2014 IEEE 28th International Parallel & Distributed Processing Symposium

An Improved Router Design for Reliable On-Chip Networks

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

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

***Abstract*—Aggressive technology scaling into the deep nanometer regime has made the Network-on-Chip (NoC) in multicore architectures increasingly vulnerable to faults. This has accelerated the need for designing reliable NoCs. To this end, we propose a reliable NoC router architecture capable of tolerating multiple permanent faults. The proposed router achieves a better reliability without incurring too much area and power overhead as compared to the baseline NoC router or other fault-tolerant routers. Reliability analysis using Mean Time to Failure (MTTF) reveals that our proposed router is *six* times more reliable than the baseline NoC router (without protection). We also compare our proposed router with other existing fault-tolerant routers such as BulletProof, Vicis and RoCo using Silicon Protection Factor (SPF) as a metric. SPF analysis shows that our proposed router is more reliable than the mentioned existing fault tolerant routers. Hardware synthesis performed by Cadence Encounter RTL Compiler using commercial 45nm technology library shows that the correction circuitry incurs an area overhead of *31%* and power overhead of *30%*. Latency analysis on a 64-core mesh based NoC simulated using GEM5 and running SPLASH-2 and PARSEC benchmark application traffic shows that in the presence of multiple faults, our proposed router increases the overall latency by only *10%* and *13%* respectively while providing better reliability.**

***Keywords*-Network-on-Chip, Reliability, Area, Power**

I. Introduction

Aggressive technology scaling [1] has enabled single chips with billion transistors. This increased availability of on-chip resources has led to the widespread use of Chip Multiprocessors (CMPs) or multicore architectures. The advent of CMPs has forced the design shift from a computation centric to a communication centric architectures. The need for an efficient and scalable communication mechanism to continue to harvest the benefits of CMPs has led to the evolution of Network-on-Chip (NoC) [2], [4] paradigm.

NoC is a packet based interconnection network that scales bandwidth with network size. It facilitates simultaneous communication between multiple cores on a single chip and plays an effective role in decoupling the intra-core computation from the inter-core communication. Major components of a NoC include routers and links.

Technology scaling [3] into deep nanometer regimes has decreased the size of a transistor and has increased the vulnerability of transistors to different fault mechanisms. Transistors are predominantly vulnerable to two kinds of faults namely, permanent and transient faults.

A permanent fault continues to affect the operation of a circuit from the time of its inception. Electromigration [5], Hot carrier degradation [6] and Time Dependent Dielectric Breakdown [7] are typical sources for permanent faults. A transient fault affects the operation of a circuit for a smaller period of time typically in the order of one clock cycle. Thermal radiation from cosmic rays [8], process variation [9] and alpha particles from the packaging material [10] are common sources for transient faults. This increased susceptibility of transistors to fault mechanisms has made reliability a front runner in the design of future systems.

NoC is neither immune to permanent and transient faults nor is unaffected by the adverse increase in 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. Faults in the NoC may result in critical problems like lost packets, network deadlock and disconnected network, which leads to severe degradation of the performance of on-chip communication [11]. In this work, we propose a fault tolerant router capable of tolerating multiple permanent faults in its pipeline. The proposed router employs minimum correction circuitry and exploits idle time of existing resources to accomplish fault tolerance.

II. Generic Network-on-Chip Router

A. *NoC Router Architecture*

Figure 1 [12] shows the architecture of a P input port, P output port router with V virtual channels per port. The control logic of a NoC router comprises of Routing Computation (RC) unit, Virtual Channel Allocation (VA) unit and the Switch Allocation (SA) unit. A central crossbar (XB) connects the input and output ports of the router.

For efficient utilization of router resources, data traverses in the NoC in the form of *flits* (flow control information units). Typically, a packet is segmented into a *head flit*, single or multiple *body flits* and a *tail flit*. Head flit allocates router resources to the packet, body flit(s) contain the payload of the packet and tail flit frees the router resources allocated to the packet.

1530-2075/14 $31.00 © 2014 IEEE
DOI 10.1109/IPDPS.2014.39

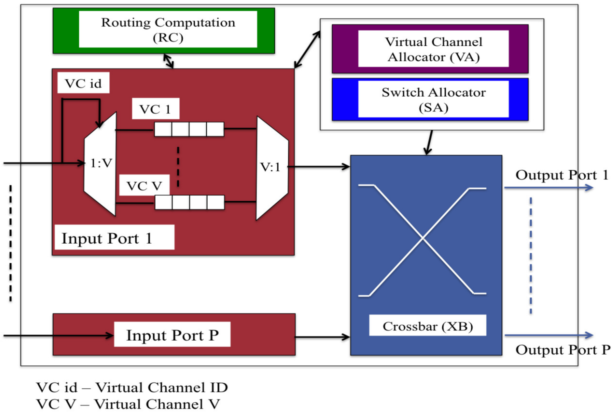


Figure 1: **Generic NoC Router Architecture**

B. NoC Router Pipeline

NoC router pipeline (Figure 2) is a 4-stage pipeline comprised of Routing Computation, Virtual Channel Allocation, Switch Allocation and Crossbar stages.

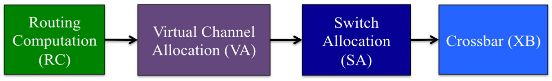


Figure 2: **Router Pipeline**

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

*2) **VA Stage**:* This is the second stage in the pipeline and is active upon completion of RC stage. This stage also operates only on head flits. Figure 3a [13] shows the architectural block diagram of a two-stage separable virtual channel allocator. In the *first stage*, based on the result of RC, every input VC with 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 VC at the downstream router compete with each other. The input VC that wins the arbitration in the second stage is allocated the VC at the downstream router.

*3) **SA Stage**:* This is the third stage in the pipeline and is active for head, body and tail flits. SA unit is responsible for determining which input VC from an input port gets to transmit a flit through the crossbar in the next cycle. Figure 3b [13] shows the architectural block diagram of a two-stage separable switch allocator. In the *first stage*, the SA unit decides which VC of an input port gets to transmit its flit through crossbar. In the *second stage*, competition between different input VCs trying to gain access to the same output port of the crossbar is resolved. The input VC that wins the arbitration in the second stage gets to transmit its flit through the crossbar in the next cycle.

*4) **XB Stage**:* This is the final stage in the pipeline and is active for head, body and tail flits. Crossbar connects the input and output ports thus facilitating flit traversal from a VC of an input port to an output port. Figure 3c shows the architecture of a p_ixp_o crossbar, where p_i and p_o are the number of input and output ports of the crossbar. SA unit is responsible for generating control signals to the multiplexers in crossbar. Input output port connections of the crossbar are configured every cycle based on the winners in SA stage.

C. Input Port Architecture

Figure 3d [12] shows the internal architecture of an input port of a router with four VCs where each VC can hold up to four flits. Each VC is associated with state fields namely ’G’, ’R’, ’O’, ’P’ and ’C’. ’G’ field indicates the status of the VC with respect to the pipeline stage in the current cycle. ’R’ field is used to store the result of RC unit. ’O’ field stores the result of VA unit that indicates which VC in the downstream router is the current packet headed to. ’P’ field indicates the read/write pointers in the VC and ’C’ field indicates the credit count.

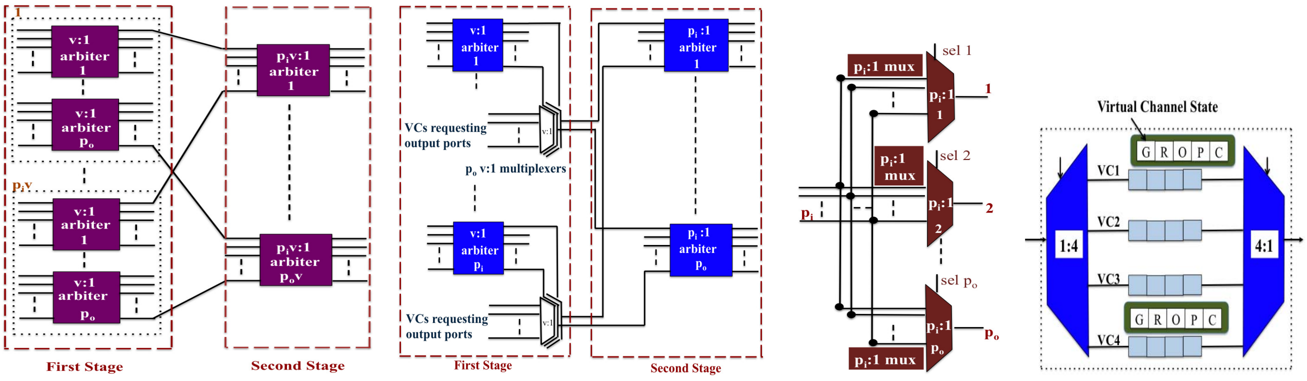
III. Related Work

Here, we provide an overview of the architectures of the fault tolerant routers presented in BulletProof [14], Vicis [15], and RoCo [16] as they also tackle the issue of tolerating permanent faults in the NoC router pipeline.

Constantinides et al. proposed the BulletProof [14] router that employs N-modular redundancy (NMR) technique to provide fault tolerance. Redundancy based techniques such as NMR demand for the existence of N copies of the protected component. Providing fault tolerance to a NoC router via NMR technique increases the silicon area required to fabricate the protected router by N times. Since the area of a design has a linear relationship with the possible number of faults, employing redundancy based technique to achieve fault tolerance is not always efficient.

Fick et al. proposed Vicis [15] methodology that can tolerate faults at network and router level. Faults at the network level are tolerated via an input port swapping algorithm and an adaptive routing algorithm that updates the routing tables to redirect traffic around faulty links. Within the router, a bypass bus is used to tolerate faults in the crossbar and low overhead Error Correcting Codes (ECC) are used to tolerate faults in the datapath.

Kim et.al proposed RoCo [16] router architecture, which enables the router to be decomposed into individual row and column components. Decomposition is facilitated by using decoupled parallel arbiters and smaller crossbars for row and column connections. Fault tolerance for routing computation stage is achieved by using look ahead routing and for switch allocation stage is achieved by sharing arbiters from



(a) Virtual Channel Allocator (b) Switch Allocator (c) Generic Crossbar (d) Input Port Architecture

Figure 3: **Baseline Router Pipeline Components**

virtual channel allocation stage. It cannot tolerate faults in virtual channel allocation and crossbar stages. 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 in a degraded fashion with the fault free component.

Our proposed router is different from these methodologies in that it provides protection for each individual stage of the pipeline. It exploits idle time of existing resources and employs minimum correction circuitry to achieve tolerance from multiple faults as explained next.

IV. Contributions

The proposed router architecture is capable of tolerating multiple permanent faults in its pipeline. It is designed by making minimal architectural modifications to the 4-stage pipeline of a baseline NoC router. These architectural modifications involve adding extra circuitry to individual pipeline stages of a NoC router and taking advantage of the inherent redundancy 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 protected router pipeline will be able to tolerate **four** permanent faults. The main contributions of this paper can be summarized as:

1) A router architecture that can tolerate multiple permanent faults in the routing pipeline.
2) Performance analysis involving area, power, latency and critical path of the proposed architecture with respect to the baseline router architecture.
3) Estimating the improvement in reliability of the proposed router architecture in comparison to the baseline router architecture using MTTF [17].
4) Comparing the reliability of the proposed router with other existing fault-tolerant NoC routers such as BulletProof [14], Vicis [15] and RoCo [16] using SPF [14].

V. Proposed Reliable NoC Router

Since each stage in the router pipeline has a distinct and very specific role, the proposed architectural modifications to each stage are tailored according to the responsibilities of that particular stage. Note that in this paper, we focus on fault tolerance and not on fault detection. We assume that faults can be detected by using one of the many existing fault detection mechanisms [18]. Also, we only consider faults in different stages of the router pipeline. Faults in the other components of a router such as multiplexers and buffers are studied in [23] and are out of scope of this paper.

A. *Routing Computation Stage*

Each input port has its own RC unit. A permanent fault in the RC unit results in the calculation of a faulty output port. Since the execution of remaining pipeline stages is dependent on the result of this stage, the entire pipeline is affected as a result of permanent fault in RC unit.

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 [24]. The fundamental logic block required for implementing XY routing protocol is a comparator. To provide fault tolerance to this stage, we propose to have a redundant RC unit for each input port. The duplicate RC unit can be turned on and used upon detection of a fault in the original unit.

B. *Virtual Channel Allocation Stage*

Since the VA unit is composed of two stages, we consider fault scenario in each stage independently.

*1) **First VA 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 unit, the $v:1$ arbiter corresponding to the output port computed by the RC unit is used to choose a single empty

VC from the available empty VCs at the downstream router connected to that output port. When an arbiter associated with an input VC is faulty, it will not be able to arbitrate for a VC at the downstream router for the corresponding output port whose arbiter is faulty. As a result, the head flit in that input VC would not be allocated a VC at the downstream router resulting in the flit being blocked at the input VC. A blocked flit in a network leads to serious ramifications. To avoid this, we propose to use the following methodology.

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 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, they can be shared between VCs. The affected VC can choose another VC it intends to borrow the arbiters from by scanning through the 'G' state field (indicates state of the VC) of all the other input VCs and picking out the first VC it encounters that is either idle or in switch allocation state. Thus, by using another virtual channel's arbiters, virtual channel allocation can be performed for the head flit residing in the affected virtual channel.

One of the two possible scenarios can arise while sharing arbiters between virtual channels of an input port. Consider the input port architecture shown in Figure 3d, and assume that VC1's arbiters are faulty and is requesting to use arbiters of VC2 in both the scenarios.

Scenario 1 - When VC1 requests to use arbiters of VC2, if the arbiters of VC2 are idle, the delay involved in borrowing the arbiters of VC2 is the same as the affect on critical path of the virtual channel allocator (Section VI-B).

Scenario 2 - When VC1 requests to use arbiters of VC2, if VC 2 is non-empty and is in VA stage like VC1, in addition to the affect on critical path, there will be an additional latency of 1 cycle. This is because; the arbiters of VC2 first perform allocation for the head flit in VC2 and on successful allocation, in the next cycle can be used for allocation for the head flit in VC1. Since the head flit in VC1 had to wait for the arbiters of VC2, the waiting induces additional latency.

Scenario 2, where two different VCs of the same input port are attempting to perform virtual channel allocation in the same cycle arises only if there was an unsuccessful virtual channel allocation encountered by one of the input virtual channels 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 (input port) into the router. Two flits cannot enter two different VCs of an input port at the same time. These flits have to come one after the other thus making the input VC the flit has entered first trigger the router pipeline earlier than the other VC where the other flit has entered in the following cycle.

This unsuccessful virtual channel allocation 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. In this high traffic rate situation, latency gets affected regardless of the fault. However, in the presence of a fault, sharing arbiters to provide fault tolerance 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.

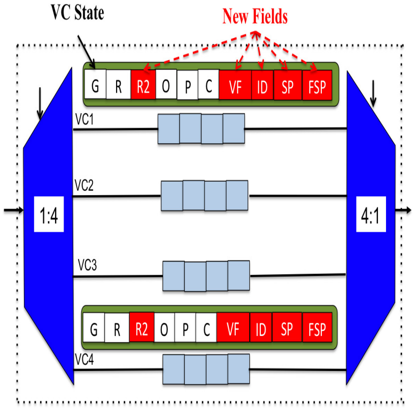


Figure 4: **Modified Input Port Architecture**

Consider a VC (e.g., VC1) that intends to use the arbiters of another VC (e.g., VC2) of the same input port. VC1 initiates the process by placing its RC result in the 'R2' field of VC2, its identification in the 'ID' field of the VC2 and setting the 'VF' (virtual channel flag) field of VC2 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 VC2 have successfully allocated an empty VC in the downstream router to the head flit in VC1, the VA unit resets the 'R2', 'ID' and 'VF' fields of VC2. 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' and 'FSP' fields are used to provide fault tolerance for switch allocator and crossbar and will be described later.

*3) **Second VA Stage:*** 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 being blocked at the current router because, the flit can be allocated another VC belonging to the required output port in the downstream router by using the associated non-faulty arbiter. However, to accomplish this, virtual channel allocation must be recomputed causing an additional latency of 1 cycle. Thus by utilizing the inherent redundant resources (multiple VCs), a fault in second stage can be tolerated without the involvement of any additional circuitry.

C. Switch Allocation Stage

Since the SA unit is composed of two stages, we consider fault scenario in each stage independently.

*1) **First SA Stage**:* The first stage (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 the winning VC traverses through the crossbar in the next cycle.

Consider the scenario when a $v : 1$ arbiter is faulty. Due to the fault, the arbiter cannot choose a VC from the associated input port and as a result the VC 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 in those VCs will be blocked.

To avoid this situation, 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 chooses an input VC as the winner without arbitration. We call this winner as default 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 be selected as the winner using bypass path as the other input. For example, consider there are *v* virtual channels namely VC1, VC2 ... VC*v* and let us say that when using the bypass path, VC2 (any VC can be considered) is always chosen as the winner. So, the inputs to the additional *2:1* multiplexer will be the output of the arbiter and the VC identification of VC2. The best way to choose the default winner is to make every input VC as default winner at different point of time. In other words, for a period of time VC1 would be the default winner followed by VC2 being the default winner for the next period of time and so on. By doing this, we can avoid the potential starvation problem that could arise from static allocation of the default winner.

When an arbiter is faulty, the *2:1* multiplexer is configured such that the value from the register (identification of VC2) is forwarded instead of the output of arbiter. If VC2 is not empty and is in SA stage, flits in VC2 can traverse through the crossbar if VC2 wins the arbitration in second stage. However, if VC2 is empty and there are flits in other virtual channels, then flits from any other VC that belongs to the same input port as VC2 can be *transferred* into VC2 by implementing a logic that can execute read/write operation.

When transferring flits from one virtual channel (e.g., VC1) to another virtual channel of the same input port (e.g., VC2), in addition to the flits, state fields of VC1 also need to be transferred into the state fields of VC2. After the transfer of flits and state fields is completed, the flits (initially in VC1) now in VC2 can traverse through the crossbar when VC2 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.

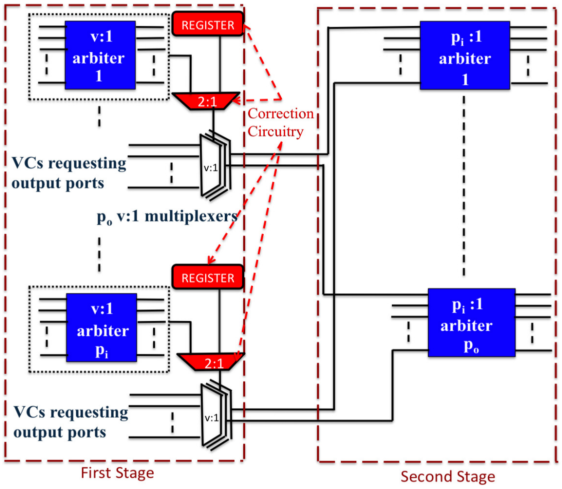


Figure 5: **Modified Switch Allocator**

*2) **Second SA Stage**:* The second stage (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 output port *x* is faulty and there exists a secondary path to reach output port *x* by using the arbiter associated with output port *y*. Now, using the fault-free arbiter associated with output port *y*, flits of any input port can reach output 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 subsection where we describe the fault tolerant methodology for crossbar. The fault tolerant methodology used for crossbar also helps in tolerating a fault in this stage.

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 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.

To provide fault tolerance to the generic crossbar architecture, we propose to have two paths to reach a specific 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.

Consider for example, output port 3 (out 3) in Figure 6. It can be reached through either multiplexer M3 or M2. When a fault affects the corresponding multiplexer (M3) of 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 if necessary in order 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. Consider the scenario where 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 output port 2 (out 2) in order to gain access to M2. To make this feasible, we add a state field named 'SP' (secondary path) 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 its execution and finds out that the output port the flits of this input VC need to go is unreachable using the regular path, it updates the 'SP' 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.

VI. Performance Analysis

The proposed router design can be applied to a router with any radix in any kind of topology. The performance analysis is conducted on a router with **5 input ports**, **5 output ports** with each input port consisting of **4 VCs**.

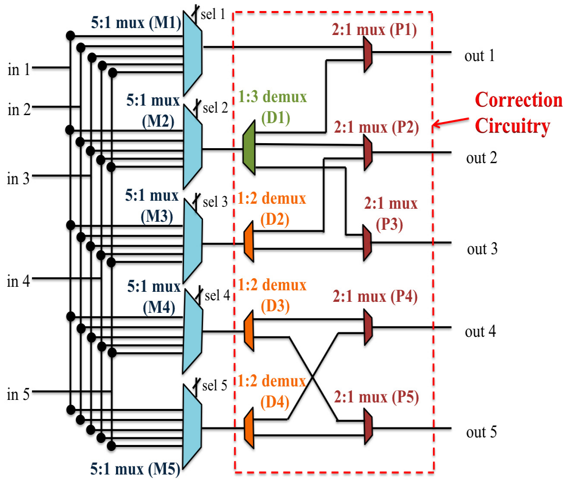


Figure 6: **Modified Crossbar**

A. Area and Power Analysis

We developed pipeline stages for the baseline and protected router in Verilog and synthesized these stages using Cadence Encounter RTL Compiler at 45nm technology. Based on the synthesis results, the correction circuitry increases the area and average power (dynamic+static) consumption of the protected router by *28%* and *29%* with respect to that of the baseline router. Incorporating fault detection mechanism [18], the resulting area and power overhead is *31%* and *30%* with respect to the baseline router.

B. Critical Path Analysis

To determine the effect of the correction circuitry on the critical path of each stage, we synthesized individual pipeline stages of both the baseline and protected router at varying clock periods. The critical path of an individual stage is calculated by finding out the specific clock period that results in zero slack time. Since RC stage employs spatial redundancy, there is negligible impact on the critical path of this stage. However, due to the correction circuitry, critical paths of VA, SA and XB stages have increased by 20%, 10% and 25% with respect to the baseline stages.

VII. Reliability Improvement Estimation Using MTTF

In this section, we estimate the reliability improvement achieved by the proposed fault tolerant router with respect to the baseline router using Mean Time to Failure (MTTF) [17]. MTTF of a component is calculated as inverse of Failures in Time (FIT) of that component (Equation 1), where FIT of a component is defined as the number of failures encountered by the component per billion hours.

$$MTTF_{component} = \frac{1}{FIT_{component}} \qquad (1)$$

A. *Failure in Time Estimation Model*

To calculate the FIT of a component, we use the architectural level reliability modeling framework presented in Shin et al [19]. This reliability framework proposes a concept referred to as Failure in time Of Reference Circuit (FORC) that facilitates designers to quantify the failure rate of a component without having to deal with its low level circuit and technology specific implementation details. This framework provides equations to compute the FIT rate of a component due to time dependent dielectric breakdown (TDDB), electromigration (EM) and negative bias temperature instability (NBTI) fault mechanisms respectively. Among these three fault mechanisms, we are interested in TDDB because the effects of EM and NBTI on a digital circuit can be recovered by moderating the stress on the circuit, but, no easy recovery process exists for TDDB.

TDDB results in the formation of a conductive path in the gate oxide resulting in leakage current through the gate [20]. The flow of current through this conductive path opposes the current of the logic state driving the effected FET resulting in slow zero to one or one to zero transitions. These slow transitions make the device susceptible to timing violations resulting in a failure. Considering a continuous device stress (100% duty cycle) on either a pFET or an nFET along the critical path, the FORC for TDDB is given by [19].

$$\boldsymbol{FORC_{TDDB} = \frac{10^9}{A_{TDDB}}.V_{dd}{}^{a-bT}.e^{-\frac{X+\frac{Y}{T}+ZT}{kT}}} \tag{2}$$

where, A_{TDDB}, a, b, X, Y and Z are fitting parameters, k is the Boltzmann's constant, T is the operating temperature of the circuit in $Kelvin$ and V_{dd} is the operating voltage in volts. Equation 2 and the above mentioned fitting parameters have been derived through a thorough set of experimental results performed by Wu et al in [20].

Using Equation 2 and the fitting parameter values obtained from [21] at an operating voltage (V_{dd}) of $1V$ and an operating temperature (T) of $300K$, we calculate the FIT of the reference circuit. With the $FORC_{TDDB}$ value calculated, the FIT due to TDDB of a field-effect-transistor (FET) can be calculated as [19]

$$\boldsymbol{FIT_{TDDB\_per\_FET} = dutycycle * FORC_{TDDB}} \tag{3}$$

Once the $FIT_{TDDB\_per\_FET}$ is calculated, FIT of a logic gate can be calculated as $number\_of\_transistors_{logic\_gate} * FIT_{TDDB\_per\_FET}$. For example, to calculate the FIT of a 2-input AND gate (FIT_{AND2}), we first calculate $FORC_{TDDB}$ and $FIT_{TDDB\_per\_FET}$ values using equations 2 and 3. Then the resulting $FIT_{TDDB\_per\_FET}$ is multiplied by the transistor count of a 2-input AND gate, i.e., *6* to get the value of FIT_{AND2}.

To calculate the FIT of a digital circuit with multiple logic gates, Sum-of-Failure-Rates (SOFR) [22] model is used. SOFR model assumes that the failure rate of a digital circuit is the sum of failure rates of the individual gates in the circuit. For example, the FIT of a logic circuit consisting of an AND gate, an OR gate and an XOR gate is calculated as the sum of FIT rates of AND, OR and XOR gates.

Using this FIT calculation methodology and assuming a SOFR [22] model, we proceed to calculate the FIT values of the individual pipeline stages of a NoC router.

B. *FIT estimation of NoC Router Pipeline stages*

FIT values of NoC router pipeline stages are calculated for a **5x5** router with each input port consisting of **4 VCs** in a **8x8** mesh topology that employs XY routing protocol. The fundamental component of VA, SA and XB stages can be deduced to be an arbiter, arbiter and a multiplexer. To implement XY routing, the fundamental component used is a comparator. The RC unit needs two comparators, one for the X-direction and one for the Y-direction. Since, the router is in a 8x8 mesh, there are 64 possible destinations and hence the RC unit needs two **6-bit comparators**.

Table I lists the fundamental components (FC) used in the design of the pipeline stages, their corresponding FIT values, the total number of FCs involved in a particular stage and the FIT value of the stage calculated based on SOFR model.

Table I: **FIT values of baseline pipeline stages**

Stage	FC	FIT_{FC}	# FCs	FIT_{stage}
RC	6-bit comparator	11.7	10	(11.7*10) = 117
VA	4:1 Arbiter 20:1 Arbiter	7.4 36.7	100 20	(100*7.4 + 20*36.7) = 1478
SA	4:1 mux 4:1 Arbiter 5:1 Arbiter	4.8 7.4 9.3	25 5 5	(25*4.8 + 5*7.4 + *9.3) = 203
XB	32-bit 5:1 mux	204.8	5	(204.8*5) = 1024

C. *FIT estimation of Correction Circuitry*

*1) **RC Stage**:* This stage employs spatial redundancy to provide fault tolerance. Since the baseline router has 5 RC units, the correction circuitry also includes 5 RC units.

*2) **VA Stage**:* New state fields namely 'R2', 'VF', and 'ID' are added to facilitate arbiter sharing. There are 20 input VCs per router and these state fields are added per VC. 'R2' field is implemented by a 3-bit D flip-flop (DFF), 'VF' field by a 1-bit DFF and 'ID' field by a 2-bit DFF.

*3) **SA Stage**:* Correction circuitry (Figure 5) for this stage involves 5 2:1 multiplexers and 5 2-bit registers. Each register is implemented as a 2-bit DFF. Two new state fields namely, 'SP' and 'FSP' are also added per virtual channel to provide fault tolerance. 'SP' field is implemented by a 3-bit DFF and 'FSP' field is implemented by a 1-bit DFF.

4) XB Stage: Correction circuitry (Figure 6) for this stage involves 5 32-bit 2:1 multiplexers, 3 32-bit 1:2 demultiplexers and 1 32-bit 1:3 demultiplexer.

Table II shows the FIT rates (per 10^9 hours) of the correction circuitry for each individual pipeline stage.

Table II: **FIT rates of Correction Circuitry**

Stage	Components	FIT
RC	10 6-bit comparators	117
VA	20 3-bit DFF ('R2'), 20 1-bit DFF ('VF'), 20 2-bit DFF ('ID')	60
SA	5 2:1 muxes, 5 2-bit DFF (register), 20 3-bit DFF ('SP'), 20 1-bit DFF ('FSP')	53
XB	5 2:1 muxes, 3 1:2 demuxes, 1:3 demux	416

D. *MTTF of Protected Router*

Using the SOFR model, the MTTF of the pipeline in the baseline router is calculated as,

$$\boldsymbol{MTTF_{baseline} = \frac{10^9}{117 + 1478 + 203 + 1024} \approx} \textbf{ 354,358 hours} \tag{4}$$

The proposed router continues to work as long as either the baseline stages or the corresponding correction circuitry are fault free. Using SOFR model, the FIT of the entire correction circuitry is $117+60+53+416 = 646$ (Table II). MTTF of a system that has two components with failure rates λ_1 and λ_2 requires only one the two components to continue to function can be calculated as

$$\boldsymbol{MTTF_{system} = \frac{1}{\lambda_1} + \frac{1}{\lambda_2} + \frac{1}{\lambda_1 + \lambda_2}} \tag{5}$$

Applying Equation 5 to the proposed fault tolerant router with $\lambda_1 = 2822$ (FIT of the baseline router pipeline) and $\lambda_2 = 646$ (FIT of the correction circuitry), MTTF of the protected router can be calculated as

$$\boldsymbol{MTTF_{protected} = \frac{10^9}{2822} + \frac{10^9}{646} + \frac{10^9}{2822 + 646} \approx} \textbf{ 2,190,696 hours} \tag{6}$$

Using equations 4 and 6, the reliability improvement in the protected router can be calculated as the ratio of MTTF of the protected router and the baseline router,

$$\boldsymbol{\frac{MTTF_{protected}}{MTTF_{baseline}} = \frac{2,190,696}{354,358} \approx 6} \tag{7}$$

From equation 7, we can prove that the protected router is approximately **6** times more reliable than the baseline router.

VIII. Reliability Comparison using SPF

Here, we study the reliability improvement of protected router in comparison to other fault tolerant routers namely BulletProof [14], Vicis [15] and RoCo [16] using Silicon Protection Factor (SPF) [14] as a metric. SPF is defined as the ratio of mean number of faults required to cause a failure and the area overhead incurred due to correction circuitry. Higher SPF value indicates higher reliability.

We estimate the SPF of a **5x5** protected router with each input port consisting of **4 VCs**. We calculate the mean number of faults to cause failure by calculating the average of minimum and maximum number of faults to cause failure.

A. *RC Stage*

A fault in the original RC unit of an input port can be tolerated by using the duplicate RC unit of that 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 one RC unit per input port. On the other hand, a minimum of **2 faults**, one in the original and the other in the duplicate RC unit of the same input port would result in failure because, routing computation can no longer be performed at that input port.

B. *VA Stage*

There are 4 VCs per input port. A packet in the VC of an input port can borrow arbiters from three other virtual channels of the same input port. So, the VA unit can tolerate 3 faults per input port, resulting in a maximum of **15 faults** that can be tolerated. If the arbiters associated with all the 4 VCs of an input port are affected by 4 different faults, it would result in failure because virtual channel allocation can no longer be performed at that particular input port. Thus, the minimum number of faults to cause failure is **4 faults**.

C. *SA Stage*

In the first stage of SA unit, if an arbiter is affected by a fault, the bypass path can be used to overcome the faulty arbiter. There are 5 arbiters in the first stage of SA unit and 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 and the other in the bypass path of the same input port would result in failure because switch allocation can no longer be performed at that input port.

Fault tolerance to the second stage of SA unit is provided by the same methodology that provides fault tolerance to crossbar. In other words, either the crossbar or the second stage of SA unit can be protected but not both. Since we considered faults in the first stage of SA unit in the calculation of SPF, we choose to consider faults in the crossbar instead of in the second stage of SA unit.

D. *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, the secondary path is used. A fault in the secondary path will result in failure. Thus, a minimum of **2 faults** will cause failure. From Figure 6, it can be deduced that a maximum of **2 faults** can be tolerated. For example, if multiplexers M2 and M4 are each affected by a fault, the crossbar is still functional with the help of correction circuitry. A fault in any other multiplexer (M1, M3 or M5) or in the correction circuitry will result in failure.

E. *SPF of Protected Router*

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

This SPF value increases further beyond 11 if the number of VCs per input is increased beyond 4. If the number of VCs per input port is decreased to 2, the SPF value is 7.

Table III shows the area overhead, number of faults to cause failure and SPF values of BulletProof, Vicis, RoCo and our proposed router. BulletProof evaluates different designs and calculates their SPF values. We choose a design that incurs approximately the same area overhead as our proposed router for doing the comparison. The authors of BulletProof and Vicis used an experimental approach through simulations to determine the number of faults to cause failure. For our router, we used a theoretical approach to calculate the number of faults to cause failure based on the fault tolerant methodology and the circuit necessary to implement the fault tolerant methodology for each pipeline stage. For RoCo, the authors did not provide the number of tolerated faults and the area overhead (N/A) of the fault tolerant methodology. Based on the proposed methodology of RoCo, we deduced the number of faults to cause failure as 5.5. Using the definition of SPF, the SPF of RoCo is < 5.5. Comparing the SPF values, we can deduce that our proposed router is more reliable than the existing methodologies.

IX. Latency Analysis

Here, we study the impact on latency caused by the correction circuitry of protected router in the process of accomplishing fault tolerance. We use GEM5 [26], a cycle accurate simulator to simulate a *8x8* mesh based NoC. GARNET [25], integrated into GEM5 is used to model the baseline router pipeline.

Table III: **SPF Comparison of our Proposed Router with BulletProof, Vicis and RoCo**

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
Proposed Router	***31%***	***15***	***11.4***

The ideal way to simulate faults is to inject them based on the FIT values described in section VII-B. Since the derived FIT values are very small, the applications need to run for a long time to inject faults. To accelerate simulations, we inject faults based on a uniform random variable with a mean of 10 million cycles. A fault is injected into a pipeline stage after 10 million cycles of its operation.

We simulated an 8x8 mesh-based NoC using GEM5 with each core associated with its own cache and directory. The cache coherence protocol used in the NoC is $MOESI\_CMP\_directory$. Figures 7 and 8 show the latency results in both the fault-free and fault injected scenario on a 8x8 NoC executing SPLASH-2 [27] and PARSEC [28] benchmark applications. Overall NoC latency has increased by 10% and 13% for SPLASH-2 and PARSEC benchmark applications respectively in the presence of multiple faults.

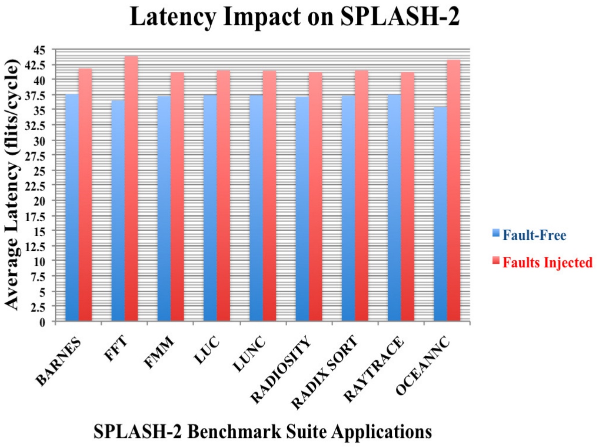


Figure 7: **Impact of faults on latency running SPLASH-2**

X. Conclusion

We considered each individual pipeline stage of a NoC router and added minimal correction circuitry to provide better fault tolerance. Synthesis results reveal that the correction circuitry results in an area and power overhead of 31% and 30% with respect to the baseline router. Reliability analysis using MTTF metric reveals that the proposed

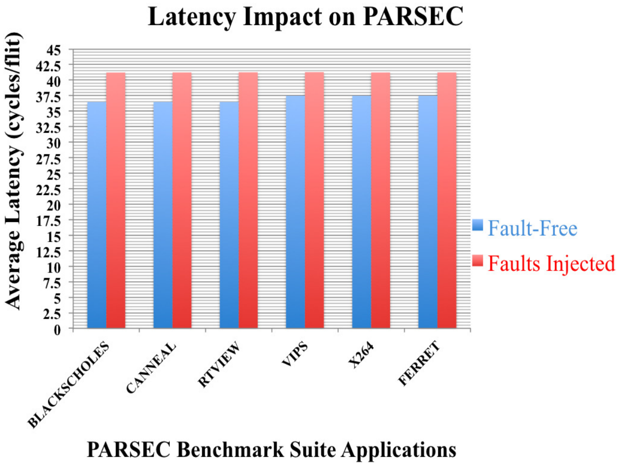


Figure 8: **Impact of faults on latency running PARSEC**

router is six times more reliable than the baseline router. Further, we use SPF as a metric to evaluate the reliability improvement of our proposed router with respect to other fault tolerant routers namely BulletProof, Vicis and RoCo. SPF calculations reveal that our proposed router has an SPF value of 11 suggesting that it provides better reliability with less overhead compared to the other fault tolerant routers.

Acknowledgment

This research was supported by NSF awards ECCS-0725765, CCF-0915537 and CNS-1318997.

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

[4] W.J. Dally and B. Towles, ”Route packets, not wires: on-chip interconnection networks,” in *Proceedings of the Design Automation Conference (DAC)*, 2001.

[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] K. Aisopos et.al, ”Enabling system-level modeling of variation-induced faults in networks-on-chips,” in *Proceedings of the Design Automation Conference (DAC)*, 2011.

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] D.P. Gaver, ”Time to failure and availability of paralleled systems with repair,” in *IEEE Transactions on Reliability*, 12(2), 30-38, 1963.

[18] A. Prodromou, et.al, ”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] J. Shin, et.al, ”A Framework for Architecture-level Lifetime Reliability Modeling,” in *Proceedings of the 37th Annual IEEE/IFIP conference on Dependable Systems and Networks (DSN)*, pp. 534-543, 2007.

[20] E.Y. Wu, et.al, ”CMOS scaling beyond the 100-nm node with silicon-dioxide-based gate dielectrics,” in *IBM Journal of Research and Development*, vol. 46, no. 2/3, pp. 287-298, 2002.

[21] J. Srinivasan, et.al, ”The case for lifetime reliability-aware microprocessors,” in *Proceedings of International Symposium on Computer Architecture (ISCA)*, pp. 276-287, 2004.

[22] K. Trivedi, ”Probability and Statistics with Reliability, Queueing, and Computer Science Applications,” Prentice Hall, 1982.

[23] J.H. Collet, et.al, ”ROBUST: a new self-healing fault-tolerant NoC router,” in *Proceedings of the 4th International Workshop on Network on Chip Architectures (NoCArc)*, 2011.

[24] Q. Yu, et.al , ”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.

[25] N. Agarwal, et.al, ”Garnet: A detailed on-chip network model inside a full system simulator,” in *Performance Analysis of Systems and Software (ISPASS)*, pp. 33-42, 2009.

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

[27] S.C. Woo, et.al, ”The SPLASH-2 Programs: Characterization and Methodological Considerations,” in *Proceedings of the 22nd International Symposium on Computer Architecture (ISCA)*, pp. 24-36, 1995.

[28] C. Bienia, et.al, ”The PARSEC Benchmark Suite: Characterization and Architectural Implementations,” in *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*, 2008.