VNU Journal of Science: Comp. Science & Com. Eng., Vol. 35, No. 1 (2019) 32–45

VNU Journal of Science:
Computer Science and Communication Engineering

Journal homepage: http://www.jcsce.vnu.edu.vn/index.php/jcsce

Original Article

An Adaptive and High Coding Rate Soft Error Correction Method in Network-on-Chips

Khanh N. Dang*, Xuan-Tu Tran

VNU Key Laboratory for Smart Integrated Systems, VNU University of Engineering and Technology, 144 Xuan Thuy, Cau Giay, Hanoi, Vietnam

Received 28 September 2018
Revised 05 March 2019; Accepted 15 March 2019

Abstract: The soft error rates per single-bit due to alpha particles in sub-micron technology is expectedly reduced as the feature size is shrinking. On the other hand, the complexity and density of integrated systems are accelerating which demand efficient soft error protection mechanisms, especially for on-chip communication. Using soft error protection method has to satisfy tight requirements for the area and energy consumption, therefore a low complexity and low redundancy coding method is necessary. In this work, we propose a method to enhance Parity Product Code (PPC) and provide adaptation methods for this code. First, PPC is improved as forward error correcting using transposable retransmissions. Then, to adapt with different error rates, an augmented algorithm for configuring PPC is introduced. The evaluation results show that the proposed mechanism has coding rates similar to Parity check's and outperforms the original PPC.

Keywords: Error Correction Code, Fault-Tolerance, Network-on-Chip.

1. Introduction

Electronics devices in critical applications such as medical, military, aerospace may expose to several sources of soft errors (alpha particles, cosmic rays or neutrons). The most common behavior is to change the logic value of a gate or a memory cell leading to incorrect values/results. Since those critical applications demand high reliability and availability due to the difficulty in maintenance, soft error resilience is widely considered as a must-have feature among them. However, according to [1], the soft error rate (SER) per gates is predictively reduced due to the shrinking of transistor size. Previously, the soft error rates of single-bit are predictively decreased by around 2 times per technology generation [2]. With the realistic analyses in 3D technology [3], the reduction is continue with small transistor sizes, 3D structure and the top layers act as shielding layers. Empirical results of 14*nm* FinFET devices show that the soft error

* Corresponding author.
Email address: khanh.n.dang@vnu.edu.vn

https://doi.org/10.25073/2588-1086/vnucsce.218

FIT (Fault In Time) rate is significantly reduced by 5-10 times from the older technologies. However, due to the increasing of integration density, the number of soft errors per chip is likely to be increased [2]. Moreover, the soft error rates in normal gates are also rising which shift the interests of soft error tolerance from memory-based devices to memory-less devices (wires, logic gates) [1]. As a consequence, the communication part needs an appropriate attention to designing soft error protection to balance the complexity and reliability.

To protect the wire/gate which plays the major role in on-chip communication from soft errors, there are three main approaches as in Fig. 1: *(i)* Information Redundancy; *(ii)* Temporal Redundancy; and *(iii)* Spatial Redundancy. While spatial and temporal redundancies are costly in terms of performance, power and area, using error correction code (ECC) and error detection (ED) is an optimal solution. Also, ECC with further forward error correction (FEC) and backward error correction (BEC) could provide a viable solution with lesser area cost and lower performance degradation. By combining a coding technique with detection feature and retransmission as BEC, the system can correct more faults. On the other hand, FEC, which temporally ignores the faults then corrects them at the final receiver, is another viable solution. Indeed, ECC plays a key role in the two mentioned solutions.

Among existing ECCs and EDs, the Parity check is one of the very first methods to detect a single flipped bit. It also provides the highest coding rate and the lowest power consumption. On the other hand, Hamming code (HM) [4] and its extension (Single Error Correction Double Error Detection: SECDED) [5] are two common techniques. This is due to the fact that those two ECCs only rely on basic boolean functions to encode and decode. Thanks to their low complexity, they are suitable for on-chip communication applications and memories [6]. On the other hand, Cyclic Redundancy Check (CRC) code is also another solution to detect faults [7]. Since it does not support fault correction, it may not optimal for on-chip communication. Further coding methods such as Bose-Chaudhuri-Hocquenghem and Reed-Solomon are exceptionally strong in terms of correctability and detectability; however, their overwhelming complexities prevent them from being widely applied in on-chip communication [7]. Product codes [8, 9], as the overlap of two or more coding techniques could also provide a much resilient and flexibility.

As previously mentioned, wires/logic gates have lower soft error rates than memories. In addition, *Magen et al.* [10] also reveals the interconnect consumes more than 50% the dynamic power. Since Network-on-Chips utilizes multiple hopes and FIFO-based design, the area cost and static power are also problematic. Therefore, we observe that using a high coding rate[1] ECC could help solve the problem. Moreover, the low complexity methods can be widely applied within a high complexity system. The soft errors on computing modules and memories are out of scope of this paper.

In this paper, we present an architecture using Parity Product Code (PPC) to detect and correct soft errors in on-chip communication. Here, we combine with both BEC and FEC to enhance the coding rate and latency. A part of this work has been published in [11]. In this work, we provide an analytical analysis for the adaptive method and provide an augmented algorithm for managing. The contributions are:

- A selective ARQs in row/column for PPC using a transposable FIFO design.

[1]Coding rate: ratio of useful bits per total bits.

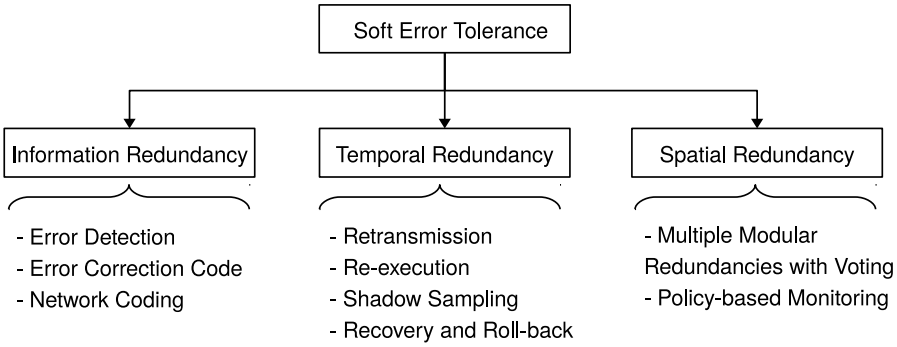

Fig. 1. Soft error tolerance approaches.

- A method to adaptively issue the parity flit.
- A method to perform go-back retransmission under low error rates.
- An adaptive mechanism for the PPC-based system with various error rates.

The organization of this paper is as follows: Section 2 reviews the existing literature on coding techniques and fault-tolerances. Section 3 presents the PPC and Section 4 shows the proposed architecture. Section 5 provides evaluations and Section 6 concludes the paper.

2. Related works

As we previously mentioned, the soft error tolerance is classified into three branches: (i) Information Redundancy, (ii) Temporal Redundancy, and (iii) Spatial Redundancy. In this work, we focus on the on-chip communication; therefore, this section focuses on the methods which tolerate soft errors in this type of medium.

For information redundancy, error correction code is the most common method. Error correcting code has been developed and widely applied in the recent decades. Among the existing coding technique, Hamming code [4], which is able to detect and correct one fault, is one of the most common ones. Its variation with one extra bit - Single Error Correction Double Error Detection (SECDED) by *Hisao* [5] is also common with the ability to correct and detect one and two faults, respectively. Thanks to their simplicity, ECC memories usually use Hamming-based coding technique [12]. Error detection only codes such as cyclic redundancy check (CRC) [13] is also widely used in digital network and storage applications. More complicated coding techniques such as Reed-Solomon [14], BCH [15] or Product-Code [8] could be alternative ECCs. Further correction of ECC could be forward (correct at the final terminal) or backward (demand repair from the transmitter) error correction. Despite its efficiency, ECC is limited by its maximum number of fault could be detected and corrected.

When ECC cannot correct but can detect the occurrence of faults, temporal redundancy can be useful. Here, we present four basic methods: *(i)* retransmission, *(ii)* re-execution, *(iii)* shadow sampling, and *(iv)* recovery and roll-back. Both retransmission [16] and re-execution [17, 18] share the same idea of repeating the faulty actions (transmission or execution) in order to obtain non-faulty actions. Due to the randomness of soft errors, this type of errors is likely to absent after

a short period. With the similar idea, shadow sampling (i.e. Razor Flip-Flop [19]) uses a delay (shadow) clock to sample data into an additional register. By comparing the original data and the shadow data, the system can detect the possible faults. Although temporal redundancy can be efficient with its simple mechanism, it can create congestion due to multiple times of execution/transmission.

Since temporal redundancy may cause bottle-necks inside the system, using spatial redundancy can be a solution [17, 20]. One of the most basic approaches is multiple modular redundancies. By having two replicas, the system can detect soft errors. Moreover, using an odd number of replicas and a voting circuit, the system can correct soft errors. Since spatial redundancy is costly in terms of area, applying them to soft error protection is problematic.

3. Parity product code

This section presents Parity Product Code (PPC) which is based on Parity check and Product code [8, 9]. While Parity check has the lowest complexity and highest coding rate among existing ECC/EDC, product code provide more flexibility for correction.

3.1. Encoding of PPC

Let's assume a packet has M-flits and one parity flit as follows:

$$P = \begin{bmatrix} F_0 \\ F_1 \\ \dots \\ F_{M-1} \\ F_P \end{bmatrix} = \begin{bmatrix} b_0^0 & b_1^0 & b_2^0 & \dots & p^0 \\ b_0^1 & b_1^1 & b_2^1 & \dots & p^1 \\ b_0^2 & b_1^2 & b_2^2 & \dots & p^2 \\ \dots & \dots & \dots & \dots & \dots \\ pb_0 & pb_1 & pb_2 & \dots & pp^i \end{bmatrix}$$

where, a flit F has N data bits and one single parity bit:

$$F_i = \begin{bmatrix} b_0^i & b_1^i & b_2^i & \dots & b_{N-1}^i & p^i \end{bmatrix}$$

Followings are the calculations for parity data:

$$p^i = b_0^i \oplus b_1^i \oplus \dots \oplus b_{N-1}^i \quad (1)$$

and

$$F_P = F_0 \oplus F_1 \oplus \dots F_{M-1}$$

Because the decoding latency is O(M), we can use a trunk of M flits instead.

3.2. Decoding of PPC

The decoding for PPC could be handled in two phases: (i) Phase 1: Parity check for flits with backward error correction; and (ii) Phase 2: forward error correction for packets. For each receiving flit, parity check is used to decide whether a single event upset (SEU) occurs:

$$C_F = b_0 \oplus b_1 \oplus \dots \oplus b_{N-1} \oplus p \quad (2)$$

If there is a SEU, C_F will be '1'. To quickly correct the flit, Hybrid Automatic Retransmission Request (HARQ) could be used for demanding a retransmission. Because HARQ may cause congestions in the transmission, we correct using the PPC correction method at the RX (act as FEC). In our previous work [11], we use the Razor-Flip Flop with Parity. However, the area and power overhead of this method are costly. Therefore, using pure FEC is desired in this method. The algorithm of decoding process is shown in Algorithm 1.

If the fault cannot be corrected, the system correct it at the receiving terminals. Parity check of the whole packet is defined as:

$$C_P = F_0 \oplus F_1 \oplus \dots \oplus F_{M-1} \oplus F_P \quad (3)$$

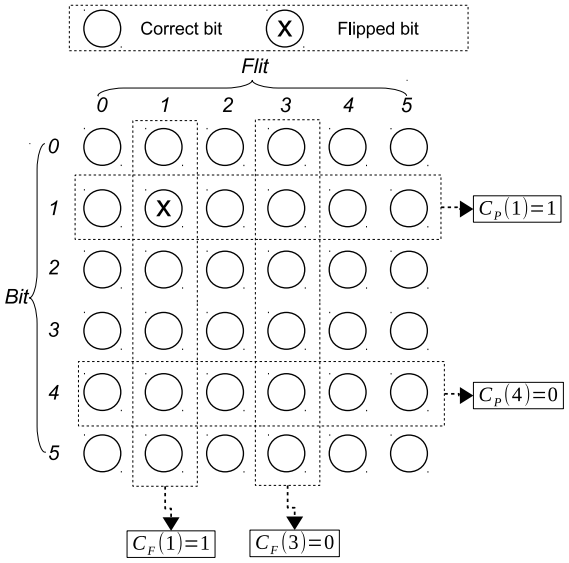

Fig. 2. Single flipped bit and its detection pattern.

Base on the values of C_F and C_P, the decoder can find out the index of the fault as in Fig. 2. The flit-parity and the index parity check of the flipped bit have the $C_F = C_P = 1$. Therefore, the decoder can correct the bit by flipping it during the reading process. Note that the FIFO has to be deep enough for M flits ($M \leq$ FIFO's depth). Apparently, PPC can detect and correct only a single flipped bit in M flits.

4. Proposed architecture and algorithm

4.1. Fault assumption

In this work, we mainly target to low error rates where there is one flipped bit in a packet (or group of flits). According to [21], the expected soft error rate (SER) for SRAM is below 10^3 FIT/Mbit (10^{-3} FIT/bit) for planar, FDSOI and FinFET[2]. Furthermore, SER could reach 6E6 FIT/Mbit in the worst case (14-nm bulk, 10-15km of attitude). Since the FIT is calculated for 10^9 hours, we can observe the realistic error rate per clock cycle is low.

[2]FIT: Failures In Time is the number of failures that can be expected in one billion (10^9) device-hours of operation.

Algorithm 1: Decoding Algorithm.

```
// Input code word flits
Input: F_i = {b^i_0, ... b^i_{N-1}, p}
// Output code word flits
Output: oF_i
// Output packet/group of flits
Output: oF_i
// Output ARQ
Output: ARQ
// Calculate the parity check
1  C_F = b^i_0 ⊕ ··· ⊕ b^i_{N-1} ⊕ p
2  SEU'_F = b'^i_0 ⊕ ··· ⊕ b'^i_{N-1} ⊕ p'
   // Correct SEUs by using RFF-w-P
3  if (C_F == 0) then
       // The original code word is correct
4      oF_i = F_i
5  else
6      if (ARQ == True) then
           // Using ARQ
7      else
           // Using FEC
8          oF_i = F_i;
9          oC_F = 1;
10 if (RX = True) then
       // Forward Error Correction Code using
          PPC
11     call FEC();
12 else
13     return oF_i;
```

Figure 3 shows the evaluation of different bit error rate with the theoritical model and Monte-Carlo simulation (10,000 cases). This evaluation is based on Eq. 4 where ϵ is the bit error rate, $P_{i,n}$ is the probability of having i faults in n bits. Note that we only calculate for zero and one fault since the two-bit error rates are extremely low. Even having two-bit error, our technique still can detect and correct thank to the transposable selective ARQ.

$$P_{i,n} = \binom{n}{i} * \epsilon^i * (1-\epsilon)^{n-i} \quad (4)$$

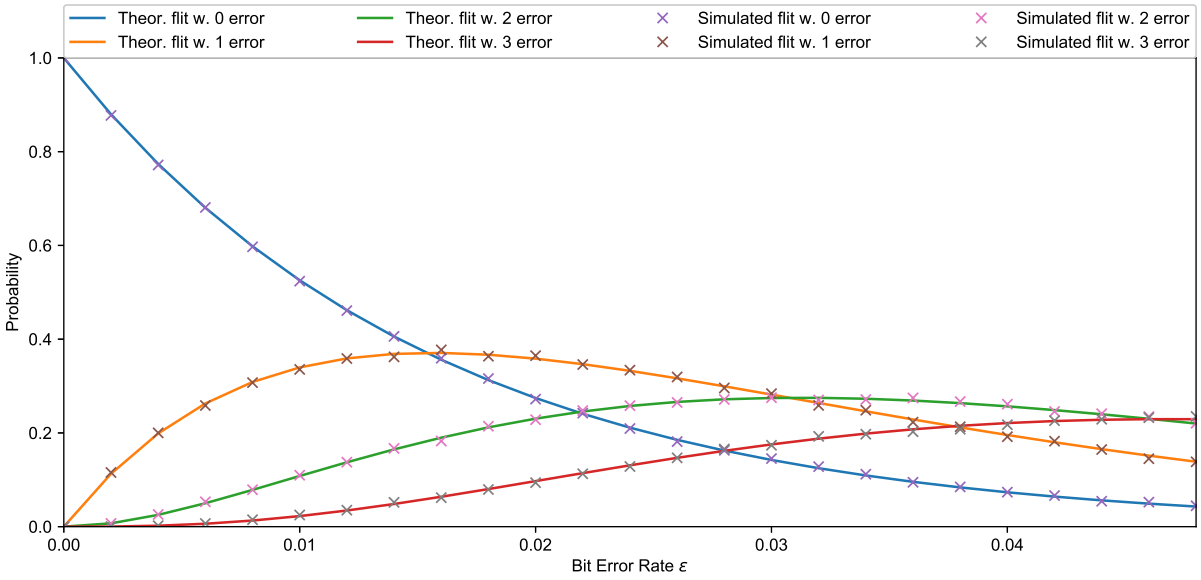

Fig. 3. Flit and packet error rate: theoretical model and Monte-Carlo simulation results. Flit size: 64-bit, packet size: 4-flits.

In summary, we analyze that BER in on-chip communication is low enough that the ECC methods such as SECDED or Hamming is overwhelmed. Providing an optimized coding mechanism could help reducing the area and power overhead. Understanding the potential high error rate is also necessary.

4.2. Transposable selective ARQ

4.2.1. Problem definition

If there are two flipped bits inside the same flit, the parity check fails to detect. On the other hand, detected faulty flits may not be corrected by using HARQ due to the fact that the flit is already corrupted at the sender's FIFO. Here, we classify errors into two types: HARQ correctable errors and HARQ uncorrectable errors. In both cases, the system relies on the correctability of PPC at the receiving terminal.

4.2.2. Proposed method

As a FEC, PPC can calculate parity check of each bit-index as in C_P. Therefore, we can further detect it by Eq. 3. If a flit has an odd number of flipped bits, a selective ARQ can help fix the data. On the other hand, if a flit has an even number of flipped bits, the C_F stays at zeros. Therefore, the decoder cannot determine the corrupted flits. However, C_P could indicate the failed indexes. Note that PPC is unable to detect the square positional faults (i.e.: faults with indexes (a,b), (c,b), (a,d) and (c,d)).

To correct these cases, the system use three stages: *(i)* Row (bit-index) Selective ARQ, *(ii)* Column (flit-index) Selective ARQ and *(iii)* Go-back-N (N: number of flits) ARQ. A go-back-N ARQ demands a replica of the whole trunk of flits (or packet) while the selective one only requests the corrupted one.

The column ARQ is a conventional method where the failed flit index is sent to TX. For the row ARQ, the bit index is sent instead. For instance if b_1^2 and b_2^2 are flipped leading to undetected SEU in F_2. By calculating the C_P, the receiver finds out that bit-index 1 and bit-index

2 have flipped bits; therefore, we can use the H-ARQs to retransmit these flits:

$$F_{ARQ_1} = \begin{bmatrix} b_1^0 \\ b_1^1 \\ \dots \\ pb_1 \end{bmatrix}$$

and

$$F_{ARQ_2} = \begin{bmatrix} b_2^0 \\ b_2^1 \\ \dots \\ pb_2 \end{bmatrix}$$

In this work, we assume that the maximum flipped bits in a flit is two. Therefore, the decoder aims to mainly use row ARQs because it cannot find out which flit has two flipped bits. The FEC and Selective ARQ algorithm is illustrated in Algorithm 2.

Algorithm 2: Forward Error Correction and Selective ARQ Algorithm.

```
// Input code word flits
Input: F_i = {b_0^i, ... b_{N-1}^i, p}
// Output code word flits
Output: oF_i
// Output ARQ
Output: ARQ
if i == 0 then
    C_P = F_i;
    regC_F = C_F
else if i < M − 1 then
    C_P = C_P ⊕ F_i;
    regC_F = {regC_F, C_F};
else
    if no or single SEU then
        P = Mask (F_i, C_P, regC_F);
        return P;
    else
        ARQ = C_P;
        // receive new flits (i ≥ N) and
           write in row indexes
        F_{i=0,...,N-1} = write_row (C_P, F_{(i≥N)})
```

4.3. Adaptive algorithm

4.3.1. Problem definition

If the error rate is low enough to cause single flipped bit in a packet, using parity flit could cost considerable power and reduce the coding rate. Therefore, we try to optimize this type of cases.

4.3.2. Adaptive F_P

PPC can perform adaptive parity flit (F_P) issuing. In this case, the receiver will check the parity of each flit as usual using Parity check. If the parity check fails, it first tries to correct using HARQ. If both techniques cannot correct the fault, receiver will send to TX a signal to request the parity flit. The parity flit is issued for each M flits as usual. If there is no fail in the parity check process, the parity flit could be removed from the transmission.

The adaptive F_P could increase the coding rate by removing the F_P; however, the major drawback is that it cannot detect two errors in the same flit.

4.3.3. Overflowing packet check

Moreover, we can extend further with a go-back retransmission instead of transposable ARQ. Assuming the maximum number of cached flits is K. Since F_P can be responsible $M > K$ flits, the correction provide by PPC is impossible and the system needs a go-back M flits retransmission. By adjusting the M value, the system can switch between go-back M-flits and PPC correction. This could be applied for low error rate cases to enhance the coding rate. The Overflowing Packet Check (OPC) could adjust the M value based on the error rate.

4.3.4. Augmented algorithm

Apparently, the original PPC, adaptive F_P and OPC are suitable for a specified error rate. To help the on-chip communication system adapt with different rates, we proposed a lightweight mechanism to monitor and adjust the proposal. We define three dedicated modes:

Algorithm 3: Augmented Algorithm for PPC.

```
   // Input: result of decoding
   Input: C_F, C_P
   // Output: modes
   Output: Mode
   // Output: M
   Output: M
1  switch Mode do
2      case Mode-1 do
3          if ∑C_P == 0 and ∑C_F == 0 then
4              M=M*2;
5          else
6              M=M/2;
7              if M == K then
8                  Mode = Mode-2;
9      case Mode-2 do
10         if ∑C_P == 0 and ∑C_F == 0 then
11             Mode = Mode-1;
12         else if ∑C_P >= 2 or ∑C_F >= 2 then
13             Mode = Mode-3;
14     case Mode-3 do
15         if ∑C_P <= 1 and ∑C_F <= 1 then
16             Mode = Mode-2;
17         else
               // Need to inform the system
```

- Mode-1: Adaptive F_P with OPC. The F_P is issued adaptively; however, after M flits, an F_P is issued to ensure the correctness of M flits.
- Mode-2: PPC standalone. Constant check the flits and packets using PPC.
- Mode-3: High error rates. The PPC decoder recognizes there are more than two faults in a packet then informs the system the high error rates situation.

Algorithm 3 shows the augmented algorithm for PPC. For each mode, the system adjusts the coding mechanism based on the output of the decoder. If there is no error detected ($C_P == 0$ and $C_F == 0$), it could switch to a higher coding rate method. Also, inside the Mode-1, the system adjusts the M value to enhance the coding rate. If there are multiple errors, the system needs to enhance the coding mechanism (i.e. reduce M value or use the original PPC). Here, we assume that both terminals have a synchronize mechanism that allows them to adjust the coding mechanism on both sides.

4.4. Proposed architecture

4.4.1. Encoding and decoding scheme

Figure 4 shows the architecture for the PPC encoding and decoding scheme. In the encoder's side, the FIFO receives data until being full. Then, the encoder transmits data through the channel with a parity bit (p) which is obtained from the 'FLIT PAR' module. On the other hand, each flit is also brought into a packet parity encoder (PACK. PAR) to obtain parity flit (F_P). This parity check flit is transmitted at the end of the packet.

At each hop of the communication, the parity check of each flit is performed. If there is a flipped bit, this module can correct using a shadow clock or ARQ.

When the flit arrives the decoder, it is checked and corrected by HARQ first. Once the flit is done, it is pushed into the FIFO and the 'PACK. PAR' module. After completing the parity value of the packet, it was sent to the controller to handle the masking process. The masking process can correct a single flipped bit; therefore, selective ARQ is used once there are 2+ faults are detected. As we previously assumed, when there are two faults in a flit, the C_P value can indicate the faulty indexes. This value will be sent back to the encoder to retransmit those indexes.

4.4.2. Transposable FIFO

To support reading and writing in both column and row (as row/column ARQ), we use a transposable FIFO (T-FIFO) architecture. Besides the normal jobs of a FIFO, it also allows randomly reading and writing by a column or row

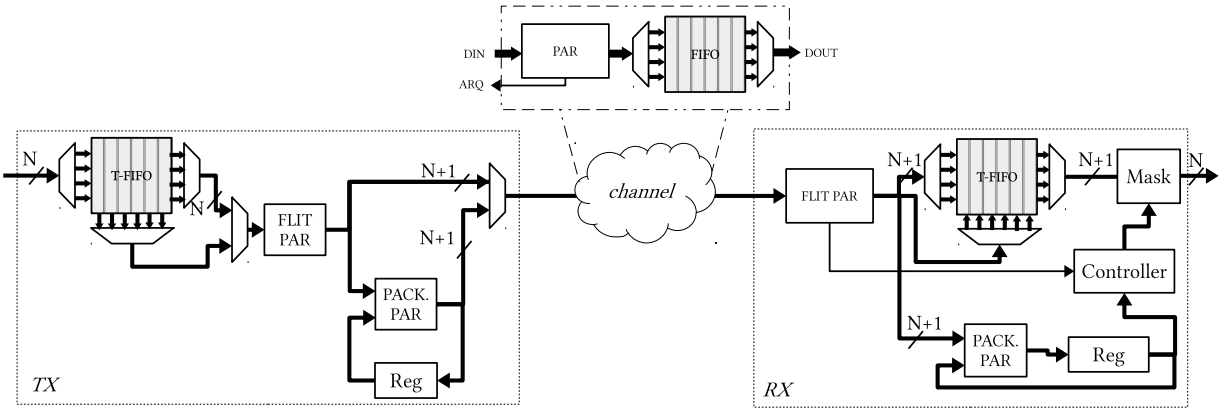

Fig. 4. PPC scheme: Parity Product Code for soft error correction.

address (which means transposable FIFO). For a bigger size, RAM-based FIFO may be utilized. A transposable SRAM [22] could be usedwith 8 transistors instead of 6 as in the traditional ones. In this work, we use a DFF-based T-FIFO.

5. Evaluation

5.1. Methodology

The architecture has been designed in Verilog HDL and synthesized using NANGATE 45 *nm* library. The design is then implemented using EDA tools provided by Synopsys. Because of the fault assumption (two faults per a group of flits), we compare the architecture to Parity check, Hamming and SECDED which are the common soft error correction methods, especially for low error rates.

5.2. Coding performance

In this section, we perform evaluation of coding rates for PPC and existing high coding rate methods (Parity, Hamming, SECDED). For a fair comparison, we only consider the coding rate at the maximum detecting and correcting capability of the methods.

5.2.1. Parity product code

Figure 5 shows the coding rate of PPC without any enhancement. The coding rate of PPC is obtained as $[NM]/[(N+1)(M+1)]$. As we can observe in this figure, PPCs with $M > 10$ has a better coding rate than both of HM and SECDED. For larger numbers of data bit-width (60+), HM and SECDED have better coding rates due to the fact that the parity check flit F_P heavily affects the overall rate. Also, smaller M values also degrade the coding rate significantly. On the other hand, Parity code outperforms the others due to the fact that it only needs one extra bit. The major drawback of Parity is lack of correctability.

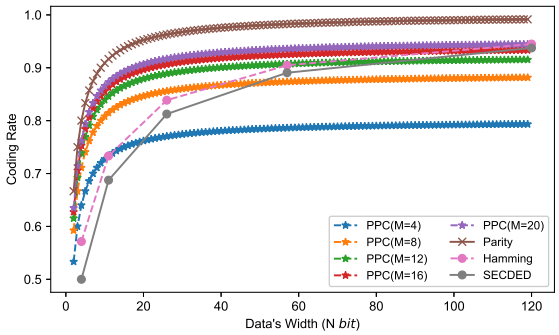

Fig. 5. Coding rates of PPC.

5.2.2. Adaptive F_P

We first evaluate the efficiency of using adaptive F_P. The results are shown in Fig. 6. The packet size is set to 4 flits and the data's width is varied from 2 to 120.

Fig. 6(a) shows the case of BER=10^{-3}, in which PPC's coding rate is reduced rapidly when increasing the data's width to be lower than both Hamming and SECDED. However, if the data's width is lower than 64-bit, PPC still outperforms both of them. Furthermore, PAR+ARQ has lower coding rate than ARQ (no-fault). Fig. 6(b) shows the case of BER=10^{-4}. In this case, PPC easily dominates both Hamming and SECDED and has a similar performance as Parity check. In comparison with the original PPC, the adaptive F_P provides an exceptional better performance, especially with no or low error rate. Please note that even we consider 10^{-4} as a low error rate, this rate is still higher than the BER we discussed in Section 4.1 where the worst case is around 6×10^6 FIT/Mbit ($\simeq$6 FIT/bit: 6 errors/bit/10^9 hours).

If the BER is reduced further to 10^{-5}, the coding rate of adaptive F_P is mostly identical to parity check.

5.2.3. Overflowing packet check

In this section, we evaluate how efficient the overflowing packet (OPC) check could be. For this evaluation, we set the buffer size is 4 while the numbers of flits for parity in the overflowing packet check are 8, 16, 32, and 64.

With high error rates (10^{-3} and 10^{-4}), we can observe the drop of coding rate in long packets. This is because the required retransmissions are occasionally needed. If the error rate drops to 10^{-5}, the coding rate is significantly better. With M=64, the coding rate is slightly lower than the Parity check which means that it is still lower than the adaptive F_P.

5.2.4. Summary

Figure 8 compares the proposed techniques. In summary, adaptive F_P offers the best coding rate among the proposed techniques. However, this method has one drawback, it can only detect and correct one flipped bit in the whole packet. The OPC version has lower coding rate, but it can detect and correct more. In order to understand the overall reliability, we investigate the reliability of these methods in the next section.

5.3. Reliability

Although coding rate could be a good measurement of the efficiency of the existing coding methods and the proposal, the reliability is also an important parameter. Reliability is defined as the probability of working without any failure. In this section, we consider soft errors are independent. Therefore, the probability of having i errors in n bits is as Eq. 4. In this case, the time to failure is calculated based on the occurrence of having i errors which are over the detection/correction threshold of the system. For each system, we assume the maximum error could be handled is e. Therefore, the reliability function R could be calculated as:

$$R = P(i \leq e) \tag{5}$$

$$= \sum_{i=0}^{e} \left(\binom{n}{i} \times \epsilon^i \times (1-\epsilon)^{n-i} \right) \tag{6}$$

Figure 9 shows the reliability results of those methods. We first consider HARQ correctable errors then the HARQ uncorrectable errors. With HARQ correctable errors, the OPC and PPC benefit from the ability to correct using HARQ. The adaptive F_P is unable to use this which leads to degradation in terms of reliability.

Without considering the HARQ correctable errors, we can observe the drop of OPC version which becomes lower than the original PPC. However, it is still higher than adaptive F_P.

Figure 10 shows high error rate cases.

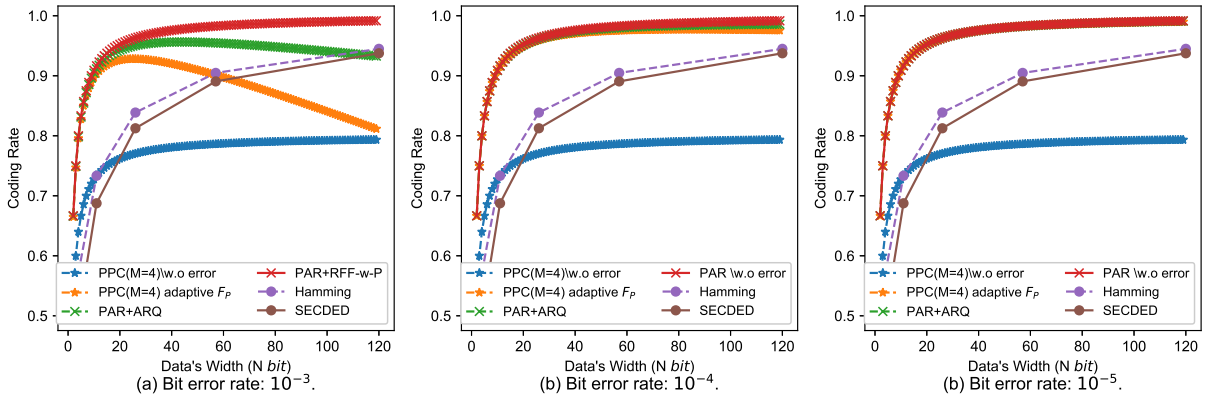

Fig. 6. Coding rates evaluation of adaptive F_P with three BERs.

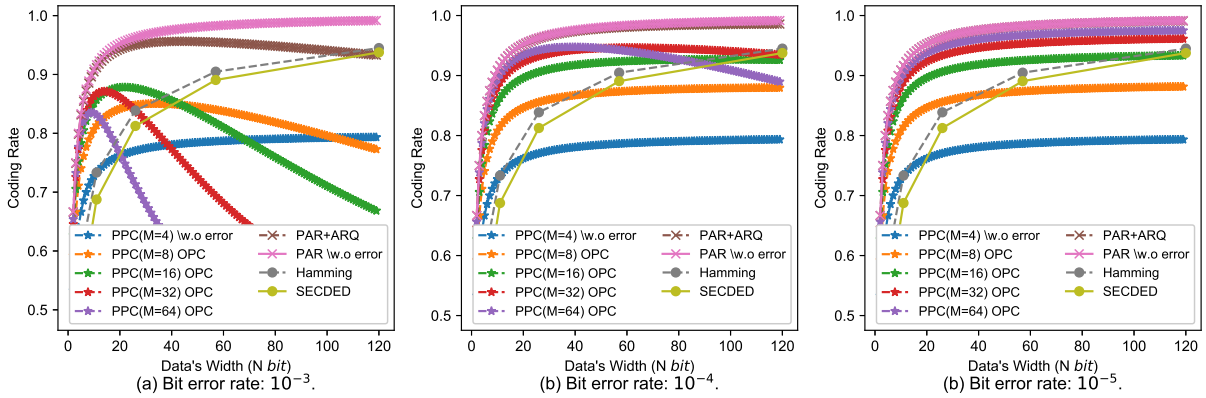

Fig. 7. Coding rates evaluation of Overflowing Packet Check with three BERs.

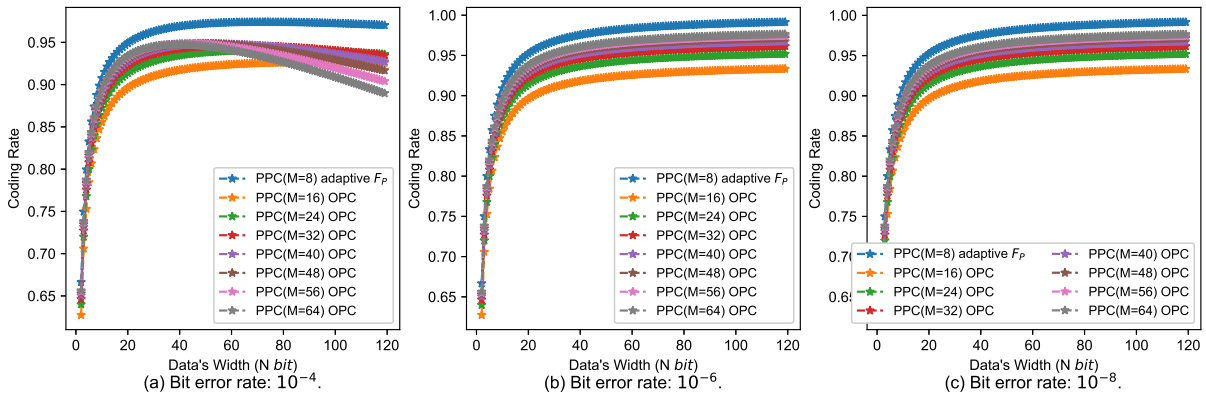

Fig. 8. Comparison of coding rate between the proposed techniques.

Table 1. Hardware complexity results of the proposal with 32-bitwidth

Design	**Module**	**Sub-module**	**Area** (μm^2)	(%)	**Power** (μW)	(%)	**Max Freq.** (MHz)
FIFO	Parity-based	33-bit, 4-slots	1111.8800		562.4571		-
	Hamming-based	39-bit, 4-slots	1300.2080		664.5005		-
	SECDED-based	40-bit, 4-slots	1331.3300		683.0590		-
Hamming		Encoder	94.1640		68.8886		2,570.69
		Decoder	234.8780		206.0509		1,369.86
SECDED		Encoder	111.7200		82.3979		2,564.10
		Decoder	253.7640		206.3793		1,250.00
PARITY		Encoder	49.4760		49.3382		2,666.67
		Decoder	51.0720		52.4233		2,380.95
Ours		Total	1856.4140	(100)	823.2660	(100)	1,273.88
	TX	Controller	344.4700	(18.6)	167.7900	(20.4)	-
		Encoder	366.8140	(19.8)	123.1230	(15)	-
		T-FIFO (32-bit)	1098.8460	(59.2)	529.6990	(64.3)	-
		Total	2107.2520	(100)	958.899	(100)	1,270.64
		Controller	495.2920	(23.5)	196.242	(20.5)	-
	RX	Mask	78.2040	(3.7)	4.066	(0.4)	-
		Decoder	303.7720	(14.4)	189.536	(19.8)	
		T-FIFO (33-bit)	1142.2040	(54.2)	518.098	(54.0)	-

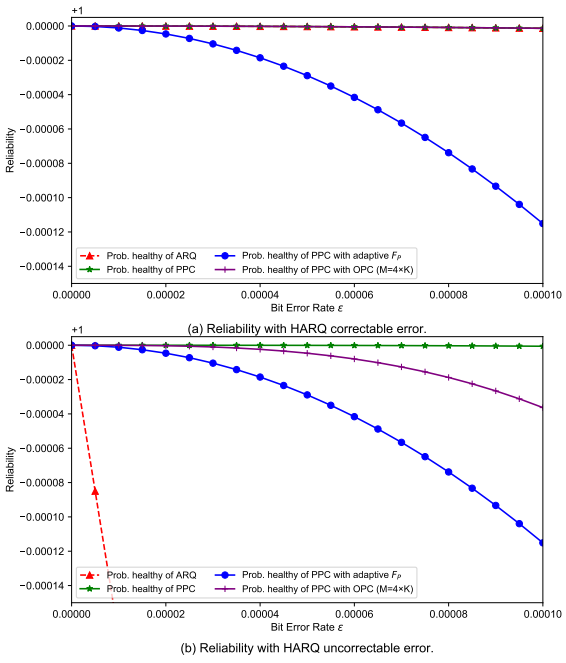

Fig. 9. Reliability comparison under low BER.

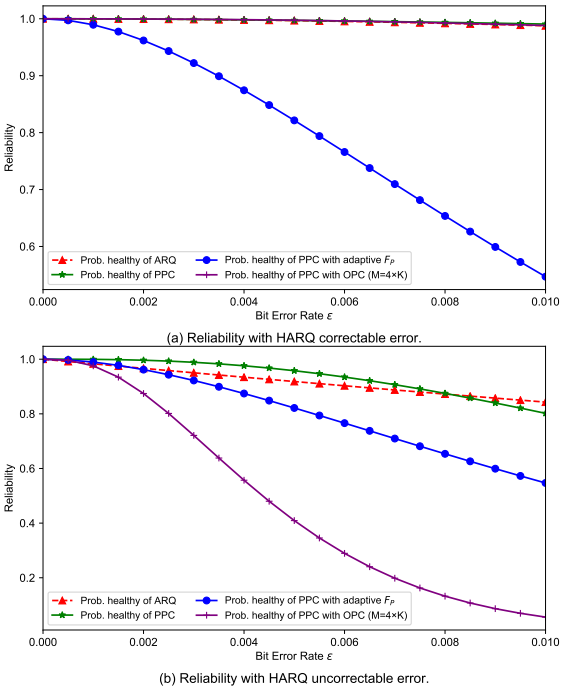

Fig. 10. Reliability comparison under high BER.

Apparently, the reliability is significantly dropped because the evaluated mechanisms are not supposed to correct multiple faults. We can observe the drop of OPC with HARQ uncorrectable error and the resilience of Parity. However, all of the evaluated methods offer low reliability in HARQ uncorrectable error case which makes them unusable.

5.4. Implementation results

In order to understand the hardware complexity of the proposed model and the other coding techniques, we implemented them with 32 data bit-width. Table 1 presents in details the hardware cost of PPC's parity modules and the sub-modules. We use NANGATE 45 *nm* library [23] and set the frequency as $500MHz$ for the power estimation.

As we can observe, the area cost of the PPC's encoder and decoder is less than 20% of the TX and RX in both area and power consumption. Most of the cost belong to the memory-based module. For instance, FIFO occupies 59.2% and 54.2% of TX's and RX's area cost. Although the complexity of encoder and decoder seems higher than other, the area cost of the FIFO can be reduced thanks to the smaller bit-width.

In comparison to three common coding techniques (Parity, Hamming and SECDED), the encoder and decoder both cost more area and power. This is because the codec requires register for calculating the C_P. The area cost and power consumption of PPC's FIFO are smaller than Hamming's and SECDED's due to the smaller bit-width. However, this overhead only impacts the multi-hop architectures such as Network-on-Chips.

Design of T-FIFO also has smaller area overhead in comparison to normal FIFOs. With 33 data bit-width, T-FIFO increases the area and power by 2.7% and 6.8%, respectively. This area is total reasonable as it provides the ability to read and write in both column and row.

6. Conclusion

In this paper, we present PPC as an error correction code and its hardware extensions for detecting and correcting soft errors in on-chip communications. A transposable FIFO is also presented to help the system retransmission by both row and column indexes. Adaptive mechanism for low error rate can help increase the coding rate of the system. Although they significantly increase the area cost, they provide one bit detect and protection with only necessity to check the parity of the data. The coding rate is also promising with better than Hamming and SECDED with large packet sizes. Also, PPC can detect more faults to inform the system.

In the future, we design of PPC with Parity could be implemented in a specific Network-on-Chip to investigate the impacts on the performance.

Acknowledgments

This work has been supported by VNU University of Engineering and Technology under Project No. CN18.10.

The preliminary part of this work was published in [11]. The authors thank colleagues for their helpful discussions and proofreading. Khanh N. Dang would like give a special thank to VNU journal team for encouraging to improve his skills in editing their style of LaTeXto match their new template.

References

[1] R. Baumann, Radiation-induced soft errors in advanced semiconductor technologies, IEEE Transactions on Device and materials reliability 5 (3) (2005) 305–316. https://doi.org/10.1109/tdmr.2005.853449.

[2] N. Seifert, B. Gill, K. Foley, P. Relangi, Multi-cell upset probabilities of 45nm high-k + metal gate SRAM devices in terrestrial and space environments, in: IEEE International Reliability Physics Symposium 2008, IEEE, AZ, USA, 2008, pp. 181–186.

[3] S. Lee, I. Kim, S. Ha, C.-s. Yu, J. Noh, S. Pae, J. Park, Radiation-induced soft error rate analyses for 14 nm FinFET SRAM devices, in: 2015 IEEE International Reliability Physics Symposium (IRPS), IEEE, CA, USA, 2015, pp. 4B–1.

[4] R. Hamming, Error detecting and error correcting codes, Bell Labs Tech. J. 29 (2) (1950) 147–160. https://www.doi.org/10.1002/j.1538-7305.1950.tb00463.x.

[5] M.-Y. Hsiao, A class of optimal minimum odd-weight-column SEC-DED codes, IBM J. Res. Dev. 14 (4) (1970) 395–401. https://www.doi.org/10.1147/rd.144.0395.

[6] S. Mittal, M. Inukonda, A survey of techniques for improving error-resilience of dram, Journal of Systems Architecture 91 (1) (2018) 11–40. https://www.doi.org/10.1016/j.sysarc.2018.09.004.

[7] D. Bertozzi, et al., Error control schemes for on-chip communication links: the energy-reliability tradeoff, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 24 (6) (2005) 818–831. https://doi.org/10.1109/tcad.2005.847907.

[8] F. Chiaraluce, R. Garello, Extended Hamming product codes analytical performance evaluation for low error rate applications, IEEE Transactions on Wireless Communications 3 (6) (2004) 2353–2361. https://doi.org/10.1109/twc.2004.837405.

[9] R. Pyndiah, Near-optimum decoding of product codes: Block turbo codes, IEEE Transactions on Communications 46 (8) (1998) 1003–1010. https://www.doi.org/10.1109/26.705396.

[10] N. Magen, A. Kolodny, U. Weiser, N. Shamir, Interconnect-power dissipation in a microprocessor, in: Proceedings of the 2004 international workshop on System level interconnect prediction, ACM, Paris, France, 2004, pp. 7–13.

[11] K. Dang, X.-T. Tran, Parity-based ECC and Mechanism for Detecting and Correcting Soft Errors in On-Chip Communication, in: Proceeding of 2018 IEEE 11th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC-2018), IEEE, Hanoi, Vietnam, 2018, pp. 1–6.

[12] L.-J. Saiz-Adalid, et al., MCU tolerance in SRAMs through low-redundancy triple adjacent error correction, IEEE Transactions on VLSI Systems 23 (10) (2015) 2332–2336. https://www.doi.org/10.1109/tvlsi.2014.2357476.

[13] W. Peterson, D. Brown, Cyclic codes for error detection, Proceedings of the IRE 49 (1) (1961) 228–235. https://www.doi.org/10.1109/jrproc.1961.287814.

[14] S. Wicker, V. Bhargava, Reed-Solomon Codes and Their Applications, first ed., John Wiley & Sons, NJ, USA, 1999.

[15] I. Reed, X. Chen, Error-control coding for data networks, first ed., Springer Science & Business Media, New York, 2012.

[16] L. Peterson, B. Davie, Computer networks: a systems approach, fifth ed., Elsevier, New York, 2011.

[17] K. Dang, et al., Soft-error resilient 3D Network-on-Chip router, in: 2015 IEEE 7th International Conference on Awareness Science and Technology (iCAST), China, 2015, pp. 84–90.

[18] K. Dang, et al., A low-overhead soft–hard fault-tolerant architecture, design and management scheme for reliable high-performance many-core 3D-NoC systems, The Journal of Supercomputing 73 (6) (2017) 2705–2729. https://www.doi.org/10.1007/s11227-016-1951-0.

[19] D. Ernst, et al., Razor: A low-power pipeline based on circuit-level timing speculation, in: Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture, Vol. 24, IEEE, CA, USA, 2003, pp. 10–20.

[20] H. Mohammed, W. Flayyih, F. Rokhani, Tolerating permanent faults in the input port of the network on chip router, Journal of Low Power Electronics and Applications 9 (1) (2019) 1–11. https://www.doi.org/10.3390/jlpea9010011.

[21] G. Hubert, L. Artola, D. Regis, Impact of scaling on the soft error sensitivity of bulk, FDSOI and FinFET technologies due to atmospheric radiation, Integration, the VLSI journal 50 (2015) 39–47. https://www.doi.org/10.1016/j.vlsi.2015.01.003.

[22] J.-s. Seo, et al., A 45nm cmos neuromorphic chip with a scalable architecture for learning in networks of spiking neurons, in: 2011 IEEE Custom Integrated Circuits Conference (CICC), IEEE, CA, USA, 2011, pp. 1–4.

[23] NanGate Inc., Nangate Open Cell Library 45 nm. http://www.nangate.com, (accessed 16.06.16) (2016).