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

Khanh N. Dang and Xuan-Tu Tran
Email: khanh.n.dang@vnu.edu.vn
VNU Key Laboratory for Smart Integrated Systems (SIS Lab),
VNU University of Engineering and Technology,
Vietnam National University, Hanoi (VNU)
Sep. 14th, 2018

Outline

• Introduction
• Proposed Architecture
• Evaluation
• Conclusion

Introduction

• Electronics devices in critical applications may expose to several soft error sources
  • Alpha particle
  • Cosmic ray
  • Neutrons

• Transistors are scaling down
  • \( \rightarrow \) lower error rate per bit causing by a particle
  • \( \rightarrow \) larger error rate per chip due to more density

• Moving to 3D-ICs:
  • Top layers act as a shield from cosmic ray

[1] "Radiation-Induced Soft Error Rate Analyses for 14 nm FinFET SRAM Devices", Soonyoung Lee et al. (Samsung Electronics)
[2] "Technology Scaling Trend of Soft Error Rate in Flip-Flops in 10 nm Bulk FinFET Technology", Taki Umemura et al. (Semiconductor R&D Center, Samsung Electronics)
Introduction (cnt.)

• Also, non-critical but high density devices may also need protection

• Tolerating soft errors for less critical devices may demand:
  • Lower complexity to be widely applied
  • Adapt to different error rate
  • Awareness of potential high defect rates

On-chip communication soft error protection

• Objectives:
  • Memory devices
  • Memory-less devices: lower error rates

• Protection Method:
  • Temporal redundancy
  • Spatial redundancy
  • Information Redundancy:
    • Error Correction Code
      • For example: Parity, Hamming, SECDED
      • Forward Error Correction
        • Temporally accept the occurrence of fault
        • Correct later at receiving terminals
      • Backward Error Correction
        • Request retransmission of sender

Context

• Soft error rates of wires are lower
  • They may not require extreme protection per bit
  • Need optimal in coding rate (ratio of useful bits in total bits)

• Soft error rates are predictably reduced while the complexity is increased
  • Lower complexity coding mechanism
  • Optimal coding rate

→ We observe the need of an adaptive coding mechanism
  • Low coding rate
  • Adaptive: accept ECC, BER and FEC

Motivation

• Among the error correction, using Parity-check could give high coding rate ($\frac{N-1}{N}$ for $N$ bits) and low complexity ($N - 1$ XOR gates); however:
  • Parity cannot correct fault without retransmission
  • If there are two flipped bits, Parity-check fails to detect

• In order to improve the Parity-check, Parity Product Code (PPC) could be used:
  • Parity for a flit and bit index (a.k.a row and column)
  • Detect 2 bits, correct 1 bit
  • Low complexity (extra $N$ XOR gates and $N$ DFFs for packet parity)
Parity Product Code: Encoding

A packet having M data flits is encoded as:

\[ P = \begin{bmatrix} F_0 \\ F_1 \\ \vdots \\ F_{M-1} \\ F_P \end{bmatrix} = \begin{bmatrix} b_0^0 & b_1^0 & \cdots & b_{M-1}^0 & p^0 \\ b_0^1 & b_1^1 & \cdots & b_{M-1}^1 & p^1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ b_0^{M-1} & b_1^{M-1} & \cdots & b_{M-1}^{M-1} & p^{M-1} \\ p_b^0 & p_b^1 & \cdots & p_b^{M-1} & pp \end{bmatrix} \]

- Each flit \( F \) consists of N data bits (\( b_0, b_1, \ldots, b_{N-1} \)) and one parity bit \( p \):
  \[ p = b_0 \oplus b_1 \oplus b_2 \oplus \cdots \oplus b_{N-1} \]

- The parity flit \( F_P \) is also the parity check of all flits:
  \[ F_P = F_0 \oplus F_1 \oplus F_2 \oplus \cdots \oplus F_{M-1} \]

Parity Product Code: Decoding

- The decoding has three stages

  **Stage 1:** Calculate the parity check of a flit
  \[ SEU_F = b_0 \oplus b_1 \oplus b_2 \oplus \cdots \oplus b_{N-1} \oplus p \]

  **Stage 2:** Calculate the parity check of a packet
  \[ SEU_P = F_0 \oplus F_1 \oplus F_2 \oplus \cdots \oplus F_{M-1} \oplus F_P \]

  **Stage 3:** Decide whether the received packet (\( \bar{P} \)) is correct, correctable or uncorrectable (detected).

Correction

- If there is one fault, both \( SEU_P \) and \( SEU_F \) are one-hot. We can correct by inversing the faulty position.
- To do the inversing, we use a masking circuit:
  \[ Mask[i, j] = 1 \text{ if } \begin{cases} \text{Correct bit} & \text{if } i = j \text{ \&\& } \text{Correct bit} \\ \text{Flipped bit} & \text{if } i = j \text{ \&\& } \text{Flipped bit} \end{cases} \]

  However, PPC stills:
  - Need retransmission for each flit as Parity-check
  - Need whole packet retransmission if the packet’s corrupted

- Decoded packet will be:
  \[ P' = \bar{P} \oplus Mask \]

Contribution

- Architecture of Parity Product Code (PPC)
- A method to correct SEU without HARQ by using dedicate Razor flip-flop with Parity (RFF-w-P).
  \( \rightarrow \) Remove retransmission
- A Selective ARQs in row/column for PPC using a transposable FIFO design.
  \( \rightarrow \) Resend the bit-index instead of whole packet.
- Adaptive mechanism for the PPC-based system with various error rates.
  \( \rightarrow \) Issuing the parity flit adaptively which can enhance the coding rates
**Outline**

- Introduction
- Proposed Architecture
  - Retransmission Optimization
  - Proposed Mechanism and Architecture
  - Adaptive algorithm
- Evaluation
- Conclusion

**Retransmission Optimization**

**Flit:**
- If there is one fault in the flit, PPC acts as parity-check and requires re-transmission.
- If the re-transmission fails to correct (i.e. soft error on the storing register), forward error to the ending terminal.
  - How can we reduce the need of re-transmission if it may not effective?

**Packet:**
- If there is one fault, PPC can correct using Masking technique.
- If there are 2+ faults, PPC requires retransmission for the whole packet.
  - Could we provide a more efficient method for retransmitting a packet?

If the error rate is lower, how can we optimize?
- More optimum techniques for PPC for lower error rate.

**Remove the retransmission using RFF-w-P**

- If the error occurs in the storing register, retransmission fails to correct it.
- Retransmission only help correct faults during the "movement" of the flit.
- Here, we use the basic shadow sampling with a trick: "using Parity check"

By using the "RFF-w-P", we can correct the soft error during transmission and forward the error of storing register to the terminal.
Efficient retransmission by T-ARQ

- If there are 2 errors on two different flits, the decode may correct by selective transmission.
  - Retransmit the whole packet
- If there are 2 error on the same flits, the decode can only recognize the faulty indexes
  - Retransmit the fault indexes'
  - T-ARQ: Transposable Automatic Retransmission Request.
- After the retransmission, the decoder can re-check the packet to ensure the integrity of the packet.
- We use a transposable FIFO which support read/write by two direction (row and column)
- For large sizes, transposable SRAM could be easily adopted.


Decoding

Algorithm 1: Decoding Algorithm.

Forward Error Correction and T-ARQ

Algorithm 2: Forward Error Correction and Selective ARQ Algorithm.

Transposable ARQ

- Assuming at the RX, there are two error which is hidden to flit-parity.
- The packet parity can find out two faulty indexes
- With T-ARQ, TX retransmits the fault indexes
- This could reduce the need of complete packet retransmission
Proposed Architecture

To detect 1 flipped bit

To detect 2+ flipped bits

Adaptive algorithm for lower error rates

• Adaptive parity flit
  • If the error rate is low, the parity flit $F_p$ could be issue only when a flit failed
  • RFF-w-P in each node check the integrity of flits, if there is a flipped bit \( \rightarrow \) send a signal to transmitting terminal (TX)
  • TX send the $F_p$ at the end of the packet if it receives a request.

• No parity bit (p):
  • No parity bit is used.
  • Only parity flit, T-ARQ is used for correction

• Go-back M flits:
  • Assume the RX has ability to store $K$-flits,
  • TX send parity flit each $M$ flits ($M>K$)
  • If RX finds out error after $M$ flits, T-ARQ cannot be executed
  • The system need to perform go-back-$M$ flits

Adaptive Parity Flit

$F_p$ is faulty \( \rightarrow \) issue $F_p$ request ($SEU_p$)

$F_p$ is not faulty \( \rightarrow \) do nothing

No parity bit (p)

• With T-ARQ, TX retransmits the fault indexes
• Since this method has high latency, we can apply it when the error rate is low enough
• Since there is no parity bit, we can disable the RFF-w-P (i.e. clock/power gating)
Outline

- Introduction
- Proposed Architecture

Evaluation

- Methodology:
  - Verilog HDL
  - NANGATE 45 nm
  - Synopsys EDA Tool
- Coding rate evaluation
- Hardware complexity evaluation
  - Area cost
  - Maximum frequency
  - Power consumption
- Comparison to low complexity ECC
  - Hamming/SECDED
  - Parity-check

Coding Rates (1): fixed packet-length (M)
Fault Model and Probability

\[ P_{0,n} = (1 - \varepsilon)^n \]
\[ P_{i,n} = n \times \varepsilon \times (1 - \varepsilon)^n \]

\( \varepsilon \) is the bit error rate, \( P_{i,n} \) is the probability of having \( i \) fault in \( n \) bits.

Coding Rates (2): adaptive parity flit

Improved by using adaptive parity flit

Detection Rate

Can detect 100% of 2 faults cases

In lower coding rate, using adaptive parity flit strongly enhance the coding rate ➔ Reaching the Parity’s coding rate

Coding Rates (3): adaptive parity flit
RFF-w-P’s hardware complexity

The area and power of RFF-w-P roughly triples the area of normal register
- Double the register sizes
- Adding two parity check and a multiplexer
- However, it could reduce one clock cycle of retransmission.

<table>
<thead>
<tr>
<th></th>
<th>Normal register</th>
<th>RFF-w-P (vs normal register)</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>Power (µW)</td>
<td>Area (µm²)</td>
</tr>
<tr>
<td>17</td>
<td>64,425</td>
<td>90,440</td>
</tr>
<tr>
<td>33</td>
<td>123,853</td>
<td>175,560</td>
</tr>
<tr>
<td>65</td>
<td>245,219</td>
<td>345,800</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Power (µW)</th>
<th>Area (µm²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>154,0858</td>
<td>(2.39)</td>
</tr>
<tr>
<td>33</td>
<td>298,318</td>
<td>(2.41)</td>
</tr>
<tr>
<td>65</td>
<td>587,9642</td>
<td>(2.40)</td>
</tr>
</tbody>
</table>

Hardware complexity breakdown

<table>
<thead>
<tr>
<th>Design</th>
<th>Module</th>
<th>Sub-module</th>
<th>Area (µm²) (%)</th>
<th>Power (µW) (%)</th>
<th>Max. Freq. (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIFO</td>
<td>Parity-based</td>
<td>32-bit, 4-ots</td>
<td>1111.8800</td>
<td>562.4571</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Hamming-based</td>
<td>96-bit, 4-ots</td>
<td>1300.2080</td>
<td>664.5885</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>SECCED-based</td>
<td>48-bit, 4-ots</td>
<td>1331.3300</td>
<td>685.0590</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Encoder</td>
<td>94.1640</td>
<td>68.8866</td>
<td>2,570.69</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Decoder</td>
<td>234.8790</td>
<td>206.0590</td>
<td>1,949.86</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Encoder</td>
<td>11.7500</td>
<td>82.3679</td>
<td>2,564.10</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Decoder</td>
<td>253.7680</td>
<td>206.3793</td>
<td>1,250.00</td>
<td>-</td>
</tr>
<tr>
<td>PARITY</td>
<td>Encoder</td>
<td>49.4760</td>
<td>49.5052</td>
<td>2,666.67</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Decoder</td>
<td>51.0790</td>
<td>52.4233</td>
<td>2,388.55</td>
<td>-</td>
</tr>
</tbody>
</table>

Receiver complexity

![Receiver complexity graph](image)

Transmitter complexity

![Transmitter complexity graph](image)
Conclusion

- We presented an implementation of parity production code with several optimizations:
  - Using RFF-w-P to remove retransmissions while ensure the correctability.
  - Using transposable FIFO (or RAM) to provide a transposable retransmission that is more efficient than retransmit the whole packet.
  - Several optimal techniques for lower error rates
    - Adaptive parity flit
    - Optional parity bit
    - Go-back-M with Parity Production Code

- The results of evaluation show:
  - Efficiency of RFF-w-P and adaptive parity flit where they significantly gains the coding rate.
  - Detection rate of PPC ensure 100% for 2 faults.
  - Design of T-FIFO cost extra area cost and power consumption but it provide more flexibility in retransmission.

Future work & Discussion

- Future work: Integrating the proposed algorithm into a system:
  - An adaptive algorithm could developed to control the operation of different mode/optimization of PPC.

- Discussion:
  - Since we use parity-check, the technique can be integrated into any parity-based ECCs
  - The latency of PPC is higher due to it need to cache a packet to decode
    - correction "on the fly" could be adopted:
      - At the node the flit faulty, it decode and provide correction
      - send the correction information at the end of packet
      - RX receive and correct right after it receive