

Reliable Design

3 Classes of design

1. Self checking systems -
faults automatically detected
by a subsystem (checker)
2. Fault-tolerant circuits -
faults do not cause system
malfunction
3. Easily testable systems -
test generation problem
is simplified.



If only $k < 2^g$ output configurations can occur during normal operation, the occurrence of any of the $2^g - k$ unallowable configurations indicates a malfunction.

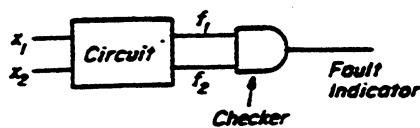
A hardware checker can be used to automatically detect such faults.

Example

(4.54)

x_1	x_2	f_1	f_2
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	0

(a)



(b)

Figure 5.1 Automatic Fault Checking for Example 5.1

Output $f_1=f_2=1$ never occurs.
Checker detects this situation
and generates fault indicator = 1
when it occurs.

k normal outputs - (valid) code words
 $2^k - q$ unallowable outputs
 $\Rightarrow$ invalid or non-code
words.

Error Detecting and Correcting Codes (4.55)

Classification - ability to detect errors

e (bit) error detecting

Any fault affecting $\leq e$ bits can be detected.

e (bit) error correcting - any fault affecting $\leq e$ bits can be corrected (set of faulty bits can be determined)

Hamming distance^(d) of a code -
minimum number of bits
in which any two code words
differ

Relation between Hamming distance (4.56) and error detecting / corrected capability of code.

d	capability
1	none
2	1-error detection, 0-error correction
3	2-error detection, 1-error correction
$\vdots$	
$e + 1$	e -error detection, $\left\lceil \frac{e}{2} \right\rceil$ -error correction
$\vdots$	
$2e + 1$	$2e$ -error detection, e -error correction

Figure 5.2 Capability of a Code with Distance d

* Smallest integer $\geq e/2$

Codes must use extra bits called check bits.

Simplest is parity check code (only one check bit).

Odd or even parity
Single bit error detecting

Example

x_1	x_2	f_1	f_2	y_e	y_o
0	0	0	1	1	0
0	1	1	0	1	0
1	0	1	0	1	0
1	1	0	0	0	1

Figure 5.3

y_e is even parity check bit
 y_o is odd parity check bit

f_1, f_2 are called information bits

$$y_o = y_e.$$

Generalized class of parity check codes
 q information bits
 c check bits

$$2^c \geq q + c + 1 \quad - \quad \text{single bit error correcting}$$

q	c
1	2
4	3
11	4
26	5
57	6
120	7

With use of codes can design circuits which automatically detect or correct errors (4.58)

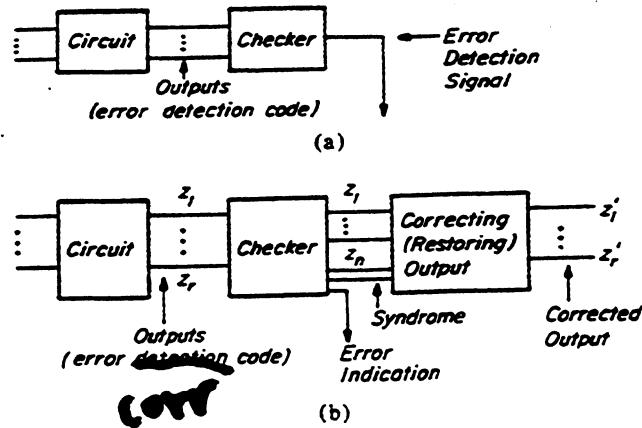


Figure 5.5 (a) Error Detecting Circuit (b) Error Correcting Circuit

Different circuits require different type of codes.

Transmission $\rightarrow$ parity check code may be adequate

Arithmetic $\rightarrow$ parity check code may not be adequate

(Cannot predict parity check bit of $A+B$ $P(A+B)$ from parity of A and parity of B $P(A), P(B)$.)

$$\begin{array}{rcl}
 A & = & 0001 \\
 B_1 & = & 0101 \\
 \hline
 A + B_1 & = & 0110
 \end{array}
 \qquad
 \begin{array}{rcl}
 A & = & 0001 \\
 B_2 & = & 0011 \\
 \hline
 A + B_2 & = & 0100
 \end{array}$$

Check bit of $A = C(A) = 1$, $C(B_1) = C(B_2) = 0$, $C(A + B_1) = 0$, $C(A + B_2) = 1$ for even parity

Figure 5.6 Addition Does Not Preserve Parity

Thus parity check bit for $A+B$ must be recomputed after addition.

Residue codes - for arithmetic operations (addition, multiplication), ~~parity~~ check bits of result can be determined from check bits of operands $\Rightarrow$ independent checking

Definition of Residue Code

4.60

Rightmost p bits are check bits

information p check
bits bits

Define number
 N

Define number
 C

$$C = N \text{ modulo } m$$

m is a parameter called
the residue of the code

$$p = \lceil \log_2 m \rceil$$

Example

Inform bits check bits

I_2	I_1	I_0	N	C	C_2	C_1
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	2	2	1	0
0	1	1	3	0	0	0
1	0	0	4	1	0	1
1	0	1	5	2	1	0
1	1	0	6	0	0	0
1	1	1	7	1	0	1

Figure 5.7 A 3-Bit Residue Code with $m = 3$

$$C = N \bmod 3$$

$$\text{Residue} = 3$$

4.62

Theorem 5.1: Let $\{a_i\}$ be a set of operands with check bits $C^i = (a_i) \bmod m$. The residue of the sum is $(\sum a_i) \bmod m$ and the residue of the product is $(\prod a_i) \bmod m$. Then

- (a) $(\sum a_i) \bmod m = (\sum (a_i \bmod m)) \bmod m$. (Check bits of sum equal sum of check bits modulo m .)
- (b) $(\prod a_i) \bmod m = (\prod (a_i \bmod m)) \bmod m$. (Check bits of product equal product of check bits modulo m .)

Proof: Let $a_i = k_{i1}m + k_{i2}$ where $0 \leq k_{i2} < m$.

- (a) Then $(\sum a_i) \bmod m = (\sum (k_{i1}m + k_{i2})) \bmod m = (\sum k_{i2}) \bmod m = (\sum (a_i \bmod m)) \bmod m$.
- (b) Then $(\prod a_i) \bmod m = (\prod (k_{i1}m + k_{i2})) \bmod m = (\prod k_{i2}) \bmod m = (\prod (a_i \bmod m)) \bmod m$. $\square$

Use of Residue Codes to Check Addition

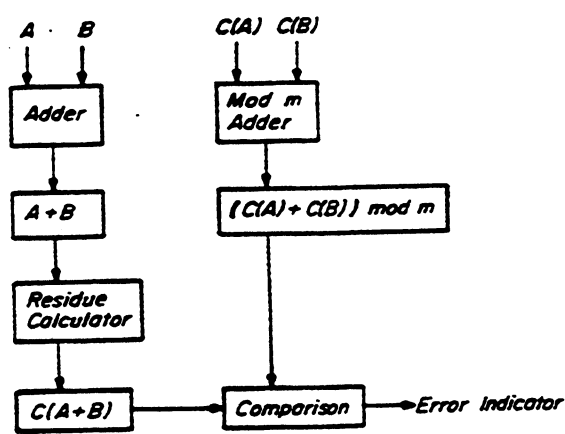


Figure 5.8

Theorem 5.2: In a residue code with m odd, all single bit errors are detected.

Proof: We must consider two cases.

Case 1: The single erroneous bit is an information bit. Then $N' = N \pm 2^i$ if bit i of the information segment is in error. Since there is just a single error $C = C'$. Thus $C' = (N') \bmod m$ if and only if $(N') \bmod m = (N) \bmod m$. This implies that $(N \pm 2^i) \bmod m = (N) \bmod m$. But $(N \pm 2^i) \bmod m = ((N) \bmod m \pm (2^i) \bmod m) \bmod m$. For m odd, $(2^i) \bmod m \neq 0$ and hence the error is detected.

Case 2: The single erroneous bit is a check bit. Then $N = N'$ and $C' = (C \pm 2^i) \bmod m$ where bit i of the check segment is in error. The error is detected unless $C' = C$. This implies that $C = (C \pm 2^i) \bmod m$ which is impossible for m odd and hence all single errors are detected. $\square$

m odd $\rightarrow$ some single bit errors may be indistinguishable

m even $\rightarrow$ some single bit errors may be undetectable

Example

$m=3$ 8 4 3 2 1
 1 1 0 | 0 0 code word 1

 check bits

error in bit 4

Results in 1 0 0 | 0 0

error is detected since
 $4 \bmod 3 \neq 0$

0 0 0 | 0 0 code word 2

error in bit 5

1 0 0 | 0 0

is detected.

However the two cases cannot be distinguished since result of single bit error is the same

4.67

Let $m = \begin{matrix} 2 & 1 \\ 4 & 3 & 2 & 1 \\ 1 & 1 & 0 & 1 \end{matrix}$ code word
check bit

error in bit 3

results in 10010

This is a valid code word

since $4 \bmod 2 = 0$

error is not detected

$\Rightarrow m$ must be odd to
detect all single bit
errors

Other types of errors

Unidirectional errors - all erroneous bits have same value

Adjacent bit errors - all bits affected by an error form a connected string of bits

Other Codes

R/m (R out of m) code - m bit words, each code word has exactly R 1's.

(Will detect all unidirectional errors)

Modified residue check code -

$m-1$ check bits, defined so total # of 1-bits in a code word is a multiple of m .

(Detects all unidirectional errors affecting $\leq m$ bits).

Berger Code

k inf bits, $\lceil \log_2 (k+1) \rceil$ check bits

Checking Circuits

Advantages of Hardware checking circuits

- (1) Intermittent faults are detected.
- (2) Errors are immediately detected upon occurrence. This prevents corruption of data.
- (3) The distribution of checkers throughout a digital system provides a good measure of the location of a fault by the location at which it is detected.
- (4) Software diagnostic program is eliminated, or at least simplified.

Disadvantages of Hardware Checking Circuits

- (1) More hardware is required, including a hardware checker.
- (2) Additional hardware must be checked or tested (*checking the checker problem*). In general, some faults cannot be automatically detected. Faults in this *hardcore* must still be tested by a diagnostic program.

Fault Tolerant Design

Redundancy techniques used to mask faults, system remains operational.
Used when repair is impossible or unfeasible.

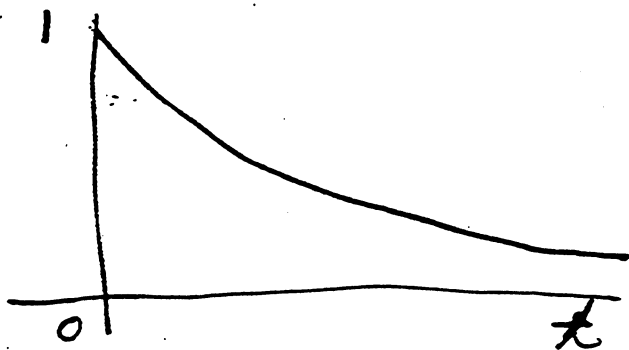
Goal: Increase expected ~~value~~ life of system (Mean time before failure)

$$MTBF = \int_0^{\infty} -t dR(t)$$

$R(t)$ = probability of survival until time t

$$MTBF = 1/\lambda \text{ (a constant)}$$

$$\Rightarrow R(t) = e^{-\lambda t}$$



Triplicated Modular Redundancy (TMR)

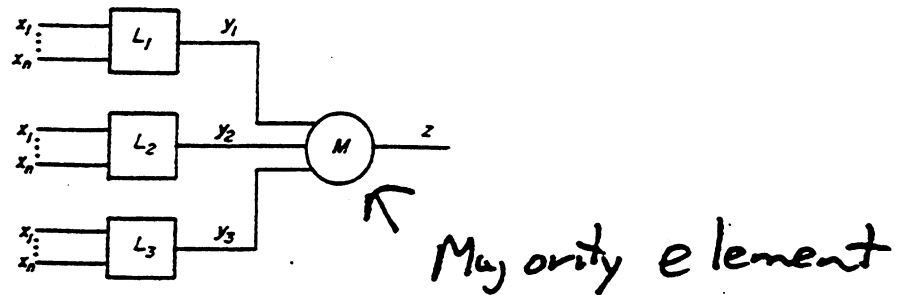


Figure 5.20 Basic TMR System

Based on error correcting code

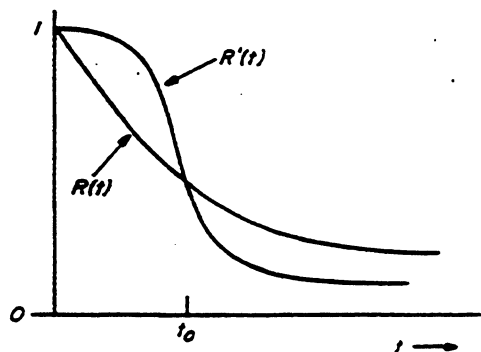
$$\begin{matrix} 0 & 0 & 0 \\ 1 & 1 & 1 \end{matrix} > \text{distance } 3$$

Assuming probability of failure of M is very small relative to L_1, L_2, L_3

$$R'(t) = \underbrace{e^{-\lambda t} e^{-\lambda t} e^{-\lambda t}}_{\text{Probability of all three of the subsystems surviving}} + \underbrace{3e^{-2\lambda t} (1 - e^{-\lambda t})}_{\text{Probability of any two of the subsystems surviving}}$$

$$= 3e^{-2\lambda t} - 2e^{-3\lambda t}$$

Two of three L_i must fail for the same input for error to occur.

Figure 5.21 Graph of Functions $R(t)$ and $R'(t)$

To find t_0

$$3e^{-2\lambda t_0} - 2e^{-3\lambda t_0} - e^{-\lambda t_0} = 0$$

$$\Rightarrow t_0 \approx .7/\lambda$$

$$MTBF = \frac{5}{6\lambda}$$

(less than that of single system)

$\Rightarrow$ increased reliability over a relatively short period of time.

4.73

TMR Concept can also be applied at the subsystem level

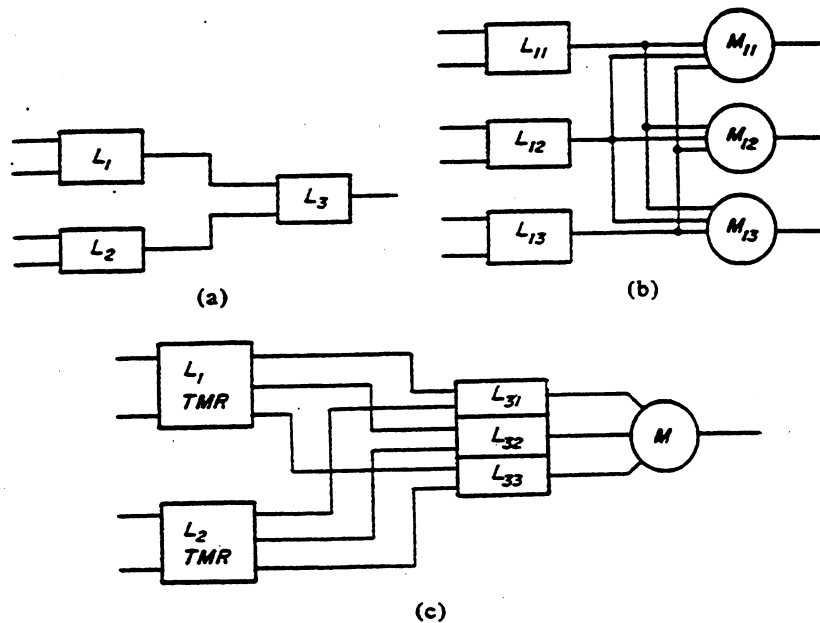


Figure 5.22 (a) A System Containing Three Modules (b) A TMR Subsystem for L_1 (c) System Designed from TMR Subsystems

This is called multiplexing.

Will also correct errors in subsystem majority gates.

Will correct multiple errors occurring in different subsystems

N Modular Redundancy

Generalization to N ~~sets~~ copies of system, N odd > 3

$N = 5, 7, 9, \dots$

Failure does not occur until

$S \geq (N+1)/2$ modules fail.

Static Redundancy

Basic System Configuration Never Changes. All copies of system active. System Reliability decreases with age.

Dynamic Redundancy

System reconfigures after each fault. System reliability fairly constant with age.

Hybrid Redundancy

Example of dynamic redundancy

3 Active Modules

$N-3$ Standby Modules (Inactive)

Detected error in an active module causes it to be automatically switched to inactive status and replaced in active status by one of the standby modules.

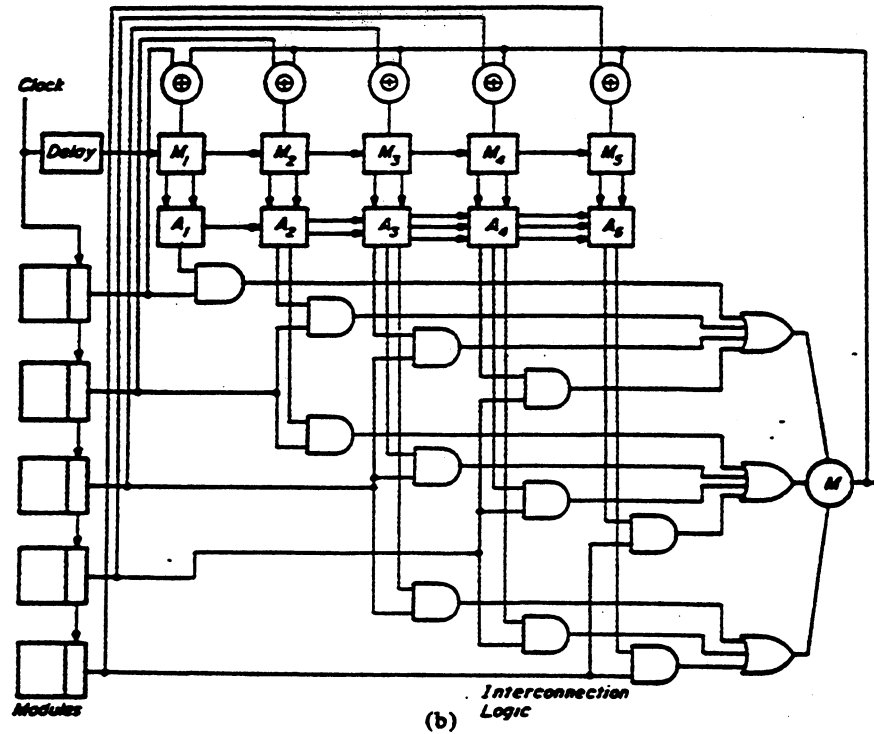
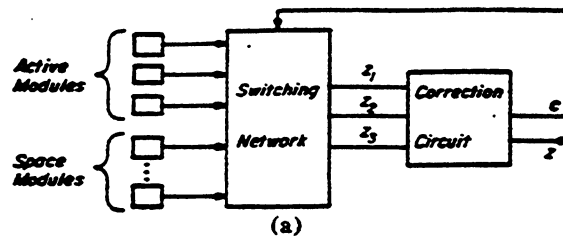


Figure 5.23

If no fault occurs in switching network system error doesn't occur until < 2 fault free copies exist.

Faults in switching network have two types of effect

1. Some good (fault-free) module cannot be switched in (to active status)

OR

2. Some faulty active status module cannot be switched out (to inactive status).

Type 1 $\Rightarrow$ equivalent to reduction of one standby module
 $N \Rightarrow N-1$

Type 2 $\Rightarrow$ system with one (permanently) faulty active module.
Next fault makes error correction impossible.

One type of fault much more serious than the other. Design system to prevent serious fault.

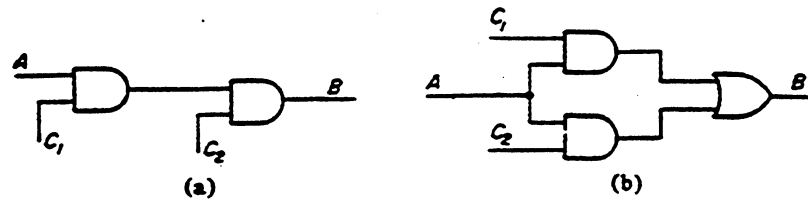


Figure 5.24 (a) Fail safe switching out circuit and
(b) fail safe switching in circuit

C_1 & C_2 independent control signals.

This is an example of fail safe design

Used when one type of fault much more costly than another.

Example

OR Intersection with two red lights
Intersection with two green lights

Fail Safe Design

(4.79)

A system is 0 fail safe if no single fault can produce an incorrect 0.

(1 fail safe if no single fault can produce an incorrect 1)

More realistically probability of incorrect 0 very very low.

Use error detecting code

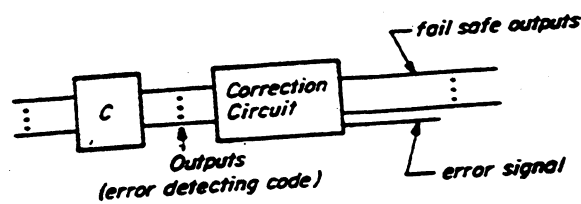


Figure 5.25 Fail Safe System

Example

4.80

x_1	x_2	x_3	z_1	z_2	z_3
0	0	0	0	1	1
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	1	0	1
1	0	0	0	1	1
1	0	1	1	1	0
1	1	0	0	0	0
1	1	1	1	0	1

(a)

Given functions z_1, z_2 define parity check bit z_3 for single bit error detection.

Want z_1 to be 0 fail safe
 Want z_2 to be 1 fail safe

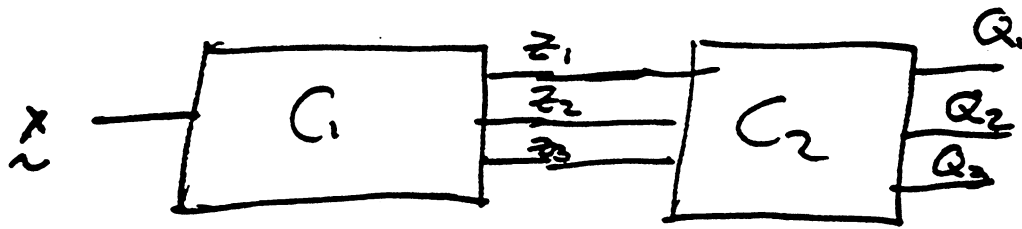
z_1	z_2	z_3	Q_1	Q_2	Q_3
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	0	1
1	0	1	1	0	0
1	1	0	1	1	0
1	1	1	1	1	1

(b)

$\begin{cases} Q_3 = 0 \text{ for even parity} \\ Q_3 = 1 \text{ for odd parity} \end{cases}$
 → error detection

Define $Q_1 = z_1$ for even parity
 $Q_1 = 0$ for odd parity

$Q_2 = z_2$ for even parity, $Q_2 = 1$ for odd parity



Must design C_1 so that for any input, any fault affects at most one output value.

System combining redundancy (4.82) and testing

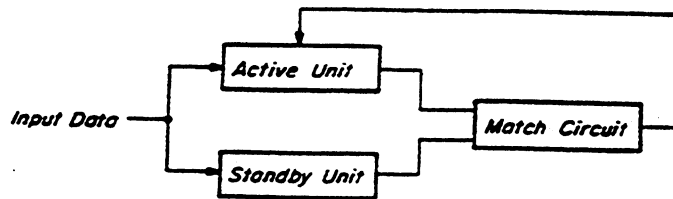


Figure 5.27

Mismatch leads to attempt to diagnose error using software diagnostic program. Successful diagnosis leads to attempt to repair faulty unit.

$$1/\lambda_1 \Rightarrow \text{MTBF}$$

$$1/\lambda_2 \Rightarrow \text{MTBR (Mean Time Before Repair)}$$

$P = \text{coverage} \Rightarrow \% \text{ of faults successfully diagnosed.}$

4.83

- S_1 State representing two fault-free modules
- S_2 State representing one fault-free module
- S_3 State representing both modules inoperative (faulty)

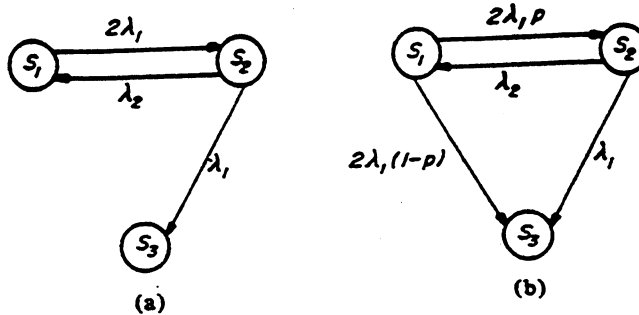


Figure 5.28 (a) Idealized Model of System (b) Model of System with Coverage p

(4.89)

$$MTBF \approx \frac{\lambda_2}{2\lambda_1^2} \left(1 + \frac{\lambda_2}{\lambda_1} (1-p) \right)$$

Assuming $1/\lambda_1 = 3$ months
 $1/\lambda_2 = 2$ hours

p	MTBF (years)
1.0	100
.995	20
.99	10
.95	2.5
.9	1

Imperfect coverage is the dominant factor in system failure when

$$1-p > \lambda_1/\lambda_2$$

Design to Simplify Testing and Test Generation

(4.88)

Observation Points

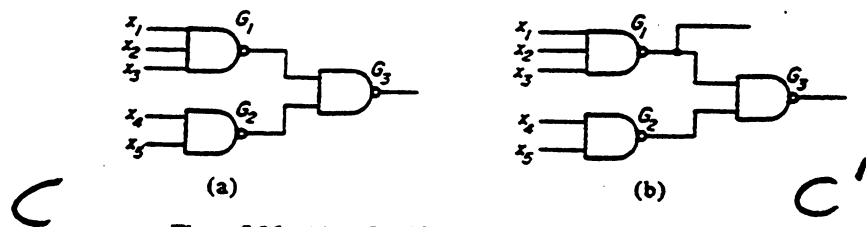


Figure 5.36 (a) A Combinational Circuit (b) The Same Circuit with Observation Point Added

C Requires 5 tests to detect all single stuck faults
 C' Requires 4 tests to detect all single stuck faults

In C to detect x_2 s-a-1

$$\Rightarrow T = x_1 \bar{x}_2 x_3 (\bar{x}_4 + \bar{x}_5)$$

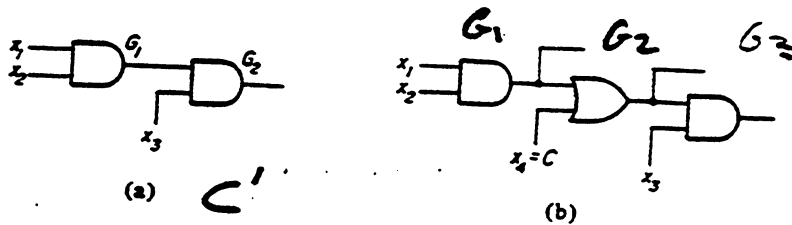
In C' to detect x_2 s-a-1

$$\Rightarrow T = x_1 \bar{x}_2 x_3$$

Test $(1, 0, 1, 1, 1)$ detects x_2 s-a-1 in C' as well as x_4, x_5 s-a-0.

Control Points

4.86



C' Requires 4 tests to detect all single stuck faults

Adding observation point at output of $G_1 \Rightarrow$ 4 tests still required.

Adding Control Signal $C + 2$
Observation points $\Rightarrow$ 3 tests sufficient

	$x_1 x_2 x_3 x_4$	$x_1 x_2 x_3 x_4$	$x_1 x_2 x_3 x_4$
G_1	$(0, 1, -, -)$ $(1, 0, -, -)$ $(1, 1, -, -)$	G_2 $(1, 1, -, 0)$ $(0, 1, -, 0)$ $(1, 0, -, 1)$	G_3 $(1, 1, 1, -)$ $(-, -, 0, 1)$ $(0, 1, 1, 0)$

Combined $(0, 1, 1, 0), (1, 0, 0, 1), (1, 1, 1, 0)$

Application to Sequential Circuits

(4.87)

Design with Embedded Shift Registers Level Scan Sensitive Design LSSD

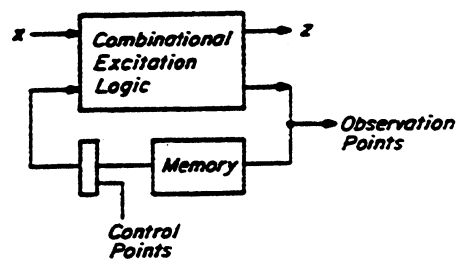


Figure 5.41 Sequential Circuit with Observable and Controllable Feedback Paths

Use Control signal C to shift from normal operation ($C=0$) to testing mode ($C=1$). When $C=1$ FF's interconnect to form a shift register to make state initialization easy.

SR Flip-Flops

4.88

$$S_1 = \bar{c}f_1(x,y) + cx_1$$

$$R_1 = \bar{c}g_1(x,y) + \bar{c}\bar{x}_1$$

$$S_i = \bar{c}f_i(x,y) + cy_{i-1}$$

$$R_i = \bar{c}g_i(x,y) + c\bar{y}_{i-1}$$

$$z_1 = \bar{c}h_1(x,y) + cy_n$$

$$z_2 = \bar{c}h_2(x,y) + c\bar{y}_n$$

} for all $i, 2 \leq i \leq n$

When $C=1$ $X_1 \rightarrow FF_1$

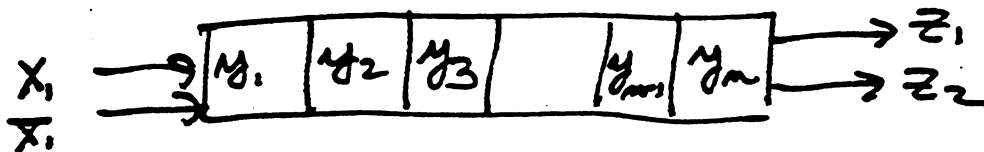
$y_1 \rightarrow y_2$

$y_2 \rightarrow y_3$

$y_{n-1} \rightarrow y_n$

$y_n \rightarrow (z_1, z_2)$

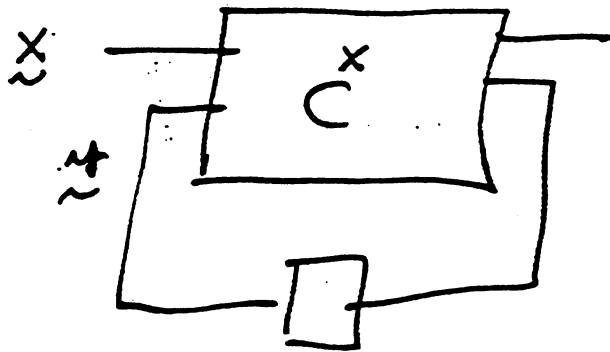
Shift Register



When $C=0 \Rightarrow$ Normal behavior

For each fault in C test pattern consists of state y and input x

4.8.9



Procedure 5.1 (Testing embedded shift register circuit):

- (1) Set $c = 1$ to switch circuit to shift register mode.
- (2) Check operation of shift register.
- (3) Select a test pattern and set shift register to desired initial state by sequentially shifting in appropriate inputs.
- (4) Set $c = 0$ to return to normal mode.
- (5) Apply test pattern input to generate incorrect state (or output) by clocking the flip-flops.
- (6) Set $c = 1$ to return to shift register mode.
- (7) Shift out final state to detect fault.
- (8) Repeat (3)-(7) for all other test patterns. $\square$