage to tolerate a single permanent fault. Assuming that each individual pipeline stage is affected by only **one** permanent fault, the PFTR pipeline will be able to tolerate **four** permanent faults. The main contributions of this paper can be summarized as:

- A fault-tolerant router architecture that can tolerate multiple permanent faults in the routing pipeline.
- Performance analysis involving area, power, latency, critical path and reliability of the proposed architecture, and comparison of improvement in reliability with other proposed fault-tolerant NoC routers such as BulletProof [13], Vicis [14] and RoCo [15].

978-1-4799-2927-6/13 $26.00 © 2013 IEEE
DOI 10.1109/SBAC-PAD.2013.32

III. Related Work

Various researchers have targeted different fault-tolerance aspects of NoC [13], [14], [15], [16], [17]. In this section, we provide a brief overview of the architectures presented in [13], [14] and [15] as they tackle the issue of permanent faults affecting a NoC router pipeline.

Constantinides et al. proposed the concept of a BulletProof [13] router that employs N-modular redundancy (NMR) technique to provide fault tolerance. NMR technique requires the existence of N copies of the protected component. As a result, the silicon area required to fabricate the protected router increases by N times. Since the area of a design has a linear relationship with the possible number of faults, employing redundancy based techniques is not always efficient.

Fick et al. proposed Vicis [14] methodology that employs port swapping algorithm to tolerate faults at ports. Further, they propose to use bypass bus to tolerate faults in crossbar and low overhead Error Correcting Codes to protect from faults that can occur in the datapath of the router.

Kim et al. proposed RoCo [15] router, which can be decomposed into individual row and column components. Since the row and column components are independent of each other, a permanent fault in one of the components does not affect the other component and the router continues to function with the fault-free component.

PFTR is different from these proposed methodologies in the aspect that it can tolerate a single permanent fault in every pipeline stage.

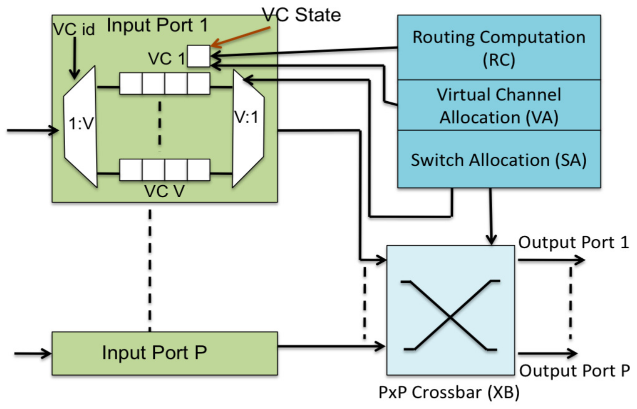
IV. Baseline Network-On-Chip Router

To better describe our proposed PFTR, we first describe the architecture and the pipeline execution of a baseline NoC router.

A. Router Architecture

Figure 1 shows the architecture of a generic NoC router. A router with *P* input ports and *P* output ports is comprised of *1:P* demultiplexer, *P:1* multiplexer, Routing Computation (RC) unit, Virtual Channel Allocator (VA), Switch Allocator (SA) and a Crossbar (XB) [11]. Each input port is comprised of *V* virtual channels (VC 1 , VC 2, ... VC V), *1:V* demultiplexer and *V:1* multiplexer. Figure 2 shows the pipeline of a NoC router.

Even though NoC is a packet based communication network, for efficient router resource utilization, packets are divided into three kinds of flits (flow control information units) namely head flit, body flit and tail flit. Head flit of a packet is responsible for allocating router resources (e.g., virtual channel) to the packet. Tail flit is responsible for de-allocating the router resources allocated for the specific packet. Body flits typically contain the payload of the packet. A packet is generally partitioned to have a single head flit, single or multiple body flits and a single tail flit.



VC id – Virtual Channel ID
VC State – Virtual Channel State
VC 1 – Virtual Channel 1

Figure 1: Generic NoC Router Architecture

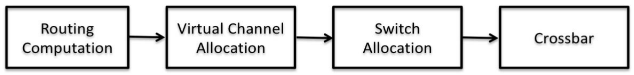


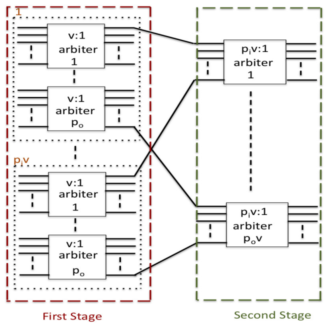
Figure 2: Generic NoC Router Pipeline

B. Router Pipeline

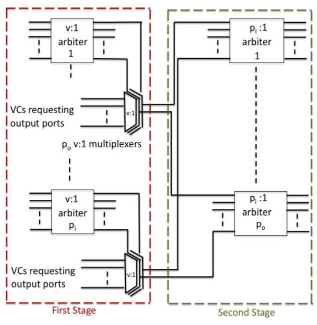
1) Routing Computation Stage: This is the first stage in the router pipeline and is active upon arrival of a head flit into the router. Using the destination information available in the head flit, the RC unit determines the specific output port of the current router through which the head flit will leave. This stage remains idle for body and tail flits.

2) Virtual Channel Allocation Stage: This is the second stage in the pipeline and is active for head flits. VA uses the result of RC as an input for performing allocation. Figure 3a shows the separable design of a two stage VA [12]. In the *first stage*, using the result of RC, each input VC that has a head flit arbitrates for an empty VC at the downstream router. In the *second stage*, head flits across different input VCs that have been allocated the same virtual channel in the downstream router compete with each other. Head flit of the input virtual channel that wins the arbitration in second stage is allocated the specific virtual channel at the downstream router. This stage remains idle for body and tail flits.

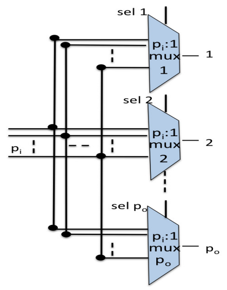
3) Switch Allocation Stage: This is the third stage in the pipeline and is active for head, body and tail flits. SA grants flits of an input virtual channel access to the output port of the crossbar. Figure 3b shows the separable design of a two stage switch allocator [12]. The *first stage* decides which virtual channel of an input port gets to transmit its flit using the crossbar. The *second stage* resolves the competition between virtual channels of different input ports trying to gain access to the same output port of the crossbar. The



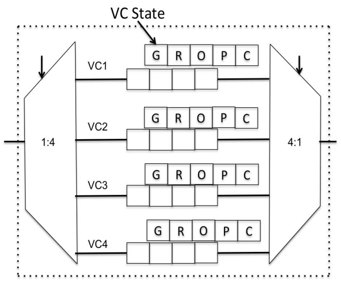
(a) Virtual Channel Allocator



(b) Switch Allocator



(c) Generic Crossbar



(d) Input Port Architecture

Figure 3: Baseline Router Pipeline Components

input virtual channel that wins this stage gets to transmit its flit through the crossbar in the next cycle.

4) Crossbar Stage: This is the fourth and the final stage in the pipeline and is active for head, body and tail flits. Figure 3c shows the generic architecture of a p_ixp_o crossbar where, p_i is the number of inputs to the crossbar and p_o is the number of outputs from the crossbar. The size of the crossbar determines the number and size of the multiplexers. The select signals to these multiplexers are controlled by the switch allocator. Based on the winners in the switch allocation stage, the multiplexers of the crossbar are configured accordingly such that the flits from the input ports travel to their respective output ports of the crossbar.

C. *Input Port Architecture*

Figure 3d shows the internal architecture of an input port of a router with four virtual channels [11]. Each virtual channel is associated with state fields namely 'G', 'R', 'O', 'P' and 'C'. 'G' field indicates the status of the VC in the current cycle. 'R' field is used to store the result of RC unit. 'O' field stores the result of VA that indicates which virtual channel in the downstream router is the current packet headed to. 'P' field indicates the read/write pointers in the virtual channel and 'C' field indicates the credit count.

V. Permanent Fault Tolerant Router (PFTR)

In this section, we consider each individual pipeline stage independently and describe the affect of a permanent fault on the stage and propose the fault tolerant methodology for that stage. Note that in this paper, our concern is focused on fault tolerance and not on fault detection. We assume that faults can be detected by using an existing fault detection mechanism [18]. Also, we only consider faults in different stages of the router pipeline. Faults in the other components of a router are studied in [24] and are out of scope of this paper.

A. *Routing Computation Stage*

Each input port has its own RC unit. Once a permanent fault manifests in the RC unit, every computation performed by it from that point would result in the calculation of a faulty output port. Since the execution of remaining stages is dependent on the result of this stage, the entire pipeline is affected as a result of permanent fault in this stage.

The architecture of the RC unit is dependent on the routing protocol employed in the NoC. In this work, we employ dimension order (XY) routing protocol in the NoC. XY routing protocol does not require routing tables [25]. To provide fault tolerance to this stage, we propose to have a redundant RC unit for each input port. Duplicate RC unit will be turned on and can be used upon detection of a permanent fault in the original RC unit.

B. *Virtual Channel Allocation Stage*

1) First Virtual Channel Allocation Stage: Figure 3a illustrates that each input VC has a set of p_o $v:1$ arbiters where, p_o is the number of output ports of the router and v is the number of VCs in the downstream router. When an input VC enters the first stage of VA, the $v:1$ arbiter corresponding to the output port computed by the RC unit is used to choose an empty VC from the available empty VCs at the downstream router connected to that output port. When a permanent fault manifests in one of the arbiters associated with an input VC, it will not be able to arbitrate for a VC at the downstream router for the corresponding output port whose arbiter is faulty resulting in the flit (packet) being blocked. To avoid this, we propose the following.

Each input VC has p_o $v:1$ arbiters that are identical to the arbiters of any other input VC. When an arbiter associated with a VC of a specific input port is affected by a permanent fault, the complete set of p_o $v:1$ arbiters is considered faulty and are not used in future computations. Instead, the affected VC, requests to use the arbiters of another VC belonging to the same input port. Since every input VC has identical set of arbiters, arbiters can be shared (temporal parallelism) between VCs. Thus, by using another VC's arbiters, virtual channel allocation can be performed for the head flit residing in the affected virtual channel.

Two scenarios are possible when a VC (VC 1) requests to use arbiters of another VC (VC 2) belonging to the same input port.

Scenario 1 - When VC 1 requests to use arbiters of VC 2, if the arbiters of VC 2 are idle, the delay involved in borrowing these arbiters is the same as the affect on critical path of VA (Section V.A).

Scenario 2 - When VC 1 requests to use arbiters of VC 2, if VC 2 is non-empty and is in VA stage like VC 1, in addition to the affect on critical path, there will be an additional latency of 1 cycle. This is because; the arbiters of VC 2 first perform allocation for the head flit in VC 2 and on successful allocation, in the next cycle can be used for allocation for the head flit in VC 1. Since the head flit in VC 1 had to wait for the arbiters of VC 2, the waiting induces additional latency.

Scenario 2 arises only if there was an unsuccessful virtual channel allocation encountered by one of the input VCs in the previous cycle. This is because; all the flits going to different VCs of the same input port have the same point of entry into the router. Two flits cannot enter two different virtual channels of an input port at the same time. These flits have to come one after the other thus making the input virtual channel the flit has entered first trigger the router pipeline earlier than the other virtual channel where the other flit has entered in the following cycle.

The unsuccessful virtual channel allocation encountered by a head flit in an input VC is not a consequence of the permanent fault but is due to the lack of empty VCs at the downstream router. This typically happens during high network traffic rate. The increase in the latency due to unsuccessful virtual channel allocation is a runtime parameter that changes based on the flow of traffic. However, in the presence of a permanent fault, sharing arbiters further increases the latency by only one more cycle.

2) Modified Input Port Architecture: Figure 4 shows the architecture of the input port with the new state fields namely, ‘R2‘, ‘VF‘, ‘ID‘, ‘SP‘ and ‘FSP‘ added to facilitate arbiters sharing between virtual channels of an input port.

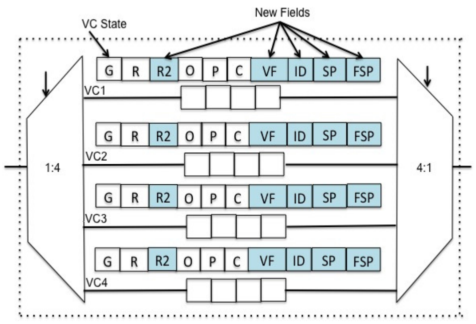


Figure 4: Modified Input Port Architecture

Consider a VC (e.g., VC 1) that intends to use the arbiters of another VC (e.g., VC 2) of the same input port. VC 1 initiates the process by placing its RC result in the ‘R2‘ field of the virtual channel (VC 2), its identification in the ‘ID‘ field of the virtual channel (VC 2) and setting the ‘VF‘ (virtual channel flag) field of the virtual channel (VC 2) to high. This field indicates whether the arbiters associated with that input VC are active for that specific input VC or are they being used by a different VC of the same input port. Once the arbiters of VC 2 have successfully allocated an empty virtual channel in the downstream router to the head flit in VC 1, the virtual channel allocator resets the ‘R2‘, ‘ID‘ and ‘VF‘ fields of VC 2. After virtual channel allocation is done, using the ‘ID‘ field, the appropriate virtual channel‘s state field is updated by the virtual channel allocator. The ‘SP‘ (secondary path) and ‘FSP‘ (secondary path flag) fields are used to provide fault tolerance for switch allocator and crossbar and will be described later.

3) Second Virtual Channel Allocation Stage: The purpose of this stage is to resolve conflicts between two different input VCs being allocated the same VC in the downstream router. This stage is comprised of a set of arbiters where each arbiter is associated with a specific VC at the downstream router. A fault in one of the arbiters in this stage will result in that specific VC at the downstream router not being allocated to any of the head flits in the current router. However, this fault does not lead to the flit (packet) being blocked at the current router because, the flit (packet) can be allocated another VC belonging to the required output port in the downstream router by using the associated non-faulty arbiter. Thus by utilizing the inherent redundant resources (multiple VCs), a permanent fault in this stage can be tolerated without the involvement of any additional circuitry.

C. Switch Allocation Stage

1) First Switch Allocation Stage: The first stage of SA (Figure 3b) is comprised of p_i $v:1$ arbiters where, p_i is the number of input ports and v is the number of VCs per input port. Each input port has an associated $v:1$ arbiter. The responsibility of an arbiter in this stage is to choose a VC from the associated input port. If the chosen VC eventually wins the arbitration in second stage, then a flit from this VC traverses through the crossbar in the next cycle.

Consider the scenario when a permanent fault has manifested in a $v:1$ arbiter. Due to the fault, the arbiter cannot choose a VC from the associated input port and as a result it cannot participate in the arbitration in second stage and hence will never win the arbitration. If the VCs of an input port never win switch allocation, the flits (packet) in the VCs of that input port will be blocked.

To avoid this, we propose to create a bypass path for each $v:1$ arbiter that can be used to choose a VC when the arbiter is faulty. When the bypass path is activated, it always chooses the same input VC as the winner. This can be accomplished by adding a *2:1* multiplexer that takes the output of the arbiter as one input and the identification of VC (stored in a register) that will always be selected as the winner using bypass path as the other input. For example, consider there are *v* VCs namely VC 1, VC 2 ... VC *v*. Let us say that when using the bypass path VC 2 is always chosen as the winner. So, the inputs to the additional *2:1* multiplexer will be the output of the arbiter and the virtual channel identification of VC 2.

Once the arbiter is faulty, VC 2 will always be chosen as winner. If VC 2 is not empty and is in SA stage, flits in VC 2 can traverse through the crossbar if it wins the arbitration in second stage. If VC 2 is empty and there are flits in other VCs, then flits from any other VC that belongs to the same input port as VC 2 can be transferred into VC 2.

When transferring flits from one VC (e.g., VC 1) to another VC of the same input port (e.g., VC 2), in addition to the flits, state fields of VC 1 also need to be transferred into the state fields of VC 2. After the transfer of flits and state fields is completed, the flits (initially in VC1) now in VC 2 can traverse through the crossbar when VC 2 wins the arbitration in second stage. Thus, with the help of transferring and using a bypass path, flits avoid being blocked and continue to traverse to their destination. Note that flits can only be transferred between two VCs of the same input port. Since reading and writing multiple flits and reading and writing the state fields can be performed in parallel, the transferring process between two input VCs incurs an additional latency of only 1 cycle. Figure 5 shows the modified switch allocator.

We used VC 2 as the default winner when the bypass path is activated for explanation. Any VC can be used as the default winner in the event of a fault in the arbiter.

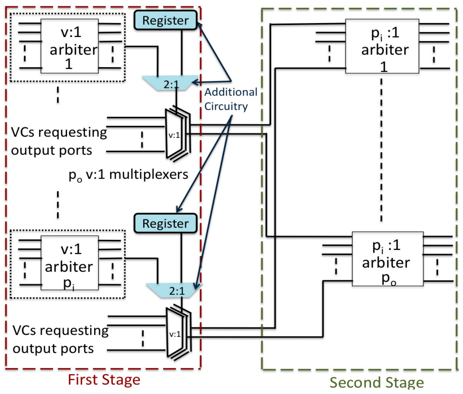


Figure 5: Modified Switch Allocator

2) Second Switch Allocation Stage: This stage of SA (Figure 3b) is comprised of p_o p_i : 1 arbiters where p_o is the number of output ports and p_i is the number of input ports. Each arbiter is associated with an output port. The input VC that wins the arbitration gets access to the associated output port of the arbiter. If the arbiter is faulty, then the input VCs cannot arbitrate for the arbiter‘s associated output port thus making the output port unreachable.

This situation can be solved by having a secondary path to reach the output port that is unreachable using the normal path. For example, consider output ports *x* and *y* of a router. Assume that the arbiter associated with port *x* is faulty and there exists a secondary path to reach port *x* by using the arbiter associated with output port *y*. Now, using the fault-free arbiter associated with port *y*, flits of any input port can reach port *x*. Details regarding the existence of a secondary path to reach an output port, how an arbiter associated with one output port helps reach another output port will be explained in the following sub-section where we describe the fault tolerant methodology for crossbar.

D. Crossbar Stage

From Figure 3c it can be inferred that each output port has an associated multiplexer that a flit from any input port needs to traverse through to reach the aforementioned output port. A permanent fault in a multiplexer blocks the passage to its associated output port. Since there is only one path to reach an output port, flits attempting to reach the output port associated with the faulty multiplexer cannot reach the output port and will get blocked.

To provide fault tolerance to the generic crossbar, we propose to have two paths to reach an output port of the crossbar. This can be achieved by using additional smaller sized demultiplexers and multiplexers. Figure 6 shows the proposed architecture for a 5x5 crossbar. For a 5x5 crossbar, the additional circuitry is composed of four demultiplexers (one 1:3 demultiplexer, three 1:2 demultiplexers) and five 2:1 multiplexers. With the help of these additional demultiplexers and multiplexers in the protected crossbar there exist two different paths to reach a specific output port.

Consider for example, out 3 in Figure 6. It can be reached through either multiplexer M3 or M2. When a fault affects the corresponding multiplexer (M3) of the out 3, using M2 and configuring the additional demultiplexer (D1) and the multiplexer (P3) accordingly, flit(s) can still reach out 3. In addition to the select signals required for the multiplexers (M1, M2, M3, M4 and M5), the select signals to these new demultiplexers (D1, D2, D3 and D4) and multiplexers (P1, P2, P3, P4 and P5) are also controlled by the switch allocator. In the fault-free scenario, the protected crossbar behaves just like the baseline crossbar. In the event of a permanent fault, the secondary path can be used to reach the appropriate output port.

For an input VC to use the secondary path, it should arbitrate for a different output port in its SA stage. Assume M3 is faulty and an input VC needs to transmit flits to out 3. To reach out 3, the input VC needs to go through M2 (secondary path). So, the input VC needs to arbitrate for access to out 2 to gain access to M2. To make this feasible, we add a state field named ‘SP‘ to every input VC. This field contains the output port the input VC needs to arbitrate for in SA stage in order to reach the correct output port.

When the RC unit finishes execution and finds out that the output port the flits of an input VC need to go is unreachable using the regular path, it updates the 'SP' (secondary path) field with the appropriate output port that should be used. The 'FSP' (secondary path flag) field is set to indicate that the secondary path needs to be used. In our example of faulty M3, the 'SP' field of the input VC is updated to hold the identification of out 2, thus arbitrating for access to out 2 and reaching out 3 using M2, D1 and P3.

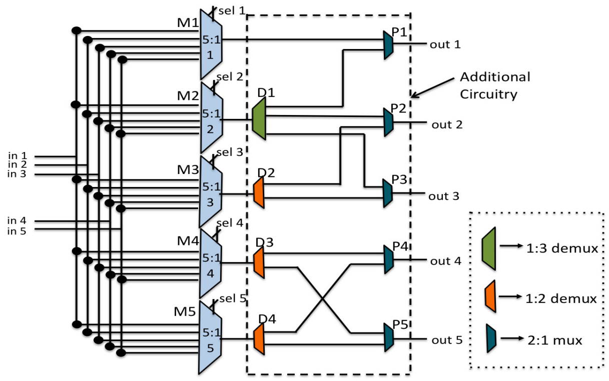


Figure 6: Modified Crossbar

VI. Performance Analysis

In this section, we present our performance analysis of PFTR (Figure 7) with respect to area, power overhead, latency impact, critical path and reliability. PFTR design can be applied to a router with any radix in any kind of topology. We choose a generic **5x5** router architecture with each input port consisting of **4 VCs**.

A. Synthesis

To study the impact of the proposed architectural modifications, we developed pipeline stages of both the baseline router and PFTR in Verilog. Using Cadence Encounter RTL Compiler, we synthesized the baseline and PFTR pipeline stages at 45nm technology. Based on the synthesis results, PFTR increases the area and average power (dynamic+static) consumption by *28%* and *29%* with respect to that of the baseline router. To detect faults, we choose to use the fault detection mechanism proposed in [18]. Incorporating fault detection mechanism into PFTR results in an area and average power overhead of *31%* and *30%* with respect to the baseline router. Thus, the area overhead incurred by the fault detection mechanism is *3%*.

1) Critical Path Analysis: To determine the affect on the critical path, we synthesized individual pipeline stages of both the baseline and PFTR at varying clock periods. We identify the critical path of a stage by finding out the specific clock period that results in zero slack time. Since for the RC stage, the RC unit is duplicated, there is negligible impact on the critical path of this particular stage. However, the critical paths of VA stage, SA stage and XB stages are increased by 20%, 10% and 25% with respect to the baseline stages.

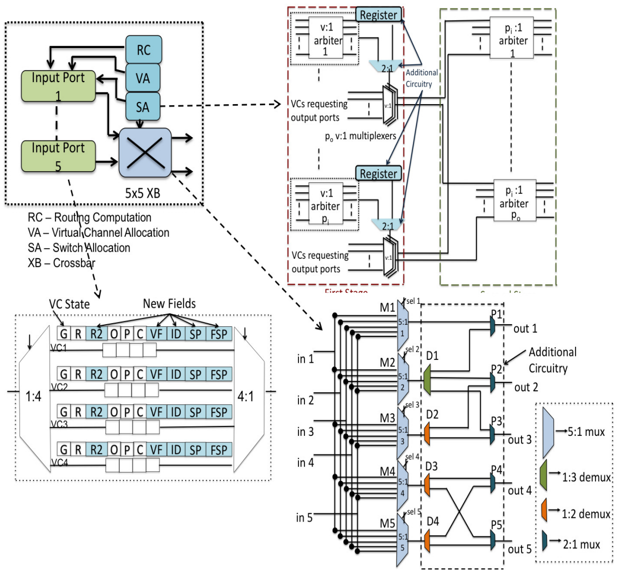


Figure 7: PFTR Architecture

B. Latency Analysis

In this section, we study the impact on latency caused due to faults into different pipeline stages. We use GEM5 [21], a cycle accurate simulator to simulate an *8*x*8* mesh based NoC with uniform random synthetic traffic. GARNET [20], integrated into GEM5 is used to model the pipeline of the baseline router. We run two different experiments as part of our latency analysis.

In the first experiment, we inject uniform random synthetic traffic at various injection rates (*0.01, 0.03, 0.05, 0.07 and 0.1packets/node/cycle*). Each packet is composed of 5 flits and each flit is 16 bytes wide. For each injection rate, we run the simulation **10** times for a period of *500,000* cycles each and calculate the average latency. The average latency is calculated as the ratio of total network latency to the total number of flits received at the end of simulation. We inject a total of *24* faults into the pipeline stages of *20* randomly chosen routers of the 8x8 NoC. Due to the faults in the pipeline stages, the respective additional circuitry is used to finish the execution of that pipeline stage. From the simulation results, we observe that the latency of the fault injected NoC has increased by *3%* (Figure 8) on average compared to the fault-free NoC.

In the second experiment, we vary the number of faults injected (*4, 8, 16, 24 and 32*) into the NoC. The traffic is injected at a rate of *0.1packets/node/cycle*. Each simulation is run **10** times for a period of *500,000* cycles each. From the simulation results, we observe that as the number of faults increase from *4* to *32*, the increase in the average latency due to the faults increases from *0.5%* to *4.5%*.

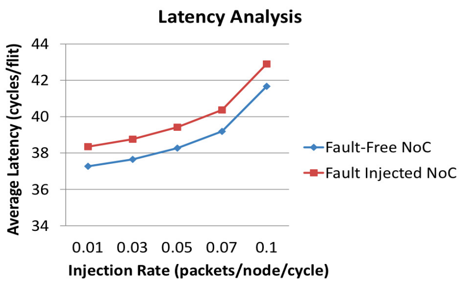


Figure 8: Impact on Latency

C. Fault Tolerance

In this section, we study the fault tolerance of PFTR in comparison to the baseline router using Silicon Protection Factor (SPF) [13], [14], [19] as the reliability metric. SPF is defined as the ratio of mean number of faults required to cause a failure and the area overhead incurred due to additional circuitry. Higher SPF value indicates higher reliability. SPF is an appropriate metric to evaluate PFTR because, it takes into account the two important parameters namely number of faults to cause a failure and the area overhead required by the protection mechanism. We estimate the SPF of a **5x5** PFTR with each input port comprised of **4 VCs**. We calculate the mean number of faults to cause failure by calculating an average of the minimum and maximum number of faults to cause failure.

1) RC Stage: To provide fault tolerance, we propose to duplicate the RC unit of each input port. Since each input port has a duplicate RC unit, the router can tolerate a maximum of **5** faults, where each fault has affected the functionality of the original RC unit of an input port. On the other hand, a minimum of **2** faults, one in the original RC unit and the other in the duplicate RC unit of the same input port would result in a failure because routing computation can no longer be performed at that particular input port.

2) VA Stage: For this stage, tolerance is achieved by borrowing arbiters from a different VC of the same input port. There are 4 VCs per input port. So, a packet in the VC of an input port can borrow arbiters from three other VCs of the same input port. So, the VA can tolerate **3** faults per input port. Thus, a maximum of **15** faults can be tolerated in the VA of a 5-input port router. If the arbiters associated with all the VCs of an input port are faulty, then virtual channel allocation can no longer be performed for a packet at that input port and will result in failure. Since there are 4 VCs per input port, the minimum number of faults that result in failure is **4**.

3) SA Stage: In the first stage, a packet in an input VC uses the arbiter associated with that input port to participate in the switch allocation. If the arbiter is affected by a fault, it can use the bypass path. There are **5** arbiters in the first stage of SA in a 5-input port router. Thus, a maximum of **5** faults, one per arbiter can be tolerated. On the other hand, a minimum of **2** faults, one in the arbiter of an input port and the other in the bypass path of the same input port would result in a failure because switch allocation can no longer be performed at that particular input port. The fault tolerance methodology for the second stage is provided by the fault tolerance methodology of crossbar. Since we considered faults in the first stage of switch allocation, in the calculation of SPF we choose to consider faults in the crossbar instead of faults in the second stage of switch allocation.

4) XB Stage: A packet in an input VC uses the regular path to reach the output port of a crossbar. If the multiplexer associated with the regular path is faulty, secondary path can be used. A fault in the secondary path will result in failure. Thus a minimum of **2** faults will cause failure. Careful observation of Figure 6 reveals that the maximum number of faults that can be tolerated is also **2**. For example, if M2 and M4 (Figure 6) are each affected by a fault, the crossbar can still remain functional with the help of additional circuitry. A fault in M1, M3 or M5 or in the additional circuitry will result in a failure.

D. SPF of the Proposed PFTR

The minimum number of faults to cause the pipeline to fail is the least of the minimum number of faults to cause failure in the individual stages of the pipeline calculated as $min\{2(RC), 4(VA), 2(SA), 2(XB)\}$, which is **2** faults. The maximum number of faults that can be tolerated by the router pipeline is calculated as the sum of the maximum faults tolerated by each individual stage, which results in $5(RC)+15(VA)+5(SA)+2(XB) = 27$ faults. Note, that this is the total number of faults that can be tolerated. An additional fault in any of the pipeline stages would result in failure. So, the maximum number of faults to cause failure is, $27 + 1 = 28$. Thus, the mean number of faults to cause failure is $(2+28)/2 = 15$ faults. The area overhead incurred by the additional circuitry is 31%. Thus, using the definition, SPF of PFTR can be calculated as $15/1.31 = 11$.

It is evident that the number of VCs has a significant affect on the SPF value of PFTR. The SPF value of PFTR increases further beyond 11 if the number of VCs per input port is increased beyond 4. If the number of VCs per input port is decreased to *2*, the SPF value of PFTR is 7.

Table I shows the area overhead, number of faults to cause failure and SPF values of existing methodologies namely BulletProof [13], Vicis [14], and RoCo [15] with respect to PFTR. We could not compare power values across the methodologies due to lack of power data in the existing methodologies. BulletProof evaluates different designs and calculates their SPF values. We choose a design that incurs approximately the same area overhead as PFTR for doing the comparison. The authors of RoCo did not provide the area overhead ("N/A") and the number of faults that can be tolerated by their design. Based on the RoCo design, we calculated the mean number of faults to cause failure as *5.5*. Using the definition of SPF, the SPF value of a design is always smaller than the mean number of faults to cause failure in the design. Hence, the SPF value of RoCo is lesser than *5.5*. Comparing the SPF values (Table I), we can conclude that PFTR has a higher SPF value than the existing methodologies indicating a better reliability.

Table I: Comparing PFTR with other fault tolerant routers

Architecture	Area	# Faults to cause failure	SPF
BulletProof	52%	3.15	2.07
Vicis	42%	9.3	6.55
RoCo	N/A	5.5	<5.5
PFTR	***31%***	***15***	***11.4***

VII. Conclusion

The focus of this work is to design a permanent fault tolerant router pipeline. We considered each individual pipeline stage of a NoC router, studied the affect of a permanent fault on that stage and proposed a fault tolerant mechanism for that stage. The proposed methodology involves adding minimal extra circuitry to provide a better fault tolerance. Synthesis results reveal that the additional circuitry results in an area and power overhead of 31% and 30% with respect to the baseline router. Silicon Protection Factor based evaluation shows that the proposed router (PFTR) is 11 times more reliable than the baseline router. PFTR also provides better reliability at lower overhead as compared to BulletProof, Vicis and RoCo methodologies.

Acknowledgment

This research was supported by NSF awards ECCS-0725765 and CCF-0915537.

References

[1] S. Borkar, "Thousand core chips: a technology perspective," in *Proceedings of the 44th annual Design Automation Conference (DAC)* pp. 746-749, 2007.

[2] L. Benini and G. Micheli, "Networks on chips: a new SoC paradigm," *IEEE Computer*, 35: pp. 70-78, 2002.

[3] K. Bernstein, "Nano-metere scale CMOS devices (tutorial presentation)," in *5th International Symposium on Quality of Electronic Design*, 2004.

[4] S. Borkar, "VLSI design challenges for gigascale integration (keynote address)," in *18th International Conference on VLSI Design*, 2005.

[5] R. Barsky and I.A. Wagner, "Electromigration-dependent parametric yield estimation," in *Proceedings of the 11th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp. 121-124, 2004.

[6] G.V. Groeseneken, "Hot carrier degradation and ESD in submicrometer CMOS technologies: How do they interact?," *IEEE Transactions on Device and Materials Reliability*, 1(1): 23-32, 2001.

[7] S. Oussalah and F. Nebel, "On the oxide thickness dependence of the time-dependent-dielectric breakdown," in *IEEE Proceedings of Electron Devices Meeting*, pp. 42-45, 1999.

[8] J. Zieglar, "Terrestrial cosmic rays," *IBM Journal of Research and Development*, 40(1): 19-39, 1996.

[9] K.J. Kuhn, "Reducing variation in advanced logic technologies: Approaches to process and design for manufacturability of nanoscale CMOS," in *IEEE Proceedings of Electron Devices Meeting*, pp. 471-474, 2007.

[10] T. May and M. Woods, "Alpha-particle-induced soft errors in dynamic memories," *IEEE Transactions on Electronic Devices*, 26(1): 2-9, 1979.

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

[12] L.S. Peh and W.J. Dally, "A delay model and speculative architecture for pipelined routers," in *Proceedings of the 7th International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 255-266, 2001.

[13] K. Constantinides, et al. "BulletProof: A defect-tolerant CMP switch architecture," in *Proceedings of the 12th International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 5-16, 2006.

[14] D. Fick, et al. "Vicis: a reliable network for unreliable silicon," in *Proceedings of the 46th annual Design Automation Conference (DAC)*, pp. 812-817, 2009.

[15] J. Kim et al. "A Gracefully Degrading and Energy-Efficient Modular Router Architecture for On-Chip Networks," in *Proceedings of the 33rd International Symposium on Computer Architecture (ISCA)*, 2006.

[16] T. Lehtonen, P. Liljeberg and J. Plosila, "Online reconfigurable self-timed links for Fault-tolerant NoC," in *VLSI Design*, pp.1-13, 2007.

[17] M. Koibuchi, H. Matsutani, H. Amano and T.M. Pinkston, "A Lightweight Fault-Tolerant Mechanism for Network-on-Chip," in *Proceedings of 2nd ACM/IEEE International Symposium on Networks-on-chip*, pp. 13-22, 2008.

[18] A. Prodromou, A. Panteli, C. Nicopoulos and Y. Sazeides, "NoCAlert: An On-Line and Real-Time Fault Detection Mechanism for Network-on-Chip Architectures," in *Proceedings of the 45th annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 60-71, 2012.

[19] K. Constantinides et al. "Architecting a Reliable CMP Switch Architecture, " *ACM Transactions on Architecture and Code Optimization (TACO)*, 4(1), 2007

[20] N. Agarwal, T. Krishna, L.S. Peh and N.K. Jha, "Garnet: A detailed on-chip network model inside a full system simulator," in *Performance Analysis of Systems and Software (ISPASS)*, pp. 33-42, 2009.

[21] N. Binkert et al. "The gem5 simulator," *SIGARCH, Computer Architecture News 39*, 2:1-7, 2011.

[22] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip," *ACM Computing Surveys (CSUR)*, 38(1), 2006.

[23] S. Vangal et al. "An 80-tile 1.28 TFLOPS network-on-chip in 65nm CMOS," in *IEEE Solid-State Circuits Conference (ISSCC)*, pp. 98,589, 2007.

[24] J.H. Collet, A. Louri, V.T. Bhat and P. Poluri, "ROBUST: a new self-healing fault-tolerant NoC router," in *Proceedings of the 4th International Workshop on Network on Chip Architectures (NoCArc)*, 2011.

[25] Q. Yu, M. Zhang and P. Ampadu, "Exploiting Inherent Information Redundancy to Manage Transient Errors in NoC Routing Arbitration," in *Proceedings of the 5th ACM/IEEE International Symposium on Networks-on-Chips (NOCS)*, pp. 105-112, 2011.