isQED Symposium 2012

Low Complexity Cross Parity Codes for Multiple and Random Bit Error Correction

M. Poolakkaparambil[1], J. Mathew[2], A. Jabir[1], & S. P. Mohanty[3]
Oxford Brookes University[1], University of Bristol[2]
University of North Texas[3]
Email: 09137484@brookes.ac.uk[1], jimson@cs.bris.ac.uk[2], ajabir@brookes.ac.uk[1], saraju.mohanty@unt.edu[3]

Presented by
Oghenekarho Okobiah, University of North Texas.

OXFORD BROOKES UNIVERSITY
University of BRISTOL
UNT UNIVERSITY OF NORTH TEXAS Discover the power of ideas.

Overview

- Motivation
- Novel Contributions
- Cross Parity Code
- Design Perspective
- Experimental Results
- Conclusion & Future Work

Motivation

- Viable solution for multiple bit error tolerance is vital in critical applications.
- Requirements in fault tolerant circuit design,
  - Low area overhead.
  - Low power dissipation.
  - Maximum fault/ error coverage.
  - No deterioration in normal circuit performance.
- Fault injection based attacks in cryptography related arithmetic circuits is a major concern.
  - Low complexity multiple error correction can be one solution.

Contributions of this Paper

- Multiple Error correction with improved fault coverage.
- Optimized for area and power.
- First known approach has been made to make a practical test bench 163-bit digit serial FF multiplier with the proposed scheme.
- Both behavioral and geometrical level implementation has been made.
- Comparison with existing known error correcting architectures.

Prior Related Research

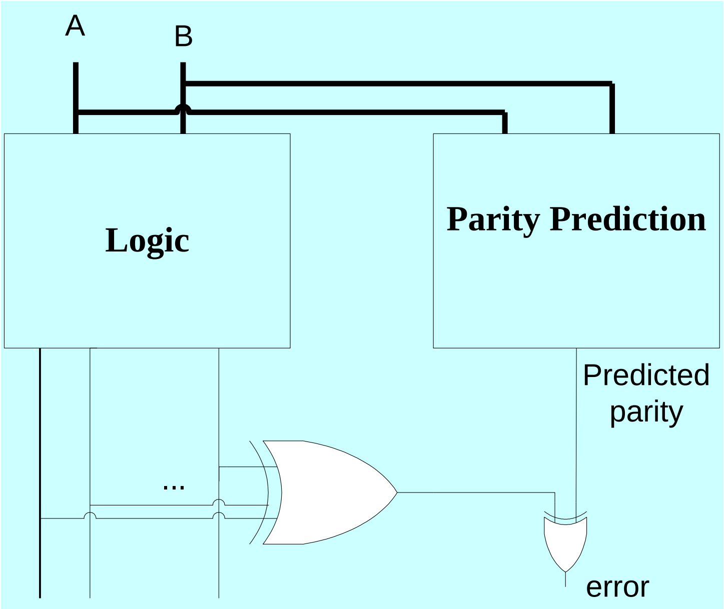

Fig 1. CED based on Parity

Ref: M. Nicoliadis , "**Carry checking/parity prediction adders and ALUs** ", IEEE Trans. VLSI Systems, vol. 11, Oct. 2003

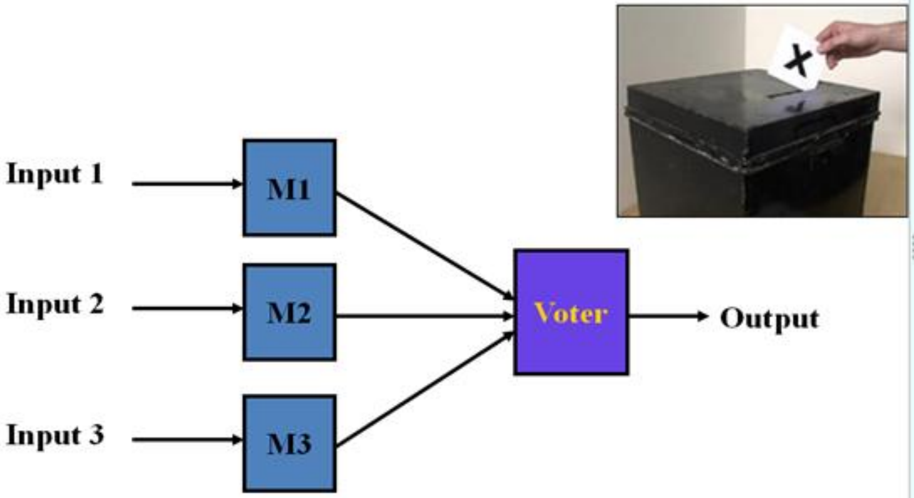

Fig 2. Triple Modular Redundancy

Prior Related Research

Research	Design	Fault Tolerance	Coverage	Improvement (%)
Mathew[11]	Hamming	Error correction	1-bit error	Y
Mathew[12]	LDPC	Error correction	1-bit and certain 2-bits	~1.5 x Y
Masoleh[2]	CED	Error correction	No correction	-
Alves[13]	CED	Error detection	No correction	-
Poolakkaparambil[13]	BCH only	Error correction	1, 2 and 3-bit errors	~3 x Y
[Proposed]	BCH & Simple Parity	Error correction	Up to certain 13-bit errors	~13 x Y

Cross Parity Code

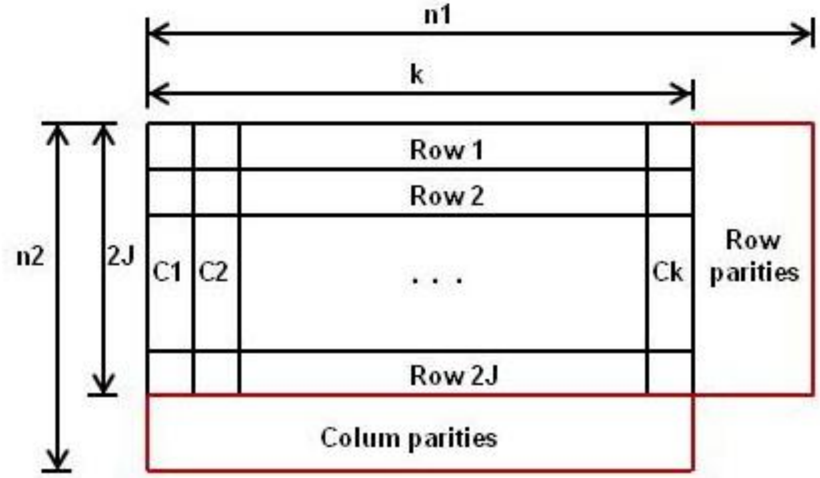

Fig 3. Cross Parity Code Encoding

- The encoding in Cross Parity Code is done similar to the product codes.
- Each row and column is encoded separately with same or different codes.
- In the proposed scheme case we use BCH codes for row and Simple parity for column.
- The decoding is done different from the classical product code.

Cross Parity Code

- The Cross Parity code row encoding can be done using any multiple error detection codes (BCH codes proved to be better in the proposed case).

C0	C1	C2	C3	C4	Ham/BCH Parity -1
C5	C6	C7	C8	C9	Ham/BCH Parity -2
C10	C11	C12	C13	C14	Ham/BCH Parity -3
C15	C16	C17	C18	C19	Ham/BCH Parity -4
CP0	CP2	CP4	CP6	CP8	
CP1	CP3	CP5	CP7	CP9	

$$P(x) = x^{n-k}\ M(x) \bmod g(x). \qquad (1)$$

$$P(x) = p_9x^9 + p_8x^8 + p_7x^7 + p_6x^6 + p_5x^5 + p_4x^4 + p_3x^3 + p_2x^2 + p_1x^1 + p_0 \qquad (2)$$

BCH Parity for Row

$$P1 = C_0 \oplus C_2 \oplus C_4 \qquad (3)$$

$$P2 = C_1 \oplus C_2 \oplus C_3 \oplus C_4 \qquad (4)$$

$$P3 = C_0 \oplus C_3 \oplus C_4 \qquad (5)$$

$$P4 = C_1 \oplus C_2 \oplus C_4 \qquad (6)$$

Hamming Parity for Row

$$CP_0 = C_0 \oplus C_{10} \qquad (7)$$

$$CP_1 = C_5 \oplus C_{15} \qquad (8)$$

$$CP_2 = C_2 \oplus C_{12} \qquad (9)$$

$$CP_3 = C_7 \oplus C_{17} \qquad (10)$$

Simple Parity for Colum

- Simple parity is used on column, as it is efficient in locating detected error (later used in correction).

Encoding and Decoding Algorithm

Cross Parity Code Algo:

Arranging the Circuit Output bits in Rows and Columns

↓

Generate parity, $P(x) = x^{n-k} \bullet M(x) \bmod g(x)$

OR

Hamming Parity, for Rows

↓

Error Detection using Row Code

&

Error location using Column Code

↓

Correction using Cross Parity Decoder

↓

Corrected bits at output

↓

End!

Design Perspective

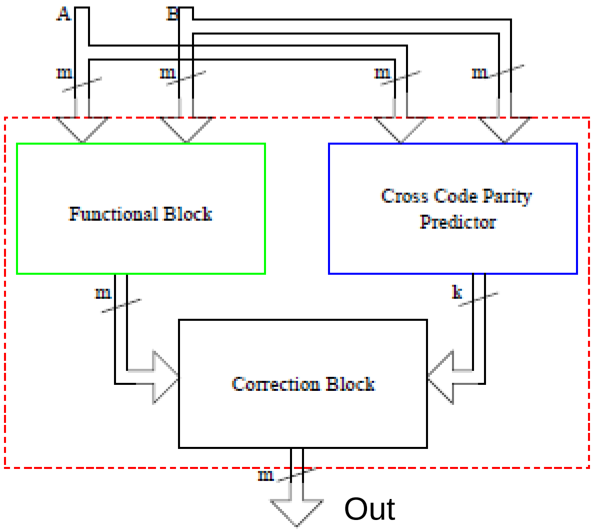

Fig 4. Block diagram of the Cross Parity Code based Error Correction Circuit

- The technique used only the error detection properties of both column and row codes.
- The row parities predict the error occurrence and the column parity information is used to locate them.
- A low complexity decoder is then used to correct the detected errors.

Design Perspective (Bit-parallel Multiplier)

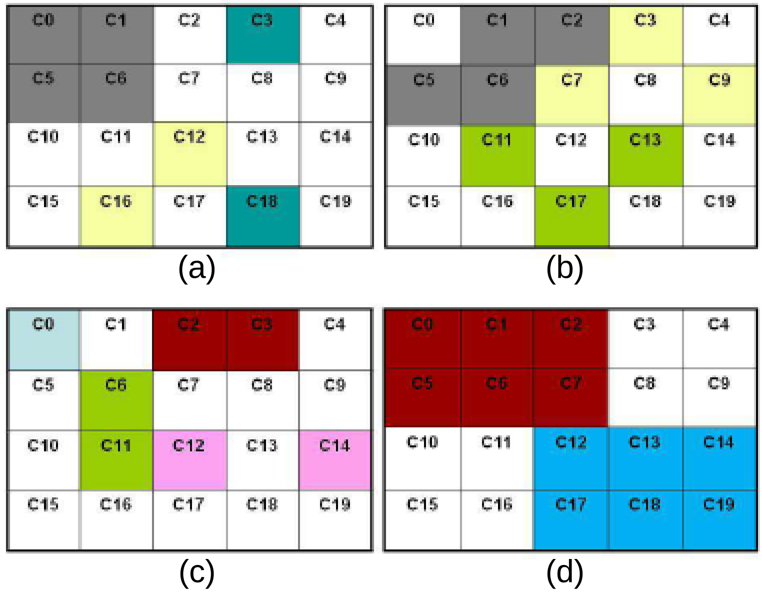

Fig 5. Certain error patterns of a 20-bit test bench multiplier circuit

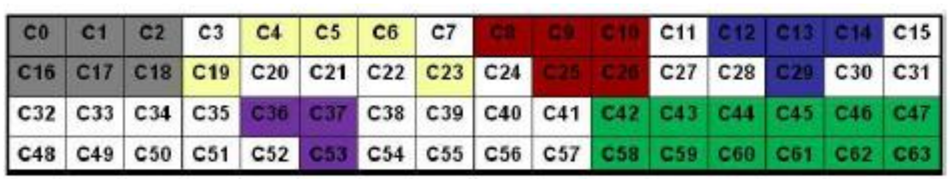

Fig 6. Certain error patterns of a 63-bit test bench multiplier circuit

Experimental Results

isQED Symposium 2012

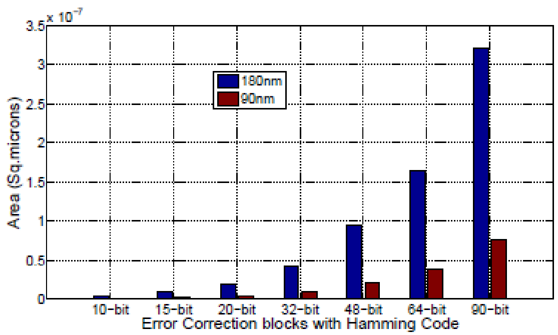

Fig 7. Error detection and correction block area of hamming cross parity

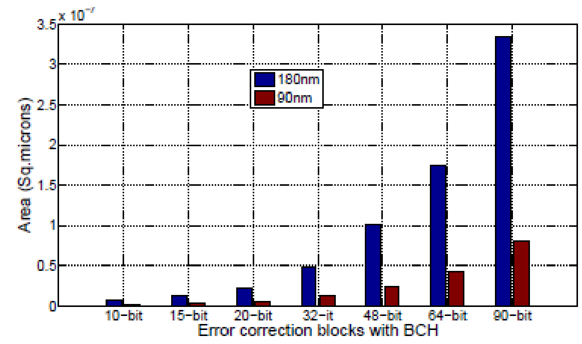

Fig 8. Error detection and correction block area of BCH cross parity code

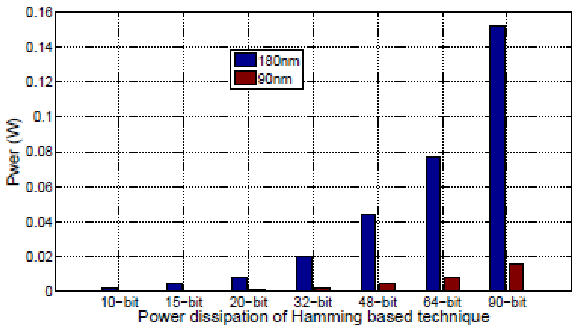

Fig 9. Power Dissipation of Hamming code based technique

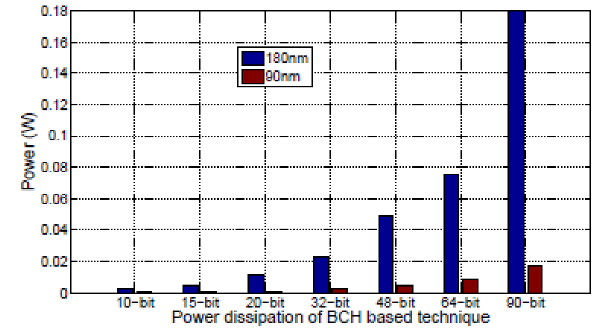

Fig 10. Power Dissipation of BCH code based technique

OXFORD BROOKES UNIVERSITY

University of BRISTOL

UNT UNIVERSITY OF NORTH TEXAS
Discover the power of ideas.

Experimental Results

TABLE I

AREA OVERHEAD COMPARISION OF VARIOUS MULTIPLIER SIZES

No. of bits	Hamming	BCH
10	142%	160%
15	123%	152%
20	121%	140%
32	108%	120%
48	105%	116%
64	104%	114%
90	101%	106%

TABLE II

COMPARISON WITH OTHER APPROACHES FOR 32-BIT MULTIPLIER

Property	Masoleh et al. 2004 [16]	Mathew et al. 2008 [12]	BCH [14]	Cross Parity (Ham)	Cross Parity (BCH)
#errors correction	single	single	3 Errors	up to 6 Errors	up to 12 Errors
Coding technique	Hamming	LDPC	Classic BCH	Hamming + Simple Parity	BCH + Simple Parity
Overhead	>100%	>100%	150.4%	108%	120.4%

Extension to Digit-Serial Multiplier

Algorithm 1:

Input : $A(x) = \sum_{i=0}^{m-1} a_i.x^i$, $B(x) = \sum_{i=0}^{m-1} b_i.x^i$, P(x).

Output : $C(x) = A(x).B(x) mod P(x)$.

Step1: $C = 0$.

Step2: $for\ i = 0\ to\ \lceil m/D \rceil$ - 1 do

Step3: $C = Bi.A + C$.

Step4: $A = A.\alpha^D$.

Step5: *end for*

Step6: return (C mod P(x))

Algorithm 1. Digit-Serial Multiplier [15]

- Due to low area overhead of the proposed scheme, they can be easily incorporated with Digit-Serial multiplier.
- Algorithm 1. is the test bench digit-serial multiplier used in this research.
- For practical design comparison, a 163-bit multiplier is used (FIPS, NIST standard).
- This believe to be first attempt reported to test a practically used digit-serial multiple error correctable design.

Experimental Results

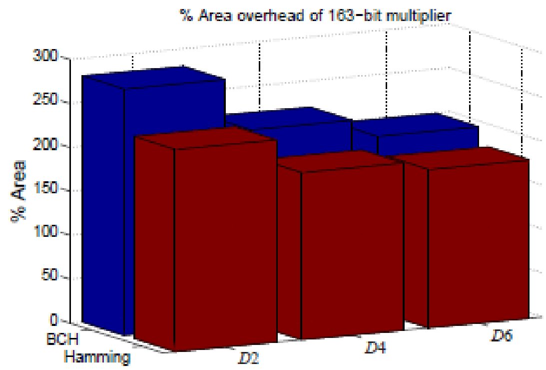

Fig 11. Area Overhead of 163-bit digit Serial Multiplier with various Digit Sizes

- Area overhead reduces with digit size 'D'
- Cross Parity technique is suitable for both digit-serial and bit-parallel architectures due to their low area overhead

- Behavioural modelling has been achieved using VHDL.
- Designs are verified for functionality.
- Synthesis & Geometrical implementation using Synopsys and Cadence SoC Encounter.

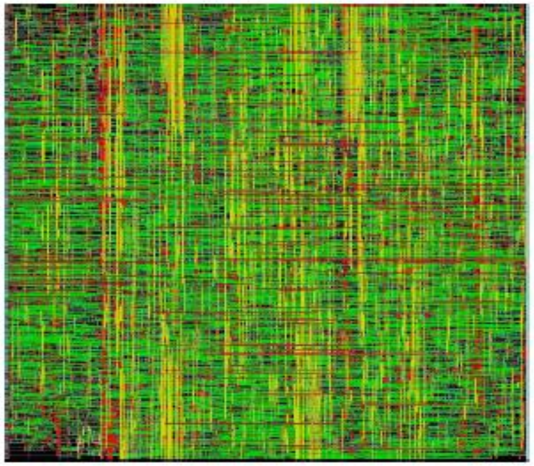

Fig 12. Layout of 163-bit Multiplier with Cross Parity ECC

Conclusions

16

- A novel multiple error correcting code has been proposed.
- The error correction scheme has been tested with a practically applicable 163-bit multiplier test bench circuit.
- The design is functionally verified using Modelsim and physical implementation has done using 180nm technology.
- Proposed method have area overhead of only 106% for a 90-bit bit-parallel circuit and 170% for a 163-bit digit serial multiplier.
- The generic property of the design allows to extend the scheme for any circuits with ‘n- inputs’ and ‘m’ outputs.
- Roughly 13x improved fault coverage w.r.t other single error correction schemes and 5x improvement w.r.t BCH with comparable area overhead.
- First known multiple error correcting design implemented for a practically used 163-bit digit serial multiplier circuit.
- Future extension include testing the proposed scheme on processor level by making the complete processor fault tolerant.

Thank you…

The presentation is available at:

http://www.cse.unt.edu/~smohanty/Presentations/Presentations.html