Microprocessors and Microsystems 38 (2014) 1037–1045

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

Variability-tolerant routing algorithms for Networks-on-Chip

Eman Kamel Gawish [a], M. Watheq El-Kharashi [b,*], M.F. Abu-Elyazeed [a]

[a] Department of Electronics and Electrical Communications Engineering, Cairo University, Giza 12613, Egypt
[b] Department of Computer and Systems Engineering, Ain Shams University, Cairo 11517, Egypt

ARTICLE INFO

Article history:
Available online 1 September 2014

Keywords:
Networks-on-Chip (NoC)
Process variations
Routing algorithms

ABSTRACT

This paper proposes variability-tolerant routing algorithms for mesh-based Networks-on-Chip (NoC). Different NoC routing algorithms are modified, from variability perspective, to route flits through links with lower failure probability. The algorithms considered in this study are XY, West-First, Negative-First, and Odd–Even routing algorithms. To evaluate our variability-tolerant routing algorithms, a cycle-accurate simulator, NoCTweak, is used to measure how tolerant the resultant NoCs are against process variations. Results reflect the efficiency of our routing algorithms to overcome the process variation problems in modern fabrication technologies. For example, variability-tolerant West-First routing algorithm achieves up to 56% reduction in NoC overall failure rate.

1. Introduction

Networks-on-Chip (NoC) have appeared as good alternatives to global interconnects because of their optimized electrical properties, such as better performance in terms of power, delay, bandwidth, and scalability, compared to buses and global interconnects. Efficient NoC designs address the issues of performance, silicon area consumption, power/energy efficiency, reliability, and variability. These issues are the fundamental design drivers for an efficient NoC design [1].

The inability to precisely control the manufacturing process might result in unpredictable behavior of both device and wire, which in turn causes performance and power variations as well as an error-prone behavior. This becomes particularly important for modern fabrication technologies with feature sizes smaller than 65 nm. The reasons for higher variation effects at smaller feature sizes can be summarized as follows:

1. The process-resulting variations become comparable to the full length or width of the device.
2. The feature size approaches the fundamental dimensions, such as the size of atoms and the wave-length of the light, which are used for patterning lithography masks.

Process variations mainly result from front-end and back-end fabrication processes. The front-end fabrication processes are those involved in the fabrication of devices, whereas back-end processes are those involved in the fabrication of interconnects. Both the front-end and the back-end fabrication processes can have either random or systematic variability effects. Systematic variation effects have spatial correlation and usually arise from lithography, Chemical Mechanical Polishing (CMP), or etching fabrication steps. These effects cause systematic variations in gate length, threshold voltage, or Line Width Roughness (LWR). Random variability effects do not have any spatial correlation and are random in nature, like Random Dopant Fluctuation (RDF), Oxide Thickness Fluctuation (OTF), or Line Edge Roughness (LER) [2]. As technology scales down, identical NoC links encompass current and delay variations due to CMOS fabrication process variations causing, error at the link receiver, which we consider a link failure [3–5].

The paths the flits are routed through on an NoC are determined by the used routing algorithm. Different paths go through different NoC links having different link delay variations, which results in different link failure probabilities. In this paper, we consider the average link failure probability that flits go through for a certain traffic pattern as an indicator of how certain NoC routing algorithms are prone to process-induced delay variations.

Routing algorithms can be classified into two types: deterministic and adaptive. In deterministic routing, a path is completely determined by its source and destination addresses. On the other hand, a routing technique is called adaptive if, given a source and a destination addresses; the path taken by a particular flit depends on dynamic network conditions (e.g., congested links due to traffic variability and minimum length to destination) [6]. In this paper, we work with one of the main deterministic routing algorithms (ordered XY routing) and three adaptive routing algorithms

* Corresponding author.
E-mail address: watheq.elkharashi@eng.asu.edu.eg (M.W. El-Kharashi).

http://dx.doi.org/10.1016/j.micpro.2014.08.002
0141-9331/© 2014 Elsevier B.V. All rights reserved.

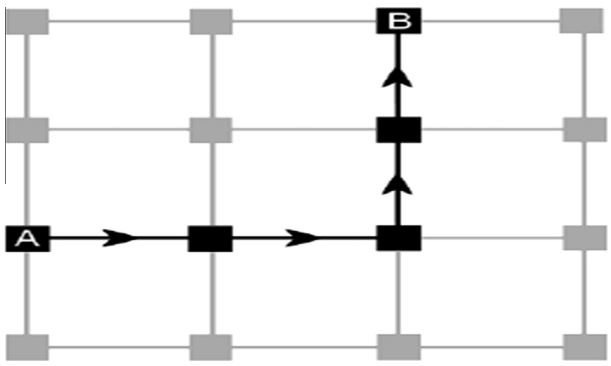

Fig. 1. XY routing from router A to router B [7].

(West-First, Negative-First, and Odd–Even minimal routing). These routing algorithms are briefed as follows:

- *XY routing*: XY routing is a dimension-order routing, which routes flits first in *x*, or horizontal direction, to the correct column and then in y, or vertical direction, to the receiver, as shown in Fig. 1. Addresses of the routers are their *XY*-coordinates [7].
- *West-First routing*: West-First routing algorithm prevents all turns to the west, so the flits going to the west must be first transmitted as far to the west as necessary. Routing flits to the west is not possible later [8]. Allowed turns in West-First routing are shown in Fig. 2(a).
- *Negative-First routing*: Negative-First routing algorithm allows all turns except turns from the positive direction to the negative direction. Flit routing to the negative directions must be done before anything else [9]. Allowed turns in Negative-First routing are shown in Fig. 2(b).
- *Odd–Even routing*: Odd–Even routing is a deadlock-free turn model, which prohibits turns from the east to the north and from the east to the south at tiles located in even columns and turns from the north to the west and the south to the west at tiles located in odd columns [10].

In this paper, we add the link failure probability to the adaptive conditions to be considered at routing (in addition to flit length, buffer size, and nearer dimension output port). We use an open source cycle accurate NoC simulator, NoCTweak [11], to simulate different traffic patterns with modified NoC routing algorithms.

The main contributions of this work are as follows:

1. Modeling, on the system level, the NoC link failure resulting from random and systematic process variations at certain technology node and mesh size.
2. Modifying XY, West-First, Negative-First, and Odd–Even routing algorithms in NoCTweak to consider link failure probability when routing, to obtain variability-tolerant routing algorithms.
3. Proposing the NoC failure rate as a measure of tolerance against process variations under certain NoC technology node, buffer size, injection rate, mesh size, traffic pattern, and routing algorithm.

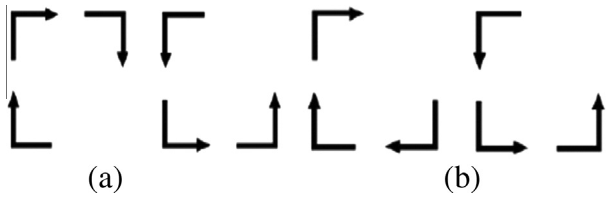

Fig. 2. Allowed turns in: (a) West-First routing and (b) Negative-First routing [8].

The rest of this paper is organized as follows. Related work is discussed in Section 2. Section 3 presents our NoC failure model. Section 4 shows the proposed variability-tolerant routing. Section 5 describes NoCTweak and analyzes the simulation results. We draw conclusions and give ideas for future work in Section 6.

2. Related work

There have been several work in the literature addressing process variations effects on NoC. Initially variability effects on NoC routers were addressed [1,12,13]. As technology scales down, interconnect delay dominates gate delay. Hence, research considered variability effects on NoC links [2–5]. Subsequently, research aimed for fault-tolerant routing algorithms [14–23].

Nicopoulos et al. presented the first comprehensive evaluation of NoC susceptibility to process variability effects and proposed an array of architectural improvements in the form of a new router design to increase resiliency to these effects [1]. By process variation exploration, Nicopoulos et al. identified the contribution of each major router stage to the overall critical path delay. The contribution to delay was used to guide the proposed modifications to improve process variation resilience without adversely affecting performance.

Sivaswamy and Bazargan tolerated variations by a variation-aware router that was optimized according to statistical critical delay path [12]. They also proposed a modification to the clock network to deliver programmable skews to different flip-flops, tolerating variations within the clock paths.

Konstantinos et al. proposed a circuit-level fault modeling tool to capture run-time process-induced random delay variations and their corresponding system-level faults. The tool points out to the router components that need resilient design [13].

Mehranzadeh and Hoodgar presented a fault-aware routing algorithm scheme called FAXY based on XY routing algorithm [6]. With FAXY routing, a flit first traverses along the *X* direction and then along the *Y* direction. When a flit traverses along the *X* direction and a link is masked due to a permanent fault, it traverses along the *Y* direction in order to increase the overall network throughput and prevent flit losses.

Wu et al. proposed an improved routing algorithm which tolerates a single link fault in 2D mesh NoC [14]. Their algorithm is deadlock-free, yet is subject to flit loss.

Ebrahimi et al. proposed deadlock-free fully adaptive routing algorithm using virtual channels along the *X* and *Y* directions [15]. Ebrahimi proposed fault-tolerant routing tolerating the fault probability resulting from random variations only. In this paper, we propose a set of variability-tolerant routing algorithms that consider link failure probability resulting from both systematic and random variations.

The turn model is originated from Glass and Ni work in [16]. They introduced three adaptive routing algorithms: West-First, North-Last, and Negative-First. These routing algorithms eliminate deadlocks without adding virtual channels by prohibiting some global turns. The turn model also results in routing algorithms that are ideal for fault tolerance, live-lock free, and highly adaptive [16–18]. Glass and Ni also proposed a turn-based fault-tolerant routing algorithm in [17] that is based on modification of the Negative-First routing algorithm. This algorithm can deal with any one-faulty-router topology. In this proposal, each routing function depends on the coordinates (Y, X) of the router, the packet destination, the input channels, and the size of the mesh.

The Position-Route method proposed in [19] is a deterministic routing algorithm. When the destination is to the west, packets are first sent in the west direction up to the column of the destination, and then, sent to either north or south. Otherwise, packets are first sent to either north or south up to the row of the destination, and then, sent to east. When faulty components exist, faulty rings are formed to surround them. Since fault rings are rectangles, some non-faulty components may be included and are considered to be faulty.

Dependable routing, where the routing algorithm used in every router can dynamically detour a faulty router or a faulty link based on local fault information, is proposed in [20]. Dependable routers also use programmable delay elements for the matched delays, whose delay values can be modified on power-on reset timing. This is useful for tolerating static and/or dynamic degradation faults. C-elements placed at the output of memory elements (e.g., flip-flops and latches) hide transient errors resulting from noise. The proposed on-chip networks contain efficient link fault detection circuits. Duplicated control circuits are also designed in order to improve transient fault tolerance. They also studied NoC failure rate when using XY routing in presence of injected transient faults. Their definition of NoC failure rate depends on number of received flits compared to some threshold at certain flit injection rate, injected transient error rate, and NoC size. Simulation results show that the duplication technique for control circuits improve fault tolerance with a low performance overhead [20].

Multiple link faults are tolerated in the routing algorithms of the multi-chip NoC platform proposed in [21]. Two routing algorithms are extended to handle the inter-cluster communication using a restricted set of links. The gateway routing that is an extension of the Glass–Ni fault tolerance routing [17] and the position route method [19] is also extended to route flits between clusters in the multi-chip NoC.

Sharifi and Kandemir proposed a variation-aware source routing algorithm for a heterogeneous NoC, where each router has a different operating latency, as a result of random process variations [23]. Their routing scheme computed the best path for each communication between a sender and a receiver, based on the inherent speed of the routers (dictated by process variations) and the current traffic pattern. The proposed variations-aware routing algorithm considered delay variations of the routers, but as technology scales down interconnect delay has proven to dominate gate delay. Hence, our variability-tolerant routing in this paper focuses on variability effects on NoC links.

Sarangi et al. modeled the effects of random and systematic processes on the probability of error for a microprocessor, mainly measuring variability in logic and SRAM components [24]. Their model provided the failure rates of micro architectural blocks, as a function of frequency, and the amount of variations. While Sarangi et al. consider only systematic and random variations resulting from front-end fabrication processes to get the normalized gate delay, in this paper we consider systematic and random variations resulting from both the front-end and the back-end fabrication processes to get the total interconnect delay variations [3].

Our work in [3] modeled systematic and random geometry variations of devices and interconnects. Using geometry variations, we calculated systematic (spatial) and random variations of the interconnect delay. We showed that process variations caused each link in NoC floor-plan to have different delay depending on its position on the die.

In this paper we estimate a process variation-induced link failure probability of each link in a mesh NoC. Then, we use the link failure probability to route flits to the links with lower probability of failure by introducing variability-tolerant routing algorithms. We modify four routing algorithms XY routing, West-First minimal routing, Negative-First minimal routing, and Odd–Even minimal routing. We simulate NoC failure rate with different traffic patterns like bit reversal (bitr), bit complement (bitc), regional, neighbor, tornado, transpose, random, and hotspot [25].

3. System level model of NoC failure

Our work in [3] shows how each link across NoC encompasses random and systematic delay variations. Link delays and their violations from timing constraints are used in this paper to estimate a probability of failure for each link across the NoC. Averaging the accumulated failure probabilities for all the links in NoC gives an indicator of the probability of link failure across NoC. Finally, simulating an NoC with different traffic patterns, routing algorithms, and mesh sizes at different technology nodes estimates the overall NoC failure rate.

3.1. NoC link failure probability

By link failure probability, we mean the probability of the link to fail to meet the design timing constraints resulting in failure or error at the receiver. The link failure probability is described by

$$P_{e(link)} = P(T_{link} > T_{nominal}) \tag{1}$$

where $P_{e(link)}$ is the link failure probability, T_{link} is the link delay, and $T_{nominal}$ is the link nominal delay that meets the delay constraints.

In case of using the statistical link design in [3], there is still a link failure probability defined as the probability the link delay fails to meet the delay constraints, in spite of using the statistical delay guard. That is:

$$P_{e(link)} = P(T_{link} > (T_{nominal} + \sigma Td_{total})) \tag{2}$$

where σTd_{total} is the total delay deviation due to process variability, which is used as statistical delay guard/margin in variability-tolerant NoC link design [3].

We define the NoC average link failure probability $P_{e(NoC)}$, as the sum of link failure probabilities across the NoC links divided by the number of links (n_{links}).

$$P_{e(NoC)} = \frac{\sum P_{e(link)}}{n_{links}} \tag{3}$$

3.2. NoC case study

The case study below describes how the proposed model in [3] is developed to estimate the failure probability for each link across an NoC.

A number of samples ($n_{samples} = 1000$) of NoCs are generated, where each sample represents a different die with different variability-induced random and spatial variations. The statistical failure probability of each link in NoC is evaluated. A case study for 4×4 mesh at 65 nm is shown in Fig. 3. The link failure probability is calculated by summing the number of times an NoC link in a floor-plan fails to meet the delay constraints due to the systematic (spatial) process variations at its position on the die, within $n_{samples}$, and dividing by the number of samples $n_{samples}$. The link failure probabilities across the NoC for the case study in Fig. 3 range from 1.6% to 3.4%. Summing the individual link failure probabilities and dividing by the number of links in NoC (n_{links}), the average NoC link failure probability is 2.37% for this case study.

We estimate the average link failure probability as the supply voltage VDD scales from 2.0 V to 0.6 V, as shown in Fig. 4. We note that, as technology scales down from 65 nm to 22 nm, the link failure probability increases. It can also be seen that, as the supply voltage decreases, the variations in threshold voltage, for example, are more pronounced, hence more links fail to meet the design timing constraints and the average NoC link failure probability increases. This agrees with the results in [24], where the error rate resulting from process variations in logic increases as the supply voltage decreases.

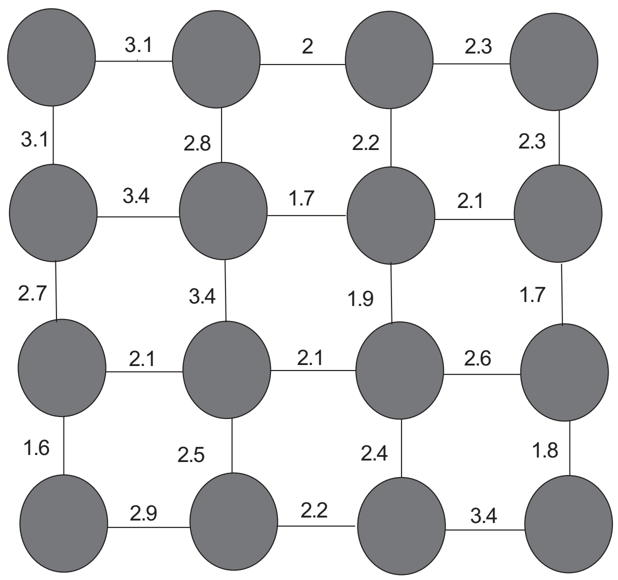

Fig. 3. NoC link failure probability percentage across a case study of 4 × 4 NoC mesh at 65 nm.

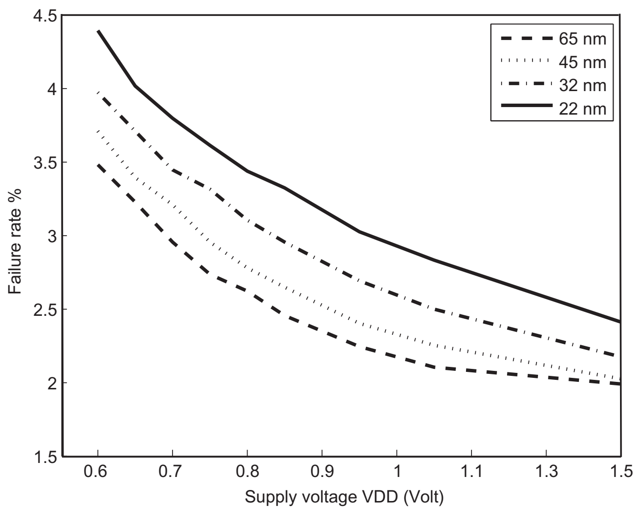

Fig. 4. Average NoC link failure rate versus supply voltage.

4. Variability-tolerant routing

We modify NoC routing algorithms to consider link failure probability when routing flits to their destinations. We work with four routing algorithms XY, West-First, Negative-First, and Odd–Even routing. The routing modifications are discussed in this section.

4.1. Variability-tolerant XY routing

The XY routing algorithm is modified to consider link failure probability when routing is to be in the *X* direction (to the east or to the west) by checking the link failure probability in the *Y* direction of the destination and routing to it, if having lower failure probability. On the other hand, if routing is to be in the *Y* direction, it does not consider link failure probability, as shown in Algorithm 1.

Algorithm 1. Variability-tolerant XY routing.

```
1: INPUTS: local x, y coordinates of the router:
   _local_x, _local_y,
2: Destination router: x, y coordinates: _dst_x, _dst_y,
3: Probabilities of link failure in x direction:
   p_x_east, p_x_west,
4: Probabilities of link failure in y direction:
   p_y_north, p_y_south.
5: OUTPUTS: a direction to route flits: LOCAL, EAST, WEST,
   NORTH, and SOUTH.
6: if _dst_x ==_local_x then        ▷ routing in Y direction is free
7:   if _dst_y ==_local_y then return LOCAL;
8:   else if _dst_y < _local_y then return NORTH;
9:   else return SOUTH;
10:    end if
11: else if _dst_x < _local_x then
12:          ▷ routing in X direction considers link failure
   probability
13:    if _dst_y ==_local_y then return WEST
14:             ▷ routing in the same row is free
15:    else if _dst_y < _local_y then
16:                ▷ if destination is up before routing to west,
17:                  ▷ compare p_x_west to p_y_north.
18:      if p_x_west < p_y_north then return WEST;
19:      else return NORTH;
20:          ▷ route to the direction with lower failure
   probability
21:      end if
22:    else
23:                ▷ if destination is down before routing,
24:                  ▷ compare p_x_west to p_y_south
25:      if p_x_west < p_y_south then return WEST;
26:      else return SOUTH;
27:      end if
28:    end if
29: else    ▷ routing to the EAST considers link failure
   probability
30:    if _dst_y ==_local_y then return EAST;
31:              ▷ routing in the same row is free
32:    else if _dst_y < _local_y then
33:              ▷ if destination is up before routing to west,
34:                ▷ compare p_x_east to p_y_north.
35:      if p_x_east < p_y_north then return EAST;
36:      else return NORTH;
37:          ▷ route in the direction with lower link failure
   probability
38:      end if
39:    else
40:              ▷ if destination is down before routing,
41:                  ▷ compare p_x_east to p_y_south
42:      if p_x_east < p_y_south then return EAST;
43:      else return SOUTH;
44:              ▷ route to the direction with lower failure
   probability
45:      end if
46:    end if
47: end if
```

4.2. Variability-tolerant West-First routing

The adaptive West-First routing is modified to consider link failure probability when routing is to be in the east direction by considering the link failure probability in the Y direction of the destination and routing to it, if it has lower failure probability. While routing in the Y or west directions are free and do not consider link failure probability, as shown in Algorithm 2.

Algorithm 2. Variability-tolerant West-First routing.

```
1: INPUTS: local x, y coordinates of the router:
   $\_local\_x,\ \_local\_y$,
2: Destination router: x, y coordinates: $\_dst\_x,\ \_dst\_y$,
3: Probabilities of link failure in x direction:
   $p\_x\_east,\ p\_x\_west$,
4: Probabilities of link failure in y direction:
   $p\_y\_north,\ p\_y\_south$,
5: Adaptive routing conditions flag: ARC.
6: OUTPUTS: a direction to route flits: LOCAL, EAST, WEST,
   NORTH, and SOUTH.
7: if $\_dst\_x ==$_local_x then    ▷ routing in Y direction is free
8:    if $\_dst\_y ==$_local_y then return LOCAL;
9:    else if $\_dst\_y < \_local\_y$ then return NORTH;
10:     else return SOUTH;
11:     end if
12: else if $\_dst\_x < \_local\_x$ return WEST;
13:                  ▷ routing to the west is free
14: else    ▷ routing to the EAST considers link failure
   probability
15:     if $\_dst\_y ==$_local_y then return EAST;
16:                  ▷ routing in the same row is free
17:        else if $\_dst\_y < \_local\_y$ then
18:               ▷ if destination is up, consider link failure
   probability
19:        if ARC and $(p\_x\_east < p\_y\_north)$ then return EAST;
20:        else return NORTH;
21:        end if
22:     else    ▷ if destination is down, consider link failure
   probability
23:        if ARC and $(p\_x\_east < p\_y\_south)$ then return EAST;
24:        else return SOUTH;
25:        end if
26:     end if
27: end if
```

Since we still prohibit routing to forbidden turns, variability-tolerant West-First routing is assumed to also be deadlock-free as West-First routing.

4.3. Variability-tolerant Negative-First routing

The adaptive Negative-First routing is modified to consider link failure probability when the routing is to route the flit to west or to the east direction, by adding link failure probability to the adaptive routing conditions, as shown in Algorithm 3.

Algorithm 3. Variability-tolerant Negative-First routing.

```
1: INPUTS: local x, y coordinates of the router
   $\_local\_x,\ \_local\_y$,
2: Destination router: x, y coordinates: $\_dst\_x,\ \_dst\_y$,
3: Probabilities of link failure in x direction:
   $p\_x\_east,\ p\_x\_west$,
4: Probabilities of link failure in y direction:
   $p\_y\_north,\ p\_y\_south$,
5: Adaptive routing conditions flag: ARC.
6: OUTPUTS: a direction to route flits: LOCAL, EAST, WEST,
   NORTH, and SOUTH.
7: if $\_dst\_x ==$_local_x then    ▷ routing in Y direction is free
8:    if $\_dst\_y ==$_local_y then return LOCAL;
9:    else if $\_dst\_y < \_local\_y$ then return NORTH;
10:     else return SOUTH;
11:     end if
12: else if $\_dst\_x > \_local\_x$ then
13:            ▷ routing to the EAST considers link failure
   probability
14:     if $\_dst\_y ==$_local_y then return EAST;
15:                  ▷ routing in the same row is free
16:     else if $\_dst\_y < \_local\_y$ then
17:               ▷ if destination is up considers link failure
   probability
18:        if ARC and $(p\_x\_east < p\_y\_north)$ then return EAST;
19:        else return NORTH;
20:        end if
21:     end if
22: else    ▷ routing to the WEST considers link failure
   probability
23:     if $\_dst\_y ==$_local_y then return WEST;
24:               ▷ routing in the same row is free
25:     else if $\_dst\_y > \_local\_y$ then
26:               ▷ if destination is down consider link failure
   probability
27:        if ARC and $(p\_x\_west < p\_y\_south)$ then return WEST;
28:        else return SOUTH;
29:        end if
30:     end if
31: end if
```

Since we still prohibit routing to forbidden turns, variability-tolerant Negative-First routing is assumed to also be deadlock-free as Negative-First routing.

4.4. Variability-tolerant Odd–Even routing

The Odd–Even routing is modified to consider link failure probability when routing is to be in the even column to the west direction, as shown in Algorithm 4.

Algorithm 4. Variability-tolerant Odd–Even routing.

```
1: INPUTS: local x, y coordinates of the router
   $\_local\_x,\ \_local\_y$,
2: Destination router: x, y coordinates: $\_dst\_x,\ \_dst\_y$,
3: Probabilities of link failure in x direction:
   $p\_x\_east,\ p\_x\_west$,
4: Probabilities of link failure in y direction:
   $p\_y\_north,\ p\_y\_south$.
5: OUTPUTS: a direction to route flits: LOCAL, EAST, WEST,
   NORTH, and SOUTH.
6: $dist\_x = \_dst\_x - \_local\_x$;    ▷ distance from local to
   destination
7: $dist\_y = \_dst\_y - \_local\_y$;
8: if $\_dst\_x ==$_local_x then    ▷ routing in Y direction is free
9:    if $\_dst\_y ==$_local_y then return LOCAL;
10:     else if $\_dst\_y < \_local\_y$ then return NORTH;
11:     else return SOUTH;
```

(continued on next page)

```
12:   end if
13: else if $dist\_x > 0$ then
14:           ▷ routing to the EAST considers link failure probability
15:   if $dist\_y == 0$ then return EAST;
16:           ▷ routing in the same row is free
17:   else if ($\_local\_x$ MOD 2) then   ▷ odd column routing is free;
18:       if $dist\_y > 0$ then return NORTH;   ▷ destination is up
19:       else return SOUTH;
20:       end if
21:   end if
22: else   ▷ routing to the WEST considers link failure probability
23:   if $dist\_y == 0$ then return WEST;
24:           ▷ routing in the same row is free
25:   else if not ($\_local\_x$ MOD 2) then
26:           ▷ even column routing considers link failure probability
27:       if ($dist\_y > 0$) and ($p\_y\_south < p\_y\_north$) then
28:           return SOUTH;
29:         else return NORTH;
30:       end if
31:   end if
32: end if
```

Since we still prohibit routing to forbidden turns, variability-tolerant Odd–Even routing is assumed to also be deadlock-free as Odd–Even routing.

5. Simulation results

We perform simulation for variability-tolerant routing algorithms at different technology nodes, mesh sizes, injection rates, buffer sizes, and traffic patterns, like bitr, bitc, regional, neighbor, tornado, transpose, random, and hotspot. Our purpose is to indicate how NoC performance is affected by link failure resulting from process variations under different routing algorithms, traffic patterns, etc.

5.1. NoCTweak

NoCTweak is a highly parameterizable, cycle-accurate, open source simulator for early exploration of performance and energy of NoC [11]. The simulator has been developed using SystemC, which allows fast modeling of concurrent hardware modules at the cycle-level accuracy. NoCTweak Version 1.0 platform is used, which simulates a 2D mesh NoC connecting multiple cores. Each NoC node consists of a processor (core and network interface) and an associated router. Each router has four buffered inputs ports, four buffer-less output ports, control logic, and crossbar switch, as shown in Fig. 5. Each processor core generates data flits and injects them into NoC through its router. Flits are routed on the NoC by a selected routing algorithm to their destinations at which the flits are immediately consumed.

NoCTweak configurable parameters include mesh size, buffer size, input voltage, link length, technology node, input traffic pattern, random seed, injection rate, and routing algorithm. The outputs of NoCTweak include average latency, average throughput, received flits, and average power consumption. Among all existing NoC simulators we found NoCTweak to be the only simulator that considers delay and power for technology nodes.

NoCTweak open source library uses 65 nm technology node. We perform scaling for delay and power at other technology nodes. Scaling is mainly done by scaling supply voltage and link length for different technology nodes by the factor S, where S is the ratio between the scaled technology node and the 65 nm node size.

The link failure probability of each link is input to NoCTweak. So, at each routing decision, the probability of link failure in the X direction (east and west) and the Y direction (north and south) are extracted based on the router position (XY coordinate). We then modified XY, West-First, Negative-First, and Odd–Even routing algorithms in NoCTweak to consider link failure probability when routing to be variability-tolerant routing algorithms.

5.2. Simulating NoC failure rate

Link failure probabilities are used by the methods we developed and embedded within NoCTweak to measure the overall NoC failure rate as an indicator of NoC tolerance to link failure induced by process variability random and systematic effects.

At the start of each simulation, the link failure probabilities of all NoC links are read from input file and mapped into NoCTweak. (This can be implemented in hardware by a simple register carrying lookup table of the link failure probabilities at each router.) Then, NoCTweak calculates NoC failure rate as the accumulated link failure probabilities of all the links that flits are routed through within NoCTweak number of simulations for certain input traffic pattern, injection rate, buffer size, mesh size, clock frequency, technology node, routing protocol, and throughput, divided by the total number of hops within the simulations. This comes in analogy with

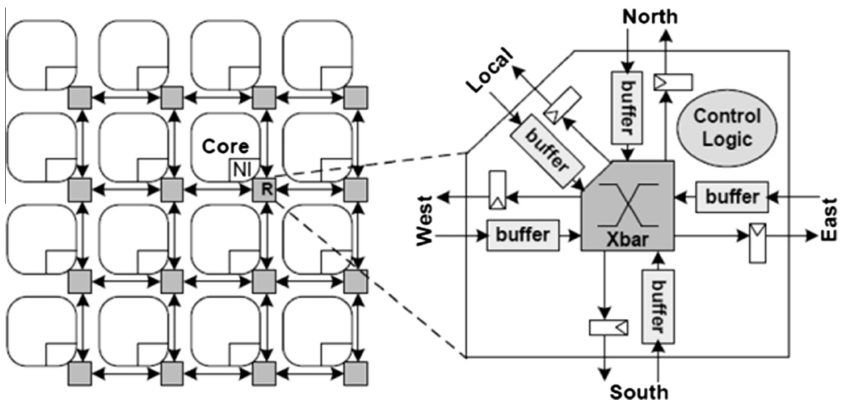

Fig. 5. NoCTweak platform for 2D mesh [11].

Table 1
NoC failure rates at different mesh sizes.

Mesh size	Latency (cycles)	Throughput (flits/cycle)	Failure rate (%)	Link power
8 × 8	30.868	0.050	1.561	1.110
10 × 10	36.795	0.050	2.020	1.359
12 × 12	42.160	0.050	2.241	1.583
16 × 16	53.127	0.050	5.303	2.035

the link fault probability by Konstantinos et al. [13], defined for each critical path as the fraction of timing violations over the total number of simulations. The NoC failure rate can be defined as:

$$\text{NoC failure rate} = 100 \times \frac{\sum^{routes}\text{link failure probability}}{n_{routes} \times n_{samples}} \times \frac{\text{flit injection rate}}{\text{throughput}} \quad (4)$$

where $\sum$ link failure probability is the sum of link failure probabilities via all the links the flits are routed through in an NoCTweak simulation for a certain input traffic pattern, n_{routes} is the total number of hops of all flits within an NoCTweak simulation, $n_{samples}$ is the total number of samples (1000) generated to estimate the link failure probability due to process variability, flit injection rate is the number of flits injected to the NoC per cycle, and throughput is the number of flits output from the NoC per cycle.

We simulated NoC failure rate for different mesh sizes at 45 nm for random uniform traffic using the original West-First routing with injection rate 0.05, buffer size of 16, and 2 GHz clock. Results are shown in Table 1.

It is shown in Table 1 that increasing the mesh size causes NoC to have higher NoC failure rates. This agrees with our work in [4], where the amount of delay variations increases as the mesh size increases. Additionally, larger mesh sizes encompass more spatial variations, and hence more links fail to meet the timing constraints. Larger size meshes mean more links, hence more average link power, and longer latency. The simulated NoC failure rate represents an indicator of how an NoC is tolerant to link failure resulting from process systematic and random variations.

5.3. Routing protocol comparison

Failure rates of XY, West-First, Negative-First, and Odd–Even routing algorithms before and after modifications to tolerate process variations, for an 8 × 8 mesh with random uniform input traffic, injection rate 0.1, and buffer size 16 at 45 nm are shown in Fig. 6. From the figure, it can be noted that:

- Variability-tolerant routing algorithms have lower NoC failure rate, than their counterpart routing algorithms, as long as the NoC is not saturated.
- In the case of Odd–Even routing, congestion partially occurs causing NoC failure rate to increase. The injection rate needs to be decreased or buffer size needs to be increased to eliminate the partial congestion of the traffic in the NoC.
- It can also be noted that, for this case study, West-First variability-tolerant routing minimizes its NoC failure rate by 56% compared to West-First routing. While XY variability-tolerant routing minimizes its NoC failure rate only by 3% compared to XY routing. As West-First routing shows the best tolerance to process variability among the routing algorithms in this case study, we focus on it for the rest of this paper.

5.4. Injection rate analysis

To make sure we study the NoC failure rate under deadlock-free network conditions, we work with non-saturated traffic conditions,

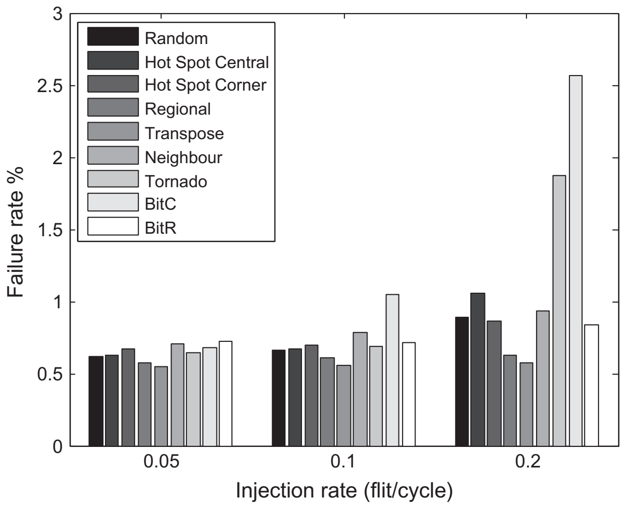

Fig. 7. NoC failure rate for variability-tolerant West-First routing at different injection rates.

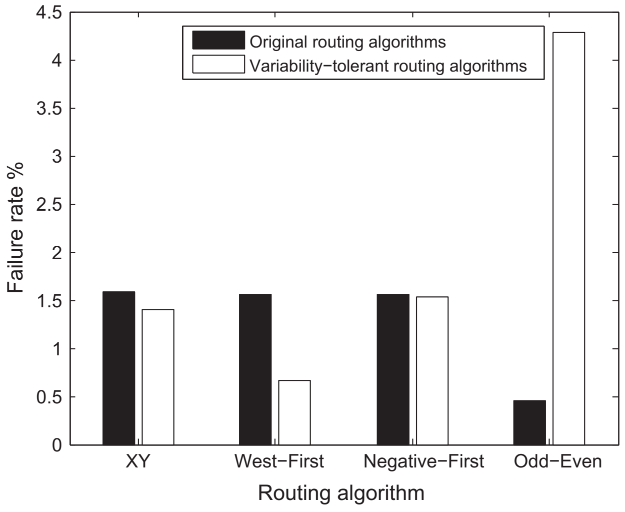

Fig. 6. NoC failure rate for different routing algorithms before and after routing modifications to tolerate process variations.

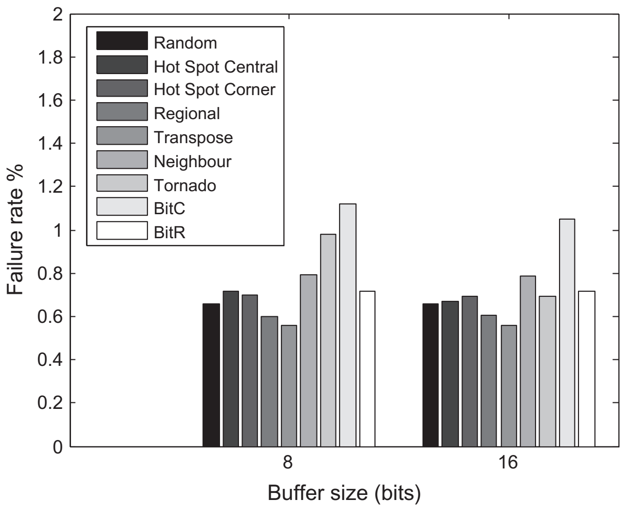

Fig. 8. NoC failure rate for variability-tolerant West-First routing at different buffer sizes.

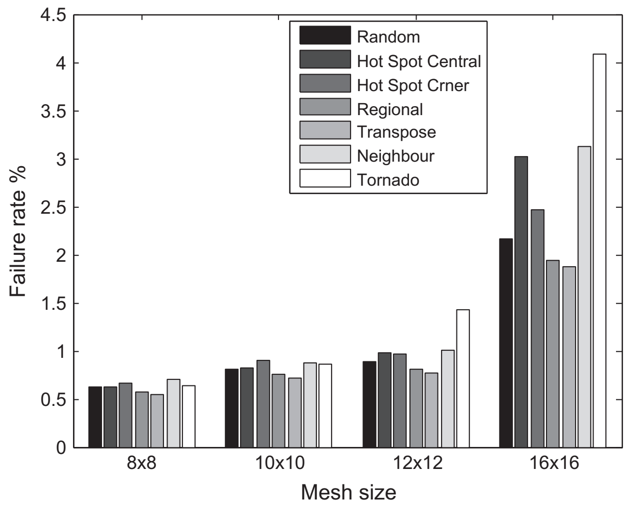

Fig. 9. NoC failure rates for variability-tolerant West-First routing at different mesh sizes.

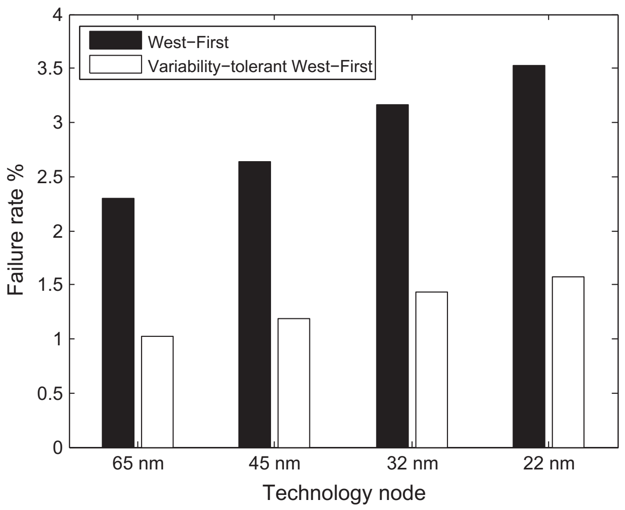

Fig. 10. NoC failure rates for West-First routing at different technology nodes.

i.e., throughput equals to the injection rate. In case of partial congestion (throughput $<$ flit injection rate), the NoC failure rate will be increased. In case of a complete congestion (throughput $= 0$), the NoC failure rate will go to $\propto$, which is an indicator of complete NoC failure.

We simulate NoC failure rate for different traffic patterns at injection rates lower than 0.2 flit/cycles (after which an 8x8 NoC at 45 nm with 2 GHz clock and 16 bits buffer size starts to saturate), with West-First variability-tolerant routing to evaluate the effects of increasing injection rate on NoC failure rate. As shown in Fig. 7, increasing the flit injection rates increases the NoC failure rates. Higher injection rates mean more traffic, more routes taken and hence larger probability of failure. It can also be noticed that, different traffic patterns leads to different NoC failure rates. Traffic patterns like bitc and tornado have the highest failure rate, while traffic patterns like neighbor and regional have the lowest failure rate.

5.5. Buffer size analysis

Simulating the 8×8 meshes NoC with variability-tolerant West-First routing at 45 nm with 0.1 injection rate and 2 GHz clock versus different buffer size is shown in Fig. 8. It is shown that increasing the buffer size decreases the NoC failure rate for some traffic patterns like tornado and bitc, which cross large distances across the NoC.

5.6. Mesh size scaling

We also simulated NoC failure rate for different mesh sizes at 45 nm with injection rate 0.05, buffer size 16, and 2 GHz clock at different traffic patterns using West-First variability-tolerant routing. Results are shown in Fig. 9. As shown in Fig. 9, the NoC failure rates increase as the mesh size increases. It can be noted that, different traffic patterns have different NoC failure rates. This agrees with the fact that different traffic patterns take different routes, hence go through different link failure probabilities. For the case in Fig. 9, the traffic patterns tornado has the highest failure rate, while the traffic pattern neighbor has the lowest NoC failure rate.

5.7. Technology node analysis

We performed NoCTweak simulation for 4×4 mesh with West-First variability-tolerant routing for random traffic, with buffer size 16 and injection rate 0.1 at 65 nm, 45 nm, 32 nm, and 22 nm technology nodes.

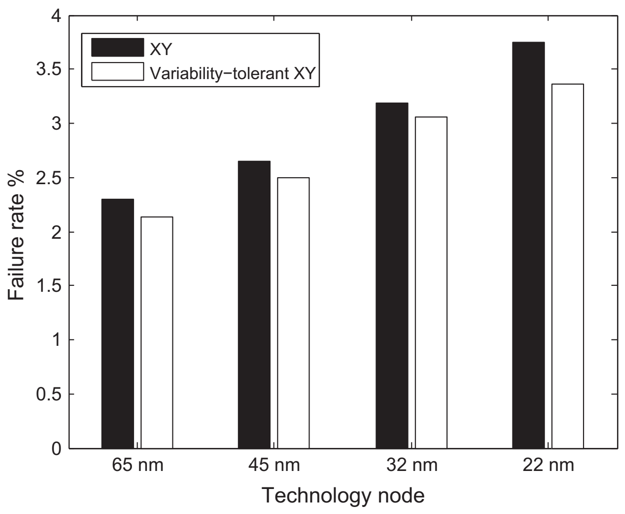

Fig. 11. NoC failure rates for XY routing at different technology nodes.

As shown in Fig. 10, as technology scales down from 65 nm to 22 nm, the NoC failure rate increases by 34%. It is also shown that, variability-tolerant West-First routing achieves lower NoC failure rates as compared to the original West-First routing at different technology nodes.

We also performed an NoCTweak simulation for 4×4 mesh with XY variability-tolerant routing for random traffic, with buffer size 16 and 0.1 injection rate at 65 nm, 45 nm, 32 nm, and 22 nm technology nodes. The estimated NoC failure rates are shown in Fig. 11. It can be note that modifying XY routing to be variability-tolerant decreases the NoC failure rates. This comes in accordance with the fact that, variability-tolerant routing selects the link with lower failure probability.

6. Conclusion and future work

This paper presented an NoC link failure system-level model for timing violations resulting from process variations. Variability-tolerant XY, West-First, Negative-First, and Odd–Even routing algorithms were also proposed in this paper, where flits are routed to links with lower failure probability in order to tolerate process variations.

Results show that variability-tolerant West-First routing achieves around 56% reduction in the overall NoC failure rate, while XY variability-tolerant routing minimizes its NoC failure rate by only 3%, for the case study of an 8×8 mesh with random uniform input traffic, injection rate 0.1, and buffer size 16 at 45 nm. We also note that, NoC failure rate increases with the increase in mesh size and injection rate or decrease in buffer size. As technology scales down, the amount of process variations increases, causing NoC failure rate to increase as well. Additionally, we have seen that, different traffic patterns have different NoC failure rates. This agrees with the fact that different traffic patterns take different routes hence go through different link failure probabilities. The traffic pattern tornado has the highest failure rate, whereas the traffic pattern neighbor has the lowest NoC failure rate.

The proposed variability-tolerant routing used static values for link failure probability, while adaptive routing algorithms, like Odd–Even routing, consider mainly dynamic network conditions. As a future work, run-time temperature/supply-voltage variations can be input to our link failure model. Changing the temperature or supply-voltage will change the amount of variations, and hence will change the link failure probability, which in turn will change dynamically the routing of flits through the NoC.

References

[1] C. Nicopoulos, V. Narayanan, C.R. Das, Network-on-Chip Architectures: A Holistic Design Exploration, Springer, Philadelphia, PA, USA, 2009.
[2] M. Orshansky, R. Nassif, D. Boning, Design for Manufacturability and Statistical Design, Springer, Philadelphia, PA, USA, 2008.
[3] E.K. Gawish, M.W. El-Kharashi, M.F. Abu-Elyazeed, Variability-tolerant NoC link design, in: Proceedings of the Fifth International Workshop on Network on Chip Architectures (NoCArc'2012), held in conjunction with the 45th Annual IEEE/ACM International Symposium on Microarcchitectures. Vancouver, BC, Canada, 2012, pp. 57–62.
[4] E.K. Gawish, M.W. El-Kharashi, M.F. Abu-Elyazeed, Variability-aware NoC geometry and topology scaling, in: Proceedings of the 2nd Saudi International Electronics, Communications and Photonics Conference (SIECPC'13), Riyadh, Saudi Arabia, 2013.
[5] E.K. Gawish, M.W. El-Kharashi, M.F. AbuElYazeed, Variability-tolerant current-mode link design for NoC, in: Proceedings of the 2013 IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing (PacRim), Victoria, BC, Canada, 2013, pp. 131–136.
[6] A. Mehranzadeh, M. Hoodgar, FAXY: fault aware routing algorithm based on XY algorithm for network on chip, Glob. J. Comput. Sci. Technol. 11 (17) (2011) 58–62.
[7] M. Dehyadgari, M. Nickray, A. Afzali-kusha, Z. Navabi, Evaluation of pseudo adaptive XY routing using an object oriented model for NoC, in: Proceedings of 17th International Conference on Microelectronics, Islamabad, Pakistan, 2005, p. 5.
[8] V. Rantala, T. Lehtone, J. Plosila, Network on Chip Routing Algorithms. Tech. Rep.; TUCS: Turku Center for Computer Science, Turku, Finland, 2006.
[9] H. Kariniemi, J. Nurmi, Arbitration and Routing Schemes for On-chip Packet Networks, Interconnect-Centric Design for Advanced SoC and NoC, Kluwer Academic Publishers, Norwell, MA, USA, 2004.
[10] J. Hu, R. Marculescu, DyAD: smart routing for networks-on-chip, in: Proceedings of the 41st Design Automation Conference, San Diego, CA, USA, 2004, pp. 260–263.
[11] A. Tran, B. Baas, NoCTweak: A Highly Parameterizable Simulator for Early Exploration of Performance and Energy of Networks on-chip, Tech. Rep.; VCL Lab, ECE Department, UC Davis, Davis, CA, USA, 2012.
[12] S. Sivaswamy, K. Bazargan, Statistical analysis and process variation-aware routing and skew assignment for FPGAs, in: ACM Transactions on Reconfigurable Technology and Systems TRETS – Special edition on the 15th International Symposium on FPGAs, vol. 1(1), 2008, pp. 58–62.
[13] A. Konstantinos, C. Chen, J. Plosila, L. Peh, Enabling system level modeling of variation induced faults in networks on chips, in: Proceedings of the Design Automation Conference DAC, New York, USA, 2011, pp. 930–935.
[14] J. Wu, Z. Zhang, C. Myers, A fault-tolerant routing algorithm for a network-on-chip using a link fault model, in: Proceedings of the Virtual Worldwide Forum on Electronic Design Automation VW-FEDA, Southampton, UK, 2011.
[15] M. Ebrahimi, M. Daneshtalab, J. Plosila, F. Mehdipour, MD: minimal path-based fault-tolerant routing in on-chip networks, in: Proceedings of the Design Automation Conference ASP-DAC 2013, Yokohama, Japan, 2013, pp. 35–40.
[16] C.J. Glass, L.M. Ni, The turn model for adaptive routing, in: Proceedings of the 19th Annual International Symposium on Computer Architecture, Gold Coast, Australia, 1992, pp. 278–287.
[17] C.J. Glass, L.M. Ni, Fault-tolerant wormhole routing in meshes, in: Proceedings of the Twenty-Third International Symposium on Fault-Tolerant Computing FTCS-23, Toulouse, France, 1993, pp. 240–249.
[18] Z. Zhang, A. Greiner, S. Taktak, A reconfigurable routing algorithm for a fault-tolerant 2D-mesh network-on-chip, in: Proceedings of the Design Automation Confrence DAC 2008, Anheim, CA, USA, 2008, pp. 441–446.
[19] Y. Fukushima, M. Fukushi, I.E. Yairi, T. Hattori, A hardware oriented fault-tolerant routing algorithm for irregular 2D-mesh network on-chip without virtual channels, in: Proceedings of the 25th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems DFT 2010, Koyoto, Japan, 2010, pp. 52–59.
[20] T. Yoneda, M. Imai, Improving dependability and performance of fully asynchronous on-chip networks, in: Proceedings of the 2011 IEEE International Symposium on Asynchronous Circuits and Systems ASYNC, Ithaca, NY, USA, 2011, pp. 65–76.
[21] T. Yoneda, M. Imai, Dependable routing in multi-chip NoC platforms for automotive applications, in: Proceedings of the 2012 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology systems DFT, Austin, TX, USA, 2012, pp. 217–224.
[22] T. Yoneda, M. Imai, Fault diagnosis and reconfiguration method for network-on-chip based multiple processor systems with restricted private memories, IEICE Trans. Inform. Syst. E97-D (9) (2013) 1914–1925.
[23] A. Sharifi, M. Kandemir, Process variation-aware routing in NoC based multicores, in: Proceedings of the Design Automation Conference DAC, New York, USA, 2011, pp. 924–929.
[24] S.R. Sarangi, B. Greskamp, R. Teodorescu, J. Nakano, A. Tiwari, J. Torrellas, VARIUS: a model of process variation and resulting timing errors for microarchitects, IEEE Trans. Semicond. Manuf. 21 (1) (2008) 3–13.
[25] W.J. Dally, B. Towles, Principles and Practices of Interconnection Networks, Morgan Kaufmann, San Francisco, CA, USA, 2004.

Eman Kamel Gawish received the Ph.D. and the M.Sc. degrees in Electronics and Electrical Communications Engineering from Cairo University, Giza, Egypt, in 2013 and 2005, respectively. She got the B.Sc. degree in Electronics and Electrical Communication Engineering from Ain Shams University, Cairo, Egypt in 2000. Currently, she is a general manager in Telecom Egypt. Her general research interests are in advanced system architectures, especially networks-on-chip (NoC), VLSI design, fabrication process variability, CAD tools, modelling and simulation. She is also interested in communication broadband, 3G, and VoIP technologies.

M. Watheq El-Kharashi received the Ph.D. degree in computer engineering from the University of Victoria, Victoria, BC, Canada, in 2002, and the B.Sc. degree (first class honors) and the M.Sc. degree in computer engineering from Ain Shams University, Cairo, Egypt, in 1992 and 1996, respectively. He is currently a Professor in the Department of Computer and Systems Engineering, Ain Shams University, Cairo, Egypt and an Adjunct Associate Professor in the Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC. His research interests include advanced microprocessor design, simulation, performance evaluation, and testability, system-on-chip (SoC), networks-on-chip (NoC), computer architecture and computer networks education, secure hardware. He published about 100 papers in refereed international journals and conferences and authored two books and 6 book chapters.

M.F. Abu-Elyazeed was born in Cairo, Egypt on February 1959. He received the B.Sc. degree (first class honors) from Cairo University, Cairo, Egypt in 1982. He also received the M.Sc. and Ph.D. degrees in Electrical Engineering from Cairo University in 1986 and 1990, respectively. From 1984 to 1993, he was a research assistant and an instructor at the Electronics and Electrical Communications Engineering Department, Cairo University. From 1994 to 1999, he was an assistant Professor at the Physics Department, Emirates University, UAE. He is presently a Professor at the Electronics and Electrical Communications Engineering, Cairo University. His research interests are in the areas of digital signal processing, circuits simulation, and fault diagnosis.