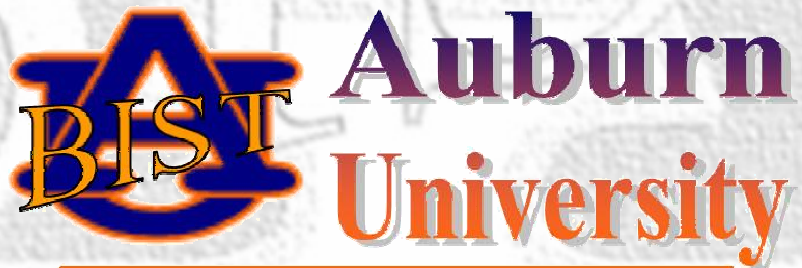


Testing Parity-Based Error Detection and Correction Circuits

Charles E. Stroud



Built-In Self-Test

Outline of Presentation

□ Testing Exclusive-OR (XOR) Gates

- ❖ Pin faults vs. gate-level faults

□ Parity Circuits

- ❖ Basic operation and design
- ❖ Testing parity trees
 - ✓ C-testable for known connections
 - ✓ Pseudo-exhaustive test set for unknown connections
 - Stuck-at & bridging fault simulation results using AUSIM

□ Hamming Circuits

- ❖ Basic operation and design
 - ✓ Use in FPGAs
- ❖ Pseudo-exhaustive test set for unknown connections
 - ✓ Stuck-at & bridging fault simulation results using AUSIM

□ Summary and Conclusions

Some Test Definitions

□ Exhaustive Test

- ❖ Apply all possible test patterns to circuit under test (CUT)

□ Pseudo-exhaustive Test

- ❖ Apply all possible test patterns to every subcircuit in CUT
 - ✓ McCluskey, Trans on Comp. '84

□ C-testable

- ❖ A circuit is C-testable if it can be tested with a constant number of test patterns, regardless of size of CUT
 - ✓ Friedman, Trans on Comp. '73

□ N -detect Test Set

- ❖ Every fault in CUT detected $\geq N$ times by N different vectors
 - ✓ Elementary logic gate, at-speed N -detect test sets ($N \geq 3$ to 5) are effective in detecting delay, bridging, and transistor faults
 - McCluskey & Tseng, ITC'00

XOR Gates

□ Elementary logic gates:

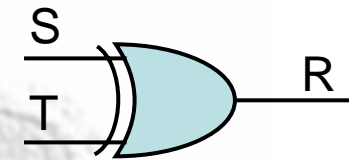
❖ AND, OR, NOT, NAND, NOR

□ XOR not considered an elementary logic gate by most designers

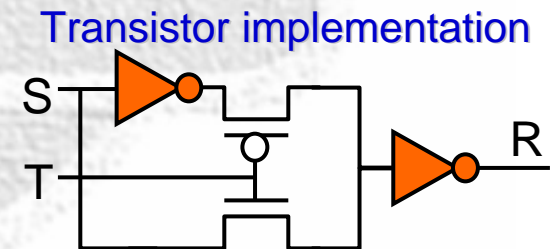
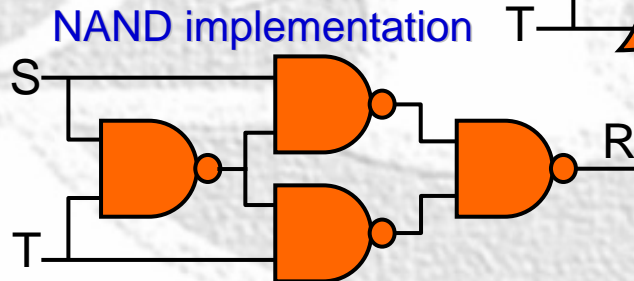
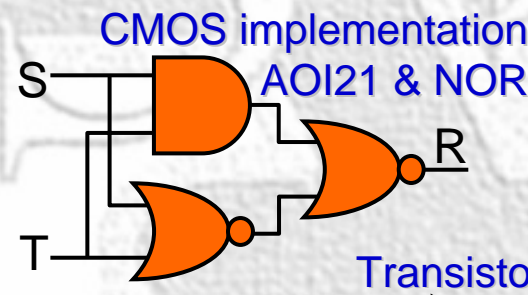
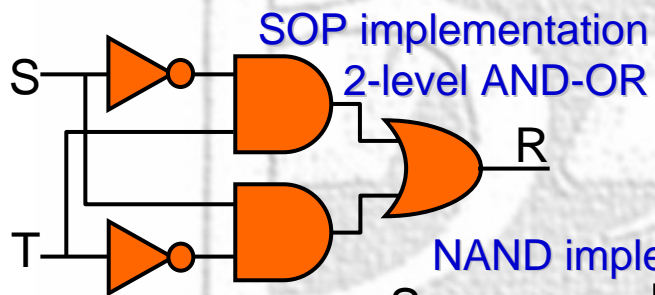
❖ Implementation requires multiple elementary logic gates

□ Note that: $S \oplus T = R$, $T \oplus R = S$, and $R \oplus S = T$

❖ Linear function



Truth Table		
S	T	R
0	0	0
0	1	1
1	0	1
1	1	0



Testing XOR Gates

Stuck-at faults

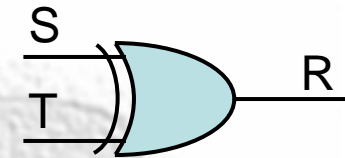
- ❖ Stuck-at-0 (sa0) and stuck-at-1 (sa1)

To detect all pin faults:

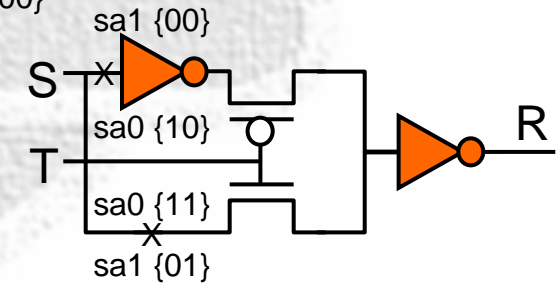
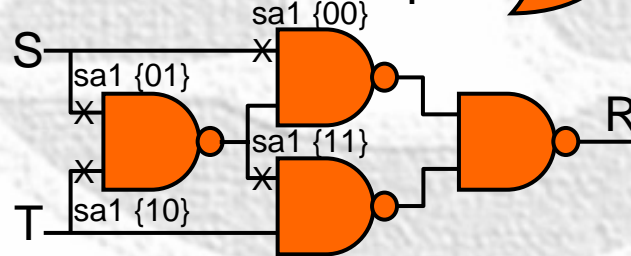
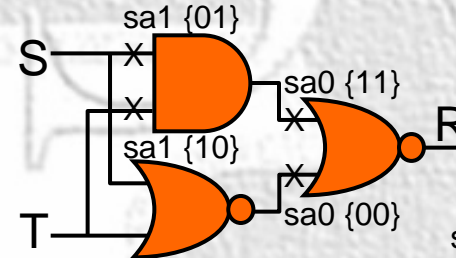
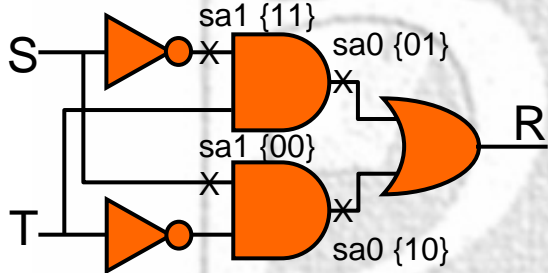
- ❖ Need 3 vectors {01, 10, and 00 or 11}
 - ✓ Mourad & McCluskey, Trans. on IE '89
- ❖ But note that any 3 vectors will work

To detect all gate-level faults:

- ❖ Need all 4 vectors



Fault-free			S		T		R	
S	T	R	sa0	sa1	sa0	sa1	sa0	sa1
0	0	0	0	1	0	1	0	1
0	1	1	1	0	0	1	0	1
1	0	1	0	1	1	0	0	1
1	1	0	1	0	1	0	0	1

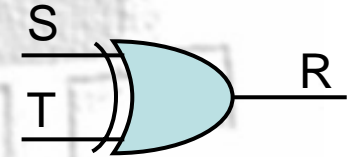


Note:

nets doubles to triples for bridging faults

Alternate View

□ **Theorem 1:** A set of test vectors that detects all single stuck-at faults on all primary inputs of a fanout-free combinational logic circuit will detect all single stuck-at faults in that circuit.



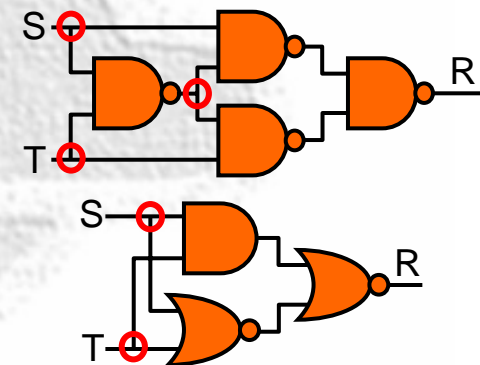
❖ No fanout in XOR when considering pin faults

□ **Theorem 2:** A set of test vectors that detects all single stuck-at faults on all primary inputs and all fanout branches of a combinational logic circuit will detect all single stuck-at faults in that circuit.

❖ 2 or 3 fanout stems in any gate-level implementation of XOR

❖ All 4 vectors needed to detect faults on fanout branches

✓ *Need exhaustive testing*



Hamming Distance

□ Distance, $d = \#$ bits different between 2 words

❖ Example, $d=3$

✓ 00110100
 ✓ 01100101

□ Used in error detection & correction codes

❖ $d =$ minimum distance between 2 valid code words

✓ Invalid code words represent error conditions

❖ $d = E + C + 1$, where $E \geq C$

✓ $E = \#$ detectable bit errors

✓ $C = \#$ correctable bit errors

❖ examples

✓ $d=1$: no detection or correction (regular data)

✓ $d=2$: 1-bit detection, no correction (parity)

✓ $d=3$: 1-bit detection & correction *or* 2-bit detection

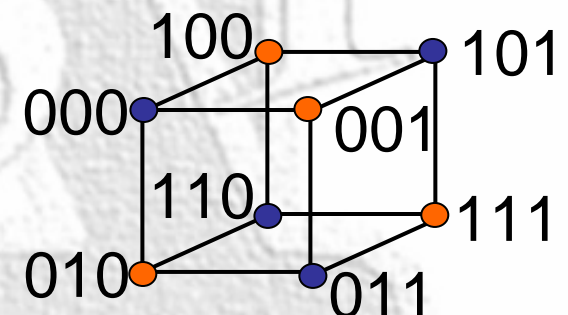
✓ $d=4$: 2-bit detection & 1-bit correction

d	E	C
1	0	0
2	1	0
3	1	1
	2	0
4	3	0
	2	1
5	4	0
	3	1
	2	2

Parity Error Detection

- ❑ Add 1 bit to create valid code words with $d=2$
- ❑ Detects single bit errors
 - ❖ Also detects all odd number bit errors
- ❑ Even parity has even # 1s in code word
 - ❖ Code word = data + parity bit
- ❑ Odd parity has odd # 1s

Data Word	Code Word	
	even parity	odd parity
00	00 0	00 1
01	01 1	01 0
10	10 1	10 0
11	11 0	11 1



3-cube for even parity

- valid code word
- invalid code word = error

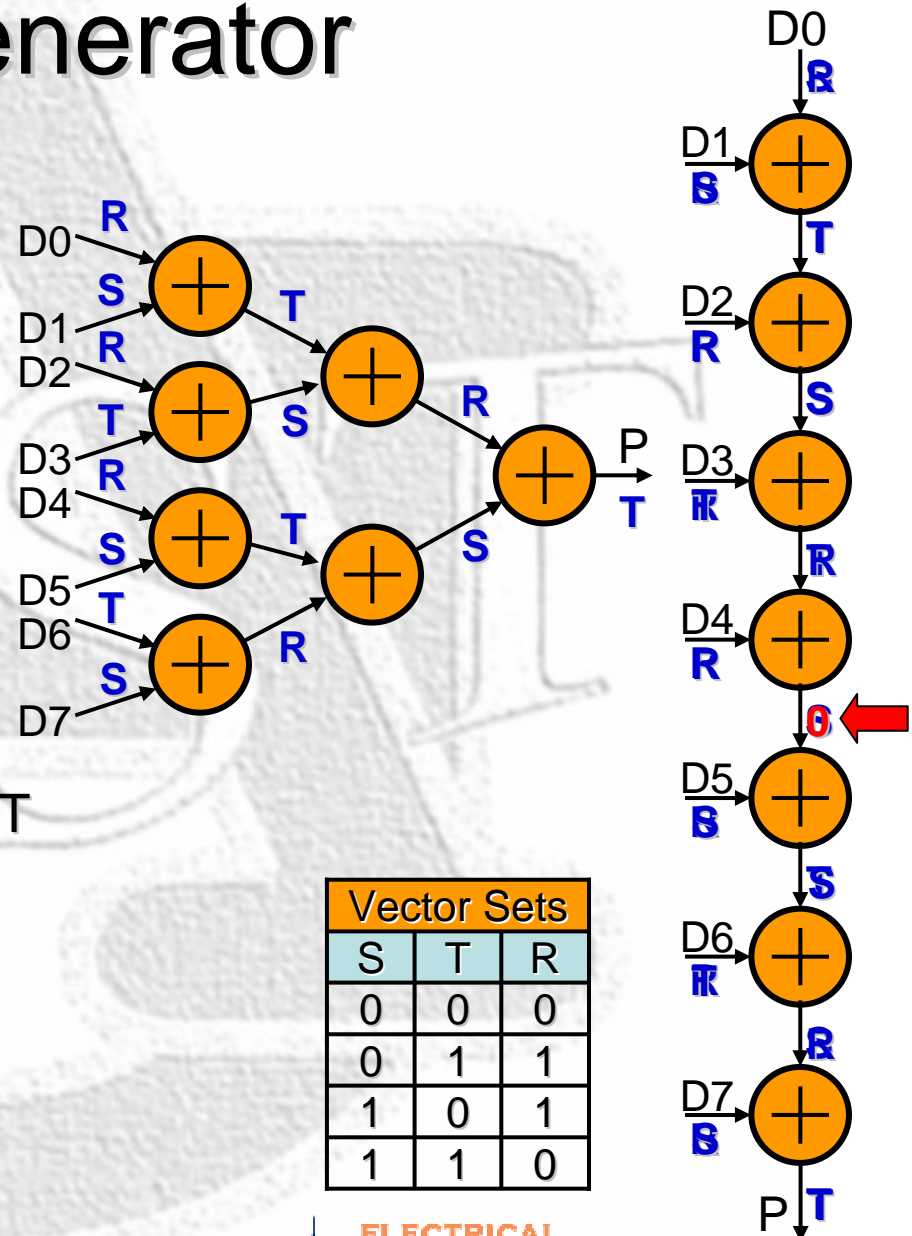
Parity Generator

❑ XOR tree to generate parity bit for N data bits

- ❖ #XOR gates = $N-1$
 - ✓ Balanced tree (#levels= $\lceil \log_2 N \rceil$)
 - ✓ Linear tree (#levels= $N-1$)
 - ✓ There are other types

❑ C-testable with 4 vectors

- ❖ All gate-level stuck-at faults
 - ✓ Mourad & McCluskey, IEEE Trans. on IE 1989
- ❖ Recall: $S \oplus T = R$, $T \oplus R = S$, & $R \oplus S = T$
- ❖ Algorithm:
 - ✓ Assign one vector set to output
 - ✓ Assign other 2 sets to inputs
 - ✓ Repeat to primary inputs
- ❖ Pseudo-exhaustive testing
- ❖ Assumes connections are known



Vector Sets		
S	T	R
0	0	0
0	1	1
1	0	1
1	1	0



Parity Generator

□ What if connections are not known?

❖ All 0s

- ✓ Applies {00} to all gates
 - Detects any sa1 pin fault

❖ Walking 1 thru 0s

- ✓ Applies {01, 10} to all gates
 - Cumulatively detects all pin faults
 - But not all gate-level faults
 - Detects all bridging faults
 - Mourad & McCluskey, TIE'89

❖ All 1s

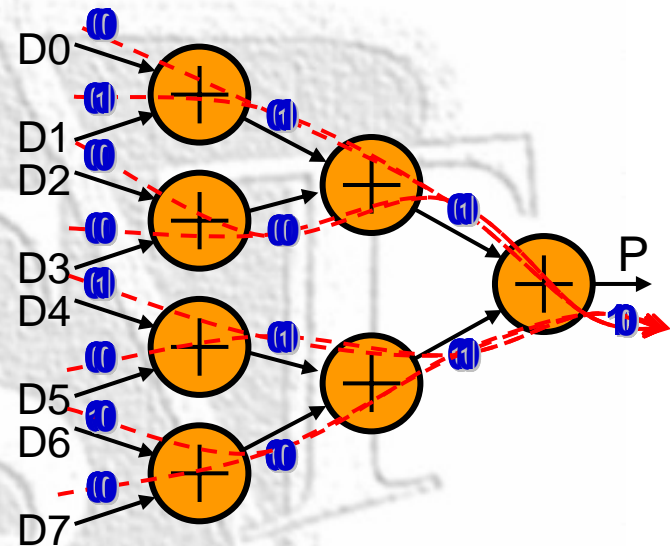
- ✓ Applies {11} to input gates
 - Cumulatively detects gate-level stuck-at faults in first level gates

❖ All combinations of two 1s in field of 0s

- ✓ Applies {00, 01, 10, 11} to all gates except output XOR gate
 - Detects all gate-level stuck-at faults in all gates except output

□ Pseudo-exhaustive test set # vectors = $\binom{N}{2} + N = \frac{N^2 + N}{2}$

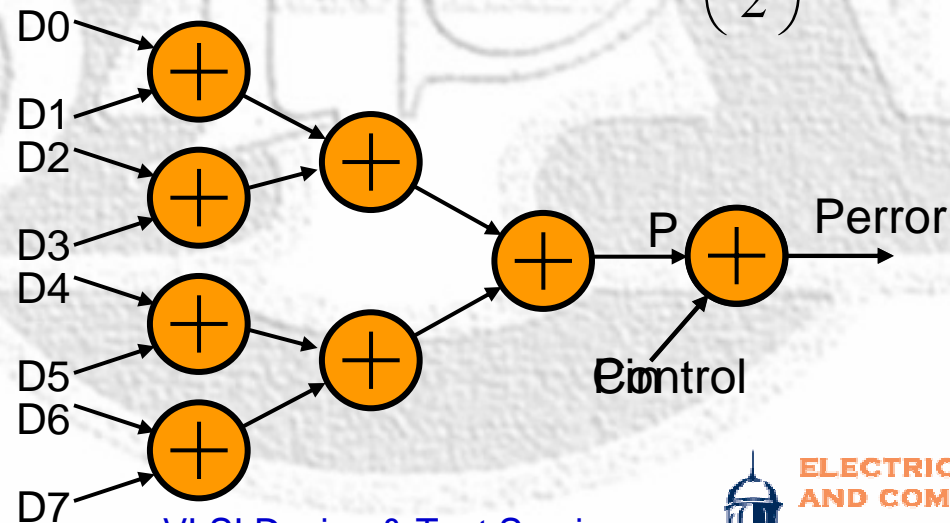
❖ Walking 1s and all combinations of two 1s in field of 0s



Parity Check

- ❑ Regenerate parity over data
- ❑ Compare regenerated parity is incoming parity
 - ❖ Mismatch indicates bit error
 - ❖ Match assumed to indicate no error
- ❑ Complete pseudo-exhaustive test set
 - ❖ Same as for generator with extra input(s) for incoming parity (and parity control)

- ❖ $N = \# \text{ data bits} + 2$ $\# \text{ vectors} = \binom{N}{2} + N = \frac{N^2 + N}{2}$



Gate-Level Fault Simulation Results

□ Example 64-bit parity tree

- ✓ 64-bit generator, 63-bit parity check, or 62-bit check w/control

- ❖ 254 collapsed pin stuck-at faults

- ❖ 504 collapsed gate-level stuck-at faults (CMOS standard cell XOR)

□ 100% fault coverage with walking 1s plus all combinations of two 1s in field of 0s

- ❖ N -detectability: $N=37$ (pin faults)

- ✓ Gate faults: $N=1$ (32 flts), $N=4$ (16 flts), $N=18$ (8 flts), $N=32$ (4 flts), $N=37$

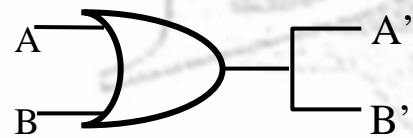
Test Pattern	# vectors	Pin faults		Gate faults	
		FC _{IND}	FC _{CUM}	FC _{IND}	FC _{CUM}
All 0s	1	50%	50%	25%	25%
Walking 1s	64	99.6%	100%	87.1%	87.5%
All 1s	1	50%	100%	25%	93.8%
Two 1s in 0s	2016	99.6%	100%	99%	100%

Bridging Faults

- ❑ Wired-AND/Wired-OR fault model (bipolar tech)
 - ❖ 1 vector {01 or 10} observing 2 outputs (A' and B'), **or**
 - ❖ Observe 1 output (A' or B') with 2 vectors {01, 10}
- ❑ Dominant fault model (more accurate for CMOS)
 - ❖ 1 vector {01 or 10} observing 2 outputs (A' and B')
 - ✓ harder to detect than wired-AND/OR (*less observable*)
 - ✓ detecting all dominant BFs \Rightarrow detects all wired-AND/OR BFs

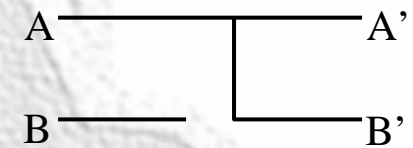


Wired-AND fault model

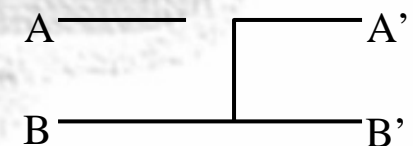


Wired-OR fault model

AB	A'B'	WAND	WOR	A _{dom} B	B _{dom} A
00	00	00	00	00	00
01	01	00	11	00	11
10	10	00	11	11	00
11	11	11	11	11	11



A dominates B model



B dominates A model

Bridging Fault Simulation Results

- Dominant bridging fault model using ordered list of nets with bridging faults on adjacent nets in list
 - ❖ 125 BFs for pin faults
 - ❖ 377 BFs for gate-level faults
 - ❖ Includes feedback BFs
 - ✓ Faults causing oscillations assumed to be detected
- 100% FC with all combinations of two 1s in field of 0s
 - ❖ 100% gate-level BF not obtained for walking 1s
 - ✓ Mourad & McCluskey, Trans on IE '89 considered only pin faults

127 pin fault nets:
 $2x(N\text{-choose-}2) = 16,002$ BFs
253 gate-level fault nets:
 $2x(N\text{-choose-}2) = 63,756$ BFs

Test Pattern	# vectors	Pin faults		Gate faults	
		FC _{IND}	FC _{CUM}	FC _{IND}	FC _{CUM}
All 0s	1	0.8%	0.8%	48.5%	48.5%
Walking 1s	64	100%	100%	82.5%	83.3%
All 1s	1	0.8%	100%	49.1%	91.8%
Two 1s in 0s	2016	98.4%	100%	100%	100%

Calculating Hamming Code

□ $H = \#$ Hamming bits

❖ $D+H+1 \leq 2^H$

✓ $D = \#$ data bits

✓ Hamming, BSTJ '50

□ $D=8$ example

❖ $H1 = D1 \oplus D2 \oplus D4 \oplus D5 \oplus D7$

❖ $H2 = D1 \oplus D3 \oplus D4 \oplus D6 \oplus D7$

❖ $H3 = D2 \oplus D3 \oplus D4 \oplus D8$

❖ $H4 = D5 \oplus D6 \oplus D7 \oplus D8$

□ Hamming distance, $d=3=E+C+1$

❖ Single bit error detection & correction (SEC)

□ Additional parity bit, $d=4=E+C+1$

❖ Parity over data & Hamming bits

❖ Double bit error detection (DED) & single bit error correction (SEC)

✓ $E=2, C=1$

C. Stroud 9/6/06

Position	1	2	3	4	5	6	7	8	9	10	11	12
Bit	H1	H2	D1	H3	D2	D3	D4	H4	D5	D6	D7	D8
Parity H1	1	0	1	0	1	0	1	0	1	0	1	0
Parity H2	0	1	1	0	0	1	1	0	0	1	1	0
Parity H3	0	0	0	1	1	1	1	0	0	0	0	1
Parity H4	0	0	0	0	0	0	0	1	1	1	1	1

Syndrome	000	001	010	011	100	101	110	111
0000	no err	H1	H2	D1	H3	D2	D3	D4
0001	H4	D5	D6	D7	D8	D9	D10	D11
0010	H5	D12	D13	D14	D15	D16	D17	D18
0011	D19	D20	D21	D22	D23	D24	D25	D26
0100	H6	D27	D28	D29	D30	D31	D32	D33
0101	D34	D35	D36	D37	D38	D39	D40	D41
0110	D42	D43	D44	D45	D46	D47	D48	D49
0111	D50	D51	D52	D53	D54	D55	D56	D57
1000	H7	D58	D59	D60	D61	D62	D63	D64

Error Type	Condition
No bit error	Hamming match, no parity error
1-bit correctable error	Hamming mismatch, parity error
2-bit error detection	Hamming mismatch, no parity error

Hamming Code Operation

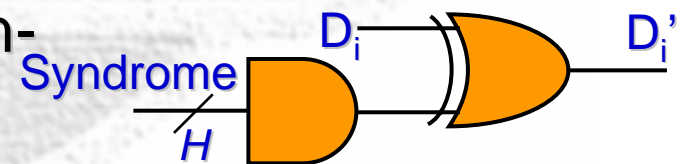
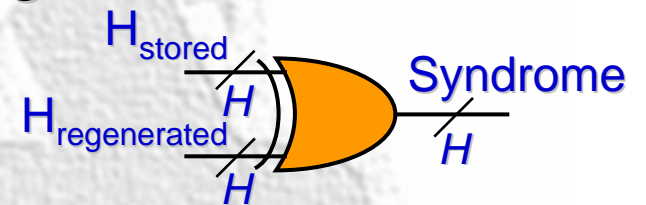
□ Example: a RAM or a hard drive

□ Input (Generate Circuit):

- ❖ Generate Hamming code for data
- ❖ Store data and Hamming bits

□ Output (Detect/Correct Circuit):

- ❖ Regenerate Hamming code for data
- ❖ Bit-wise XOR with stored Hamming bits
 - ✓ Non-zero syndrome indicates
 - Error detection
 - Bit position of error bit
 - Flip that bit to correct
- ❖ Use extra parity to determine non-correctable double bit error
 - ✓ Can disable correction circuit



Error Detection and Correction

Single bit error examples

❖ D3 is erroneous

✓ Changes H2 and H3

➤ Syndrome = 0000 110 = bit 6

❖ D6 is erroneous

✓ Changes H2 and H4

➤ Syndrome = 0001 010 = bit 10

❖ Odd number of bits change

✓ Overall parity bit error (SEC)

Position	1	2	3	4	5	6	7	8	9	10	11	12
Bit	H1	H2	D1	H3	D2	D3	D4	H4	D5	D6	D7	D8
	1	0	1	0	1	0	1	0	1	0	1	0
	0	1	1	0	0	1	1	0	0	1	1	0
	0	0	0	1	1	1	1	0	0	0	0	1
	0	0	0	0	0	0	0	1	1	1	1	1

Double bit error example

❖ D3 and D6 are erroneous

✓ Changes H3 and H4 (not H2)

➤ Syndrome = 0001 100 = bit 12

- Indicates error in D8

❖ Even number of bits change

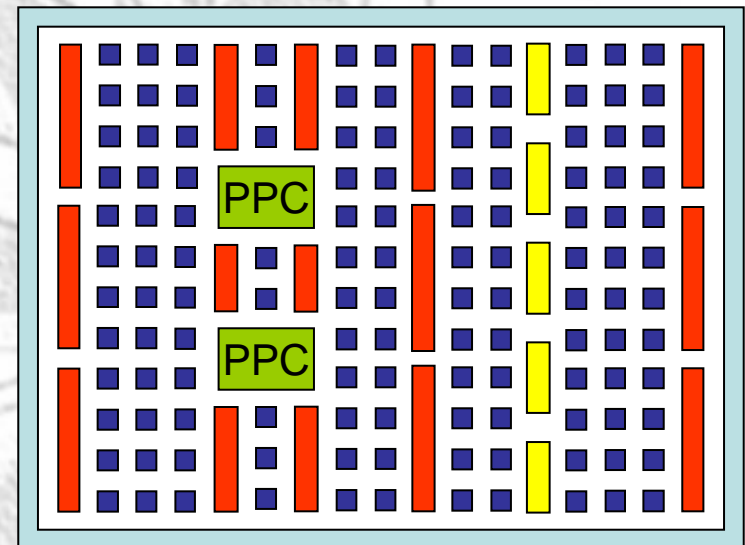
✓ No overall parity error (DED)

Syndrome	000	001	010	011	100	101	110	111
0000	no err	H1	H2	D1	H3	D2	D3	D4
0001	H4	D5	D6	D7	D8	D9	D10	D11
0010	H5	D12	D13	D14	D15	D16	D17	D18
0011	D19	D20	D21	D22	D23	D24	D25	D26
0100	H6	D27	D28	D29	D30	D31	D32	D33
0101	D34	D35	D36	D37	D38	D39	D40	D41
0110	D42	D43	D44	D45	D46	D47	D48	D49
0111	D50	D51	D52	D53	D54	D55	D56	D57
1000	H7	D58	D59	D60	D61	D62	D63	D64



Xilinx Virtex 4 FPGAs

- Contain 48 to 552 18K-bit dual-port RAMs
 - ❖ Program from 16Kx1-bit RAM to 512x36-bit RAM
 - ❖ Can operate as 24 to 276 36K-bit RAMs with ECC
 - ✓ 512x72-bit RAMs
 - ✓ Hamming code
 - ✓ 64-bit data
 - ✓ 7-bit Hamming
 - Single error correction
 - ✓ 1-bit overall parity
 - Double error detection
 - ❖ Can also operate as FIFOs
 - ✓ With ECC mode
 - ✓ Or without ECC

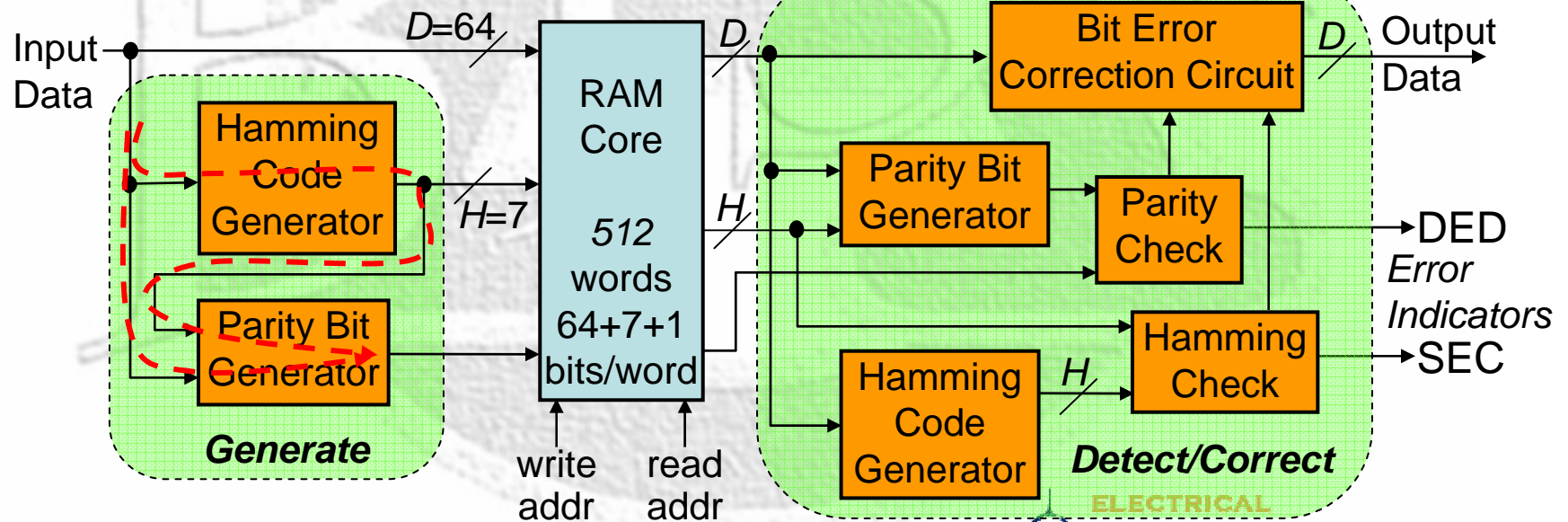


- =DSPs
- =PLBs
- =Block RAMs/FIFOs
- =I/O Buffers

Xilinx Virtex-4 ECC RAM

- ❑ Separate Hamming code generators
 - ❖ Separate write & read ports
- ❑ Reconvergent fanout in Generate circuit
- ❑ No direct observability or control of Hamming or parity bits to detect faults in FT circuit

Syndrome	000	001	010	011	100	101	110	111
0000	no err	H1	H2	D1	H3	D2	D3	D4
0001	H4	D5	D6	D7	D8	D9	D10	D11
0010	H5	D12	D13	D14	D15	D16	D17	D18
0011	D19	D20	D21	D22	D23	D24	D25	D26
0100	H6	D27	D28	D29	D30	D31	D32	D33
0101	D34	D35	D36	D37	D38	D39	D40	D41
0110	D42	D43	D44	D45	D46	D47	D48	D49
0111	D50	D51	D52	D53	D54	D55	D56	D57
1000	H7	D58	D59	D60	D61	D62	D63	D64



Testing ECC Circuitry

- **Init:** initialize RAM with vectors with Hamming bit errors
 - ❖ Then read out to test Detect/Correct circuit
 - ✓ Note: 72 inputs to Detect/Correct circuit vs. 64 inputs to Generate circuit
- Collapsed pin stuck-at faults
 - ❖ Generate circuit = 1076
 - ❖ Detect/Correct circuit = 2035
- Collapsed gate-level stuck-at faults (CMOS standard cell XOR)
 - ❖ Generate circuit = 2112
 - ❖ Detect/Correct circuit = 3359

Circuit	Vectors	# Vectors	Pin fault detection	Gate fault detection	Cum. FC
Generate	all 0s; walk 1-thru-0s	65	100%	87.7%	87.7%
	all 1s	1	50%	26.5%	93.9%
	walk two 1s-thru-0s	2016	99.9%	99.6%	100%
Detect & Correct	Output of ECC generate vectors	2082	56%	58.4%	58.4%
	<i>Init:</i> all 0s; walk 1-thru-0s; all 1s; all Hamming values w/ data=0s;	321	100%	95.2%	98.1%
	<i>Init:</i> walk two 1s-thru-0s	2556	73.5%	71.9%	100%

5 configurations of FPGA block RAM contents

C. Stroud 9/6/06

VLSI Design & Test Seminar

Testing ECC Circuitry

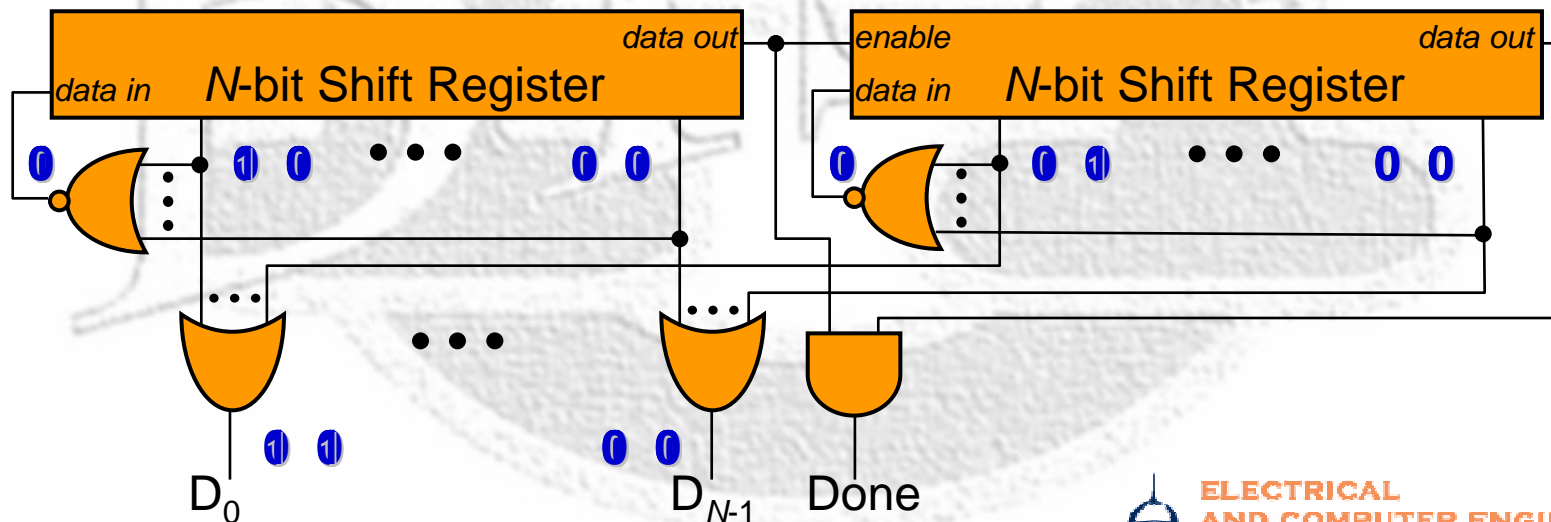
- Assuming dominant bridging fault model
 - ❖ Recall: detecting all dominant BFs detects all wired-AND/OR BFs
- Number of pin bridging faults
 - ❖ Generate circuit = 527
 - ❖ Detect/Correct circuit = 821
- Number of gate-level bridging faults
 - ❖ Generate circuit = 1583
 - ❖ Detect/Correct circuit = 2165

Note: *using ordered list of nets with pair-wise faulting of nets in order of list and applying both combinations of each net dominating the other*

Circuit	Vectors	# Vectors	Pin BF detection	Gate BF detection	Cum. FC
Generate	all 0s; walk 1-thru-0s	65	100%	83.7%	83.7%
	all 1s	1	17.5%	46.2%	91.9 %
	walk two 1s-thru-0s	2016	99.6%	99.9%	100%
Detect & Correct	Output of ECC generate vectors	2082	78.9%	79.1%	79.1%
	<i>Init:</i> all 0s; walk 1-thru-0s; all 1s; all Hamming values w/ data=0s;	321	100%	92.5%	97%
	<i>Init:</i> walk two 1s-thru-0s	2556	95.6%	85.7%	100%

Test Pattern Generator

- Use TPG for reprogrammable PLAs
 - ❖ From *Designer's Guide to BIST*
 - ❖ Two N -bit shift registers with reset implemented in PLBs
 - ❖ Generates $(N+1)^2$ vectors as shown
 - ✓ All 0s
 - ✓ Walking 1 through field of 0s
 - ✓ All combinations of two 1s in field of 0s
 - ❖ Total unique vectors = $\binom{N}{2} + N + 1 = \frac{N^2 + N + 2}{2}$



Summary and Conclusions

□ Parity error detection circuits

❖ Previous algorithms for 100% fault detection

- ✓ **But** only for known XOR connections in parity tree

- ✓ **And** only detects all pin-level bridging faults

 - not gate-level bridging faults

❖ Pseudo-exhaustive test set:

- ✓ Walk a 1 through a field of 0s , **and**

- ✓ All combinations of two 1s in a field of 0s

 - Detects all gate-level stuck-at and bridging faults in parity tree

 - Independent of XOR connections



Summary and Conclusions

□ Hamming code error correction circuits

❖ **Problem:** detecting faults in circuit designed to tolerate faults

❖ **Solution:** initialize RAM with Hamming error conditions

✓ Same pseudo-exhaustive test set for parity tree

➤ Detects all gate & bridging faults in Hamming code generator circuit

✓ Add all Hamming bit values with data bits = all 0s

➤ Cumulatively detects all gate & bridging faults in error detect/correct correction circuit

❖ **Question:** Is there a formal proof?