Parity-based ECC and Mechanism for Detecting and Correcting Soft Errors in On-Chip Communication

Khanh N. Dang and Xuan-Tu Tran,
SISLAB, University of Engineering and Technology
Vietnam National University Hanoi (VNU), Hanoi, 123106, Vietnam
Email: {khanh.n.dang,tutx}@vnu.edu.vn

***Abstract*—Soft errors are expecting to be accelerated with the shrinking of feature sizes due to low operating voltages and high circuit density. However, soft error rates per single-bit is expectedly reduced with technology scaling. With tight requirements for the area and energy consumption, using a low complexity and high coding rate error correction code (ECC) to handle soft errors in on-chip communication is necessary. In this work, we use Parity Product Code (PPC) and propose several supporting mechanisms to detect and correct soft errors. First, PPC can work as a parity check to detect single event upset (SEU) inside each flit. Then, to reduce the needed retransmission, a Razor flip-flop with parity check (RFF-w-P) is proposed to work with PPC. Since PPC can act like forward error correction (FEC), we also present a selective transmission in bit-indexes by using a transposable FIFO. Therefore, the proposed mechanism not only guarantee single error detection/correction but also provide 2+ error correction as FEC. The proposed work also reduce the area cost of FIFO in comparison to traditional coding methods and adapt too multiple error rates.**

I. Introduction

Electronics devices in critical applications such as medical, military, aerospace may expose to several sources of soft error (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 value/result. Unfortunately, those critical applications demand high reliability and availability due to the difficulty in maintenance. To ensure the overall reliability and availability of the system, soft error resilience is widely considered a must-have feature among those applications.

According to [1], the soft error rate per gates is predictively reduced due to the shrinking of transistor size. The soft error rates of single-bit are decreased by $\sim$2x per technology generation [2]. However, due to the reducing of operating voltages, which create vulnerabilities to supply voltage noises, and the increasing of integration density, the number soft errors per chip is likely to be increased [2]. Moreover, the soft error rates in normal gates are also rising which shifts the interests of soft error tolerance from memory-based devices to memory-less devices (wires, logic gates) [1]. As a consequence, tolerating soft error need to be featured in both memory and memory-less modules. Also, the detectability and correctability for a single memory cell or gate need to be simpler than traditional methods (i.e. redundancies, Hamming code, ...) to conserve the cost. A good trade-off between the additional area cost, the performance degradation and the protectability given by the soft error resilient method need to be carefully considered.

To protect the wire/gate from soft errors, we summarize the existing method into three main approaches as in Fig. 1: (i) Information Redundancy; (ii) Temporal Redundancy and (iii) Spatial Redundancy. Among these solutions, using error correction code (ECC) with further forward and backward corrections is a viable solution with lesser area cost and lower performance degradation. By encoding the data to obtain a codeword, corrupted data can be detected or corrected in the receiving terminal. By combining a coding technique with detection feature and retransmission which is call backward error correction, the system can further detect and correct more fault. On the other hand, forward error correction (FEC), by temporally ignoring and correcting the faults at the final receiver is another solution. Indeed, ECC plays a key role in the two mentioned solutions.

Among several existing ECCs, parity check is one of the very first methods to detect single flipped bit. Also, Hamming code (HM) [3] and its extension (Single Error Correction Double Error Detection: SECDED) [4] are two common techniques. This is due to the fact the those two ECCs only rely on basic boolean functions to encode and decode. Thanks to their low complexities, they are suitable for on-chip communication application and memories. On the other hand, Cyclic Redundancy Check (CRC) code is also another solution to detect faults. Since it does not support fault correction, it may not optimal for on-chip communication. Further coding methods such as BCH or Reed-Solomon is exceptionally strong; however, their complexities are overwhelming that prevent them from being widely applied in on-chip communication.

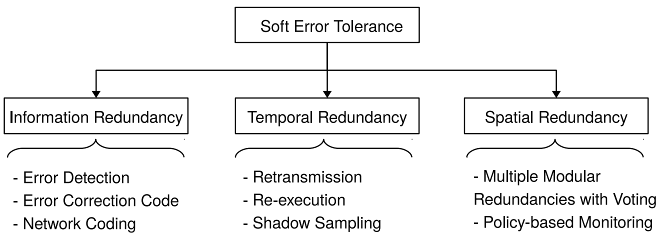


Fig. 1. Soft error tolerance approaches.

As previously mentioned, soft error rates of wires/logic gates are lower than memory. Also, soft error per bit rates of both SRAM and DRAM are likely to be reduced [1]. Therefore, we observe that using a low complexity error correction code may be suitable for future technologies. Moreover, with a low complexity coding method, it can be widely applied within a high complexity system without being considerably degraded in terms of area cost, power consumption, and performance.

In this paper, we present an architecture using Parity Product Code (PPC) to detect and correct soft errors in on-chip communication. In each transmission, the system can act like parity check to obtain low complexity and high coding rate. Furthermore, in cooperative with Razor flip-flop with Parity (RFF-w-P), the system can help correct SEU without using ARQ. If RFF-w-P cannot correct the faults, it will be corrected later using the PPC's correction or selective ARQs. The contribution is as follows:

- Architecture of encoder and decoder using Parity Product Code (PPC) that offers one fault detection in a flit and one fault correction in a packet (or trunk of flits).
- A method to correct SEU without HARQ by using dedicate Razor flip-flop with Parity (RFF-w-P).
- A Selective ARQs in row/column for PPC using a transposable FIFO design.
- Adaptive mechanism for the PPC-based system with various error rates.

The organization of this paper is as follows: Section II reviews the existing literature on coding techniques and fault-tolerances. Section III presents the proposed PPC and section IV shows the proposed architecture. Section V provides evaluation and Section VI concludes the paper.

II. 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 reliability; therefore, this section focuses on the method to tolerate soft error 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 [3], 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* [4] 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 [5]. Error detection only codes such as cyclic redundancy check (CRC) [6] is also widely used in digital network and storage applications. More complicated coding techniques such as Reed-Solomon [7], BCH [8] or Product-Code [9] 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. Despites 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 (iii) recovery and roll-back. Both retransmission [10] and re-execution [11] share the same idea of repeat the faulty action (transmission or execution). Due to the randomness of soft errors, this type of error likely to absent after a short period. With the similar idea, shadow sampling (i.e. Razor Flip-Flop [12]) use a delay (shadow) clock to sample data into an additional register. Then, by comparing the original data and the shadow data, the system can detect the possible fault. 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. 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; the most efficient method is to create spares and use them as replacements for faulty elements.

III. Parity Product Code

This section presents Parity Product Code (PPC) which is based on parity check and product code to inherit the benefits of both techniques [13], [9]. We first present the fault assumption. Then, the encoding and decoding process is presented.

A. 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). In addition, the selective ARQ can guarantee two flipped bits per packet. For higher error rates, using different ECCs with the help of ARQ could be a proper solution. Also, using multiple sets of code and interleaving could help handle more faults.

B. Encoding of PPC

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

$$P = \begin{bmatrix} F_0 \\ F_1 \\ \dots \\ F_{M-1} \\ F_P \end{bmatrix} = \begin{bmatrix} b^0_{0,0} & b^0_1 & b^0_2 & \dots & p^0 \\ b^1_0 & b^1_1 & b^1_2 & \dots & p^1 \\ b^2_0 & b^2_1 & b^2_2 & \dots & p^2 \\ \dots\dots & \dots\dots & \dots\dots & \dots\dots & \dots\dots \\ pb_0 & pb_1 & pb_2 & \dots & pp^i \end{bmatrix}$$

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

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

Followings are the calculation for parity data:

$$p = b_0 \oplus b_1 \oplus \dots \oplus b_{N-1} \tag{1}$$

and

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

Note that we can use a trunk of flits instead of a whole packet. Because the decoding process requires caching the whole M flits, using a small value of M could reduce the decoding latency.

C. 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 decider whether a SEU occurs:

$$\text{SEU}_F = \text{Parity}(F) = b_0 \oplus b_1 \oplus \dots \oplus b_{N-1} \oplus p \qquad (2)$$

If there is a SEU, SEU_F will be '1'. To quickly correct the flit, Hybrid Automatic Retransmission Request (HARQ) could be used for demanding retransmission. Because HARQ may cause congestions in the transmission, we use these two method: (i) Razor FF with Parity (see Section IV-A) and (ii) Correcting using the PPC correction method at the RX (act as FEC). 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:

$$\text{SEU}_P = \text{Parity}(P) = F_0 \oplus F_1 \oplus \dots \oplus F_{M-1} \oplus F_P \qquad (3)$$

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

D. Transposable Selective ARQ

If there are two flipped bits inside a flit, the parity check fails to detect. On the other hand, detected faulty flits may not be corrected by using ARQ or RFF-w-P due to the fact that the flit is already corrupted at the sender's FIFO. In both cases, the system relies on the correctability of PPC at the receiving terminal (RX).

As a FEC, PPC can calculate parity check of each bit-index as in SEU_P. Therefore, we can further detect it by Eq. 3. If a flit has an one (or an odd number) of flipped bits, a selective flit-index ARQ can help fix the data. On the other hand, if a flit has an even number of flipped bits, the SEU_F stays at zeros. Therefore, the decoder cannot determine the corrupted flits. However, SEU_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)). However, this cases is out of scope of this paper.

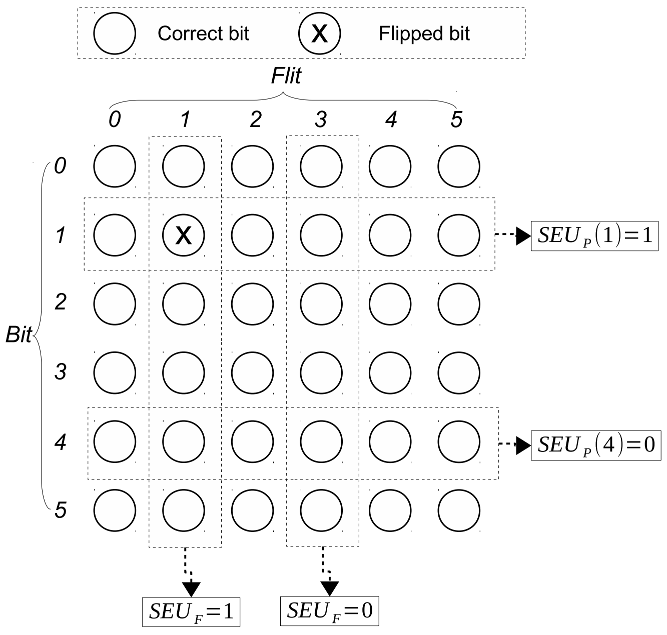


Fig. 2. Single flipped bit and its detection pattern.

Algorithm 1: Decoding Algorithm.

```
// Input code word flits
Input: $F_i = \{b^i_0, \dots b^i_{N-1}, p\}$
// Input code word flits from the shadow FIFO
Input: $F'_i = \{b'^i_0, \dots b'^i_{N-1}, p'\}$
// Output code word flits
Output: $oF_i$
// Output packet/group of flits
Output: $oF_i$
// Output ARQ
Output: $ARQ$
// Calculate the parity check of both flits
$SEU_F = b^i_0 \oplus \dots \oplus b^i_{N-1} \oplus p$
$SEU'_F = b'^i_0 \oplus \dots \oplus b'^i_{N-1} \oplus p'$
// Correct SEUs by using RFF-w-P
if ($SEU_F == 0$) then
    // The original code word is correct
    $oF_i = F_i$
else if ($SEU'_F == 0$) then
    // The shadow code word is correct
    $oF_i = F'_i$
else
    // both data is incorrect
    if ($ARQ == True$) then
        ARQ = False; /* Already do ARQ */
        $oF_i = F_i$;
        $oSEU_F = 1$;
    else
        ARQ = True;
        $oSEU_F = 0$;
if ($RX = True$) then
    // Forward Error Correction Code using PPC
    call FEC();
else
    return $oF_i$;
```

To correct the even number of fault case, the system has two options: (i) Full ARQ, and (ii) Selective ARQ. A full ARQ demands a replica of the whole trunk of flit (or packet) while the selective one only request the corrupted one. Based on two cases, there are two options: (i) column (or flit-index) ARQ and (ii) row (bit-index) ARQ. 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 is flipped leading to undetected SEU in F_2. At the SEU_P, the receiver find out that index 1 and 2 having flipped bit; therefore, we can use the ARQs to demand those flits:

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

and

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

In this work, we assume that the maximum flipped bit in a flit is two. There, the decoder aim to mainly use row ARQs because it cannot find out which flit has two flipped bit. 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: $F_i = \{b_0^i, \dots b_{N-1}^i, p\}$
// Output code word flits
Output: $oF_i$
// Output ARQ
Output: $ARQ$
1  if $i == 0$ then
2      $SEU_P = F_i$;
3      $regSEU_F = SEU_F$
4  else if $i < M - 1$ then
5      $SEU_P = SEU_P \oplus F_i$;
6      $regSEU_F = \{regSEU_F, SEU_F\}$;
7  else
8      if no or single SEU then
9          P = Mask ($F_i, SEU_P, regSEU_F$);
10         return P;
11     else
12         ARQ = $SEU_P$;
           // receive new flits ($i \geq N$) and write in row
              indexes
13         $F_{i=0,\dots,N-1}$ = write_row ($SEU_P, F_{(i \geq N)}$)
```

IV. Proposed Architecture

This section presents the architecture to perform the PPC scheme. We first present the Razor Flip-flop with parity bit which could help reduce the ARQ stage. Then, the architecture of encoding and decoding scheme is presented. Further enhancements are presented in last two sub-sections.

A. Razor flip-flop with Parity

To help reduce ARQ, we use Razor FF [12] for minoring the register with a parity bit as in Figure 3. Furthermore, we add circuit for detecting and correcting SEU. The shadow flip-flop samples data using a delay clock s_clk which is a generated clock from the system one (clk) with a small delay. The output of these flip-flops is checked using a parity check module (PAR). If only one of them fails the parity check, the other is used. If both original data and shadow data are failed ($SEU_F = 1$), a ARQ signal or FEC will be used. In the next retransmission of ARQ, if both registers repeat the same failure meaning that the sender's data may be corrupted, the FEC process is used. In this fashion, instead of keeping ARQ, it forward to the RX. The silent or uncorrectable faults are corrected later using the transposable selective ARQ (see Section III-D).

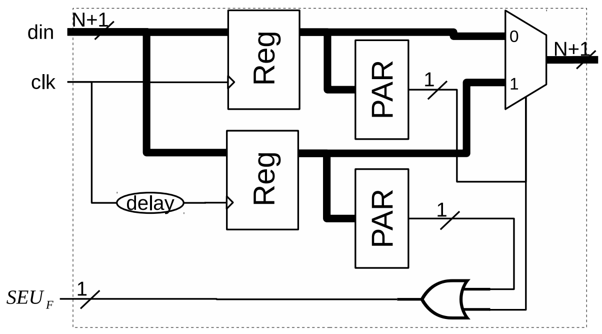


Fig. 3. RFF-w-P: Razor Flip-flop with Parity Check.

A single transient fault could be easily corrected using this method. Therefore, the HARQ process is removed which help reduce one clock cycle in the transmission process.

B. 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 the parity flit (F_P). This parity check flit is transmitted at the end of the packet.

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

When a flit arrives the decoder, it is checked and corrected by RFF-w-P first. Once this flit is done, it is pushed into the FIFO and the 'PACK. PAR' module. After completing the parity value of a packet, it was sent to the controller to handle the masking process. The masking process can correct a single flipped bit; therefor, selective ARQ is used once there are two faults are detected. As we previously assume, when there are two faults in a flit, the SEU_P value can indicate the bit-indexes of the faults. This value will be sent back to the encoder in order to retransmit the data belonging those indexes.

Note that we can implement PPC with or without RFF-w-P. In this case, retransmissions are required if the decoder detects data corruptions.

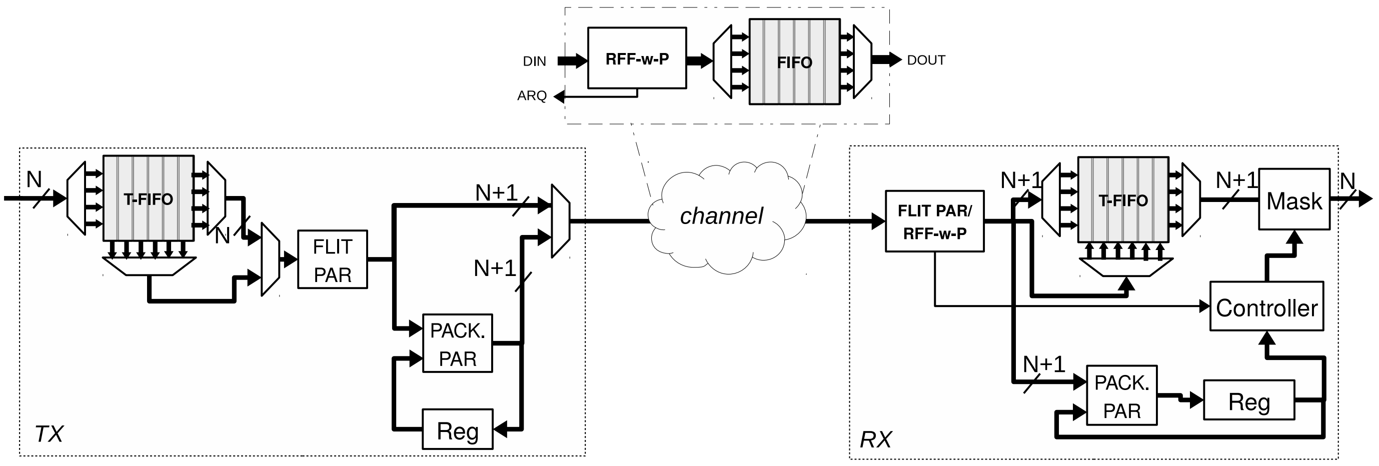


Fig. 4. PPC scheme: Parity Product Code for soft error correction.

C. Transposable FIFO

To support reading and writing in both column and row (as bit-index and flit-index ARQ), we use a transposable FIFO (T-FIFO) architecture. Besides the normal jobs of a FIFO, it also allows reading and writing by a column or row addresses. For a bigger size, RAM-based FIFO may be utilized. A transposable SRAM could be use [14] with 8 transistors instead of 6 as in the traditional ones. In this work, we use a DFF-based T-FIFO.

D. Additional modes of PPC

Depend on the fault rate situation, the system can adapt the coding scheme to reduced the latency and redundant bits. This part present additional modes of PPC.

1) Adaptive F_P issue: If the error rate are low enough, PPC can perform optional parity flit (F_P). In this case, each intermediate node (i.e. router in a network-on-chip) will check the parity of each flit as usual using Parity check or RFF-w-P. If the parity check fails, it first try to correct using RFF-w-P or HARQ. If both techniques cannot correct the fault, the node will send to TX (sending terminal) a signal to request the parity flit F_P. 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. This mode is only efficient for low error rates which have one flipped bit per packet.

2) go-back M flits: Moreover, we can extend further with a go-back retransmission instead of transposable ARQ. Assuming the maximum number of cache-able flits is K. Since F_P can be responsible $M > K$ flits, the correction by PPC is impractical and the system need 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 omit the correction process.

V. Evaluation

A. Methodology

The architecture is designed in Verilog HDL and synthesized using NANGATE 45 nm library using EDA tools by Synopsys. Because of the fault assumption (one/two faults per a group of flits), we compare the architecture to HM, SECDED, and PAR+ARQ (parity check with ARQ) which are common soft error correction methods.

B. Coding performance

Figure 5 shows the coding rate of PPC and others without ARQ. Coding rate is to present the ratio of useful bit in total transmitting bit. The coding rate of PPC is obtained as $[NM]/[(N+1)(M+1)]$ (N: flit's width, M: packet's length). As we can observe in this figure, PPCs with $M > 10$ have better coding rates than both HM and SECDED with less than 30 data bitwidth. For larger numbers of data bit-width (60+), HM and SECDED gain their coding rates due to the fact that the parity check flit F_P heavily impact 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 it only needs one extra bit. The major drawback of Parity is it only detect without correct faults.

$$P_{0,n} = (1-\epsilon)^n \tag{4}$$
$$P_{1,n} = n * \epsilon * (1-\epsilon)^n \tag{5}$$

Moreover, Figure 6 shows the evaluation with two different bit error rates (10^{-3} and 10^{-4}). This evaluation is based on Eq. 4 where $epsilon$ is the bit error rate, $P_{i,n}$ is the probability of having i fault in n bits. Here, we assume bit error probability in independent. Note that we only calculate for zero and one fault to investigate the efficiency of adaptive parity flit F_P. For PAR, HM, and SECED, n is data's bit-width while for PPC, we use n as data's bit-width multiply by number of flits per packet (M). In this evaluation, PPC with adaptive F_P can reach the coding rate of Parity in small

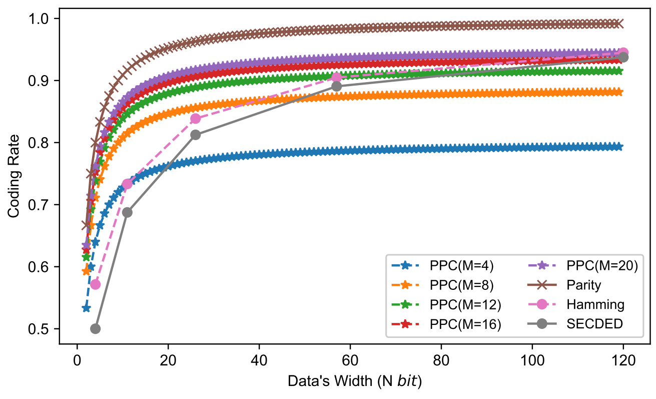


Fig. 5. Coding rates of PPC.

data widths or error rates. Once the probability of fault is increased, the PPC's coding rate is reduced rapidly to lower both Hamming and SECDED. On the other hand, PAR+ARQ has lower coding rate than ARQ (no-fault) and ARQ+RFF-w-P. Also, at lower error rate, PPC with adaptive F_P has coding rates closing to PAR+ARQ and outperforms both SECDED and HM.

C. Fault detection for packet

This section investigate the fault detection ability of PPC for a packet. This could help the system understand the channel's quality and go-back-M flits protocol. In order to study the detection ability for a packet, we perform a 10,000 cases Monte-Carlo simulation represented in Fig. 7. Monte-Carlo simulation is performed by randomizing the fault position in the channel and calculating the averaged value of the results. Here, we define a packet is faulty by having 2+ faults on either column or row check.

With PPC (N=2, M=2), the results show that the average number of detected faults is 3.6370. However, the in-depth analysis using Monte-Carlo simulation also points out that only 56.69% and 36.05% of 3-faults and 5-faults cases, respectively, were detected. In fact, the results show that more than 99% of 4+ fault patterns have been detected. The three faults pattern detection rates of 4×4, 8×8 and 16×16 are 82.39%, 93.91% and 98.14%, respectively.

With ability to detect multiple faults, the decoder can either choose to perform the selective or the whole packet retransmission.

D. 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 I presents in details the hardware cost of PPC's parity modules and the sub-modules. We use NANGATE 45 nm library and set the frequency as 500MHz for the power estimation. Maximum frequencies are investigated separately.

As we can observe, the area cost of the PPC's encoder and decoder are less than 18% of the TX and RX in both area and power consumption. Most of the cost consumption belong to the memory-based module. For instance, FIFO occupies 60.2% and 45.2% of TX's and RX's area cost and RFF-w-P occupies 20.5% and 25.5% of RX's area cost and power. Although its complexity of encoder and decoder seem higher than other, the area cost of the FIFO in the intermediate nodes can be reduced thanks to the smaller bitwidth.

In comparison to three common coding techniques (Parity, Hamming and SECDED). The encoder and decoder of PPC both cost more area and power. This is because the codec requires register for calculating the SEU_P.

Design of T-FIFO also has smaller area overhead in comparison to the normal FIFO. With 33 data bitwidth, 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/write in both column/row.

E. Comparison

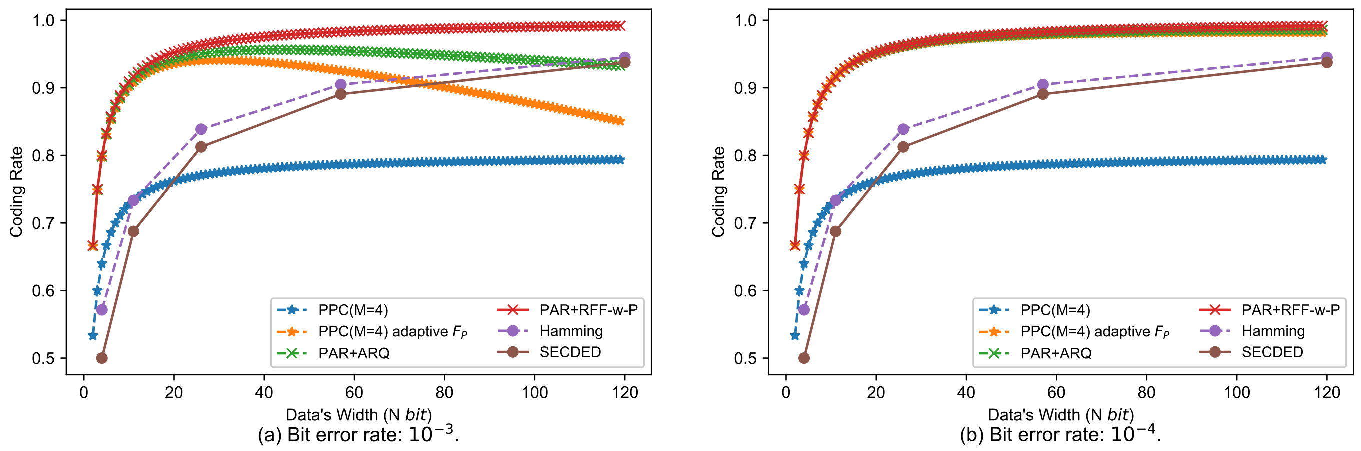
1) Razor register with parity: Table II presents the hardware cost of the RFF-w-P in comparison to a normal register. As can be seen, the area costs are nearly $3.4\times$ the normal register's because it needs to double the register and use two parity check module. Moreover, thank to its resilience to transient faults, it can help reduce one clock cycle for the retransmission. In comparison, RFF-w-P demands less area cost ($1.41\times$) than SECDED's decoder with normal registers ($2.05\times$) while providing one-bit correction. Also, power consumption of RFF-w-P is higher because it use double registers. However, SECDED uses extra bit (i.e. 40 instead of 33), therefore the area and power overhead of the its system could be higher.

2) Hardware complexity: Figure 8 and 9 show the hardware complexity for the transmitter (TX), receiver (RX) and FIFO area. Beside Hamming, SECDED, Parity (PAR) and PPC, we also consider the RFF-w-P as an option to study the complexity of these configurations.

The area cost of PPC's TX and RX are both higher than Parity, Hamming and SECDED. This is due to the encoding and decoding process of PPC both need registers. The design of T-FIFO is also slightly complicated than normal FIFO with less than 7% overheads in terms of area and power.

Considering the FIFO's area of the intermediate node in on-chip communications, the FIFO of PPC and PAR (no RFF-w-P) cost less area cost than SECDED and Hamming. The 16-bit version of on-chip communication parts using PPC and PAR without using RFF-w-P costs 14.04% and 17.65% less than Hamming and SECDED, respectively. This is due to more bitwidth in Hamming and SECDED (20 and 21 bit instead of 17 bit).

To provide more protection, RFF-w-P could be implemented with extra hardware cost. With 16, 32 and 64 data width-bit, FIFO with RFF-w-P (in PPC+RFF-w-P and PAR+RFF-w-P) are 50.15%, 53.28%, and 55.20% larger than normal FIFO, respectively. The area cost of PPC's TX is increased by less than 30% to be able to has RFF-w-P. Note that RFF-w-P works



(a) Bit error rate: 10^{-3}. (b) Bit error rate: 10^{-4}.

Fig. 6. Coding rates evaluation with different bit error rates.

TABLE I
HARDWARE COMPLEXITY RESULTS OF PPC AND OTHER METHODS WITH 32-BITWIDTH.

Design	Module	Sub-module	Area (μm^2)	(%)	Power (μW)	(%)	Max Freq. (MHz)
Intermediate node's 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
PPC	TX	Total	1827.9520	(100)	823.2660	(100)	1,273.88
		Controller	322.9240	(17.7)	167.7900	(20.4)	-
		Encoder	359.1000	(19.6)	123.1230	(15)	-
		T-FIFO (32-bit)	1100.7080	(60.2)	529.6990	(64.3)	-
	RX	Total	2528.3301	(100)	1.220×10^3	(100)	1,270.64
		Controller	467.6280	(18.5)	194.7110	(16)	-
		Mask	78.2040	(3.1)	4.0350	(0.3)	-
		Decoder	280.8960	(11.1)	167.4860	(13.7)	
		T-FIFO (33-bit)	1142.2040	(45.2)	600.7490	(42.3)	-
		RFF-w-P	517.1040	(20.5)	310.3160	(25.5)	-

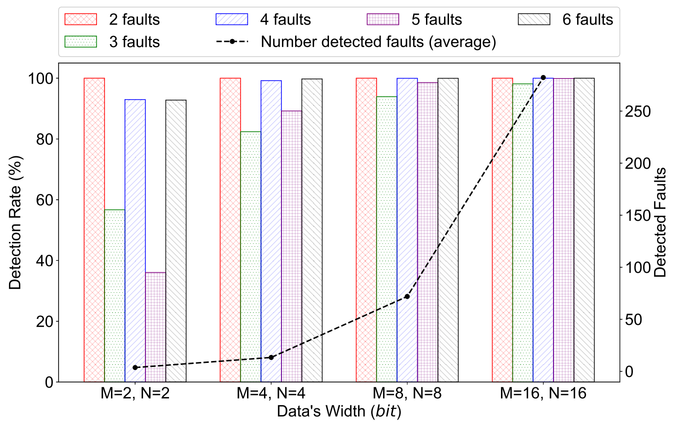


Fig. 7. Fault detection ability for packet of PPC.

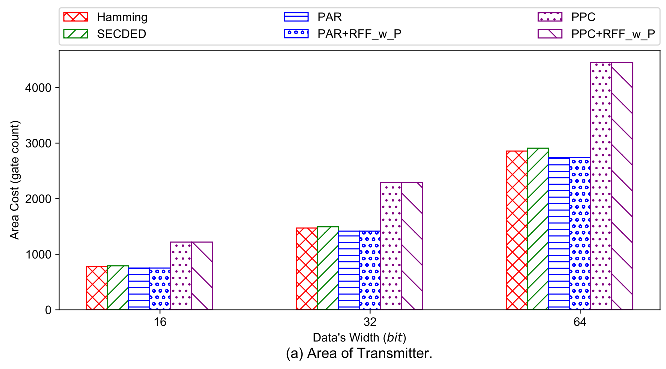
TABLE II
HARDWARE COMPLEXITY OF RFF-W-P IN COMPARISON TO NORMAL REGISTERS.

	Normal register		RFF-w-P (vs normal register)	
N	Power (μW)	Area (μm^2)	Power (μW)	Area (μm^2)
17	64.425	90.4400	154.0858 (2.39)	304.5700 (3.37)
33	123.8534	175.5600	298.318 (2.41)	592.3820 (3.37)
65	245.2194	345.8000	587.9642 (2.40)	1169.0700 (3.38)

as a pre-stage of FIFO therefore the area cost could be reduced by integrating RFF-w-P to the FIFO.

VI. Conclusion

In this paper, we adopt PPC as an error correction code and its hardware extensions for detecting and correcting soft errors in on-chip communications. We further Razor flip-flop with Parity to help detect and correct soft errors. A transposable FIFO is also presented to help the system retransmission by



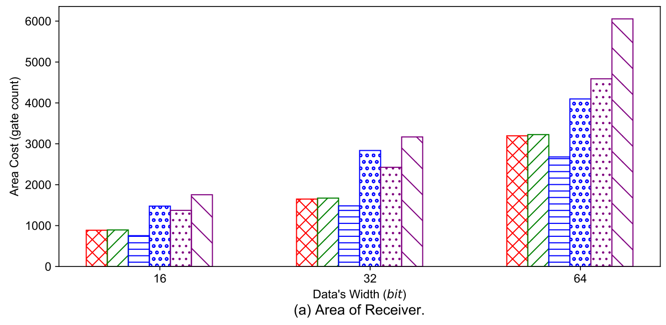


Fig. 8. Hardware complexity result in equivalent NAND2x1 gate counts.

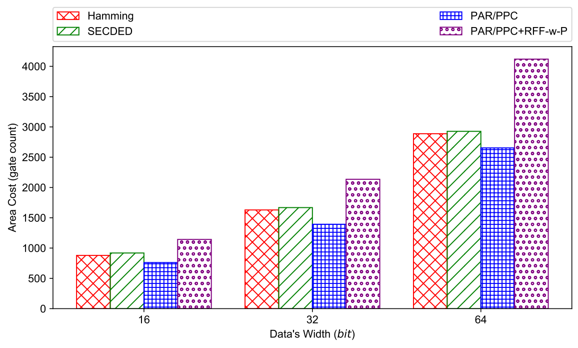


Fig. 9. FIFO complexity result in equivalent NAND2x1 gate counts.

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 need to check the parity of the data. The coding rate is also promising with better than Hamming and SECDED with large packet sizes or in low error rates. Also, PPC can detect more faults to be about to inform the system.

In the future, the design of PPC and Razor flip-flop with Parity could be implemented in a specific Network-on-Chip to investigate the impacts on the performance. Because this method is based on Parity check, RFF-w-P could be combined with Hamming or SECDED to obtain a stronger protection.

ACKNOWLEDGMENT

The work is supported by University of Engineering and Technology, Vietnam National University, Hanoi under the project CN18.10.

REFERENCES

[1] R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Transactions on Device and materials reliability*, vol. 5, no. 3, pp. 305–316, 2005.

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

[3] R. W. Hamming, "Error detecting and error correcting codes," *Bell Labs Tech. J.*, vol. 29, no. 2, pp. 147–160, 1950.

[4] M.-Y. Hsiao, "A class of optimal minimum odd-weight-column SEC-DED codes," *IBM J. Res. Dev.*, vol. 14, no. 4, pp. 395–401, 1970.

[5] L.-J. Saiz-Adalid *et al.*, "MCU tolerance in SRAMs through low-redundancy triple adjacent error correction," *IEEE Trans. VLSI Syst.*, vol. 23, no. 10, pp. 2332–2336, 2015.

[6] W. W. Peterson and D. T. Brown, "Cyclic codes for error detection," *Proceedings of the IRE*, vol. 49, no. 1, pp. 228–235, 1961.

[7] S. B. Wicker and V. K. Bhargava, *Reed-Solomon codes and their applications*. John Wiley & Sons, 1999.

[8] I. S. Reed and X. Chen, *Error-control coding for data networks*. Springer Science & Business Media, 2012, vol. 508.

[9] F. Chiaraluce and R. Garello, "Extended Hamming product codes analytical performance evaluation for low error rate applications," *IEEE Trans. Wireless Commun.*, vol. 3, no. 6, pp. 2353–2361, 2004.

[10] L. L. Peterson and B. S. Davie, *Computer networks: a systems approach*. Elsevier, 2007.

[11] K. N. Dang *et al.*, "Soft-error resilient 3d network-on-chip router," in *2015 IEEE 7th International Conference on Awareness Science and Technology (iCAST)*, Sept 2015, pp. 84–90.

[12] 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*. IEEE Computer Society, 2003, p. 7.

[13] R. M. Pyndiah, "Near-optimum decoding of product codes: Block turbo codes," *IEEE Trans. Commun.*, vol. 46, no. 8, pp. 1003–1010, 1998.

[14] J.-s. Seo, B. Brezzo, Y. Liu, B. D. Parker, S. K. Esser, R. K. Montoye, B. Rajendran, J. A. Tierno, L. Chang, D. S. Modha *et al.*, "A 45nm cmos neuromorphic chip with a scalable architecture for learning in networks of spiking neurons," in *Custom Integrated Circuits Conference (CICC), 2011 IEEE*. IEEE, 2011, pp. 1–4.