

BCH Code Based Multiple Bit Error Correction in GF Multiplier Circuits

P. Mahesh¹ & A. M. Jabir : Oxford Brookes University
J. Mathew² & D. K. Pradhan : Bristol University
S. P. Mohanty³ : University of North Texas
E-mail: 09137484@brookes.ac.uk¹, jimson@cs.bristol.ac.uk², saraju.mohanty@unt.edu³

Presented by: Oleg Garitselov : University of North Texas







Overview



- Motivation
- Background
- Galois Arithmetic Circuits
- BCH Code Based Error Correction
- Design Steps
- Experimental Results
- Conclusion & Future Work





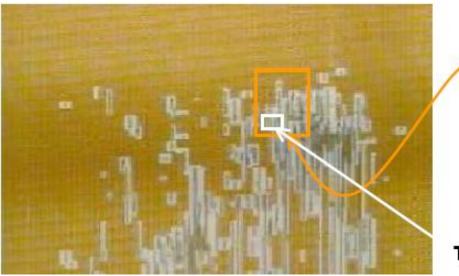




Motivation



Fault attacks on Crypto. Hardware Laser on the AL-AH registers



AL-AH registers

Targetted region

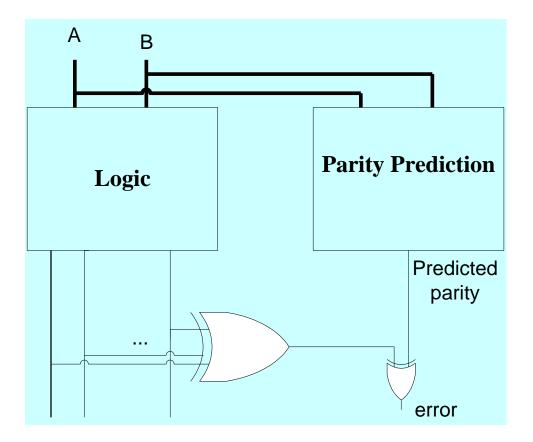






Previous Work





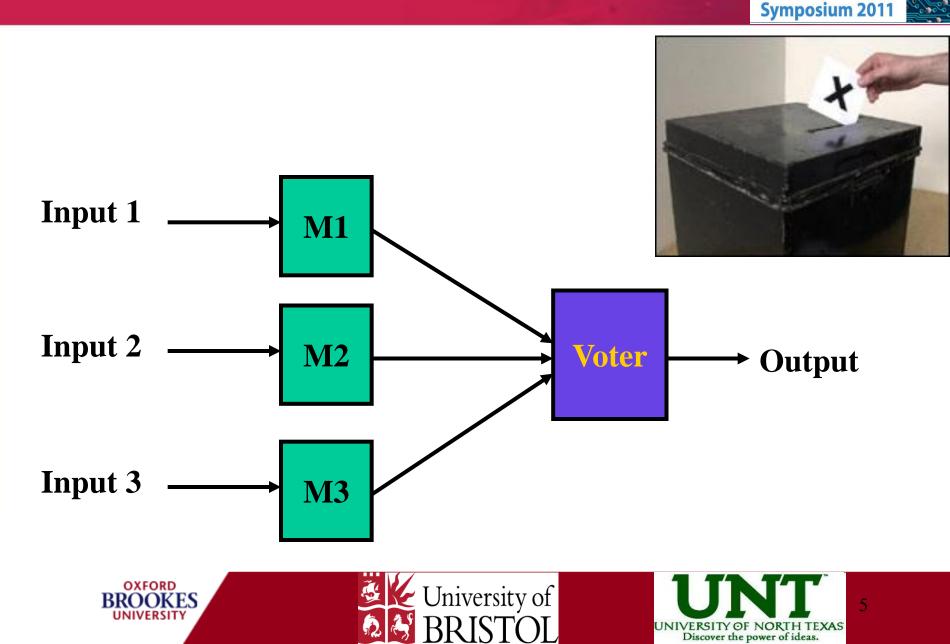
Ref: M. Nicoliadis , "Carry checking/parity prediction adders and ALUs ", IEEE Trans. VLSI Systems, vol. 11, Oct. 2003







Hardware Redundancy (TMR)



isQED





- Also called *Galois Field*, denoted by GF(*N*)
- $N = p^k$, p is a prime number, and k is an integer
- Each element is a *k*-tuple
- There are exactly *N* elements in the field (0, 1,..., *N*-1) or (0, 1, α , α^2 , α^3 ,..., α^{N-2}) where, α is a primitive element







Finite Field



- Two operators
 - '+' called addition operation forming Abelian Group
 - '.' called multiplication operation forming Abelian group.
- Additive and multiplicative identities 0 and 1 respectively
- Additive and multiplicative inverse exists for each element









$$[0, 1, \alpha, \alpha^2] = [0, 1, \alpha, \beta]$$

Elements can be represented in GF(2ⁿ) as 2-tuples over GF(2).











Let $\alpha^2 = \beta$ Thus we have 4 elements 0, 1, α and β

| GF(4) | 2-tuple | of GF(2) | Polynomial Representation | | |
|-------|-----------------------|------------|--|--|--|
| | a ₁ | a 0 | <i>a</i> ₁ <i>x</i> + <i>a</i> ₀ | | |
| 0 | 0 | 0 | 0 | | |
| 1 | 0 | 1 | 1 | | |
| α | 1 | 0 | x | | |
| β | 1 | 1 | (<i>x</i> +1) | | |





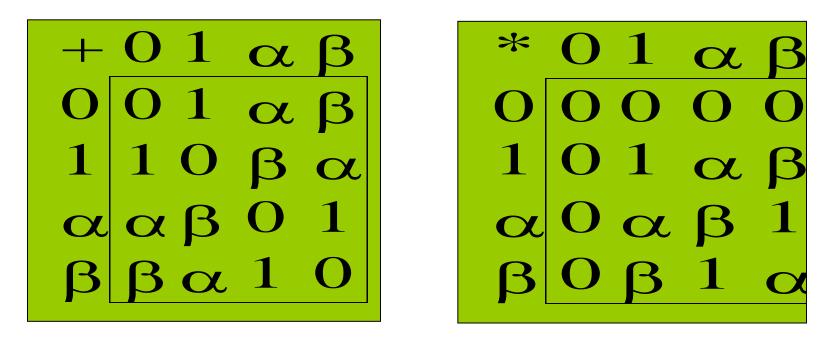


9

Addition and Multiplication



• Addition and Multiplication in GF(4)



> β is multiplicative inverse of α , and vice versa.







Generation of GF(2^m)



- Let us generate GF(8) with PP p(x) = x³ + x + 1.
- Let α be a root of p(x), i.e. $p(\alpha) = 0$.
- Then $\alpha^3 + \alpha + 1 = 0$, i.e. $\alpha^3 = \alpha + 1$.

| α^{0} | = | 1 | 0 | \leftrightarrow | [0,0,0] |
|--------------|---|---|------------------------------------|-------------------|---------|
| $lpha^1$ | = | α | α | \leftrightarrow | [0,1,0] |
| α^{2} | = | $lpha^2$ | α^2 | \leftrightarrow | [1,0,0] |
| α^{3} | = | $\alpha + 1$ | $\alpha^3 = \alpha + 1$ | \leftrightarrow | [0,1,1] |
| $lpha^4$ | = | $\alpha^2 + \alpha$ | $\alpha^4 = \alpha^2 + \alpha$ | \leftrightarrow | [1,1,0] |
| α^{5} | = | $\alpha^3 + \alpha^2 = \alpha^2 + \alpha + 1$ | $\alpha^5 = \alpha^2 + \alpha + 1$ | \leftrightarrow | [1,1,1] |
| α^{6} | = | $\alpha^2 + 1$ | $\alpha^6 = \alpha^2 + 1$ | \leftrightarrow | [1,0,1] |
| 0 | _ | | $\alpha^7 = 1$ | \leftrightarrow | [0,0,1] |









Multiplication is done by using polynomial mod a primitive polynomial P(x). i.e. $\alpha(x) \cdot \beta(x) \mod P(x)$

Let
$$P(x) = x^2 + x + 1$$

Thus $\alpha \cdot \beta = x (x + 1) = x^2 + x$

So
$$(x^2 + x) \mod x^2 + x + 1 = 1$$

 $\alpha \cdot \beta = 1$
 $\alpha^{-1} = \beta, \beta^{-1} = \alpha$
 $\alpha \operatorname{and} \beta \operatorname{are inverse of each}$
 $\alpha \operatorname{other}$











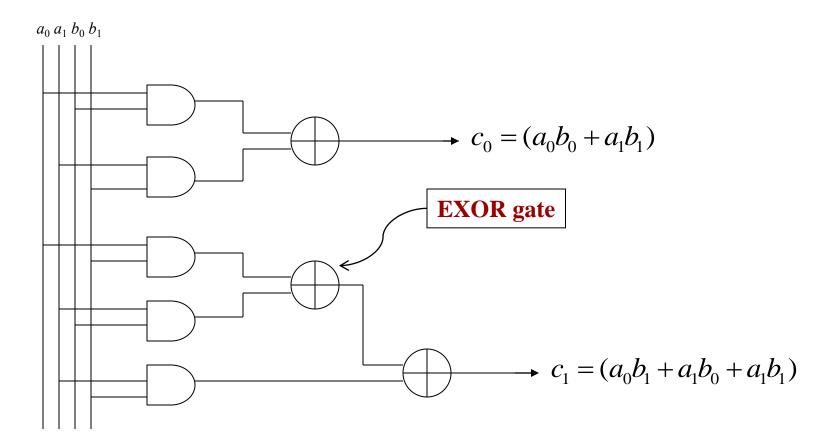


Figure:2 GF(4) multiplier

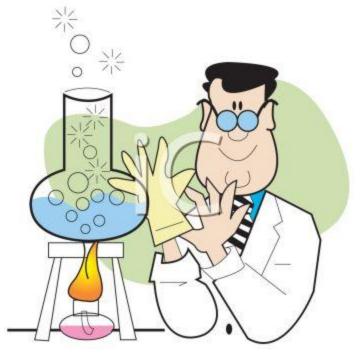








Multiple Bit Error Correction Using BCH Codes

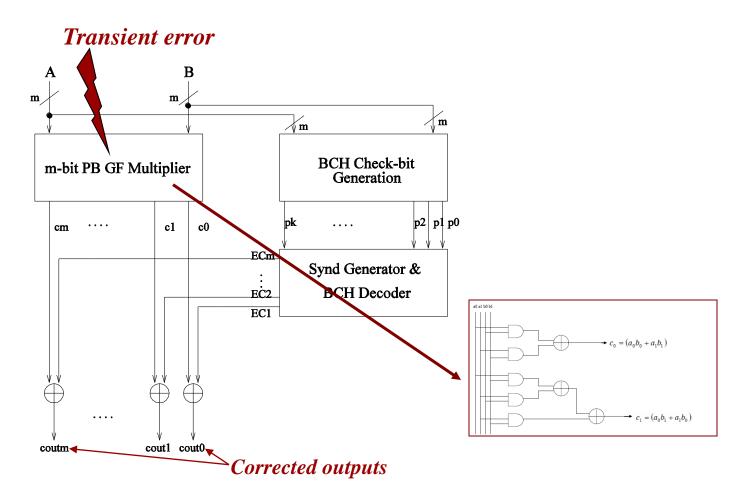








Multiple Bit Error Correction Using BCH Codes: The Design Architecture



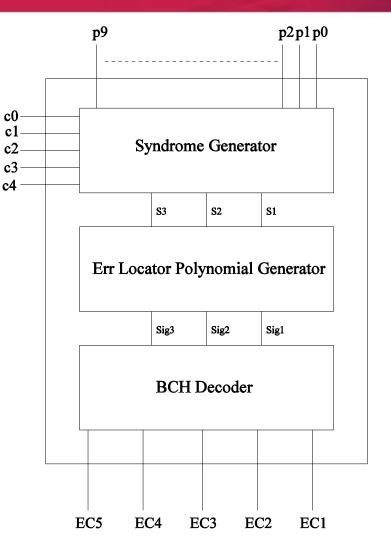
OXFORD BROOKES UNIVERSITY





Multiple Bit Error Correction Using BCH Codes: Synd. Generator and BCH Decoder



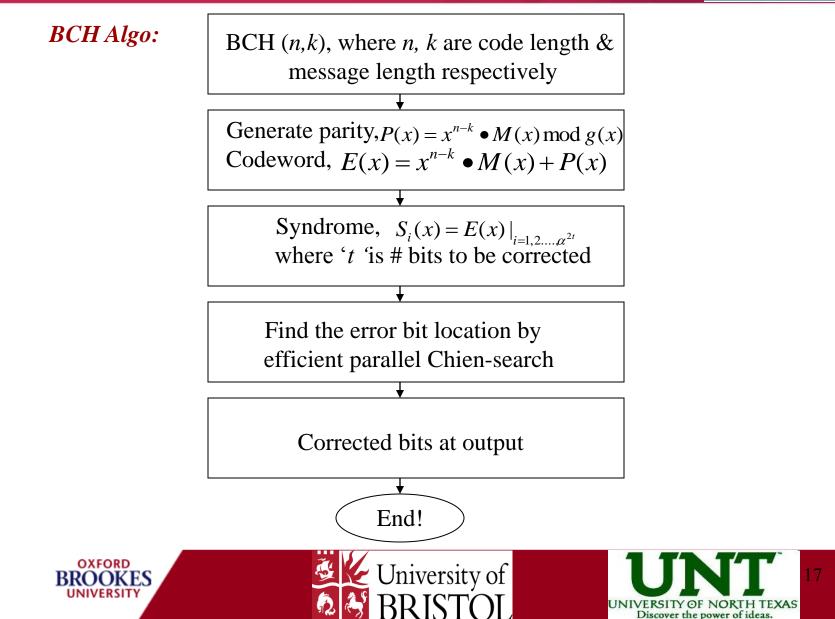






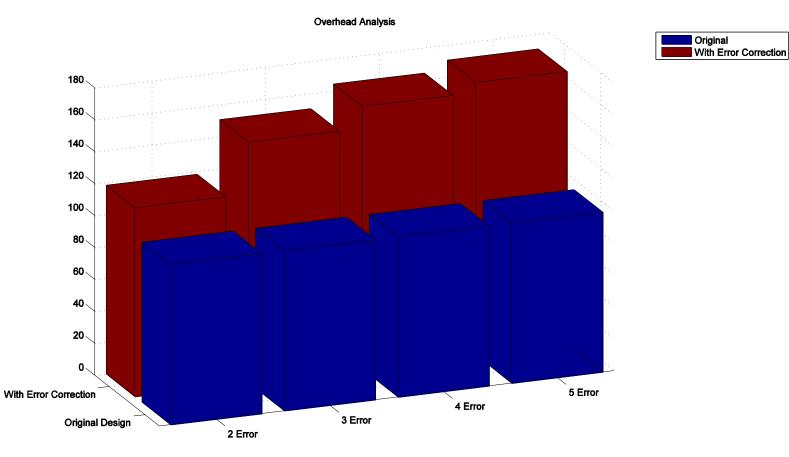


Multiple Bit Error Correction Using BCH



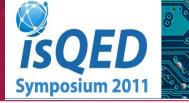
Experiments & Results







Experiments & Results



| Property | [12] | [10] | Proposed | Proposed | Proposed |] |
|--------------------|---------|--------|----------|----------|----------|---|
| #errors correction | single | single | 3 Errors | 4 Errors | 5 Errors | ← |
| Coding technique | Hamming | LDPC | BCH | BCH | BCH | 1 |
| Overhead | >100% | 100% | 150.4% | 164.04% | 170.4% |] |

Area comparison with other techniques

| | | Maga | | | | | |
|-----------------------|---------------------------------------|---|---|---|---|---|------------------------|
| | /test_bch_top/b | 0000000000001111 | 000000000000000000000000000000000000000 | 21111 | Bits 1.2 & 16 are | erroneous | |
| | /test_bch_top/c_out | 000000000000000000000000000000000000000 | 000000000000000000000000000000000000000 | | in the second | | |
| | /iest_bch_top/luut1/a | 000000000000000000000000000000000000000 | 000000000000000000000000000000000000000 | 20000 | 1-1 | | |
| | /test_bch_top/kut1/b | 0000000000001111 | 000000000000000000000000000000000000000 | 21111 | | | |
| | /test_bch_top/last1/c_out | 000000000000000000000000000000000000000 | 000000000000000000000000000000000000000 | 200000000000000000000000000000000000000 | 100000000000000000000000000000000000000 | 000000000000000000000000000000000000000 | |
| | | | | 110000000000000000000000000000000000000 | 2000000000011000 | 0111000000000000000 | 010100000000 |
| | /best_bch_top/uut1/t_p | 000000000000000000000000000000000000000 | 0.0000000000000000000000000000000000000 | 2000 | | | |
| | /test_bch_top/uut1/t_s1 | Gerrected | 00000 | 10110 | 00101 | 110 100 | 111101 |
| | //est_bch_top/uut1/t_s3 | odioprected | 66650 | I11000 | 111000 | 201010 | 2000:10 |
| | /best_bch_top/uut1/t_s5 | | uquu - | 01000 | 10010 | 200100 | 200011 |
| | /test_bch_top/uut1/sig1 | 00000 | 00000 | 10110 | 00101 | 10100 | 111101 |
| | /test_bch_top/luut1/sig2num10 | 00000 | 90000 | 111001 | 110001 | 111101 | 10110 |
| | //test_bch_top/wut1/tig2num11 | 00000 | 00000 | 11111 | 20001 | 01100 | 01001 |
| | /test_bch_top/laut1/sig2num1 | 00000 | 00000 | 110111 | 110011 | 201090 | 201010 |
| | /test_bch_top/ku/t1/sig2den10 | 00000 | 00000 | 10101 | 11111 | 11000 | 200111 |
| | //test_bch_tsp//uut1/sig2den1 | 00000 | 00000 | 01101 | 200111 | 10010 | 200101 |
| | /test_bch_top/Lut1/sig2den | 00000 | 00000 | 01111 | 01100 | 2000.10 | 10111 |
| Modelsim simulation 🔍 | //test_bch_top/kut1/sig2 | 00000 | 00000 | 200011 | 201010 | (10000 | 2000.10 |
| | /test_bch_top/Lut1/sg3num1 | 00000 | 00000 | huu | 200111 | 20111 | 11111 |
| 1 | /iest_bch_top;kut1/sig3 | 00000 | 00000 | 110010 | 20000 | 110101 | 111010 |
| results | /test_bch_top;luut1/t_cor1 | 0 | | + | | | |
| | <pre>/test_bch_top/aut1/t_cor2</pre> | 0 | | | | | |
| | /test_bch_top(last1/t_cor3 | 0 | | | | | Street, Street, Trans. |
| | <pre>/test_bch_top,luut1/t_cor4</pre> | 0 | | | Bit positions | corresponds to err | oneous bits |
| | /test_bch_top/Luit1/t_cor5 | 0 | | | | | |
| | /test_bch_tsp,but1/t_cor6 | č | | | | | |
| | /test_bch_top;laut1/t_cor7 | ě | | | | | |
| | /htst_bch_top/kut1/t_cor8 | 0 | | | | | |

[10] Fault Tolerant Bit Parallel Finite Field Multipliers Using LDPC Codes, J. Mathew, J. Singh, A. M. Jabir, M. Hosseinabady, D. K. Pradhan, IEEE 2008.
 [12] Single Error Correctable Bit Parallel Multipliers Over GF(2ⁿ), J. Mathew, A. M. Jabir, H. Rahaman, D. K. Pradhan, IET Computer Digital Tech., Vol. 3, Iss. 3, pp. 281-288, 2009.







Conclusion & Future Work



- Multiple bit error correction with compromise over slightly higher area
- For a fixed number of bits to be corrected, percent area overhead reduces with larger and more practical multipliers
- Highly parallel and efficient Chien-search block
- This scheme can be easily extendable to GF multiplier of any size



















Questions?







