

Lab 3

Part 1: Error Detection/Correction Circuit

- Implement an error detection/correction circuit by modifying the Robertson's multiplier circuit you have designed in Lab 2

Part 2: Adder Implementations

- Modify the single-bit xor-based adder of your text to derive 2 new adder forms.
- Measure the delays of each adder

Error Detection/Correction

The Code

<u>Value</u>	<u>Codeword</u>	<u>Value</u>	<u>Codeword</u>
0	11001000011	5	01001110100
1	11000100101	6	00101111000
2	10100101001	7	11000010110
3	01100101010	8	10100011010
4	01010110010	9	10010011100

Invariants of The Code

- ***Number of 1's: each codeword consists of exactly 6 zeros & 5 ones.***
 - Capability of detecting any *odd* number of errors
- ***Sum of indices: If we assign an index of 1 to 11 to each bit and perform a summation of indices where a 1 appears and subtract the indices where a 0 appears, the result is -4.***

Example:

Indices:	11 10 9 8 7 6 5 4 3 2 1
Codeword 1:	1 1 0 0 0 1 0 0 1 0 1
Indices for 1s:	$1+3+6+10+11 = 31$ (Sum1)
Indices for 0s:	$2+4+5+7+8+9 = 35$ (Sum2)
Sum1-Sum2 =	-4

Error Detection & Correction

- ***Invariant #1***

- Detects any *odd* number of errors

- ***Invariant #2***

- Detects any single bit errors
- Detects any 2-bit errors
- Pinpoints the position of the corrupted bit

Reason: *With single bit error, the resulting sum will deviate from -4 by **twice** as much as the index value*

Examples:

Indices: 11 10 9 8 7 6 5 4 3 2 1

Codeword 7: 1 1 0 0 0 1 1 0 1 1 0

Indices for 1s in the least significant 5 bits: $2+3+5+6+10+11=37$ (Sum1)

Indices for 0s in the least significant 5 bits: $1+4+7+8+9 = 29$ (Sum2)

Sum1-Sum2 = 8

$(8 - (-4)) / 2 = 12/2 = 6$ (index of error)

Other Properties of The Code

Value Embedding property

- For each codeword i , the difference in the least significant 5 bits between the sum of 1-bit indices and the sum of 0-bit indices, plus 9 divided by 2, equals i

Indices:	11	10	9	8	7	6	5	4	3	2	1
Codeword 7:	1	1	0	0	0	0	1	0	1	1	0
Indices for 1s in the least significant 5 bits:	$2+3+5=10$ (Sum1)										
Indices for 0s in the least significant 5 bits:	$1+4 = 5$ (Sum2)										
$((\text{Sum1}-\text{Sum2})+9)/2 = 7$ (equal to the index of the codeword)											

Indices:	11	10	9	8	7	6	5	4	3	2	1
Codeword 9:	1	0	0	1	0	0	1	1	1	0	0
Indices for 1s in the least significant 4 bits:	$3+4+5 = 12$ (Sum1)										
Indices for 0s in the least significant 4 bits:	$1+2 = 3$ (Sum2)										
$((\text{Sum1}-\text{Sum2})+9)/2 = 9$ (equal to the index of the codeword)											

Circuit Functionality

- ***Phase 1: Checking Invariant 1***
 - Multiplicand: 1
 - Multiplier: the input codeword
 - Accumulator: the result obtained by checking the invariant
- ***Phase 2: Checking Invariant 2 and the value-embedding property simultaneously***
 - Multiplicand: bit index implemented by a counter
 - Multiplier: the input codeword
 - To obtain the embedded value, only the first 5 steps are needed.
- ***Correcting single bit error***
 - Add 4
 - Compute the absolute value of the accumulator using an absolute value evaluation logic
 - Divide the absolute value by 2 via smart connection of signal lines
 - Flip the erroneous bit using a decoder together with xor gates

Adder Functionality

$$C_{out} = xy + xC_{in} + yC_{in}$$
$$S = (x \oplus y) \oplus C_{in}$$

x	y	C _{in}	S	C _{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Adder Derivations

- *One can modify the boolean equations such that the adders can share gates*

$$C_{out} = xy + xC_{in} + yC_{in}$$

$$C_{out} = xy + (x + y)C_{in}$$

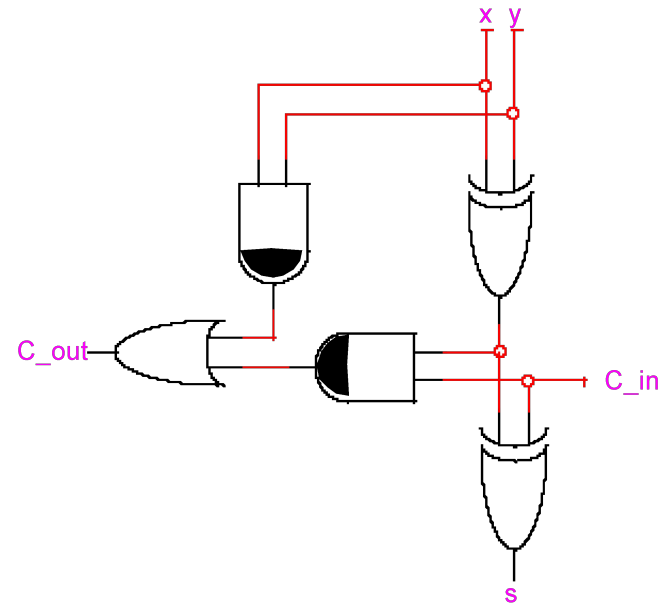
$$C_{out} = xy + (xy + x'y + xy')C_{in}$$

$$C_{out} = xy + xyc_{in} + (x'y + xy')C_{in}$$

$$C_{out} = xy + (x'y + xy')C_{in}$$

$$C_{out} = xy + (x \oplus y)C_{in}$$

$$s = (x \oplus y) \oplus C_{in}$$



Adder Implementation Requirements

- *Design 2 additional adders*
 - *One should contain 6 NAND gates and 2 OR gates*
 - *The other should contain 4 AND gates, 3 OR gates, and 2 inverters*
 - *Report the carry-to-carry delay*

Name	Delay (ns)	Name	Delay (ns)
2-input AND	2.4	2-input NAND	1.4
2-input OR	2.4	2-input NOR	1.4
2-input XOR	4.2	2-input XNOR	3.2
Inverter	1.0		