OASIS NoC Survey

The H. Vu
The University of Aizu,
Graduate School of Computer Science and Engineering,
Adaptive Systems Laboratory,
Fukushima-ken, Aizu-Wakamatsu-shi 965-8580, Japan
E-mail: d8182106@u-aizu.ac.jp

Abstract

Three-Dimensional-Network-on-Chip (3D-NoC) architectures were considered as promising solutions to tackle the interconnection issues in the systems, with high-performance and low interconnect-power. However, it is easier for them to suffer from some faults caused by various manufacturing and design factors. Fault-tolerant is a challenge needs to be considered carefully. Therefore, this report presents some mechanisms and algorithms to tackle the fault-tolerance in these systems. In addition, OASIS*-NoC background and a hardware complexity evaluation of a 2x2x2 OASIS-NoC system are also explained.

*OASIS is a reliable Network-on-Chip project which taken place at Adaptive Systems Laboratory, The University of AIZU.

1 Introduction

As 3D-NoC architectures started to show their high performance and energy efficiency compared to 2D-NoC systems, questions about their reliability to sustain their performance growth begun to arise. The fault appearances may be caused by some main factors: (1) the complex nature of 3D-IC fabrics and the continuing shrinkage of semiconductor elements and (2) the single-point-failure nature of NoCs. In fact, it is predicted that on a future 100-billion transistor chip, 20-billion transistors will be malmanufactured and a further 10-billion will fail during operation [20]. Thus it is important to find the answers for these questions.

Many works have been conducted to handle fault-tolerance in 3D-NoC systems. Some of the works focused on permanent faults occurring in Through-Silicon-Vias (TSVs). Others worked on link failure in general by adopting fault-tolerant routing algorithms. Some other works tried to address fault issues in the router level.

This report shows some techniques composed to alleviate the effect of faults on 3D-NoC systems. Two fault-tolerant algorithms, called Look-Ahead-Fault-Tolerant (LAFT) and Hybrid-Look-Ahead-Fault-Tolerant (HLAFT), are proposed to deal with the link failure issue. The Others were Random-Access-Buffer, Traffic-Prediction-Unit and Bypass-Link-on-Demand, which focused on tackling deadlock recovery in input-buffers and congestion in the crossbar. In addition, OASIS-NoC background and evaluations of a 2x2x2-3D-OASIS system are also illustrated.

2 OASIS-NoC Background

2.1 NoC overview

In recent years, there has been an increase in the number of cores on a single chip. This trend is inevitable because we have a huge demand to process big data, however, it also deals with challenges. One of them is how to connect the cores in the systems. While it is difficult for conventional systems such as shared-bus and Point-to-Point to meet requirements, Network-on-Chips (NoCs) [5, 9, 1] were considered to be a promising method which can overcome the challenge mentioned above. This mechanism provides a higher bandwidth and more enhanced performance because one of main reasons is that NoC systems can process in parallel. It also meets the requirements of scalability parallelism integration, power consumption. The explanations of NoC's characters are presented in next sections.

2.2 Topology

NoC Topology is a way which connects Processing Elements (PEs) in the network. The topology plays an integral part in the NoC system because it impacts the performance and power consumption.

There are some different topologies for NoC systems. Each one of them has both advantages and drawbacks. Topology choice depends on the target application requirements such as area and power.

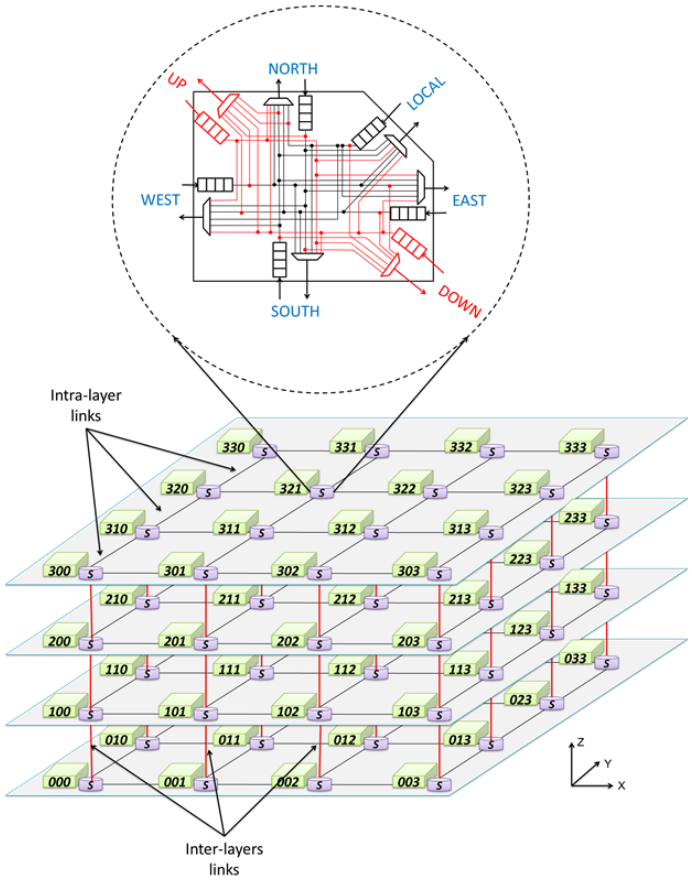

Figure 1: 4x4x4 3D-NoC mesh topology [6].

The Mesh [21, 22] topology is considered to be the most common is used for NoC systems, It is recommended for few-ten-PEs applications because of better application scalability and

also simpler implementation. One other reason is that the hops in the Mesh are equal. This property leads to a reduction in the latency.

Figure 1 illustrates a 4x4x4 3D-NoC mesh topology. With the Mesh, the simplest way to create a 3D-NoC is overlapping 2D-Mesh-NoC layers. All of elements of a 2D can remain when converted into 3D, except the router. The router architecture in 3D-NoC systems, simply adds two additional ports for up link and down link when converted from the 2D. This character is evidence for the scalability of the Mesh.

2.3 Switching Scheme

A switching technique determines how to transfer a message between source and destination nodes. It consists of two main kinds: circuit switching and packet switching.

Circuit switching mechanisms have no requirements for buffering, repeating and regenerating during transferring messages. The first step is path setup. The channel will be reserved by sending a probe into the network. Once path setup is complete, other messages need to wait until the current message transfer is complete. This technique is suitable for transmitting a large amount of data with full hardware bandwidth. However, the main drawback of this method is high overhead latency.

Packet switching is frequently used in NoC systems. In this switching method, each message should be split into several packets. There exist four main switching methods in packet switching: Two mechanisms based on allocation of resources in the packet; Store-and-Forward (SF) and Virtual-Cut-Through (VCT); Others are Wormhole Switching (WH) and Virtual Channel with resource allocation in units of flit.

Store-and-Forward technique was used in the first prototypes and designs of NoC. The SF requires the depth of FIFO buffers in the router to be equal to the packet size. This requirement leads to the systems having higher area and power consumption. In addition, another drawback of this technique is high per-hop latency, because the entire incoming packet is written into the buffer then transferred.

Virtual-Cut-Through, unlike SF, flits of a packet can be forwarded to the next router before the tail flit has came to current router. However, this is started when the buffer space of the next router is large enough for the entire packet. In addition, This mechanism still requires large buffers.

Wormhole switching is considered to be the most common technique in NoC design based on allocating resources in to flits and is also used in the OASIS-NoC system [23]. This method is different compared to both SF and VTC. Unlike SF, the buffer size in the WH is not required to host an entire packet. In practice, the buffer size selection depends on the computation speed of the router in each node. However, the main drawback of this technique is the blocking issue caused by a packet being divided into separate flits, which can be at different routers.

Virtual Channel technique [15] was proposed to deal with the blocking problem in wormhole. This method is based on sharing the same physical channel, and leads to improved link use. VC splits the input-buffer into smaller queues which are independent each other and are managed by an arbiter. When a blockage happens in one VC, the other ones are not affected, and they continue asking requests from their corresponding output-channels. In this fashion, non-blocked requests are served and their slots are freed to host other incoming fits. There are two important things to mention in this technique: (1) instead of using fair bandwidth arbitration, *winner-take-all* bandwidth location reduces the average latency of virtual-channel flow control with no throughput penalty (2) when increasing the buffer storage available, increasing the numbers of virtual channel is better than increasing the numbers of flits to each virtual channel.

2.4 Flow Control

Flow control methods are resource management strategies, where resources includes all nodes from source to destination, such as channel bandwidth or buffer capacity [16]. A good flow-control technique provides an efficient resource allocation. There are three flow control techniques, ON/OFF, Credit-based, and ACK/NACK, presented in this subsection.

2.4.1 Credit-based

In Credit-based, the upstream node holds a count of the number of free flit buffers in each virtual channel downstream [16]. This count is decreased one unit when upstream sends a flit. In contrast, when a downstream node sends a flit and frees a slot buffer, it sends a credit to the upstream node in order to increase a appropriate count. A drawback of this method is the large overhead caused by sending the number of control signals.

2.4.2 On/Off flow control

In order to efficiently use channel bandwidth, On/Off technique reduces the amount of control signaling by using an *on/off* signal. This signal is sent by the downstream node in order to permit whether the upstream node can be continuously transmitted or not. Unlike sending a credit in the Credit-based method, the downstream sends an *off* signal only when the control bit in upstream node is on and the number of free slots in the buffers downstream node is below the threshold, and vice verse. This leads to a reduction overhead and also improve channel bandwidth.

2.4.3 ACK/NACK flow control

Unlike both Credit-based and On/Of methods, the upstream node in ACK/NACK keeps no information about buffer availability of the downstream node. Therefore, it does not spend time for waiting and calculating the buffer information. At first, the upstream node sends a flit that does not worry about whether input buffers of downstream node are full or not. On the one hand, if the downstream node is available, it receives this flit and then sends an *ACK* signal back to source node. When receiving the signal, the source node frees the buffer slot. On the other hand, if the destination node sends a *NACK* signal, then the source node re-sends the flit.

2.5 Router Architectures

A router is the main component of a NoC system and consists of registers, switches, function units, and control logic [16]. It plays a crucial role in implementing the routing and flow control functions required to buffer and forward flits to their destination. In this subsection, a typical virtual-channel router is explained.

Figure 2 depicts a virtual-channel router architecture. The diagram can be divided into two groups: (1) the *datapath*, plays a role in storage and movement of a packet's payload, and consists of an input unit for each input port, an output unit for each output port, and a switch connecting the input units to the output units. (2) the *control plane* includes the rest of blocks which are route computation, virtual channels, and the switch allocator. It is responsible for coordinating the movement of the packets through an input port to an output port.

Modern routers are pipelined and work at the flit level. The operation of this router can be explained as flowing: At the first step, incoming flits of a packet are buffered in the virtual channels of the input buffer. The VC allocator unit plays a role as as control unit to determine which virtual channel holds the incoming flits. Then, control blocks perform route computation

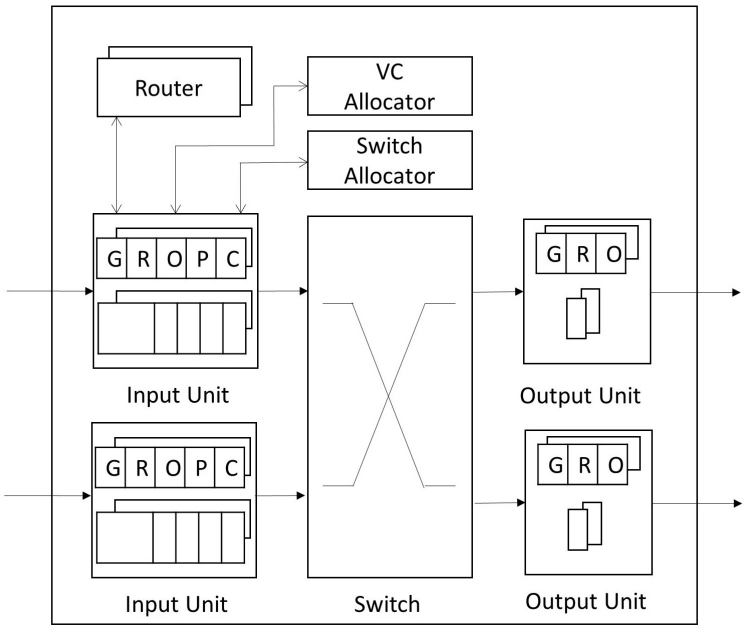

Figure 2: Virtual-Channel router block diagram [16]

to determine the output port where the packet can be forwarded. Once a route has been determined and a virtual channel allocated, each flit of the packet is forwarded over this virtual channel by allocating a time slot on the switch and output channel using switch allocator and forwarding the flit to the appropriate output. Finally, the output unit forwards the flit to the next router in the packet's path.

Based on the architecture mentioned above, a router architecture [10] was proposed to be applied for 3D-NoC systems, as shown in Fig. 3.

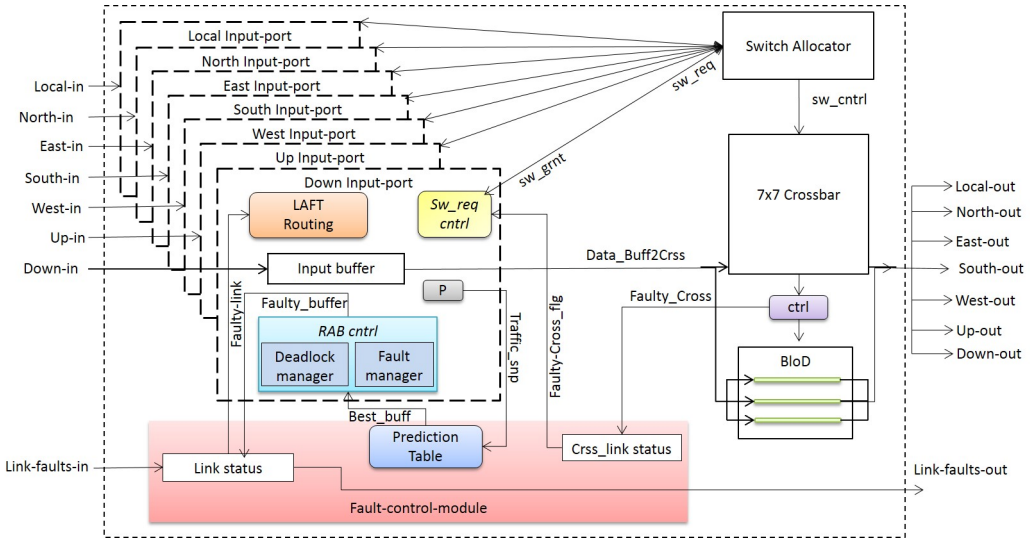

Figure 3: 3D-Fault-Tolerant-OASIS-NoC router architecture [10]

There are some differences between the routers in the Fig 3 and the Fig 2. The first one is that, at each input port, the virtual channels are replaced by only one buffer. In order to deal with deadlock in this buffer, The Random Access Buffer and The Traffic Prediction Unit mechanisms are proposed. Moreover, a Hybrid Look Ahead Fault Tolerant routing mechanism is also proposed to tackle faults appearing in the interconnection network. The second one

is the switch unit which is changed into a 7x7 crossbar, and the Bypass Link on Demand method is proposed to confront permanently faulty links crossbar. These proposed techniques are explained in the next sections.

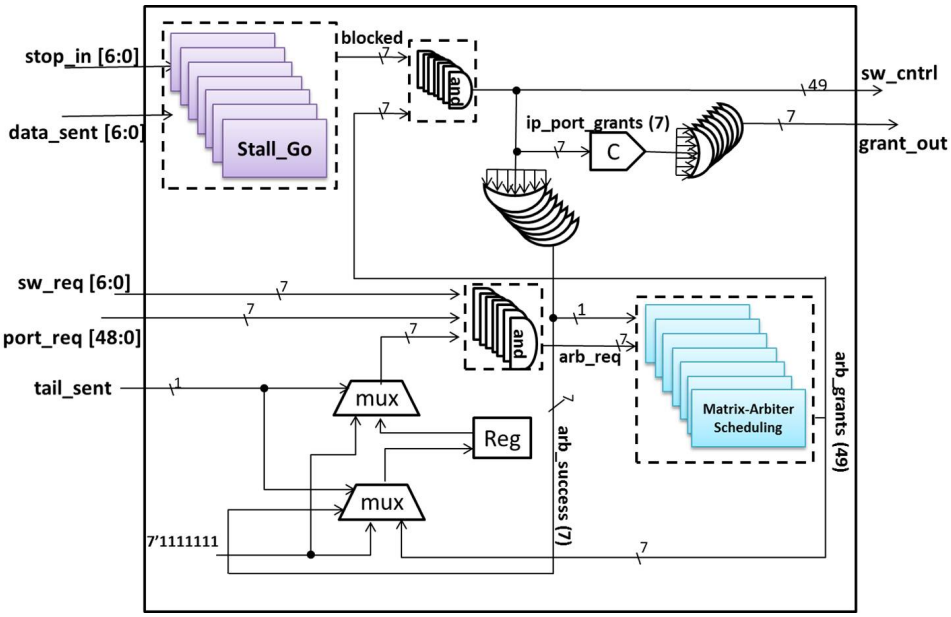

Figure 4: Switch allocator block diagram [10]

The switch allocator unit is responsible for deciding when a flit will be transferred from a granted input port to an allocated out port. This unit is composed of two main components: *Stall-Go flow control* and *Matrix Arbiter Scheduling*, as shown in the Fig. 4.

The *Stall-Go flow control module* is designed to prevent buffer overflow and packet dropping in the input buffers. This technique uses two control signals: *nearly-full* which is sent from the downstream node to inform that the buffer is almost full and only one slot is available to host the last flit and *data_sent* which is sent from the crossbar to indicate the flit is transferred, and one slot in the buffer is released.

Matrix Arbiter Scheduling module adopts a least recently served priority scheme [23] to calculate the requests from input ports. First, this module accepts all requests which are sent from the input ports via two main signals: *sw_req* (switching request) and *port_req* (desired output port). After that, it computes and grants access to the request with the highest priority.

The switch allocator will send two main signals: *sw_cntrl* to the crossbar for allocating the crossbar channel, and *grnt_out* to the requested input port. Finally, the flit will be read from the granted input buffer, and then transferred to the allocated output port through the crossbar.

2.6 Packet Format

In packet switching, *messages* need to be split into *packets* in order to improve the efficiency of resource allocation. Packet size needs to be considered carefully because it impacts on performance and functionality of a flow control mechanism. Depending on flow control technique, a *packet* may be split into *flits*.

Figure 5 illustrates a *flit* format which applies for 3D-OASI-NoC systems. The first bit, *tail*, indicates whether flit is the *tail* flit or not. The next three bits show the *Next_port* which is used by the Look Ahead Fault Tolerant routing algorithm to determines the direction that the flit will be sent. Unlike the conventional flit format, each 3D-OASIS-NoC [11] flit will contains the address of destination node (*X-dest, Y-dest, Z-dest*) because this is to apply for Random

Access Memory and Hybrid Look Ahead Fault Tolerant techniques. This difference guarantees the flits can be transmitted to the destination without following the head flit.

Figure 5: 3D-OASIS-NoC flit format

Finally, the remaining-64 bits are dedicated to store the playload. The size of the playload can be changed to satisfy the requirements of particular applications.

In some cases, a flit maybe divided into *phits* which are the unit of information transferred through a channel in a single clock. Although *phits* are not allocated the resource, they play a role in finding the boundaries between flits.

2.7 Network Interface

Network Interface (NI) is an element connecting a network terminal and interconnect network. It plays an important role in NoC operation because a bad NI can increase bottleneck occurrence and latency in the system. There are three types of NI, Processor-network interface, Shared-memory interfaces, and Line-card interface. A processor-network interface is designed to connect a processor (core or processing element) and a router. A good NI should minimize processor overhead and be safe. In this subsection, the Processor-network interface is presented.

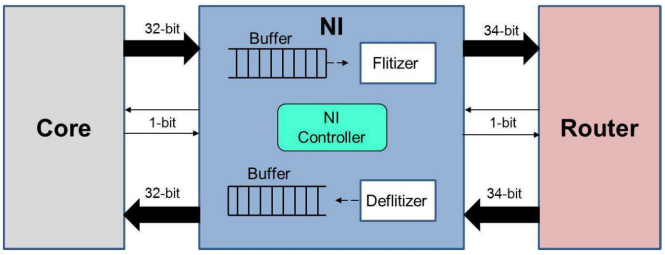

Figure 6: A network interface for distributed routing [24].

Figure 6 depicts the block diagram of a network interface which is based on the two register interface model. In order to send a message from the core to the router, first, the core sends the message to the NI. After that, the message is written in the buffer (writing buffer). Then, the message is transformed into flits by *Flitizer* unit. Finally, these flits will be transmitted to the router. On the other hand, there are the corresponding steps for transferring all flits of a packet from the router to the core.

The NI structure is shown in Fig. 7. The structure consists of two parts: (1) one of them is dedicated to transferring packets from the core to the router, included C2R-Buffer, C2R controller, Flitizer (2) the other, includes three remaining units, designed for transmitting from the router to the core.

Buffers are the FIFO (First-In-First-Out) buffer. The buffer depth is equal to 16, and the size of each slot is 32-bits. The buffers are controlled by corresponding controllers with *Add-Write, Write-En* signals for writing packets into the buffers, and*Add-Read, Read-En* signals for reading from the buffers.

Flitizer and Deflitizer units are implemented for transforming a packet into flits, and flits into a packet, respectively. In Flitizer unit, as shown in the Fig. 7, the difference between

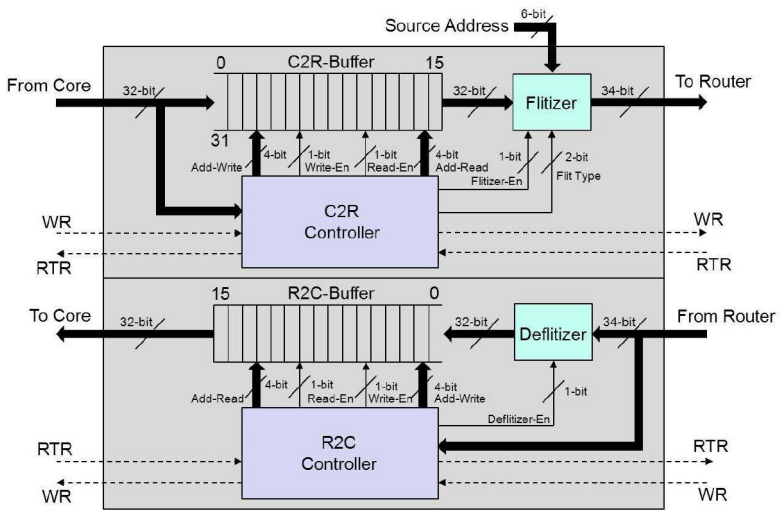

Figure 7: Internal structure of Network Interface [24].

data input link and data output link is that the output link has 2-bits more than the other. The 2-bits unit is dedicated the header of flit, corresponding, and there are four types of flit ("00","01" for header flit; "10" for body flit, "11" for end flit). The 2-bits will be calculated and sent from C2R controller. In Deflitizer, on the other hand, incoming flits from the router will be converted into a packet before storing in the R2C-Buffer.

Controllers will control the operation of the other units in the NI system. They used *Wr* (Write) and *RTR* (Ready to Receive) signals to handshake with the core and the router.

This NI was designed and prototyped on an Altera FPGA board [24]. However, in order to implement with OASIS-NoC system, this NI may be changed because there is a difference in flit format between this NI and the OASIS-NoC system.

2.8 Routing Algorithm

Routing algorithms are the techniques to select a path from a source node to a destination node in a particular topology. There are some factors to determine a good routing algorithm: load balance, which a good routing algorithm balances loads cross the network channels. If this factor is guaranteed, improving the throughput of the network; minimal path, this factor leads to reduce the numbers of hope and the overall latency of a message converse; In addition, abilities to work in the presence of faults and adapt to the current traffic in the network need to be taken consideration.

Routing algorithms can be classified into three types in terms of how they select between the set of possible paths from source to destination nodes.

Deterministic routing algorithms do not worry about the diversity of path in the case which there are multiple possible paths. This may not guarantee load balance. These methods, however, are quite common used because they are easier to implement and deal with deadlock.

Oblivious algorithms do not consider any information about the network's present state. This character makes it simple to implement and analyze.

Adaptive algorithms, unlikely to oblivious algorithms, adapt to the state of the network by using the state information in making routing decisions. This information may consist of statuses of a node or link, the length of queues for network resources, and historical channel load information [16]. This technique can respond with network's states, however, this may increase the latency and also reduce system throughput.

3 Reliable OASIS-NoC

3.1 Reliability Challenges

3D-NoC systems have more advantages such as high performance, low power consumption by ten to a hundred time. However, they also deal with challenges, which almost of them are derived from natural characters of 3D systems.

TSV footprints act as vertical links to connect layers when overlapping 2D-NoC layers. They may be faced with yield problems and these defects are very difficult to predict and prevent.

Heat is also a problem that needs to be considered in 3D-Noc. This problem is caused by a high density of components in the system. This factor has a huge impact on power dissipation.

These factors mentioned above may cause faults in 3D-NoC. In addition, predicting and preventing these faults are not easy. Therefore, it is necessary to propose mechanisms to alleviate the influence of faults, thereby increasing the reliability of the system.

3.2 Fault-tolerant Architecture

According to [18], input-buffers and crossbars occupy the largest area in the 3D-NoC system that can reach the 80% and 10%, respectively. As consuming the largest portion of the router area, the fault occurrence probability is very high in the input-buffers and crossbar if we assume that the faults' distribution is proportional to the area distribution. Therefore, the fault occurrence in input-buffers and crossbar need to be taken into consideration carefully. This section presents some mechanisms to mitigate the fault influence on input-buffers and crossbars.

3.2.1 Random-Access-Buffer Scheme for Deadlock-Recovery

The Random-Access-Buffer (RAB) [2, 4, 12, 3] is similar to Virtual-Channels (VCs), but it is much simpler and less complex. The random property which means the flits can be transferred in different paths and they do not follow the head flit because all of the flits contain the address of destination node. First, when RAB detects a flit as being the cause of deadlock in the buffer by using a timer, the request of the flit is dropped. Then, it looks for another flit whose required out-port is available, this flit is suggested to the Switch-allocator to serve. As a result, the technique ensures the input-buffer operation may be not interrupted.

Figure 8 depicts an example of RAB. At Fig. 8 (a), after period of time, if the request being processed (P1 North) is not severed, a flag is issued informing the presence of a deadlock, its request is dropped. Next, A buffer-controller (BC) at each input-port reads the head of the next packet in the buffer. If required out-port of the packet is available (P2 East at Fig. 8 (b)), BC sends a request to the Switch-allocator to be served. When the request is granted the flit is read from buffer and releases the slot for receiving a new incoming packet (Fig. 8 (c)). After a new packet is written in the buffer, the blocked packet is checked again (Fig. 8 (d)). The BC receives a grant for the direction requested (North) and the packet is read from the buffer.

In addition, there are some extend sub-mechanisms added to RAB to enhance its efficiency. The first is Status Register (SR), including an array of n 2-bit items (where n is the buffer depth). For RAB, this register plays an important role in avoiding not only overwriting but also *transient, intermittent* and *permanent* faults in the input-buffer. Moreover, In order to reduce the dynamic power in the system, a power management scheme is also proposed. This scheme is based on clock-gating to turn-off the clock in some specific components which, in some periods, are in a waiting state and not performing any computation. This technique is applied to three components: faulty input-ports, local-routing modules, and RAB circuits.

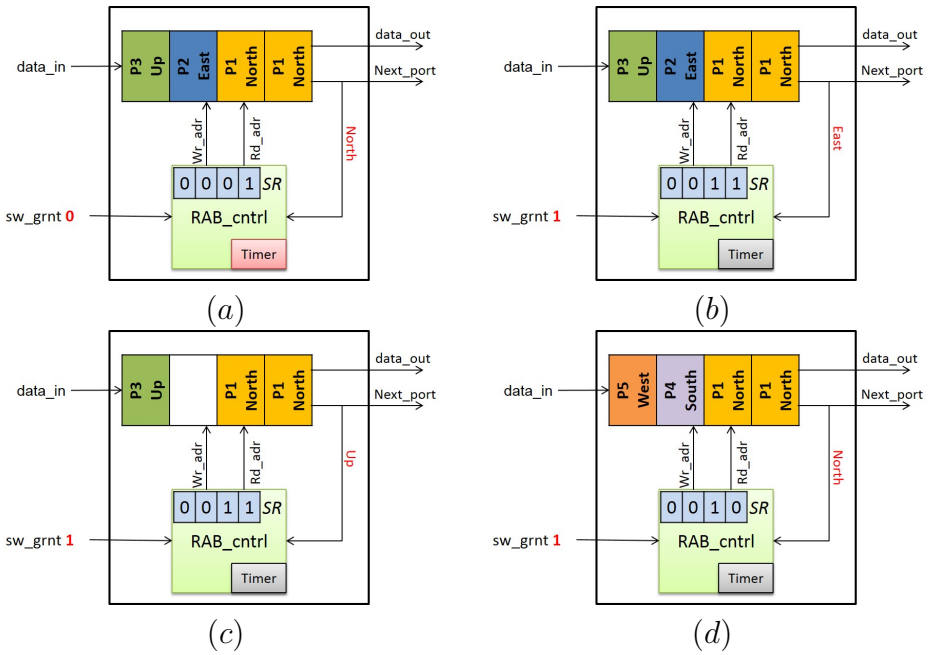

Figure 8: Example of deadlock-recovery with Random-Access-Buffer [2].

3.2.2 Traffic-Prediction Unit

In order to enhance the performance and fully exploit the entire router resources, Traffic-Prediction Unit (TPU) is proposed based on sharing the input-buffers resources among all the input-ports. This unit provides information about the best input-buffer in the router for receiving incoming flits.

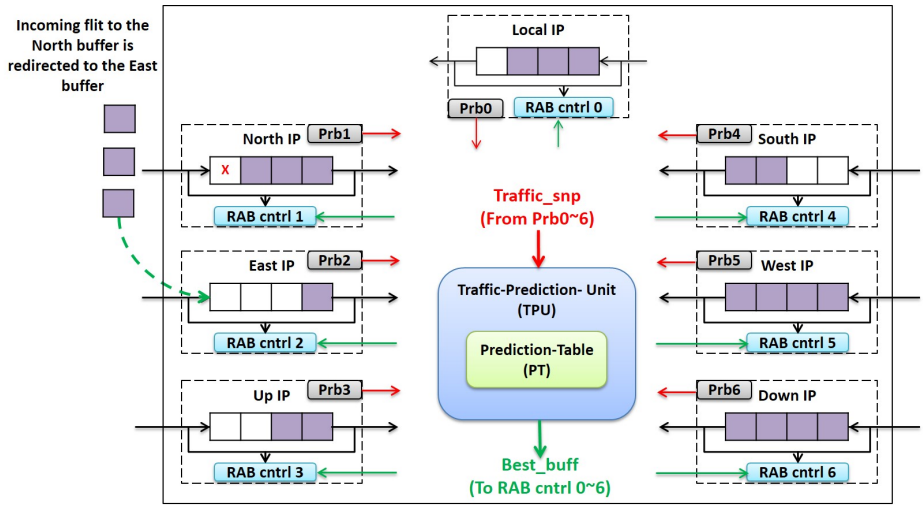

Figure 9: Simplified example explaining the use of the Traffic-Prediction-Unit (TPU) [6].

Figure 9 illustrates a simplified example explaining the TPU. The information about traffic load (red signals) is collected by monitoring probes (Prb) attached to each input-port. The collected information is used and calculated to select the best input-buffer (East IP). The best-buffer (*Best-buff*) signal is sent to the corresponding RAB controller for receiving incoming flits. As the Figure shows, Incoming flits are redirected to the East buffer instead of North one. In addition, in order to guarantee that packet's flits will arrive in order with no additional

hardware/power overhead nor performance degradation, only header flits of packet can be redirected.

Apart from advantages such as dealing with fault and overflow in input buffers, this technique also guarantees the load balancing between the input buffers.

3.2.3 Bypass-Link-on-Demand

The Bypass-Link-on-Demand mechanism [13, 14] is proposed to solve permanent faults in crossbar by adding escape channels whenever the number of faults in the crossbar increases. The *ctrl* unit, shown in the Figure 10, manages to check the crossbar link status. In the case where a fault is detected in one or several links, it sends flags to the Fault-control-module (FCM) which disables the faulty crossbar links and enables the appropriate number of bypass channels. As shown in Fig. 10, two bypass-links are used what there are two-permanent (Red links) and one-transient (Green link) fault links (Fig. 10 (a)), compared to only one (Fig. 10 (b)) when the transient link is recovered.

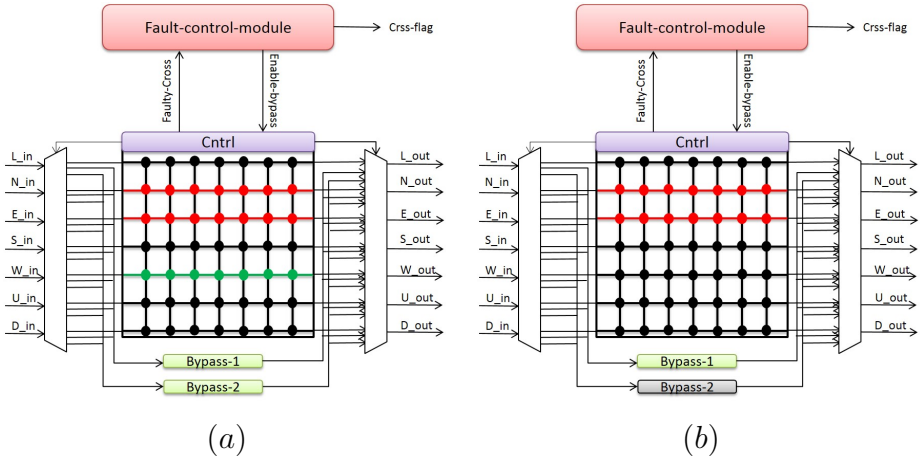

Figure 10: Example of Bypass-Link-on-demand [6].

In addition, the number of Bypass-links is very important, and it should be minimized as much as possible to reduce the area and power overhead. Therefore, exploiting the unused crossbar links already existing in the system is the good way to achieve area and power savings while keeping the performance at its peak.

Extension: Apart from mechanisms which mentioned in this section, an other router architecture [17], named SER-3DR, is proposed to void soft errors in 3D NoC systems. Evaluation results show that SER-3DR is able to achieve a high level of transient error protection.

3.3 Fault-Tolerant Routing Algorithms

This section illustrates efficient routing algorithms to alleviate fault effects on the 3D-NoCs, called Look-Ahead-Fault-Tolerant (LAFT) and Hybrid-Look-Ahead-Fault-Tolerant (HLAFT) algorithms. LAFT brings more advantages when compared to Look-Ahead-XYZ routing algorithm. However, HLAFT is expected to outperform LAFT because it tackles drawbacks of LAFT.

3.3.1 Look-Ahead-Fault-Tolerant Routing Algorithm

Conventional XYZ routing [19] compares the addresses between current and destination nodes to determine the *Output-Port*. The computed information is sent to Switch-Arbiter

requesting access of the selected out-port. This communication technique may bring high latency because flits should pass through all four stages in pipeline (as shown in Fig. 11 (a)) at each hop, thus, degrading the system throughput.

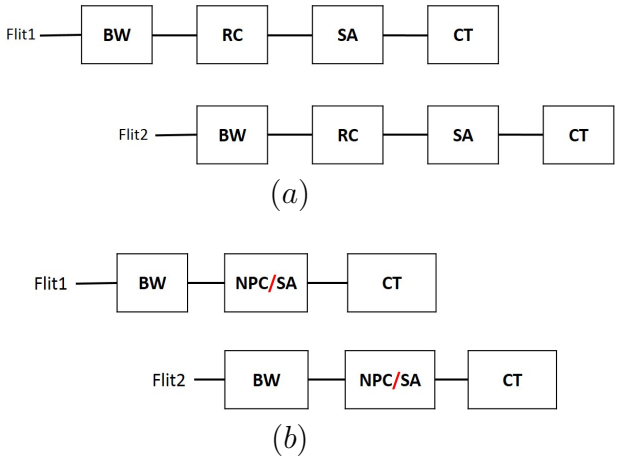

Figure 11: The routing router pipeline stages: (a) Conventional XYZ (b) Look-Ahead-XYZ [6].

In order to improve throughput, the Look-Ahead-XYZ routing algorithm (LA-XYZ) [8, 7] parallelizes Routing Calculation (RC) and Switch-Arbiter (SA) stages. As a result, there are only three stages in the pipeline (as shown in Fig. 11 (b)), therefore, reducing the system latency. However, this algorithm lacks support for fault tolerance in NoCs.

Algorithm 1: Look-Ahead-Fault-Tolerant routing algorithm (LAFT) [6]

```
   // Destination address
   Input: X_dest, Y_dest, Z_dest
   // Current node address
   Input: X_cur, Y_cur, Z_cur
   // Next-port identifier
   Input: Next-port
   // Link status information
   Input: Fault-in
   // New-next-port for next node
   Output: New-next-port
   // Calculate the next-node address
1  Next← Next-node (X_cur, Y_cur, Z_cur, Next-port);
   // Read fault information for the next-node
2  Next-fault← Next-status (Fault-in, Next-port);
   // Calculate the three possible directions for the next-node
3  Next-dir← poss-dir (X_dest, Y_dest, Z_dest, Next_x, Next_y, Next_z);
   // Evaluate the diversity number of three minimal paths
4  Div_1 ← path-div (X_dest, Y_dest, Z_dest, poss − dir_1);
5  Div_2 ← path-div (X_dest, Y_dest, Z_dest, poss − dir_2);
6  Div_3 ← path-div (X_dest, Y_dest, Z_dest, poss − dir_3);
   // Evaluate the New-next-port direction
7  if (|Next-dir| > 1) then
8  |   if (Div_1==Div_2==Div_3) then
9  |   |   New-next-port ← min-congestion (poss − dir_1, poss − dir_2, poss − dir_3);
10 |   else
11 |   |   New-next-port ← max-diversity (poss − dir_1, poss − dir_2, poss − dir_3);
12 else
13 |   if (Next-dir == 1) then
14 |   |   New-next-port← Next − dir_1;
15 |   else New-next-port← nonminimal (X_dest, Y_dest, Z_dest, X_cur, Y_cur, Z_cur, Fault-in);
```

The benefit of LAFT outweighs the LA-XYZ because it supports fault-tolerant NoCs. There are three phases in this technique. Firstly, the same as in LA-XYZ, LAFT calculates the next

node address depending on the next-port identifier read from the flit and current node address. Secondly, LAFT compares z, y, z coordinates of both next and destination nodes concurrently, in order to get information of three possible directions for a minimal routing. During that time, LAFT also gets the information about the next node fault status. Finally, the routing selection is performed to choose the out-port of next node.

For this decision, there is a set of prioritized conditions to guarantee fault tolerance and high performance either in the presence or absence of faults. The selected direction should be one with:

1. **Minimal path**, and it is given the highest priority in the routing selection.
2. **The largest next hop path diversity**.
3. **The least congestion status**, it is given the lowest priority.

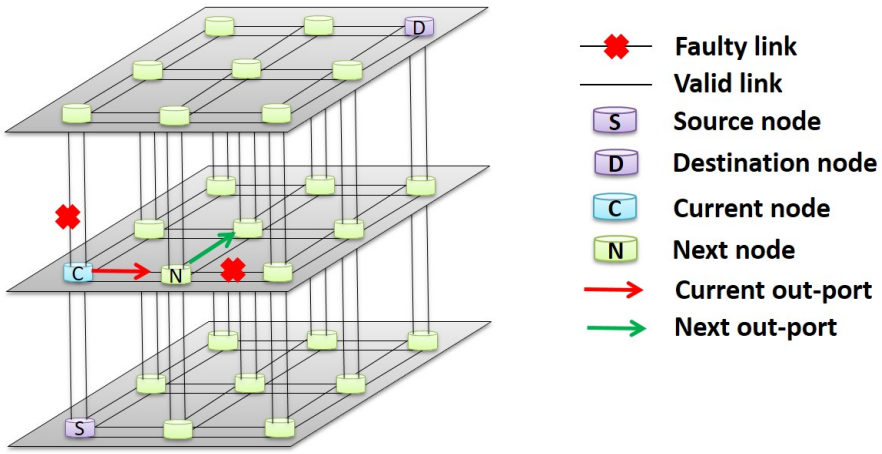

Figure 12: Look Ahead Fault Tolerant routing algorithm example [6].

Fig. 12 is an example of LAFT. Is is important to mention that both the North and the Up directions of the Next node (Labelled N) are minimal possible paths. However, the diversity value of the North is equal to 3 (East, North, Up), while the figure for the Up is only equal to 2 (North, Up). Therefore, Next out-port is the North one (Green arrow).

3.3.2 Hybrid-Look-Ahead-Fault-Tolerant Routing Algorithm

Applying LAFT reduces the router latency and also improves system performance. However, in some cases, LAFT may not select the best route. Fig. 13 (a) is an example for this case. Choosing the North route leads to the node which all the minimal possible links are faulty.

As illustrated in Fig. 13 (b), when the flit arrives to the node labeled **C** and having a current out-port North precalculated in the previous node, the fault-control checks the fault status of the links of the next-node in the north direction. When performing the checking, all the possible minimal paths are found faulty; thus, Loc-RC is triggered and computes Up as the new best route (since East is faulty). Then, the newly calculated Up direction is sent to the LA-RC module which issues North as the Next-out-port.

In order to deal with issues which mentioned above, HLAFT is proposed. A Local-Routing-Calculation module (Loc-RC) is added to the router. Unlike LAFT, after calculating the next node, if all minimal possible paths of next node are faulty, a flag is issued to activate Loc-RC. Then Loc-RC recalculate the output-port by verifying the status of the current and neighboring nodes links.

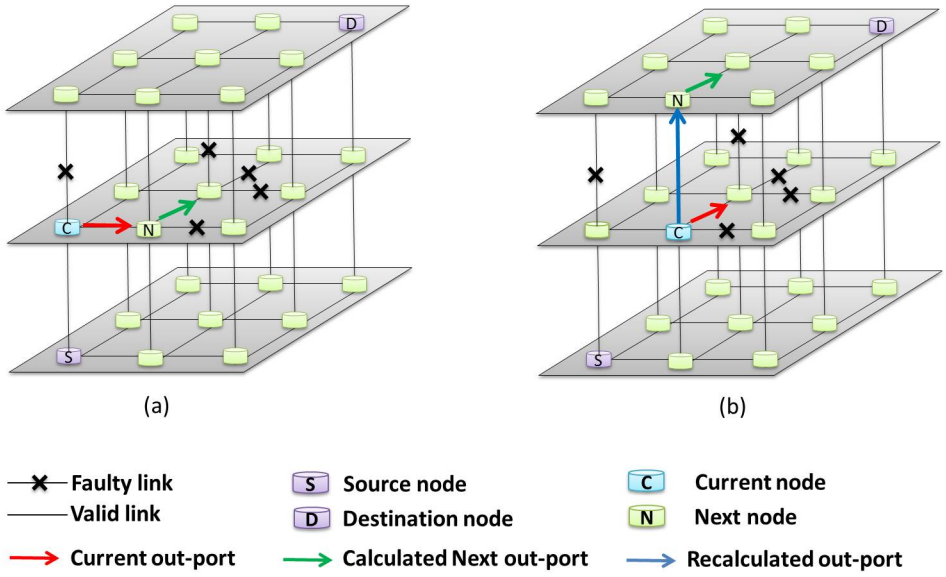

Figure 13: Example of fault-tolerant routing: (a) Look-ahead routing (LAFT) (b) Hybrid routing (HLAFT) [6].

Fig. 13 (b) illustrates the advantage of HLAFT. Applying this mechanism, the Up direction (Green arrow) is re-selected instead of the North one (Red arrow) because, following the red direction, all minimal possible paths are faulty. In addition, more deep understanding is illustrated in the algorithm.

Algorithm 2: Hybrid-Look-Ahead-Fault-Tolerant routing algorithm (HLAFT) [6]

```
   // Destination address
   Input: X_dest, Y_dest, Z_dest
   // Current node address
   Input: X_cur, Y_cur, Z_cur
   // Next-port identifier
   Input: Next-port
   // Link status information
   Input: Fault-in
   // New-next-port for next node
   Output: New-next-port
   // Calculate the next-node address
1  Next← Next-node (X_cur, Y_cur, Z_cur, Next-port);
   // Read fault information for the next-node
2  Next-fault← Next-status (Fault-in, Next-port);
   // Calculate the three possible directions for the next-node
3  Next-dir← poss-dir (X_dest, Y_dest, Z_dest, Next_x, Next_y, Next_z);
   // Evaluate the flag
4  if (|Next-dir| == 0) then
5  |   flag← 1;
6  else flag← 0;;
   // Evaluate the New-next-port
7  if (flag == 1) then
   // Trigger local-routing
8  out-port← local-routing (Fault-in, X_cur, Y_cur, Z_cur, X_dest, Y_dest, Z_dest);
   // Execute LAFT with the new recomputed out-port
9  New-next-port← LAFT-routing (X_cur, Y_cur, Z_cur, X_dest, Y_dest, Z_dest, out-port, Fault-in);
10 else
   |   // Execute LAFT with the inputed Next-port identifier
11 |   New-next-port ← LAFT-routing (X_cur, Y_cur, Z_cur, X_dest, Y_dest, Z_dest, Next-port, Fault-in);
```

4 Evaluation

4.1 Evaluation methodology

A network system was designed in Verilog HDL and compiled by using Quartus Prime 15.1 tool. The Look-Ahead-XYZ routing algorithm is evaluated in terms of area utilization, power consumption, and speed.

A simple technique based on the Random Number Generator transmission method is selected to perform the network system. A couple of nodes with the longest path in the network is selected, the 000 and 111 nodes play roles in source and destination nodes, respectively. In this simulation, the source node sends to the destination node ten flits where the playload (16 bits) of each flit is a random member. The system configuration parameters are illustrated as the Table 1.

Table 1: System configuration parameters.

Parameters / System	3D-OASIS-NoC
Topology	Mesh
Network size	2x2x2
Packet size	10 flits
Buffer Depth	4
Witching	Wormhole-like
Flow control	Stall-Go
Routing	LA-XYZ
Flit size	33 bits
Data size	16 bits
Target Device	Altera Stratix IV

4.2 Evaluation results

4.2.1 Hardware complexity evaluation

Area consumption: the Table 2 shows the data about hardware configuration parameters. The number of logic elements is equal to 5794, at 2% total, while the figures for the registers and the pins are 6608 and 546, respectively.

Table 2: Complexity evaluation result.

System/ Parameters	Logic utilization	Total registers	Total pins
2x2x2-3D-OASIS	5794	6608	546

Power evaluation: The results are shown in the Fig. 14, where total power consumption is 1150.55 mW. The proportion of power consumption in the core static is the highest, at 79%, while the figures for the core dynamic and the I/O are, at 11% and 10%, respectively.

Speed: In Slow 850mV 100C model, the maximum frequency meets 148.52 MHz compared to 157.31 MHz in Slow 850mV -40C Model. The Slack factor is 4.192 in Fast 850mV 100C model, while the figure for Fast 850mV -40C model is higher, at 4.650.

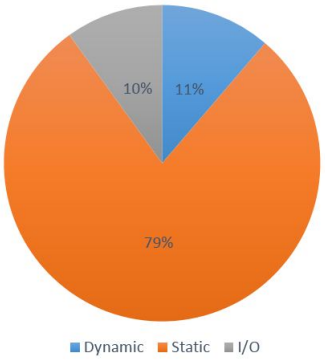

Figure 14: Power analyzer

4.2.2 Performance evaluation

In the evaluation, the latency (end to end) and the throughput of the network system are calculated, with the clock is set 125 (MHz). First, the latency is performed from sending the first flit to the time when all of the flits are received at the destination node. The latency of this network system is 111.2 ns/flit. On the other hand, the throughput of the network system reaches about 0.144 flit/cycle.

5 Conclusion

The proposed mechanisms manage to avoid the system failure while ensuring graceful performance degradation, minimizing the additional hardware complexity and remaining power-efficient.

In tackling failure link issue, the LAFT algorithm reduces the communication latency and enhances the system performance while maintaining a reasonable hardware complexity and ensuring fault tolerance in links. Moreover, a proposed algorithm, called Hybrid-Look-Ahead-Fault-Tolerant (HLAFT), takes advantage of both local and look-ahead-routing to enhance the performance of 3D-NoC systems while ensuring graceful performance degradation.

Apart from routing algorithms mentioned above, some other techniques are also proposed to handle the fault occurrence in the input-buffers and crossbar. The RAB is used as a light-weight solution to recover from the deadlock in input-buffers. Moreover, this technique has added a TPU to further reduce the latency thanks to faults appearing in buffer-slots. In addition, the congestion in crossbar is relieved by a technique, called Bypass-Link-on-Demand. Furthermore, the hardware complexity and network performance evaluations of a 2x2x2-3D-OASIS system are also illustrated.

References

[1] A. Ben Ahmed, K. Mori, and A. Ben Abdallah. Onoc-spl: Customized network-on-chip (noc) architecture and prototyping for data-intensive computation applications. In *4th International Conference on Awareness Science and Technology*, pages 257–262, Aug 2012.

[2] Akram Ben Ahmed and Abderazek Ben Abdallah. Graceful deadlock-free fault-tolerant routing algorithm for 3d network-on-chip architectures. *Journal of Parallel and Distributed Computing*, 74(4):2229 – 2240, 2014.

[3] Abderazek Ben Abdallah and Akram Ben Ahmed. OASIS 3D-Router Hardware Physical Design. Technical report, Adaptive Systems Laboratory, Division of Computer Engineering, School of Computer Science and Engineering, University of Aizu, 2014.

[4] Abderazek Ben Abdallah, M. Nakamura, Akram Ben Ahmed, M. Meyer, and Y. Okuyama. Fault-tolerant router for highly-reliable many-core 3d-noc systems. In *The 3rd International Scientific Conference on Engineering and Applied Sciences (ISCEAS 2015)*, 2015.

[5] Abderazek Ben Abdallah and Masahiro Sowa. Basic Network-on-Chip Interconnection for Future Gigascale MCSoCs Applications: Communication and Computation Orthogonalization. In *JASSST2006*, 2006.

[6] Akram Ben Ahmed. *High-throughput Architecture and Routing Algorithms Towards the Design of Reliable Mesh-based Many-Core Network-on-Chip Systems*. PhD thesis, Graduate School of Computer Science and Engineering, University of Aizu, March 2015.

[7] Akram Ben Ahmed and Abderazek Ben Abdallah. LA-XYZ: low latency, high throughput look-ahead routing algorithm for 3D network-on-chip (3D-NoC) architecture. In *Embedded Multicore Socs (MCSoC), 2012 IEEE 6th International Symposium on*, pages 167–174. IEEE, 2012.

[8] Akram Ben Ahmed and Abderazek Ben Abdallah. Low-overhead Routing Algorithm for 3D Network-on-Chip. In *Networking and Computing (ICNC), 2012 Third International Conference on*, pages 23–32, Dec 2012.

[9] Akram Ben Ahmed and Abderazek Ben Abdallah. Graceful deadlock-free fault-tolerant routing algorithm for 3D Network-on-Chip architectures. *Journal of Parallel and Distributed Computing*, 74(4):2229–2240, 2014.

[10] Akram Ben Ahmed and Abderazek Ben Abdallah. Adaptive fault-tolerant architecture and routing algorithm for reliable many-core 3d-noc systems. *J. Parallel Distrib. Comput.*, 93(C):30–43, July 2016.

[11] Akram Ben Ahmed, Abderazek Ben Abdallah, and Kenichi Kuroda. Architecture and design of efficient 3D network-on-chip (3D NoC) for custom multicore SoC. In *Broadband, Wireless Computing, Communication and Applications (BWCCA), 2010 International Conference on*, pages 67–73. IEEE, 2010.

[12] Akram Ben Ahmed, Achraf Ben Ahmed, and Abderazek Ben Abdallah. Deadlock-Recovery Support for Fault-tolerant Routing Algorithms in 3D-NoC Architectures. In *Embedded Multicore Socs (MCSoC), 2013 IEEE 7th International Symposium on*, pages 67–72. IEEE, 2013.

[13] Akram Ben Ahmed, Michael Meyer, Yuichi Okuyama, and Abderazek Ben Abdallah. Adaptive Error-and Traffic-aware Router Architecture for 3D Network-on-Chip Systems. In *IEEE Proceedings of the 8th International Symposium on Embedded Multicore/Manycore SoCs (MCSoC-14)*, pages 197–2014, 2014.

[14] Akram Ben Ahmed, T. Ochi, S. Miura, and Abderazek Ben Abdallah. Run-Time Monitoring Mechanism for Efficient Design of Application-Specific NoC Architectures in Multi/Manycore Era. In *Complex, Intelligent, and Software Intensive Systems (CISIS), 2013 Seventh International Conference on*, pages 440–445, July 2013.

[15] W. J. Dally. Virtual-channel flow control. *IEEE Transactions on Parallel and Distributed Systems*, 3(2):194–205, Mar 1992.

[16] W. J. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, 2004.

[17] Khanh N. Dang, Michael Meyer, Yuichi Okuyama, Xuan-Tu Tran, and Abderazek Ben Abdallah. A soft-error resilient 3d network-on-chip router. In *IEEE 7th International Conference on Awareness Science and Technology*, 2015.

[18] Andrew DeOrio, David Fick, Valeria Bertacco, Dennis Sylvester, David Blaauw, Jin Hu, and Gregory Chen. A reliable routing architecture and algorithm for nocs. *Trans. Comp.-Aided Des. Integ. Cir. Sys.*, 31(5):726–739, May 2012.

[19] B. S. Feero and P. P. Pande. Networks-on-chip in a three-dimensional environment: A performance evaluation. *IEEE Transactions on Computers*, 58(1):32–45, Jan 2009.

[20] S. Furber. Living with failure: Lessons from nature? In *Eleventh IEEE European Test Symposium (ETS'06)*, pages 4–8, May 2006.

[21] Shohei Miura, Abderazek Ben Abdallah, and Kenichi Kuroda. PNoC: Design and Preliminary Evaluation of a Parameterizable NoC for MCSoC Generation and Design Space Exploration. In *The 19th Intelligent System Symposium (FAN 2009)*, pages 314–317, 2009.

[22] Kenichi Mori, Abderazek Ben Abdallah, and Kenichi Kuroda. Design and evaluation of a complexity effective network-on-chip architecture on FPGA. In *Proc. of The 19th Intelligent System Symposium (FAN 2009)*, pages 318–321, 2009.

[23] Kenichi Mori, Adam Esch, Abderazek Ben Abdallah, and Kenichi Kuroda. Advanced design issues for OASIS network-on-chip architecture. In *Broadband, Wireless Computing, Communication and Applications (BWCCA), 2010 International Conference on*, pages 74–79. IEEE, 2010.

[24] Ryuya Okada. Architecture and Design of Core Network Interface for Distributed Routing in OASIS NoC. Master's thesis, The University of Aizu, 2012.