This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2020.2991749, IEEE Transactions on Computers

ECDR^2: Error Corrector and Detector Relocation Router for Network-on-Chip

Letian Huang, Chikun Yuan, Junshi Wang, Masoumeh Ebrahimi, Xuan Xie, Qiang Li

Abstract—Network-on-chip (NoC) is commonly used in modern many-core systems due to their high bandwidth and flexibility. As the manufacturing process keeps scaling, the reliability challenge in NoCs increases as well. The error correction code (ECC) is widely adopted in error correction NoCs to improve the data correctness. At the same time, extra stages are introduced in the router pipeline to improve the error correction capability. As a result, conventional error correction routers suffer from high network latency. Motivated by this limitation, i.e., we remove the extra pipeline stages delicately introduced for error correction. We propose an error correction router, called error corrector and detector relocation router (ECDR2), whose architecture optimizes the pipeline flow of the router. As a result, it can achieve both low latency and high error correction. Experimental results show that, compared with the baseline design, ECDR^2 obtains to 13.67% and 39.4% less average latency under the uniform traffic pattern and Dedup benchmark, respectively, in an 8×8 mesh NoC. The circuit area of ECR is also 7.9% less than that of the baseline design under 45-nm technology.

Index Terms—Network-on-Chip, Error Correction, Latency, Reliability

1 Introduction

Networks-on-Chip (NoCs) are commonly adopted in Multi-Processor System-on-Chip (MPSoC). NoCs bring the main advantages of high bandwidth, scalability, and flexibility [1]. In smaller technology nodes, NoCs suffer from the reliability problem caused by the soft error, crosstalk, and aging, among others [2]. Errors may occur on the delivered data and affect the reliability of the entire system. Therefore, it is crucial to embed the error correction capability in NoCs and enhance the reliability of data delivery.

The error correction code (ECC) is a widely used error correction method in NoCs [3], [4], [5], [6], [7]. The basic principle of ECC is recovering the erroneous data by adding redundant bits to the original data. The original data is first converted to the well-designed codewords by *encoders*. Then, *correctors* and *decoders* check the correctness of codewords to remove errors. *Correctors* correct the detected errors while *decoders* convert the codewords to the original data. Consequently, *decoders* are typically used at the end of the data transmission while *correctors* are used during the transmission (*e.g.*, in routers).

The pipeline stages in NoC routers are used to improve the network throughput. On the other hand, error correctors are used in routers to ensure the correctness of data. However, the circuit path delay of the corrector is as large as the critical path of a router. Thereby, to implement correctors and at the same time keep high clock frequency, traditional error correction routers introduce extra pipeline stages [3], [4], [5], [6], resulting in extended network latency.

Besides error correction, there are several other mechanisms to detect or tolerate faults in NoC. Built-in self-test (BIST) is one of these mechanisms, that sets the test controller, pattern generators, and wrappers in the routers [8]. Using BIST, faulty components in NoCs can be detected. Fault-tolerant routing algorithms [9] also help to achieve higher reliability in NoCs that aim to deliver packets to their destinations correctly through rerouting packets. It is also possible to diagnose the router components by setting some operation rules, such as in NoCAlert [10]. NoCAlert is a rule-based mechanism, which provides a comprehensive on-line and real-time fault detection scheme for NoC error detection. To diagnose faults in the control components, NoCAlert uses some rules to detect whether the packet's output is illegal for the current inputs. In this paper, we only focus on the ECC-based soft error correction mechanism. Most of the other mechanisms are orthogonal to ECC and can be easily combined with the proposed error corrector and detector relocation router architecture.

In this brief, we propose an error correction router architecture named error corrector and detector relocation router ECDR^2. ECDR^2 brings the main advantages of low network latency, high reliability, and low hardware overhead. Compared with conventional error correction routers, the proposed ECDR^2 neither introduces extra stages to the router pipeline nor affects the clock frequency. The design of ECDR^2 is based on RTL synthesizable Verilog HDL, and the evaluation platform is built using SystemVerilog. Different routers, including the proposed ECDR^2 and the baseline router, are evaluated under different synthetic traffic patterns as well as the PARSEC benchmark [11]. Results show that, compared with the baseline router, on average, ECDR^2 reduces the network latency by 14.57% under six traffic patterns and by 30.84% under nine PARSEC benchmarks.

This paper is organized as follows. Section 2 describes the related work. In Section 3, we first introduce the generic router architecture without error correction capability, then we propose an error correction router as the baseline design. Section 4 presents the principle and architecture of ECDR^2. Section 5 gives the experimental results and analysis, and Section 6 concludes the paper. The main contribution of this brief includes:

- Propose the ECDR^2 and its micro-architecture that provides low latency and high reliability for NoC design.
- Carry the evaluation that proves the high performance, high reliability, and low cost of ECDR^2.

2 Related Work

Classical NoC routers utilize 4 pipeline stages [12]. To reduce the number of stages, techniques such as speculative switch allocation (SSA) [13] and lookahead routing calculation (LRC) [14] can be applied. Each of these techniques can reduce one pipeline stage, and thus a 2-stage router can be implemented. There are also 1-stage routers such as the dimension-sliced router [15] and virtual output queued (VOQ) router [16]. The dimension-sliced router [15] cannot support adaptive routing while the VOQ router [16] decreases the clock frequency due to the high fanout in the output port.

- *Letian Huang, Qiang Li are with School of Electronic Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China.*
- *Chikun Yuan was with University of Electronic Science and Technology of China, Chengdu, China. He is now with Guangdong OPPO Mobile Telecommunications Co., Ltd., Dongguan, China.*
- *Junshi Wang was with University of Electronic Science and Technology of China, Chengdu, China. He is now with Beijing Zhaoxin Electronic Technology Co., Ltd., Beijing, China.*
- *Xuan Xie is with School of Automation Engineering, University of Electronic Science and Technology of China, Chengdu, China.*
- *Masoumeh Ebrahimi is with Royal Institute of Technology, Stockholm, Sweden and University of Turku, Turku, Finland.*
- *Corresponding author: Letian Huang, E-mail: huanglt@uestc.edu.cn.*

Manuscript received April 19, 2005; revised August 26, 2015.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2020.2991749, IEEE Transactions on Computers

In the field of error correction routers, remarkable works have been published to improve the NoC reliability. However, most of these works are based on the classical 4-stage router architectures, which usually lead to a higher network latency than the designs with the lower number of pipeline stages. Moreover, extra pipeline stages are usually introduced to achieve error correction capability. But extra pipeline stages increase the latency further.

The ECC-based fault-tolerant NoCs are generally divided into two categories: End-to-End (E2E) and Hop-to-Hop (H2H). In the E2E scheme, packets are encoded at the source and decoded at the destination [7], and it can be implemented in both software and hardware levels. Nevertheless, by applying this scheme, errors may accumulate during the NoC data delivery, and they grow beyond the capacity of the employed ECC to correct them at the destination. The H2H scheme, on the other hand, detects and corrects errors at every router, which provides higher reliability at the cost of more area overhead. The proposed ECDR^2 utilizes the H2H scheme to achieve higher reliability.

Daniele Rossi *et al*. [3] proposed two ECC implementation methods in NoCs with different quality of service (QoS) levels. One of them targets a lower overhead router design while the other one aims at a higher reliability router. The *low-overhead method* encodes the flits before they leave a router, then corrects and decodes the flits when they enter a router. This method has limited reliability, as data is only protected on the links, and errors in routers are not concerned. In the *reliable method*, flits are encoded at the network interface (NI) before they are injected into the network. Then, correctors, located at every router port, detect errors and correct the flits. This method can protect the data during the whole data transfer in routers and links. Although the two mentioned methods provide different QoS levels, the circuit delay of ECC codecs is long, and so they introduce at least one extra pipeline stage for error correction.

An error control scheme was proposed in [4] that integrates link error recovery and buffer error correction. This router implements two encoder and decoder pairs. One is for the link error correction and another one is for the buffer error correction. A flit recovery algorithm is also presented in this work to enhance reliability further. However, the network latency is not well considered in this work as implementing four ECC codecs in one router needs several extra pipeline stages. As a result, although offering high reliability, this scheme greatly increases the latency and degrades the network performance.

Lu Wang *et al*. proposed a fault-tolerant router architecture with the ability to detect permanent faults [5]. In this architecture, ECC is used to not only correct errors but also detect permanent faults in the buffer. Flits are encoded upon entering a buffer, and their code redundancy is stored in another buffer, which is specially designed for this purpose. When flits leave the buffer, they are encoded again and compared with the previously generated code to detect permanent faults. The error correctness also takes place in the comparison phase. This work adopts a generic 2-stage router as a baseline while its fault-tolerant version requires an extra stage for error correction, and thus it leads to a 3-stage pipeline.

Dongkook *et al*. proposed a hop-by-hop re-transmission router in [6]. The router has a 3-flit-depth re-transmission buffer per virtual channel. The buffers maintains the flits until the downstream router correctly receives them. Nevertheless, this architecture has large design complexity and hardware overhead due to the re-transmission control and the employed buffers. More importantly, this router adopts one extra pipeline stage for error detection.

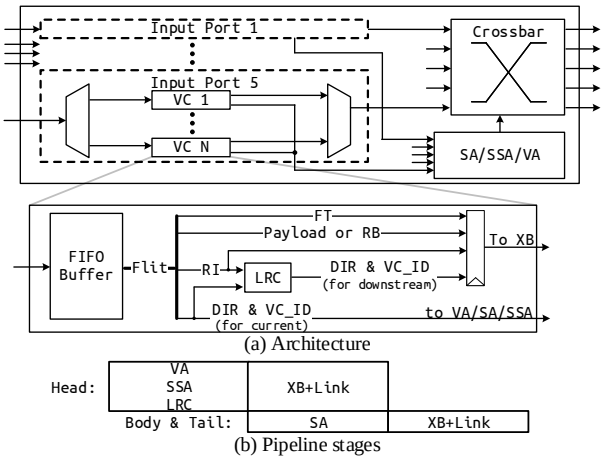


Fig. 1: A generic 2-stage router without error correction ability

Pavan Poluri *et al*. proposed an error correction router without adding extra pipeline stages in [17]. In this router, the routing computation (RC) is executed twice on a packet. After the execution of the standard RC, the second RC with the error correction capability is executed in the next pipeline stage. The execution is in parallel with the virtual channel allocation unit. The results of two RCs are compared with each other to determine the correctness of the standard RC. By executing the second RC in parallel with virtual channel allocation (VA), the router does not introduce an extra stage. However, this work has two main drawbacks as low latency and low reliability. This method is implemented on the 4-stage pipeline router which is not a proper indicator of a good performance. Even though the method does not add an extra stage, the latency of the router is still high. Besides, this method only protects the routing information (RI), but not the data carried by packets.

In summary, conventional ECC-based NoC routers have one characteristic in common — they adopt extra pipeline stages to implement error correction [3], [4], [5], [6]. However, the number of pipeline stages in the high-performance routers are strictly limited, and adding an extra stage to the pipeline can significantly increase the network latency. Our work is motivated by this drawback of the error correction routers. We design ECDR^2 based on a 2-stage low-latency router without adding any extra pipeline stages for error correction. All packet's data is protected to achieve high reliability.

3 Generic and Baseline Router

In this section, we first describe the architecture of a generic 2-stage router without the error correction capability [13], [14], in which our ECDR^2 architecture is developed based on that. Then, we explain the baseline design that is a 3-stage error correction router with one pipeline stage dedicated to error correction. This type of routers was widely adopted in previous works [3], [4], [5], [6]. For the ease of reading, Table 1 lists the frequently used acronyms in this paper.

TABLE 1: Acronym list

Acronym	Description
LRC	Lookahead Routing Computation
FT	Flit Type (head, body, and tail)
RI	Routing Information (packet source, destination)
DIR	Direction (indicate packet output port)
RB	Reserved Bits (flags, packet ID, size, and so on)
OHC	One-Hot Check
RDC	Redundancy (of ECC/EDC)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2020.2991749, IEEE Transactions on Computers

3.1 Generic 2-stage router without error correction ability

The primary function of an NoC router is to receive packets from the input ports and forward them to the output ports. Fig.1 (a) shows the architecture of a generic 2-stage router. The router has five input ports and five output ports. Each input port is equipped with N virtual channels (VCs). In order to reduce pipeline stages, the generic router utilizes the speculative switch allocator (SSA) [13] and LRC [14]. The essential components of the generic router are as follows:

Lookahead Routing Computation (LRC): LRC is the complement of RC. The basic function of RC is deciding the packet's output direction (DIR) for the current router. To reduce the pipeline stages, LRC generates DIR for the downstream router. In this case, the DIR for a given router is generated one hop ahead and arrives simultaneously with the head flit. If a router utilizes virtual channels, LRC also generates the VC number for the downstream router. Therefore LRC can be executed in parallel with allocators because LRC of the upstream router has already generated the information required by allocators (i.e., DIR and VC ID). LRC only performs on the head flits as other flits do not carry routing information (RI). The input signals of LRC are the packet's RI and DIR from the upstream router.

Virtual channel Allocation (VA): VA in a given router is responsible for allocating VC in the downstream router for the new incoming packet. VA arbitrates between VCs that are requesting the same downstream VC. Once a VC is granted, it holds the grant until the packet is entirely sent out. VA is only executed for the head flit, and other flits follow the head. The input signals of VA are DIR and the VC number.

Switch Allocation (SA): Multiple VCs on an output port share the same physical link, and the link can transfer only one flit in a single cycle. Consequently, SA is used to arbitrate between VCs that are requesting the same output port. Generally, VA and SA are executed in sequence due to their logical dependence. First, the VA should be granted, and then the SA request can assert.

Speculative Switch Allocation (SSA): To achieve low latency, SSA [13] is adopted in this generic router. The switch allocation request can be marked either as speculative or non-speculative. When a packet starts performing allocations, the VA request and the speculative SA request assert at the same time. If both VA and SSA are succeeded, the requests for the virtual channel and switch allocation will be granted for the head flit in a single cycle. If only VA is succeeded, the packet must request SA again in the next cycle. However, the SA request is no longer marked speculative since the packet is already allocated an output virtual channel. On the other hand, if the VA fails, no matter SSA fails or not, the packet must request VA and SSA in the next cycle.

The SSA method works well under low traffic conditions as fewer conflicts result in better speculation. However, there is not any performance penalty due to speculation failure. The reason is that none-speculative SA requests are set prior to speculative SA requests. So none-speculative packets, which have already been allocated an output VC, will always win SA to speculative packets, which are expecting an output VA. The input signal of SA and SSA is the output port ID, which is determined by the packet's DIR.

Fig.1 (b) shows the pipeline stages of the generic 2-stage router. In the first stage of the router, LRC, VA, and SSA work in parallel on the head flit, and SA is executed on the body and tail flits. In the second stage of the router, flits are sent to the crossbar for the link traversal.

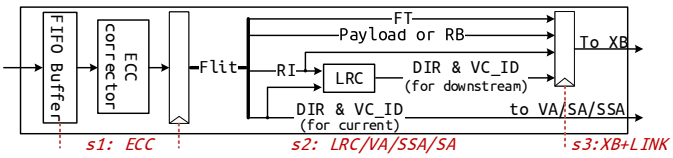


Fig. 2: Baseline 3-stage router with error correction ability

3.2 Baseline 3-stage router with error correction ability

We build the baseline error-correction router by adding an extra pipeline stage to the generic router. The VC architecture of the baseline 3-stage router is illustrated in Fig. 2 (a). Other components such as allocators and crossbar are the same as those in the generic router and are not shown in the figure. To obtain the error correction ability, the generic router adopts an error correction unit between the buffer and LRC. A set of registers is placed after the ECC unit to store the corrected flits. These registers are necessary to perform the error correction in a single pipeline stage and to prevent the long circuit delay. As a result, the first stage of the baseline router is consumed by error correction. Except for the corrector, the VC structure of the baseline router is the same as the one in a generic 2-stage router.

In the baseline error-correction router, the allocator and LRC units are executed in parallel after the ECC stage. As mentioned before, only RI and DIR are needed for the execution of LRC and allocators (instead of the whole packet), and they occupy a few bits in the head flit. By considering these observations, we design our ECDR^2 architecture and merge the error correction, detection, LRC, VA, SA, and SSA into one stage, while keeping clock frequency high.

4 Error Corrector and Detector Relocation Router

The purpose of ECDR^2 is to eliminate the extra pipeline stages dedicated to error correction. In ECDR^2, we use both ECC and EDC to protect NoC flits and to avoid long circuit delay. This section first explains the ECC and EDC methods which are necessary to support ECDR^2. Then we present the principle of ECDR^2, followed by its detailed micro-architecture.

4.1 ECDR^2 ECC/EDC code methods

In an NoC packet, there are three different types of flits (*i.e.*, head, body, and tail), each containing different data. As shown in Fig 3, the head flit contains RI, DIR, and VC_ID. Other bits in the head flit are reserved for the packet size, flags, and packet ID. There are also some spare bits which are called reserved bits (RB) in this brief. Body and tail flits, on the other hand, carry payloads. All flits use 2 bits to indicate the flit type (FT). For packet routing, the critical data are RI, DIR, VC_ID, and FT, while payload and RB are non-critical data.

The idea of ECDR^2 is to treat critical and non-critical data differently from reliability, overhead, and decoding delay perspectives. The two main aspects in ECC/EDC are as follows: (1) reliability, where higher reliability means lower coding efficiency (*i.e.*, the ratio of protection bits to the coded bits is

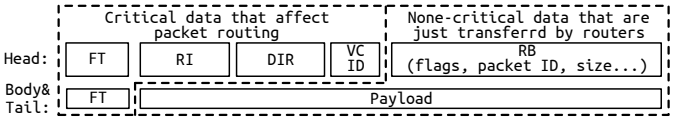


Fig. 3: NoC packet structure

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2020.2991749, IEEE Transactions on Computers

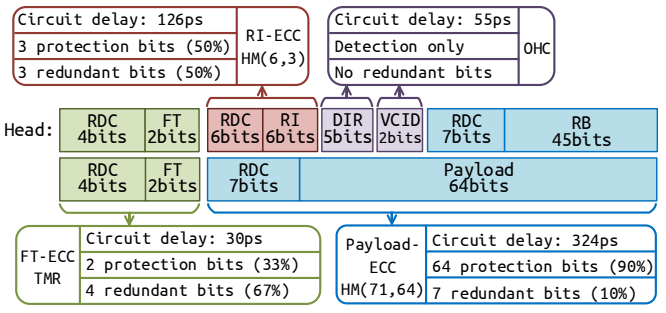


Fig. 4: The design example of ECDR^2 coding methods. RDC refers to redundancy for each ECC.

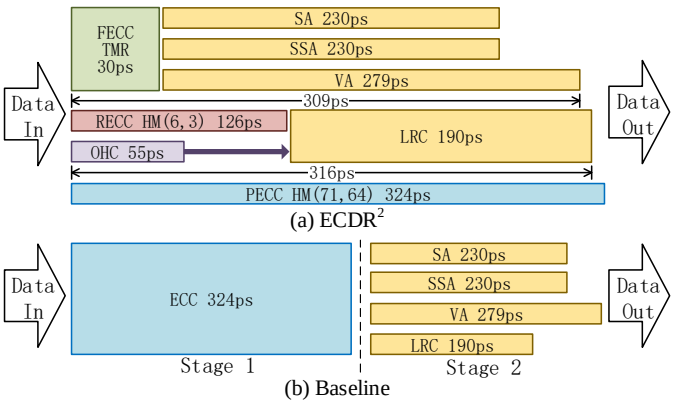


Fig. 5: The locations of ECC corrector and detector

high); (2) decoding delay, where more protection bits result in longer decoding delay. Considering these two aspects, ECDR^2 protects critical data with high-reliable and short-delay codes, and the non-critical data with relatively lower-reliable and longer-delay codes. By using short-delay codes for critical data, no extra pipeline stage would be needed while keeping the clock frequency high. On the other hand, codes with shorter protection bits are used for non-critical data to reduce redundancy and hardware overhead. A design example of different code methods in ECDR^2 is shown in Fig. 4 where the circuit delay is obtained from *Synopsys Design Compiler* under TSMC 45-nm technology.

ECC for FT (FT-ECC): The head flit determines the VC in the downstream router, and the tail flit indicates the end of a packet. An erroneous flit type (FT) may lead to incomplete packets, packet missing, and even deadlock. The case is especially catastrophic when a body or tail flit is erroneously seen as the head flit. To achieve good protection, ECDR^2 adopts triple modular redundancy (TMR) to code FT.

ECC for RI (RI-ECC): Routing information (RI) is usually used to indicate the destination of a packet. Erroneous RI may result in sending the packet in a wrong direction, which in turn may also lead to packet missing or deadlock. The width of RI is 6 bit in this design example, so we adapt two HM(6,3) codes to protect it.

One-hot check (OHC): One-hot coding is used to code DIR and VC_ID due to its simplicity. One-hot coding can be used to request VA and SA allocations without the need for any decoding, and thus it prevents long critical delay. Moreover, one error at any bit of a one-hot codeword makes the code no longer one-hot. Consequently, changing form a one-hot code to another one-hot code needs at least two erroneous bits. As a result, the one-hot code innately has one-bit error detection ability. The design example utilizes OHC to check the correctness of DIR and VC_ID.

ECC for Payload and RB (Payload-ECC): Payload and reserved bits (RB) are only used at the destination node; however, they occupy most of the bits in a packet. By assuming a 64-bit width 5-flit packet, the payload occupies 256 bits, and the reserved data occupies 45 bits. Therefore, we protect the payload and RB by ECC with high coding efficiency, which is HM(71,64) code, to minimize the flit width and storage overhead. The HM(71,64) code has 64 protection bits and 7 redundancy bits. Although RB has only 45 bits, HM(71,64) can also be applied to it by setting the other 19 bits to a default constant value. In this case, the encoder, corrector, and decoder needed for HM(71,64) can be reused for both the payload and RB, which reduce the hardware overhead.

The use of these four codes (*i.e.*, FT-ECC, RI-ECC, OHC, and Payload-ECC) does not impose a large overhead. The reason is that FT-ECC, RI-ECC, and OHC only protect very few bits in the packet, and their correctors are light-weight accordingly. Moreover, Payload-ECC is always needed in hop-to-hop (H2H) error correction routers, and it is not specific to ECDR^2. In other words, critical data are protected with low-efficiency codes, but they occupy only a few bits in a packet; Payload and RB occupy most of the bits in a packet, but their coding efficiency is high. As a result, the code method in ECDR^2 introduces minimal encoding redundancy bits that saves the hardware overhead.

4.2 Eliminating the extra pipeline stage for error correction

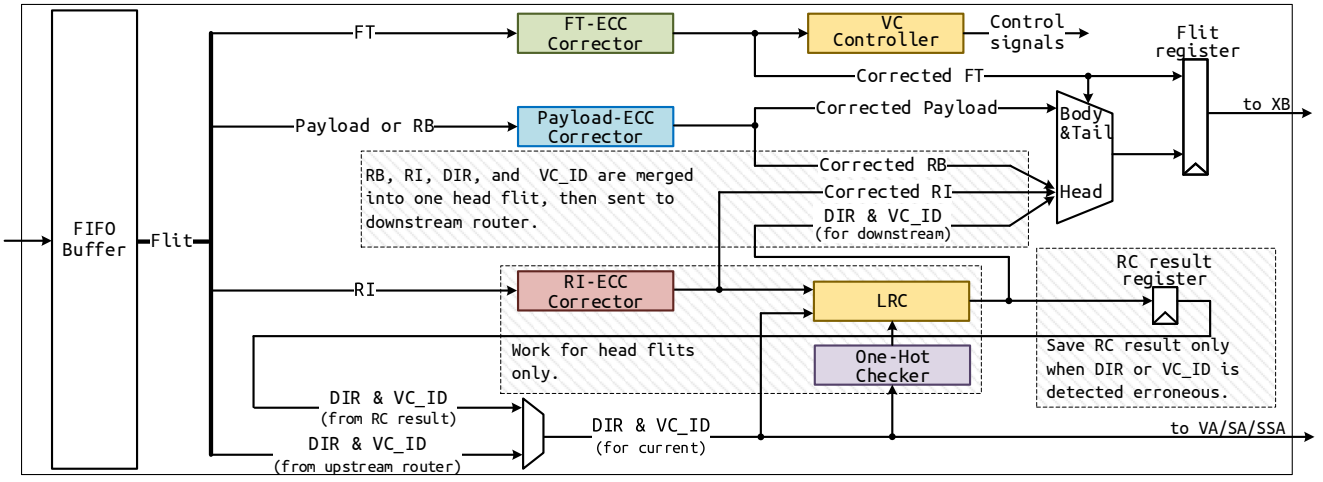
Based on the proposed code methods, ECDR^2 can eliminate the extra pipeline stage by merging error corrector, detector and *routing units* (*i.e.*, LRC, VA, SA, SSA) into one stage. Fig. 5 (a) illustrates the principle of ECDR^2, which shows the locations of error corrector and detector with regard to the routing units. In the Fig. 5, we show the delay of each block. Furthermore, the length of each block represents its critical path delay.

Different units need specific packet information to be used as their input signals. For example, since the VC allocation function differs depending on the flit type (FT), the VC controller needs the error-free flit type to control the allocations request correctly. As a result, the FT-ECC correction is executed before allocators to ensure the correct allocation operation. In the same manner, the RI-ECC corrector and OHC unit are set prior to LRC unit since LRC needs RI and DIR as its inputs. The payload and RB are none-critical data, which do not affect packet routing, and thus they are not needed by any routing units in a router. Consequently, the Payload-ECC corrector runs in parallel with other units.

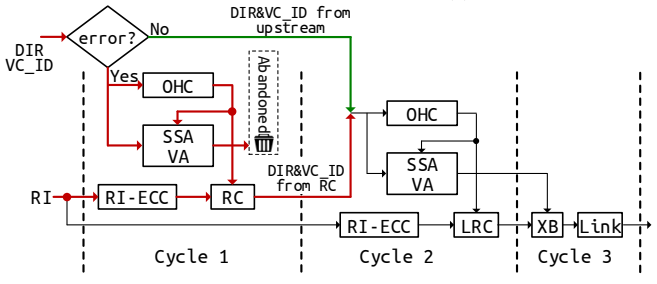
Although FT-ECC, RI-ECC, and OHC units are set before routing units, their critical path delay is short because of the specially designed code methods. As can be seen in Fig.5 (a), the summed delay of FT-ECC and VA is 309ps while that of RI-ECC and LRC is 316ps. Both are shorter than Payload-ECC critical path delay that is 324ps; therefore, ECDR^2 does not increase the critical path delay of the entire router. In contrast, as shown in Fig.5 (b), in the baseline router, the ECC units occupy one extra stage to prevent a long critical-path delay. Otherwise, if the ECC correction and the other components are simply placed into one stage, the accumulated delay of ECC and VA will be 603ps, which results in low clock frequency and degraded performance. This is the main reason why traditional routers introduce extra pipeline stages for error correction.

ECDR^2 is built based on the fact that the critical data occupies a few bits in a packet, and short-delay codes are used to protect them. By merging corrector, detector, and routing

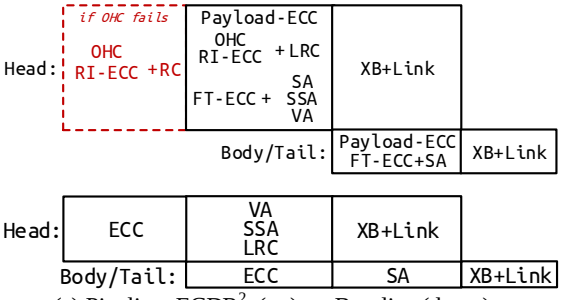
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2020.2991749, IEEE Transactions on Computers



(a) Micro-architecture of a virtual channel



(b) Operations to handle erroneous DIR or VC_ID



(c) Pipeline: ECDR^2 (up) *vs.* Baseline (down)

Fig. 6: Overview of ECDR^2 micro-architecture

units into the same pipeline stage, ECDR^2 eliminates the extra stage dedicated for error correction. As a result, protecting critical data with short-delay codes realizes a 2-pipeline error-correction router while keeping the clock frequency high. Moreover, protecting payload and RB the with small-redundancy code minimizes the width of the coded flit.

4.3 ECDR^2 micro-architecture

Fig. 6 (a) illustrates the micro-architecture of a VC in ECDR^2. The units out of VC remain the same as the generic router. As shown in Fig. 6 (a), FT, RI, payload, and RB are sent to their corresponding correctors. The VC controller receives the corrected FT and controls other units in the VC according to the flit type.

If the VC is currently handling a body or tail flit, the Payload-ECC corrector receives the payload from the buffer and sends the corrected data to the *flit register* through a 2-to-1 multiplexer. The corrected FT is also sent to the flit register to form a complete flit. The flit register is placed between packet routing and link traversal, dividing them into two stages and thus preventing long critical path delay. The RI-ECC corrector, LRC unit, and one-hot checker are not executed on the body and tail flits.

The protection of the head flit is different from body and tail flits. First, we discuss the ECC assuming that there is no error in DIR and VC_ID because they are protected by EDC rather than ECC. A head flit is composed of FT, RI, DIR, VC_ID, and RB. The RB is corrected by the Payload-ECC corrector, while RI is corrected by the RI-ECC corrector. The DIR and VC_ID for the downstream router are calculated by the LRC unit. These data form the new head flit, which is stored in the flit register and sent to the downstream router through the crossbar in the next stage.

In addition to the bits corrected by ECC, the other bits such as DIR and VC_ID are protected by EDC (one-hot code) which has no correction ability. The one-hot checker is used to detect whether any error exists in DIR or VC_ID. Nevertheless, the one-hot code can only detect errors but cannot correct them. Thereby, it is impossible to recover the data when errors occur. A straight-forward method is to drop the packet and re-transmit it, but it decreases the network performance and reliability. To address this issue, ECDR^2 adopts a similar method as [9]. To successfully forward the packet with erroneous DIR or VD_ID, ECDR^2 executes the standard RC, instead of lookahead, to generate the correct DIR and VC_ID for the current router. Because RC and LRC have almost the same logic function, the LRC unit can be reused to execute the standard RC.

Fig. 6 (b) shows the cases for erroneous and correct one-hot code with red and green lines, respectively, while the black lines are used by both cases. If DIR or VC_ID has an error, OHC detects the error and informs the LRC unit in Cycle 1. Then, the LRC unit changes its function to execute the standard RC (instead of lookahead), *i.e.*, calculate DIR and VC_ID for the current router. Meanwhile, the VA and SSA work in parallel with OHC and RC. However, at the end of Cycle 1, the results of allocators are abandoned as they are made based on the erroneous DIR and VC_ID. Otherwise, if DIR and VC_ID have no error, Cycle 1 is skipped. In Cycle 2, DIR and VC_ID are used for allocators and LRC, no matter they are newly generated in Cycle 1 or from the upstream router. To achieve the above function, a set of registers, named RC result register, is used to store RC results as shown in Fig.6 (a). When errors are detected, this register stores the results of the standard RC. In this case, the erroneous DIR and VC_ID are corrected by an additional clock cycle. The circuit area overhead is negligible due to the

reuse of the LRC unit.

Fig.6 (c) shows the pipeline stages of the ECDR^2 and baseline router, differentiated by the head and body/tail flits. Regarding the head flit, the stage count varies depending on whether OHC fails or not. For ECDR^2, only when OHC fails, the standard RC is executed, and thus it introduces an extra stage. Many operations are executed in the second stage, and they are highly paralleled. Since the Payload-ECC corrector does not feed other components, it works in parallel with others. RI-ECC and OHC run prior LRC to ensure correct RI and detect faults in DIR. The FT-ECC corrector is executed before allocators as the allocation function differs depending on the flit type. Regarding the body/tail flits, in the first stage, the FT-ECC corrector and SA are executed in parallel with the Payload-ECC corrector as they do not have any data dependence. In the second stage, the flit is transferred to the downstream router via the crossbar and link. As for the baseline router, the stage number is fixed three. In most cases (i.e., there is no error in RI), ECDR^2 has one less stage than the baseline router.

5 Experiment Results

5.1 Experiment setup

To evaluate the network latency and error correction capacity of ECDR^2, we have performed RTL simulations. As introduced in the Sub-section 3.2, our baseline design is a 3-stage error correction router [3], [4], [5], [6]. We have also evaluated the generic 2-stage router to provide comprehensive results. To have fair error correction capacity, both ECDR^2 and the baseline router adopt the same code method, as shown in Fig.4. However, the baseline router has no scheme to reuse the LRC unit, so packets with erroneous DIR or VC_ID are dropped by the baseline router.

We have used synthesizable Verilog HDL to design all the routers and SystemVerilog to build the simulation platform. The simulated NoC utilizes 8×8 mesh topology and adopts the XY deterministic routing algorithm [18]. All routers have two VCs per input port, and the buffer in each VC has fore slots. The comparison matrix includes network reliability, network latency, and circuit area. The latency of a packet is defined as the number of cycles from when the first flit is injected to the network and the last flit is ejected from the network. The network latency is calculated as the average latency of all packets.

To measure the packet latency, we have used six synthetic traffic patterns as well as various traffic traces from the PARSEC benchmark [11]. For six traffic patterns, 1000 packets are used to warm up the NoC, then 10000 packets are injected to each node for latency measurement. For the PARSEC traffic traces, we have used Simplescalar [19] and ESYNet [20] to build and simulate a 64-node many-core system equipped with an 8×8 mesh interconnection. The 16 routers located at the west and east borders of the mesh are connected to the memory controllers, while each of the other 48 routers is connected to an Alpha 21264 processing core. Each processing core has a private 32 KB L1 cache, and the entire system has 8 MB shared L2 cache, distributed in the whole mesh.

TABLE 2: Percentage of erroneous packets

(a) Baseline

Error Rate	1E-5	1E-4	2.5E-3	5E-3	7.5E-3	1E-2
Total	0.366%	4.25%	12.94%	34.55%	47.60%	66.14%
Payload	0.365%	4.14%	12.65%	33.76%	46.73%	64.60%
RI	0.011%	0.11%	0.29%	0.79%	0.87%	1.54%

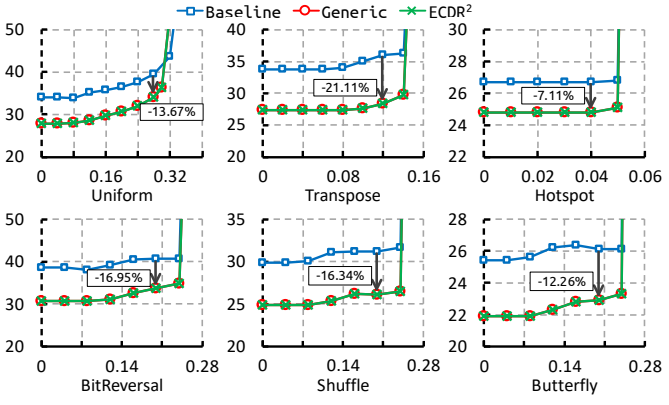
(b) ECDR^2

Error Rate	1E-5	1E-4	2.5E-3	5E-3	7.5E-3	1E-2
Total	0.005%	0.288%	1.71%	6.90%	15.75%	29.16%
Payload	0.005%	0.288%	1.71%	6.89%	15.67%	28.61%
RI	0.000%	0.000%	0.003%	0.014%	0.083%	0.55%

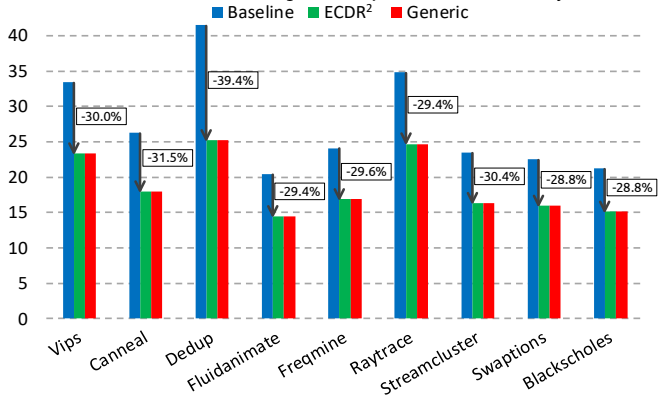
5.2 Network reliability

To evaluate the network reliability, the simulation platform should have the ability to inject errors to NoCs. In our simulation platform, errors are injected by flipping values of data bits. We define a parameter named *error rate* that is the probability of flipping a bit value in one cycle. Note that the error injection in different bits is uncorrelated, meaning that errors can be injected into multiple bits in one cycle.

Table 2 reports the percentage of erroneous packets in the baseline and ECDR^2 routers under the uniform traffic pattern and the packet injection rate of 0.1 flit/cycle/router. The payload row shows the percentage of packets that reach their correct destinations but carry erroneous payload or RB, while RI row shows the percentage of packets that reach wrong destinations or have been dropped in NoC. The table does not represent the generic router, because the generic router has no error correction ability, and it encounters deadlock even when the error rate is very low. As can be seen from the table, ECDR^2 leads to lower erroneous packets than the baseline router in all error rates. There are two main reasons for the better reliability of ECDR^2. First, ECDR^2 adopts the reuse of the LRC unit which can protect packets with the erroneous DIR and VC_ID while the baseline router drops these packets. Second, the extra pipeline stage in the baseline router results in



(a) Average latency under synthetic traffic patterns (in clock cycle). The horizontal ordinate is the packet injection rate in flit/cycle/router



(b) Average latency under PARSEC benchmark suits (in clock cycle)

Fig. 7: Average latency in the error-free environment

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2020.2991749, IEEE Transactions on Computers

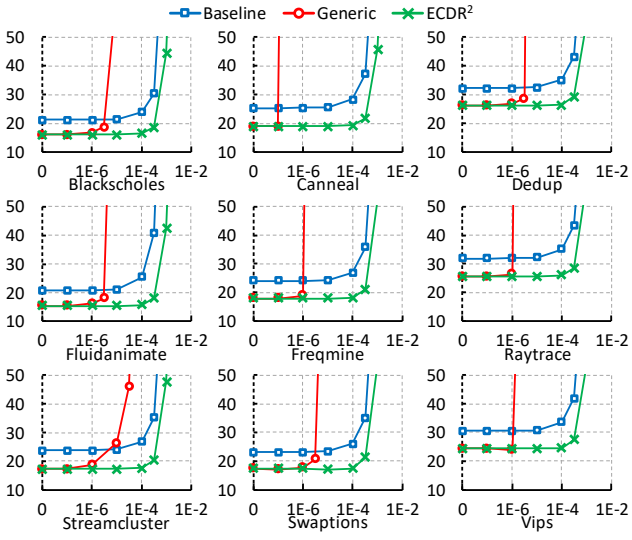


Fig. 8: Average latency (in clock cycle) under PARSEC benchmark suits and for different error rates

a longer data path. The longer data path introduces more errors because errors are injected into every data bits in NoC. As a result, more errors are injected into the baseline router during the same period. Also, as can be observed from this table, the erroneous count in RI is significantly smaller than the payload. Since ECDR^2 provides stronger protection for RI, deadlock can be prevented when the error rate is high.

5.3 Network latency in the error-free environment

Both the router architecture and the error correction ability affect the network latency. To evaluate the pipeline stage elimination, we only compare the network latency caused by the router architecture difference in this subsection. The simulations are carried in an error-free environment, *i.e.*, no error is injected.

Fig.7 (a) illustrates the network latency under different traffic patterns. It can be seen that the generic router and ECDR^2 show almost the same latency characteristics. The reason is that the architectures of both routers are identical except the error correction components which are not utilized in the error-free environment. Fig.7 (a) also shows that the latency of the baseline router is always lager than ECDR^2. The reason lies behind the extra pipeline stage in the baseline router, which adds one cycle for delivering a flit in a router. As a result, a packet experiences several cycles larger latency than in ECDR^2 depending on the distance between its source and destination node and the network traffic.

The packet latency under the PARSEC benchmark in an error-free environment is shown in Fig.7 (b). Similar to the case of synthetic traffic, the generic router and ECDR^2 show a similar latency while the latency of the baseline router is always larger. The most obvious benchmark is *Dedup*, on which ECDR^2 has 39.4% less latency than the baseline router. *Dedup* has relatively heavy traffic among all benchmarks, especially in systems with many cores [11]. The shorter path in the generic and ECDR^2 leads to lower traffic and thus the network latency. On average, ECDR^2 shows 30% less latency than the baseline routers.

5.4 Network latency with error injected

To understand the effect of error correction on the latency, we inject errors into the network. To fairly evaluate the packet latency when errors are injected, we adopt the retransmission scheme to ensure 100% packet delivery. The source node resends the packet if it is not correctly received by the destination node. The latency of the retransmitted packet is calculated as the number of cycles from when the packet is first injected into the network until the correct one is received by the destination. Obviously the retransmission of packets significantly increase the average packet latency.

Fig. 8 illustrates the network latency under various PARSEC benchmark suits and for different error rates. When the error rate is low, the network latency is rather stable since few packet re-transmissions occur. The latency values are similar to that of the error-free environment in Fig.7 (b). The latency of the baseline router is larger than both the ECDR^2 and generic routers as it adopts one more pipeline stage.

As the error rate increases, the number of re-transmitted packets grows, and the network latency increases consequently. In Fig. 8, we define *failure point* as the point of the error rate, where the latency begins to raise. The *failure point* can characterize the error correction capability of the routers. A lower *failure point* means the worse error correction capability. As can be seen from the Fig. 8, under all nine benchmarks, the *failure point* of ECDR^2 is higher than the baseline and generic routers. This is compatible with the results in Table 2 that shows ECDR^2 has better error correction capability, so that the number of re-transmitted packets grows lower. The *failure point* of the generic router is much lower than both ECDR^2 and the baseline router, due to the absence of the ability to correct errors. Because the increasing number of re-transmitted packets caused by errors, which also creates congestion and even deadlock in the network. The *failure point* of the *Canneal* workload is relatively low because its workload is the heaviest among all the other workloads. The heavy workload (i.e., more packets) leads to more errors and a larger probability of deadlock, which is seen as infinite latency in the simulation. The *Streamcluster* workload is relatively light, so there is a low probability of encountering deadlock, and thus the corresponding *failure point* is higher than the other benchmarks. Unlike the generic router, the latency of ECDR^2 and the baseline router increase more smoothly because of their error correction capability.

5.5 Hardware overhead

We have synthesized the RTL design of three routers with TSMC 45-nm technology. Table 3 shows the hardware comparison of different routers. The circuit area and power consumption of ECDR^2 are smaller than the baseline by 8.61% and 12.21%, respectively. The reason is that the baseline router has an extra pipeline stage which requires a set of registers to store the pipelined data. Although ECDR^2 utilizes more logic gates for error correction and LRC reuse, their hardware overhead is small compared with the extra registers of the baseline router. The generic router has the lowest area and power consumption as it does not support error correction. Regarding the clock frequency, the critical paths of all three routers are in VA. Considering this critical path, the baseline router reaches the maximum frequency of 1.47GHz, while ECDR^2 reaches 1.32GHz. The main reason is that ECDR^2 reads flits from the FIFO buffer and uses it for VA in the same clock cycle. The delay of reading from the buffer increases the critical path delay of ECDR^2. In the baseline router, buffer read and VA are placed into two different stages, thus it does not increase the critical delay.

Table 4 lists the average latency in the unit of *ns* when all the routers run at. Although the max frequency of ECDR^2 is

TABLE 3: Hardware overhead

Designs	Area(μm^2)	Ratio(%)	Power(mW)	Ratio(%)	Critical delay(ns)	Max frequency(GHz)	Ratio(%)
Baseline	51367	100.00	49.630	100.00	0.68	1.47	100.00
ECDR^2	46947	91.39	43.571	87.79	0.76	1.32	89.79
Generic	37951	73.88	34.412	69.33	0.70	1.43	97.27

TABLE 4: Average latency of PARSEC benchmark suits (in ns)

Benchmark	Baseline	ECDR^2		Generic	
	Latency	Latency	RED.(%)	Latency	RED.(%)
Vips	22.7	17.7	22.13	16.3	28.27
Canneal	17.9	13.7	23.44	12.6	29.49
Dedup	28.2	19.1	32.27	17.6	37.62
Fluidanimate	13.9	11.0	21.07	10.1	27.30
Freqmine	16.4	12.9	21.33	11.9	27.54
Raytrace	23.7	18.7	21.07	17.2	27.30
Streamcluster	16.0	12.4	22.25	11.5	28.39
Swaptions	15.3	12.2	20.35	11.2	26.64
Blackscholes	14.4	11.5	20.42	10.6	26.70

decrease by 10.21%, it still achieves more than 20% latency reduction under all benchmarks compared with the baseline. This is mainly due to the eliminated pipeline stage. The clock frequency of the proposed router and baseline router can both scale lower (for example, 1 GHz) without any timing problem. In this case, if both routers work at the same frequency, the proposed router will achieve a significantly better performance than the baseline router due to the stage elimination.

The max frequency of the proposed router is 1.32GHz. When targeting a much lower frequency (for example, 0.7 GHz), the pipeline needs a redesign to have fewer stages due to the positive timing slack. On the other hand, if a higher clock frequency (for example, 1.5GHz) is needed by the SoC specification, the pipeline also must be redesigned to have more stages for timing closure, which will introduce larger network latency and power consumption. In other words, when the clock frequency is scaled out of the applicable range, a pipeline redesign is needed, and the proposed router microarchitecture cannot be directly applied in this situation. Nevertheless, the proposed design methodology of this paper can also be referenced in the pipeline redesign even if the clock frequency needs to be scaled.

6 Conclusion

In this brief, we proposed the error correction router, named ECDR^2, without adding any extra stage to the router pipeline. We presented the router micro-architecture that adopts a 2-stage pipeline, rather than the 3-stage pipeline used by the traditional error correction routers. We merged different methods to provide full error coverage for NoC packets. RTL simulations show that the proposed ECDR^2 has a better network performance and reliability than the baseline router. The circuit area and power consumption of the proposed ECDR^2 are also lower than the baseline router as it does not have an extra pipeline stage and the required registers. Compared with the baseline router, the proposed ECDR^2 shows better network latency, reliability, and hardware overhead. Even if the microarchitecture design could not be applied directly, this design methodology can also be referenced in pipeline redesign if the frequency needs to be scaled.

Acknowledgment

The authors would like to thank the anonymous reviewers for their constructive and helpful suggestions and comments. This work was supported by the National Natural Science Foundation of China (NSFC) under the Grants No.61701095, No.61534002, No.61761136015. This work was also supported by VR, Sweden.

References

[1] L. Benini and G. D. Micheli, "Networks on chips: A new soc paradigm," *IEEE Computer*, vol. 35, no. 1, pp. 70–78, 2002.
[2] M. Radetzki, C. Feng, X. Zhao, and A. Jantsch, "Methods for fault tolerance in networks-on-chip," *Acm Computing Surveys*, vol. 46, no. 1, pp. 1–38, 2013.
[3] D. Rossi, P. Angelini, and C. Metra, "Configurable error control scheme for noc signal integrity," in *On-Line Testing Symposium, 2007. IOLTS 07. 13th IEEE International*, pp. 43–48, IEEE, 2007.
[4] Q. Yu, B. Zhang, Y. Li, and P. Ampadu, "Error control integration scheme for reliable noc," in *IEEE International Symposium on Circuits and Systems*, pp. 3893–3896, 2010.
[5] L. Wang, S. Ma, C. Li, W. Chen, and Z. Wang, "A high performance reliable noc router," *Integration the Vlsi Journal*, vol. 58, p. 583–592, 2017.
[6] D. Park, C. Nicopoulos, J. Kim, N. Vijaykrishnan, and C. R. Das, "Exploring fault-tolerant network-on-chip architectures," in *International Conference on Dependable Systems and Networks (DSN'06)*, pp. 93–104, June 2006.
[7] S. Shamshiri, A. Ghofrani, and K. T. Cheng, "End-to-end error correction and online diagnosis for on-chip networks," in *IEEE International Test Conference*, 2011.
[8] B. Aghaei, A. Khademzadeh, M. Reshadi, and K. Badie, "A new bist-based test approach with the fault location capability for communication channels in network-on-chip," *Journal of Electronic Testing*, vol. 33, no. 3, pp. 1–13, 2017.
[9] J. Liu, J. Harkin, Y. Li, and L. Maguire, "Low cost fault-tolerant routing algorithm for networks-on-chip," *Microprocessors & Microsystems*, vol. 39, no. 6, pp. 358–372, 2015.
[10] A. Prodromou, A. Panteli, C. Nicopoulos, and Y. Sazeides, "Nocalert:an on-line and real-time fault detection mechanism for network-on-chip architectures," in *Ieee/acm International Symposium on Microarchitecture*, pp. 60–71, 2012.
[11] C. Bienia, *Benchmarking Modern Multiprocessors*. PhD thesis, Princeton University, January 2011.
[12] W. J. Dally and B. P. Towles, *Principles and Practices of Interconnection Network*. Morgan Kaufmann Publishers Inc., 2004.
[13] L. S. Peh and W. J. Dally, "A delay model and speculative architecture for pipelined routers," in *The Seventh International Symposium on High-Performance Computer Architecture*, pp. 255–266, 2001.
[14] M. Galles, *Spider: A High-Speed Network Interconnect*. IEEE Computer Society Press, 1997.
[15] J. Kim, "Low-cost router microarchitecture for on-chip networks," in *Ieee/acm International Symposium on Microarchitecture*, pp. 255–266, 2009.
[16] S. T. Nguyen and S. Oyanagi, "A low cost single-cycle router based on virtual output queuing for on-chip networks," in *Digital System Design: Architectures, Methods and TOOLS*, pp. 60–67, 2010.
[17] P. Poluri and A. Louri, "A soft error tolerant network-on-chip router pipeline for multi-core systems," *IEEE Computer Architecture Letters*, vol. 14, no. 2, pp. 107–110, 2015.
[18] W. Zhang, L. Hou, J. Wang, S. Geng, and W. Wu, "Comparison research between xy and odd-even routing algorithm of a 2-dimension 3x3 mesh topology network-on-chip," in *Wri Global Congress on Intelligent Systems*, pp. 329–333, 2009.
[19] T. Austin, E. Larson, and D. Ernst, "Simplescalar: an infrastructure for computer system modeling," *Computer*, vol. 35, no. 2, pp. 59–67, 2002.
[20] J. Wang, Y. Huang, M. Ebrahimi, L. Huang, Q. Li, A. Jantsch, and G. Li, "Visualnoc: A visualization and evaluation environment for simulation and mapping," in *ACM International Workshop on Many-Core Embedded Systems*, pp. 18–25, 2016.