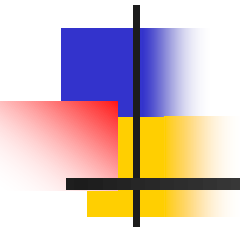


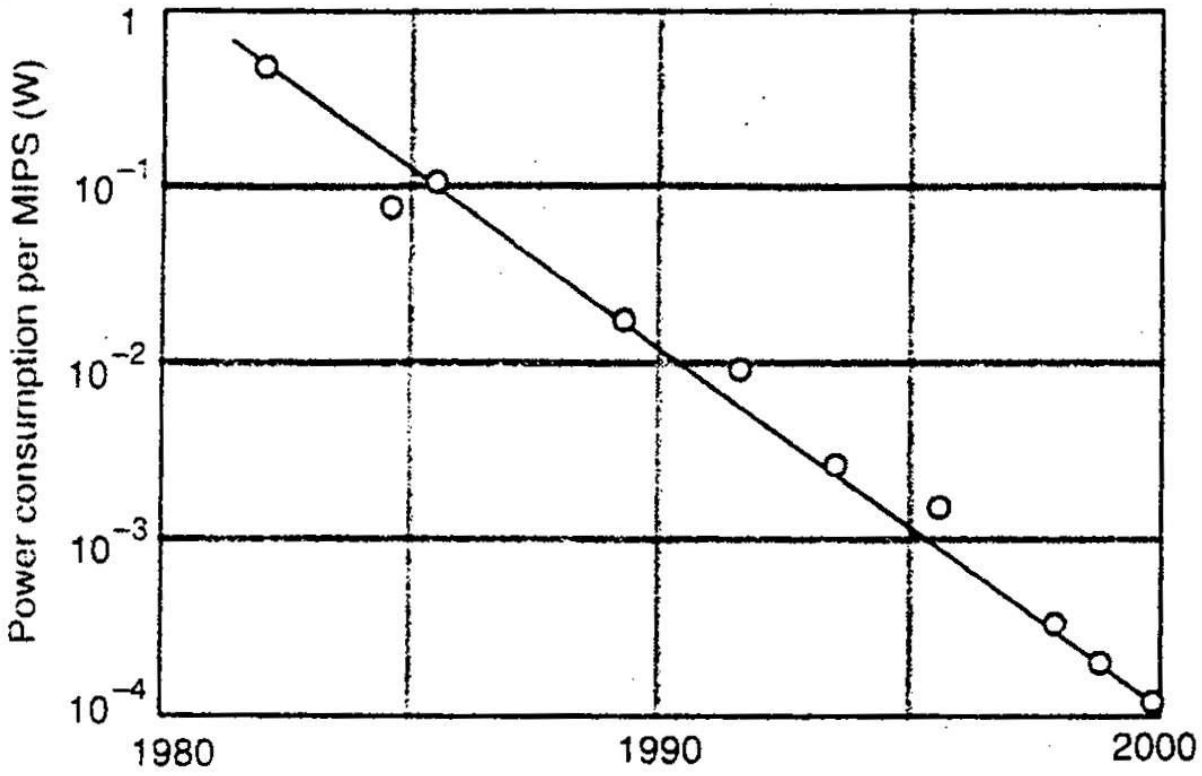
數位IC設計

Low-Power Design

Low-Power Design

General digital system：**Speed** and **Cost**

Portable Devices：***Power consumption*** , Speed and Cost



Power consumption trend in DSPs [Raba98].

Power Consumption

The average power consumption in CMOS：

$$P_{avg} = \alpha \ f \ C V^2$$

V : **Supply voltage**

C : **Capacitance**

f : **Clock frequency**

α : **Average number of 0-to-1 transitions**

Power consumption of a 32-bit bus operating 5V and 100MHz ?

If a capacitance of 30 pF per per and α=0.2

$$P_{avg} = \alpha \ f \ C V^2 = 0.2 \times (10^8)(32 \times 30 \times 10^{-12})5^2$$

Reduction of Switching Activity (1/2)

$$P_{avg} = \alpha \ f \ C \ V^2$$

It is not easy to reduce C and V (speed penalty).

Reduction the switching activity α

1. 2's complement number ⇒ Sign-magnitude number

2. General code counter ⇒ Gray code counter

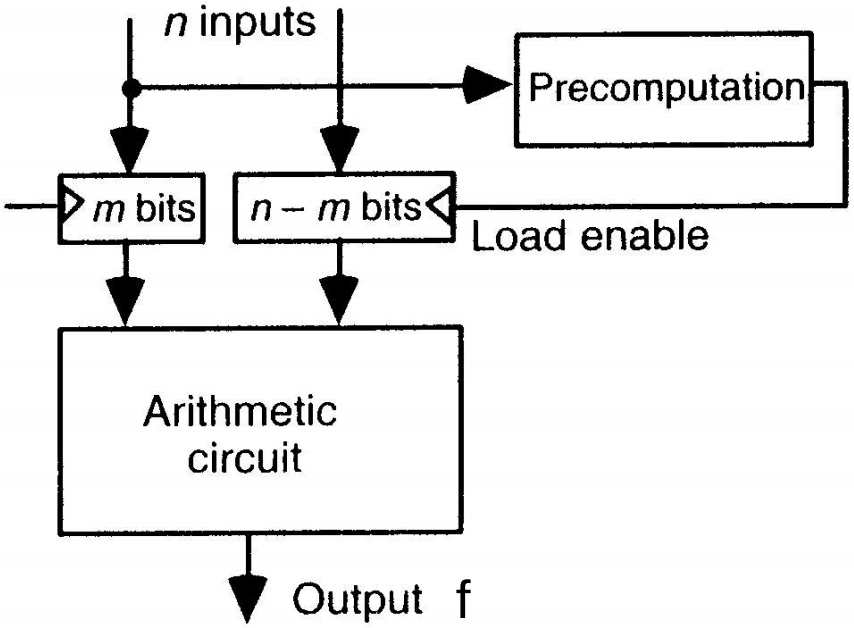
$$b_0 b_1 b_2 ... b_n$$
$$g_0 = b_0; \ g_k = b_k \oplus b_{k-1}$$

3. Reordering of operations

separate operations into two groups of positive and negative values respectively

Reduction of Switching Activity (2/2)

4. Precomputation



output *f* can be determined by using *m* bits of the *n* bits

5. Algorithm optimization

Reduction of Power Waste (1/5)

Reduction of power waste

1. Reduce the number of arithmetic operations

$ab+bc \Rightarrow a(b+c)$ **less multiplication**

$15a \Rightarrow 16a-a$ **less multiplication, why ? shifter**

$(a+b\,j)(c+d\,j)=(ac-bd)+(ad+bc)\,j$

4 multipliers and 3 adders

$\Rightarrow (a+b\,j)(c+d\,j)=[c(a+b)-b(c+d)]+[c(a+b)-a(c-d)]\,j$

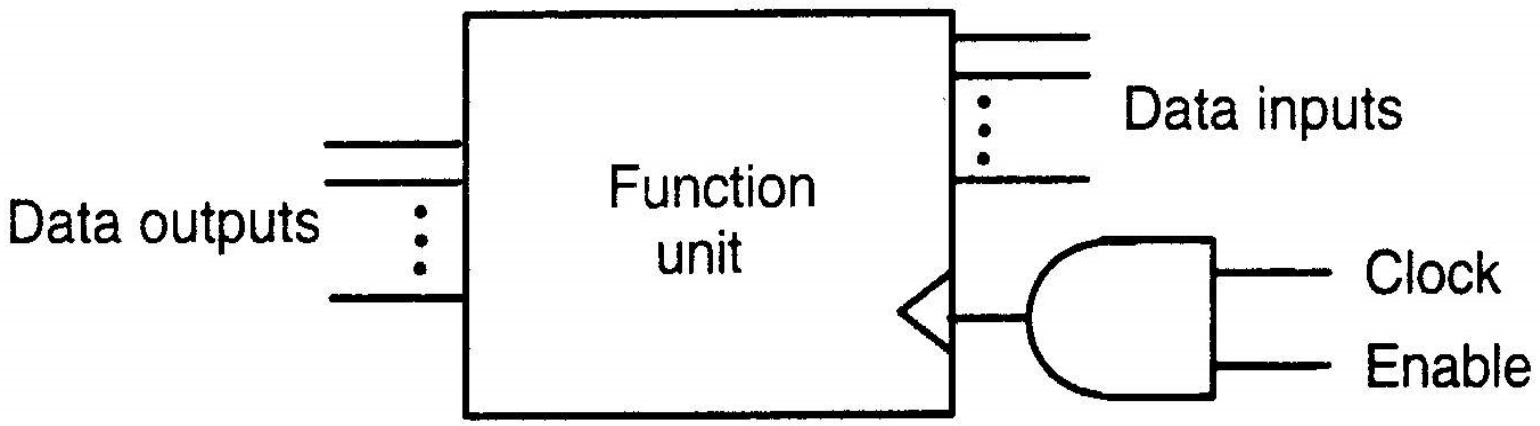
3 multipliers and 6 adders

less power but more delay

Please draw their flow graph

Reduction of Power Waste (2/5)

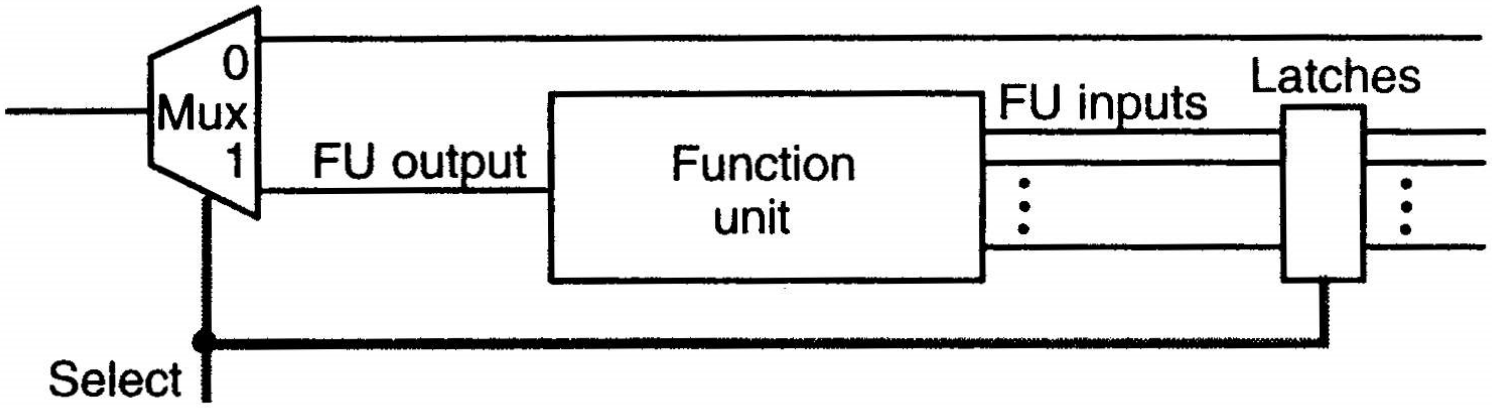
2. Disable or turn off those unused units



Saving power through clock gating.

Reduction of Power Waste (3/5)

3. Guarded evaluation

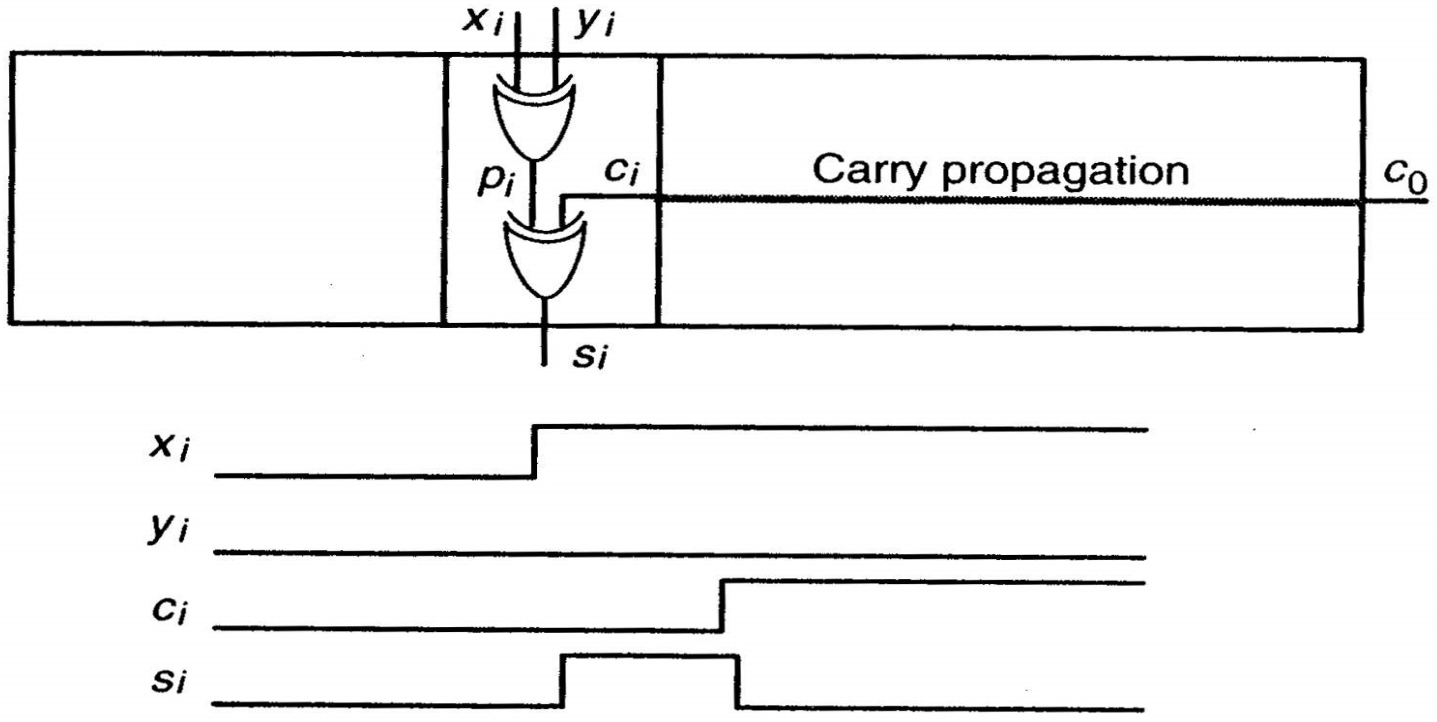


Saving power via guarded evaluation.

When select signals is high, the latches become transparent; Otherwise, the earlier inputs to function unit are preserved.

Reduction of Power Waste (4/5)

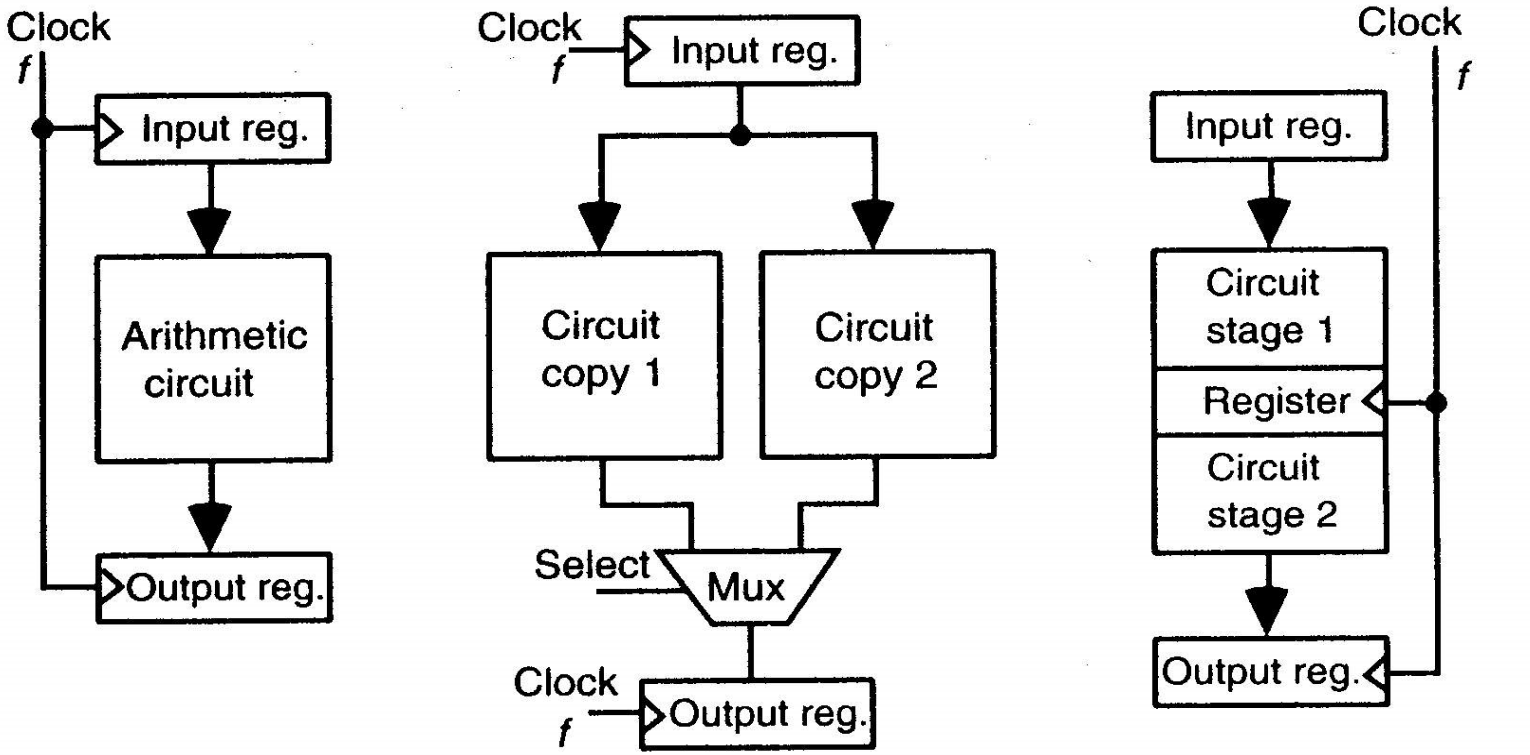
4. Eliminate the glitch



Example of glitching in a ripple-carry adder.

Reduction of Power Waste (5/5)

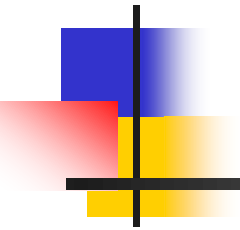
5. Power reduction via parallelism and pipelining



Frequency = f
Capacitance = C
Voltage = V
Power = P
(a)

Frequency = $0.5f$
Capacitance = $2.2C$
Voltage = $0.6V$
Power = $0.396P$
(b)

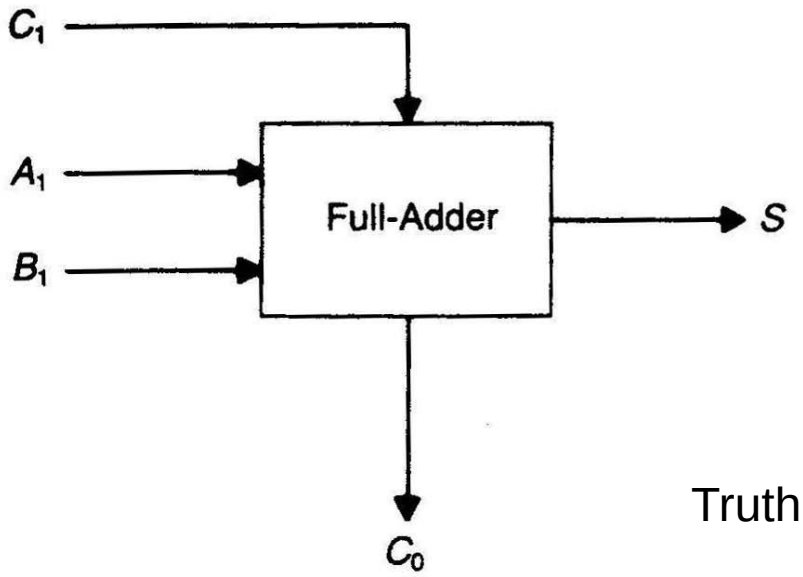
Frequency = f
Capacitance = $1.2C$
Voltage = $0.6V$
Power = $0.432P$
(c)



VLSI系統設計與高階合成

Chip Testing

Fault Definition (1/2)



A_1	B_1	C_1	S	C_0
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

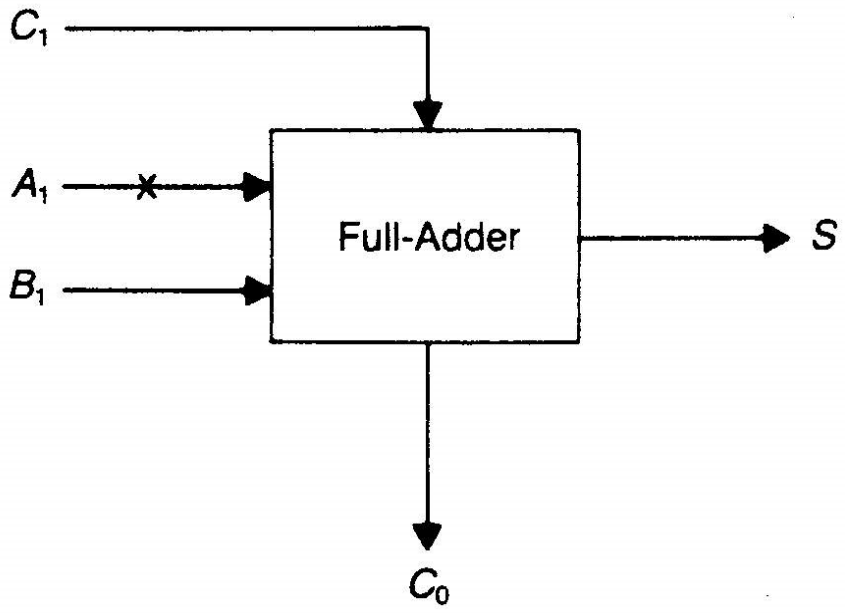
Truth table for the fault-free circuit

Fault: a physical defect, imperfection, or flaw that occurs within some hardware and software component

Error: the manifestation (result) of a fault

Failure: the result of a error -- a system performs one of its functions incorrectly

Fault Definition (2/2)



A_1	B_1	C_1	S	C_0	
0	0	0	1	0	
0	0	1	0	1	error
0	1	0	0	1	
0	1	1	1	1	
1	0	0	1	0	
1	0	1	0	1	
1	1	0	0	1	
1	1	1	1	1	

A short occurs in line A_1 (**fault - become permanent 1**)

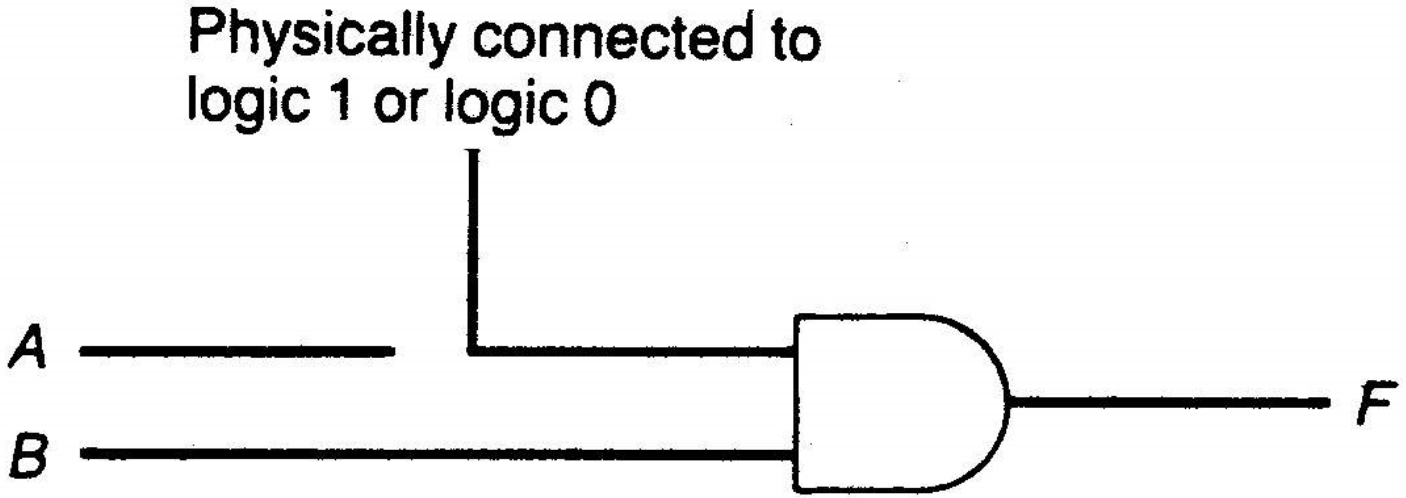
The circuit performs correctly for the input combinations 100, 101, 110, and 111, but not for 000, 001, 010, and 011 (**error**).

If the output of the circuit is controlling a relay, a **failure** might occur.

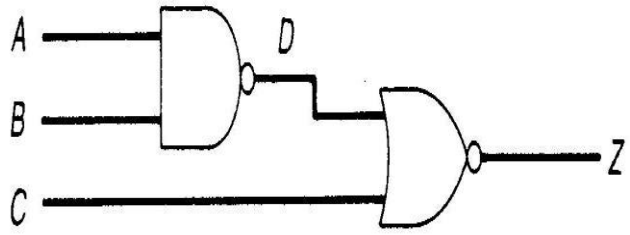
Fault Model (1/4)

The logical stuck-fault model:

stuck-at-0 (s-a-0) and stuck-at-1 (s-a-1)



Fault Model (2/4)



Truth table of a fault-free circuit

A	B	C	Z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Truth table if *C* has a s-a-1 fault

A	B	C	Z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

$Z(L_0)$ represents the output when the stuck-fault L_0 is present in the circuit

Truth table showing the response of the fault-free and faulty circuits for the circuit

A	B	C	Z	$Z(A_0)$	$Z(B_0)$	$Z(C_0)$	$Z(D_0)$	$Z(Z_0)$	$Z(A_1)$	$Z(B_1)$	$Z(C_1)$	$Z(D_1)$	$Z(Z_1)$
0	0	0	0	0	0	0	1	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	1	0	1	0	0	0	1
0	1	1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	1	0	0	1	0	0	1
1	0	1	0	0	0	0	0	0	0	0	0	0	1
1	1	0	1	0	0	1	1	0	1	1	0	0	1
1	1	1	0	0	0	1	0	0	0	0	0	0	1

Fault Model (3/4)

Truth table showing the response of the fault-free and faulty circuits

Truth table showing the response of the fault-free and faulty circuits for the circuit

A	B	C	Z	$Z(A_0)$	$Z(B_0)$	$Z(C_0)$	$Z(D_0)$	$Z(Z_0)$	$Z(A_1)$	$Z(B_1)$	$Z(C_1)$	$Z(D_1)$	$Z(Z_1)$
0	0	0	0	0	0	0	1	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	1	0	1	0	0	0	1
0	1	1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	1	0	0	1	0	0	1
1	0	1	0	0	0	0	0	0	0	0	0	0	1
1	1	0	1	0	0	1	1	0	1	1	0	0	1
1	1	1	0	0	0	1	0	0	0	0	0	0	1

Fault detection table (identify the patterns that detect specific faults)

Fault detection table for the circuit

A	B	C	A_0	B_0	C_0	D_0	Z_0	A_1	B_1	C_1	D_1	Z_1
0	0	0	0	0	0	1	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	1	0	1	0	0	0	1
0	1	1	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	1	0	0	1	0	0	1
1	0	1	0	0	0	0	0	0	0	0	0	1
1	1	0	1	1	0	0	1	0	0	1	1	0
1	1	1	0	0	1	0	0	0	0	0	0	1

Fault Model (4/4)

The reduced fault detection table for the circuit

The reduced fault detection table for the circuit

A	B	C	F	C_0	A_1	B_1
0	1	0	0	0	1	0
1	0	0	0	0	0	1
1	1	0	1	0	0	0
1	1	1	0	1	0	0

F represents the equivalent faults A_0, B_0, Z_0, C_1, and D_1.

Use ATPG (Automatic Test Pattern Generation)

The transistor stuck-fault model:

VLSI Testing (1/3)

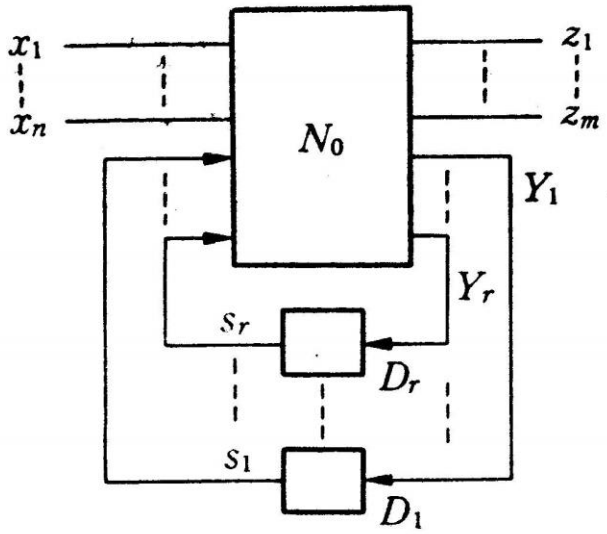
Fault Simulation:

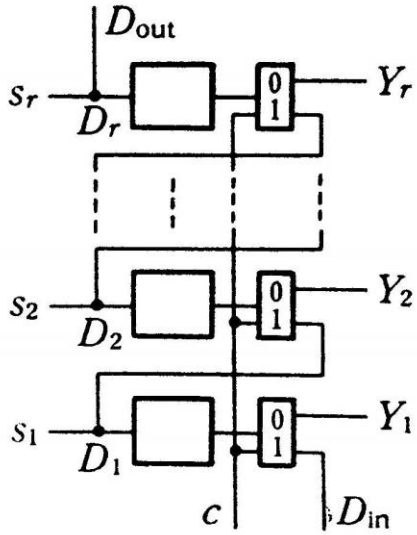
➡ **ATPG (Automatic Test Pattern Generation)**

Design for Testability (DFT)

➡ Controllability and Observability

1. Scan-based design



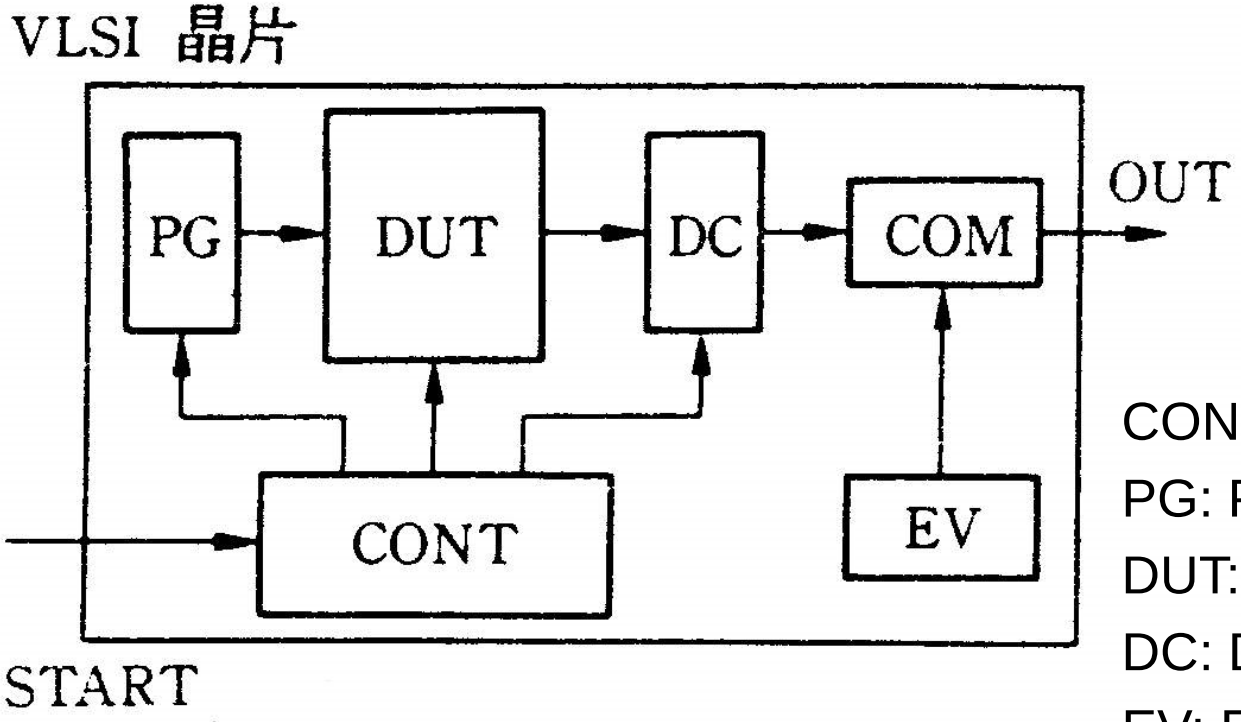


c=0, normal mode

c=1, testing mode ➡ Scan it and scan out

VLSI Testing (2/3)

2. Build-In Self –Test (BIST)



CONT: Controller

PG: Pattern generator

DUT: Device under test

DC: Data compression

EV: Expected value reg.

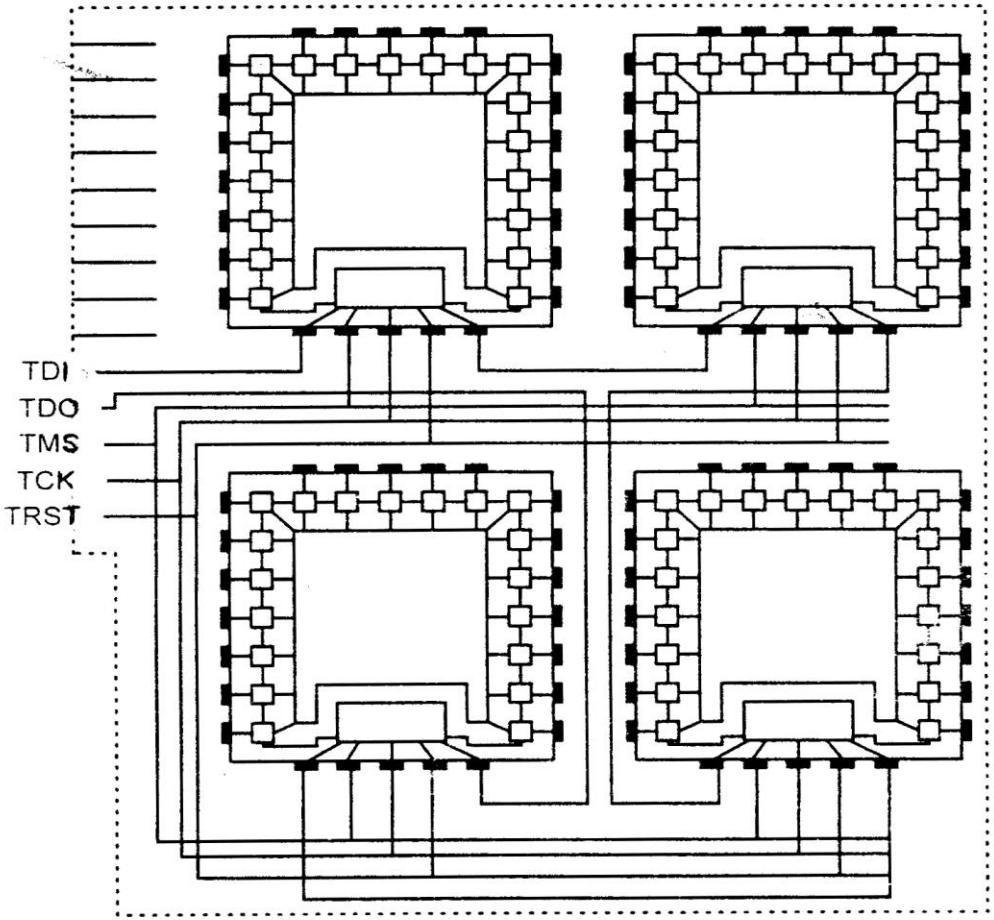
COM: Comparator

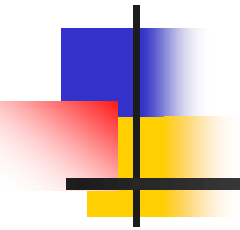
VLSI Testing (3/3)

Boundary Scan:

TCK: test clock TMS: test mode signal

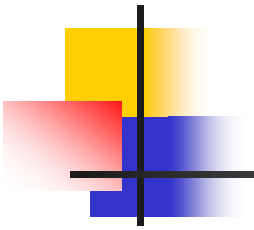
TDI: test data in TDO: test data output TRST: optional





VLSI系統設計與高階合成

Fault-Tolerant Design



Fault Tolerance (1/2)

How to combat faults ?

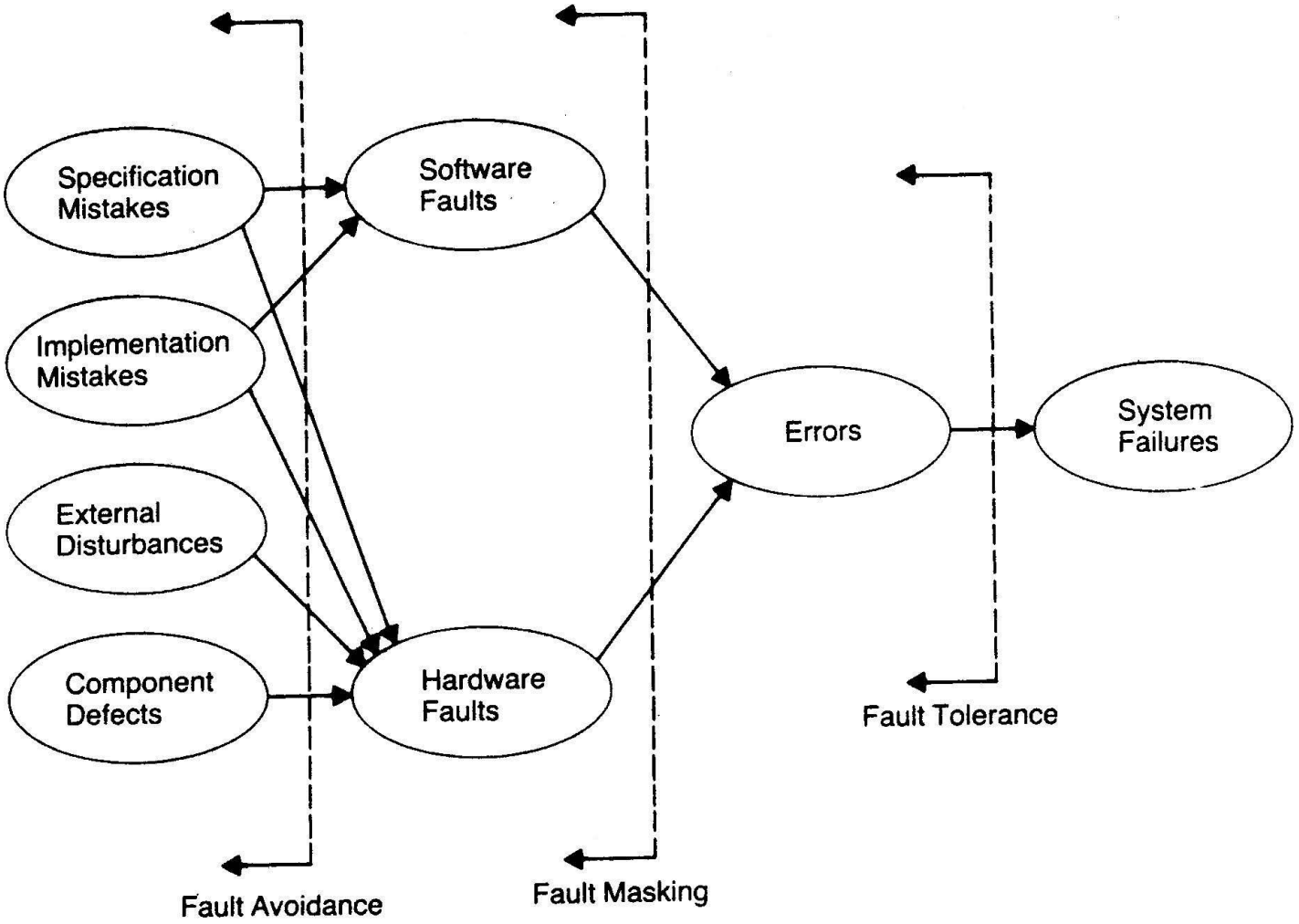
Fault avoidance: any technique that is used to prevent faults

Fault masking: any technique that is used to prevent faults in a system from introducing errors into the system

Fault tolerance: the ability of a system to continue to perform its tasks after the occurrence of faults. The ultimate goal of fault tolerance is to prevent system failures from ever occurring.

Fault Tolerance (2/2)

fault detection, fault location, fault containment, fault recovery



Design of Fault Tolerance

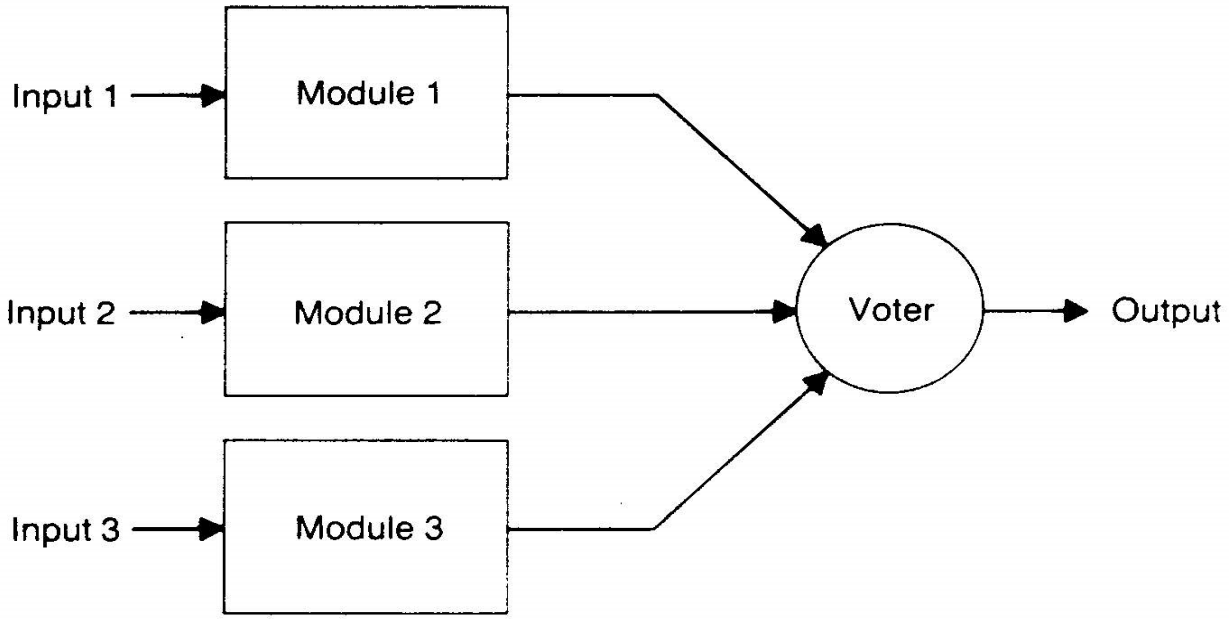
Design of Fault Tolerance:

1. **Hardware Redundancy**
2. **Information Redundancy**
3. **Software Redundancy**
4. **Time Redundancy**

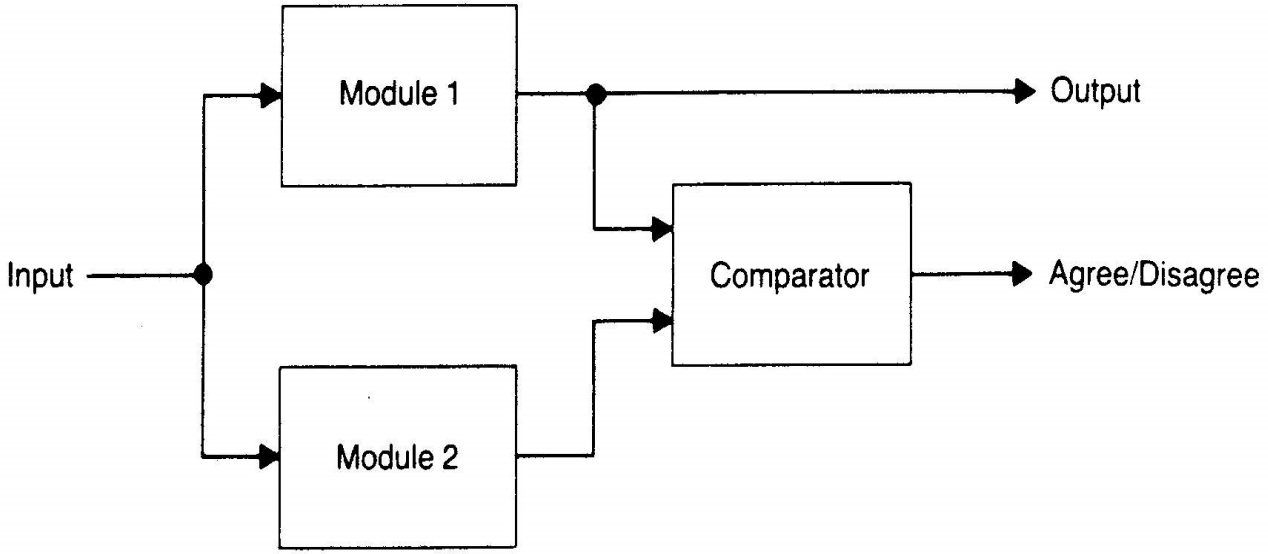
Redundancy: the addition of information, resources or time beyond what is needed for normal system operation

Hardware Redundancy (1/3)

Passive Hardware Redundancy (fault masking)



Active Hardware Redundancy



Hardware Redundancy (2/3)

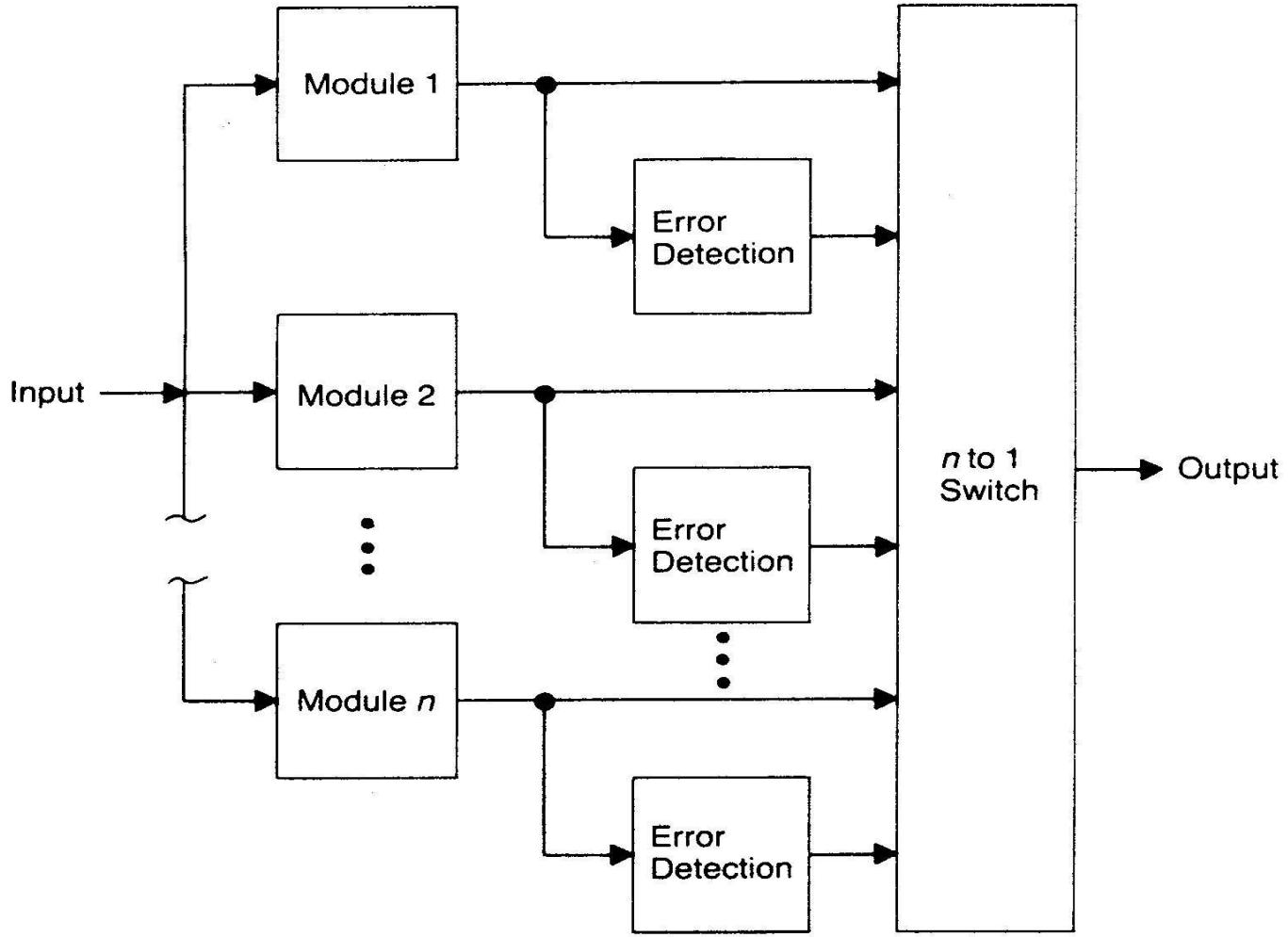
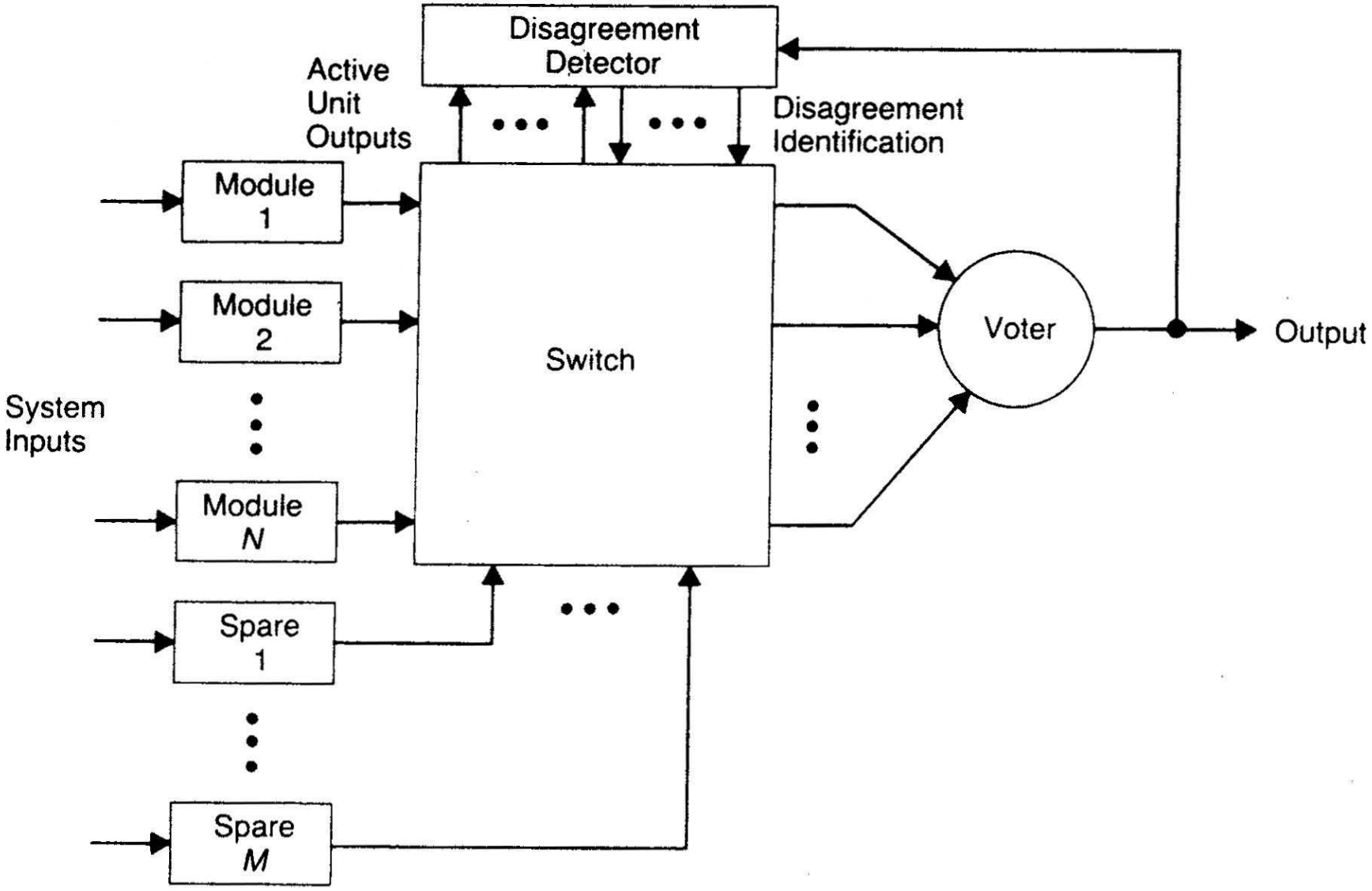


Fig. 3.14 In standby sparing, one of n modules is used to provide the system's output, and the remaining $n - 1$ modules serve as spares. Error detection techniques identify faulty modules so that a fault-free module is always selected to provide the system's output.

Hardware Redundancy (3/3)

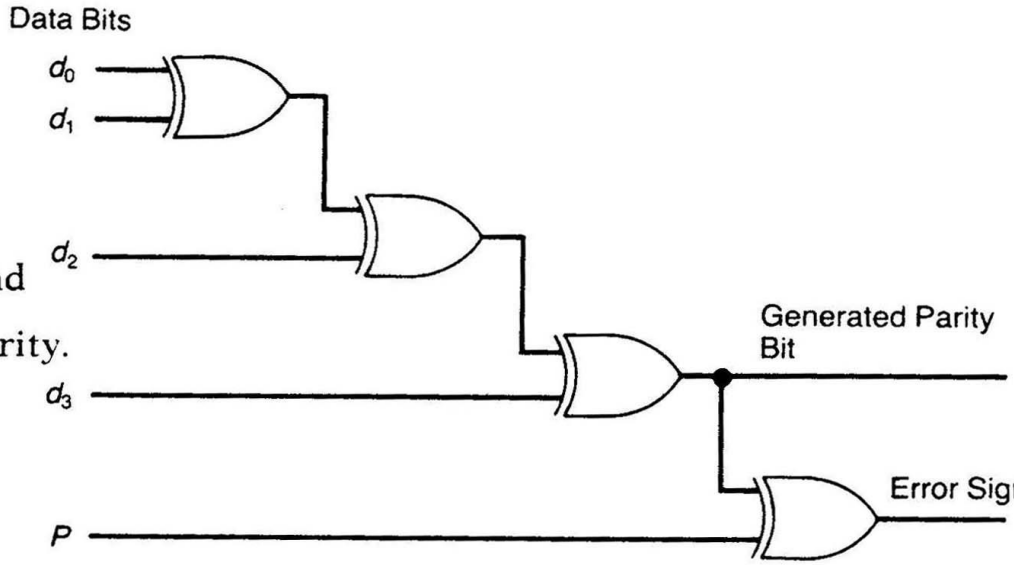


Information Redundancy (1/4)

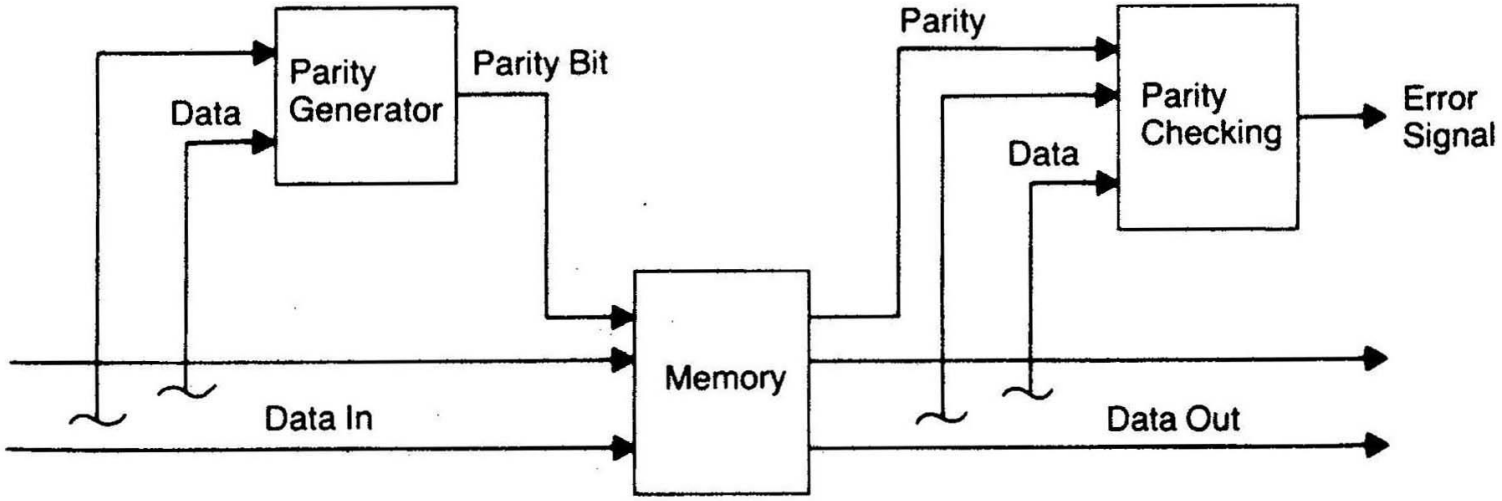
TABLE 3.3 Odd and even parity codes for BCD data

Decimal digit	BCD	BCD odd parity	BCD even parity
0	0000	0000 1	0000 0
1	0001	0001 0	0001 1
2	0010	0010 0	0010 1
3	0011	0011 1	0011 0
4	0100	0100 0	0100 1
5	0101	0101 1	0101 0
6	0110	0110 1	0110 0
7	0111	0111 0	0111 1
8	1000	1000 0	1000 1
9	1001	1001 1	1001 0
		↑ Parity bit	↑ Parity bit

Information Redundancy (2/4)

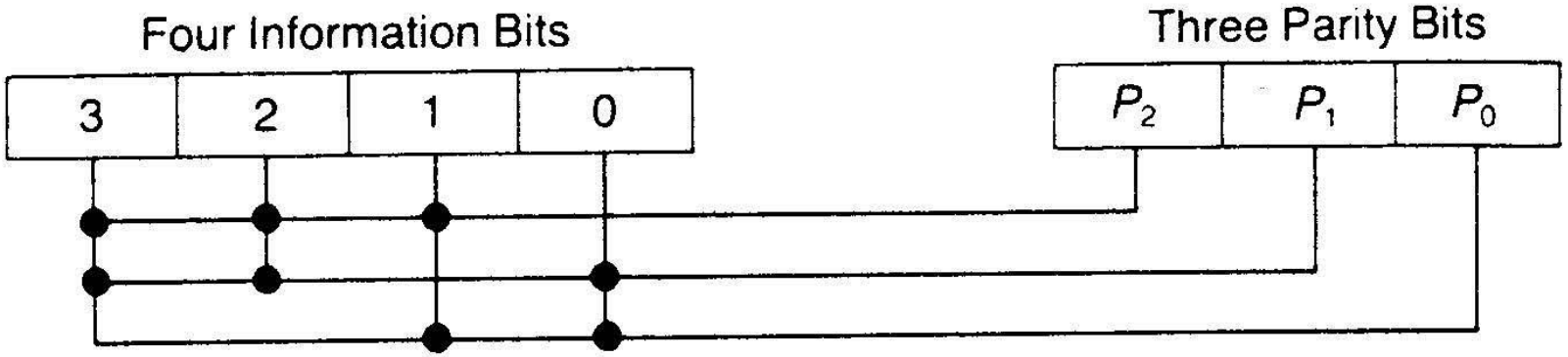


A 4-bit parity generation and checking circuit for even parity.



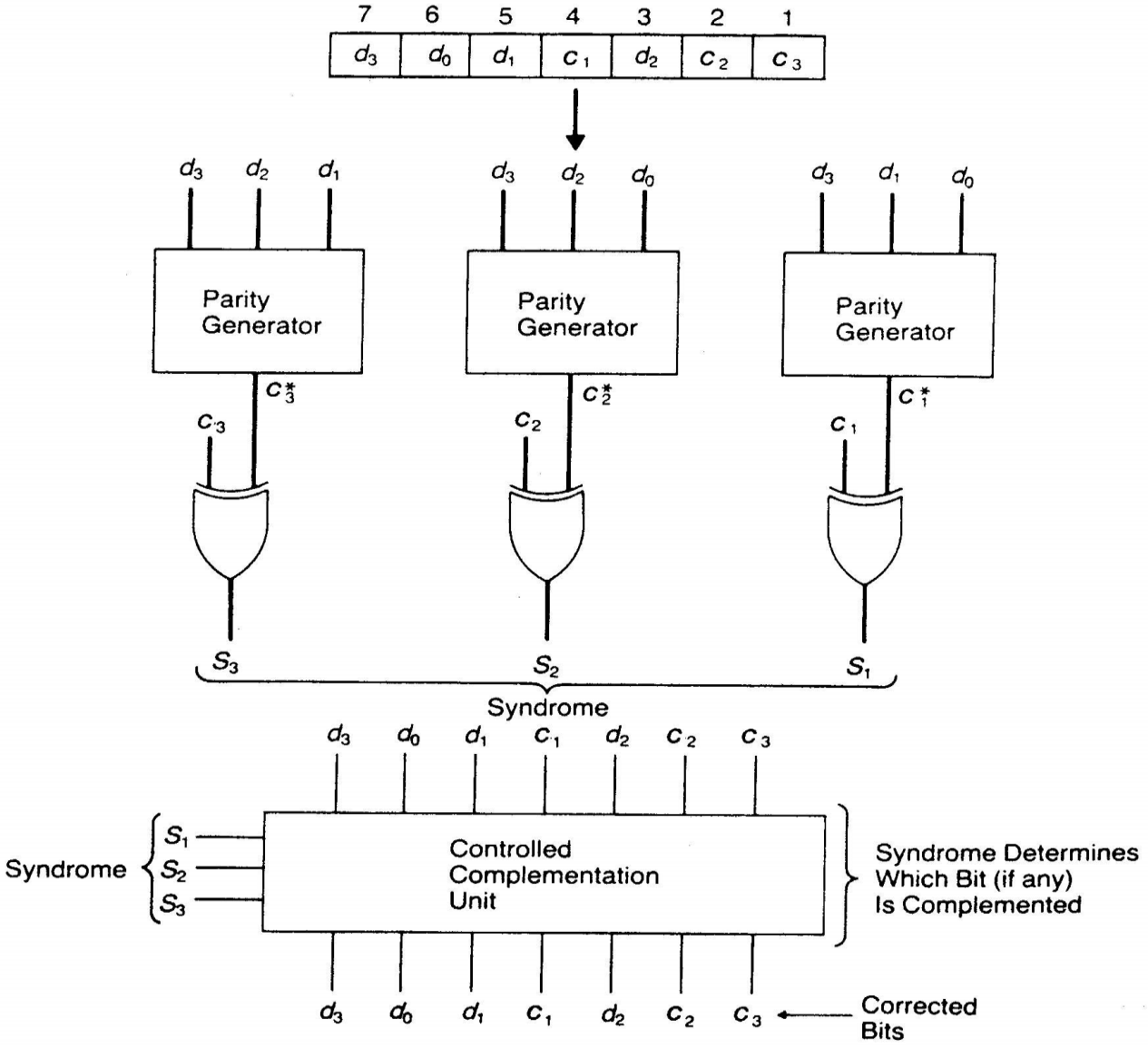
Information Redundancy (3/4)

Hamming code (error correcting code)



Bit in Error	Parity in Error		
3	P_2	P_1	P_0
2	P_2	P_1	
1	P_2		P_0
0		P_1	P_0
P_2	P_2		
P_1		P_1	
P_0			P_0

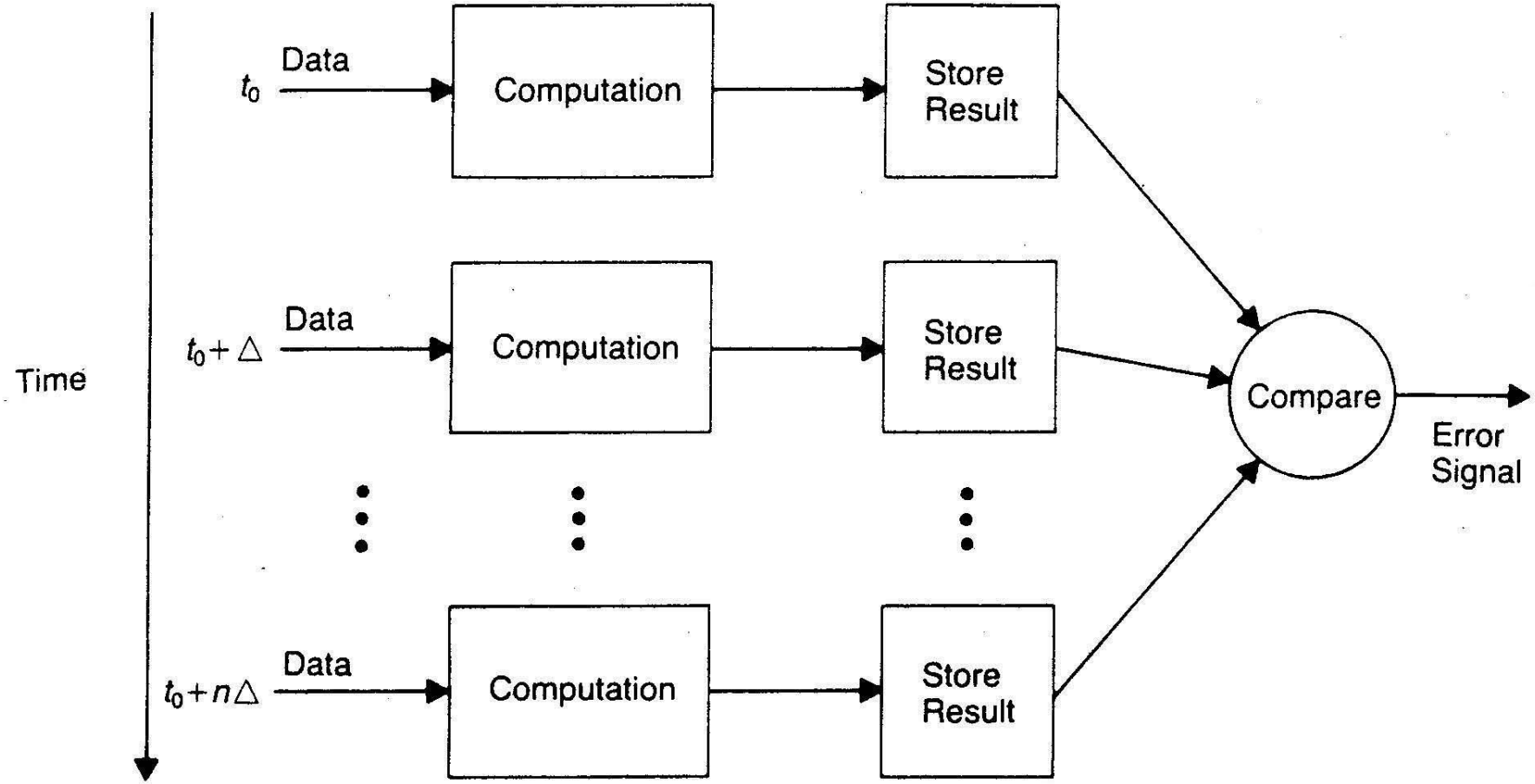
Information Redundancy (4/4)



Time Redundancy (1/2)

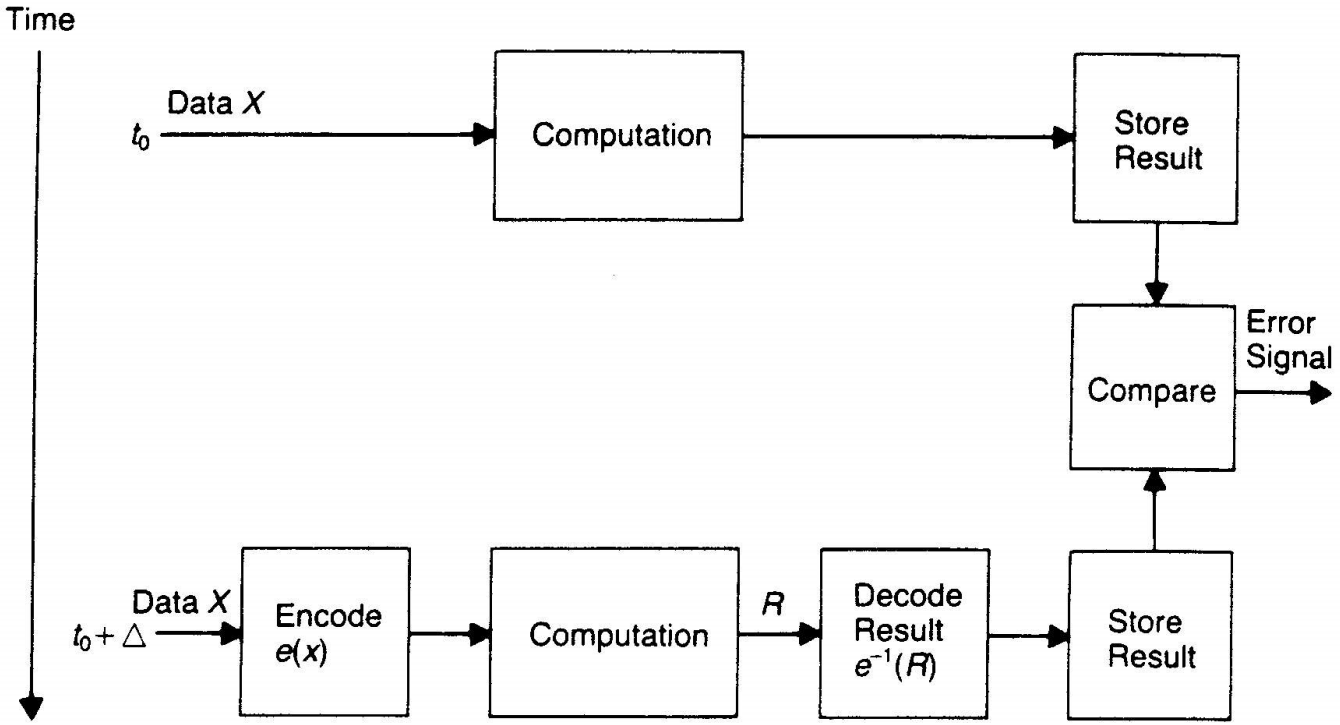
Time Redundancy

Transient fault detection



Time Redundancy (2/2)

Permanent fault detection



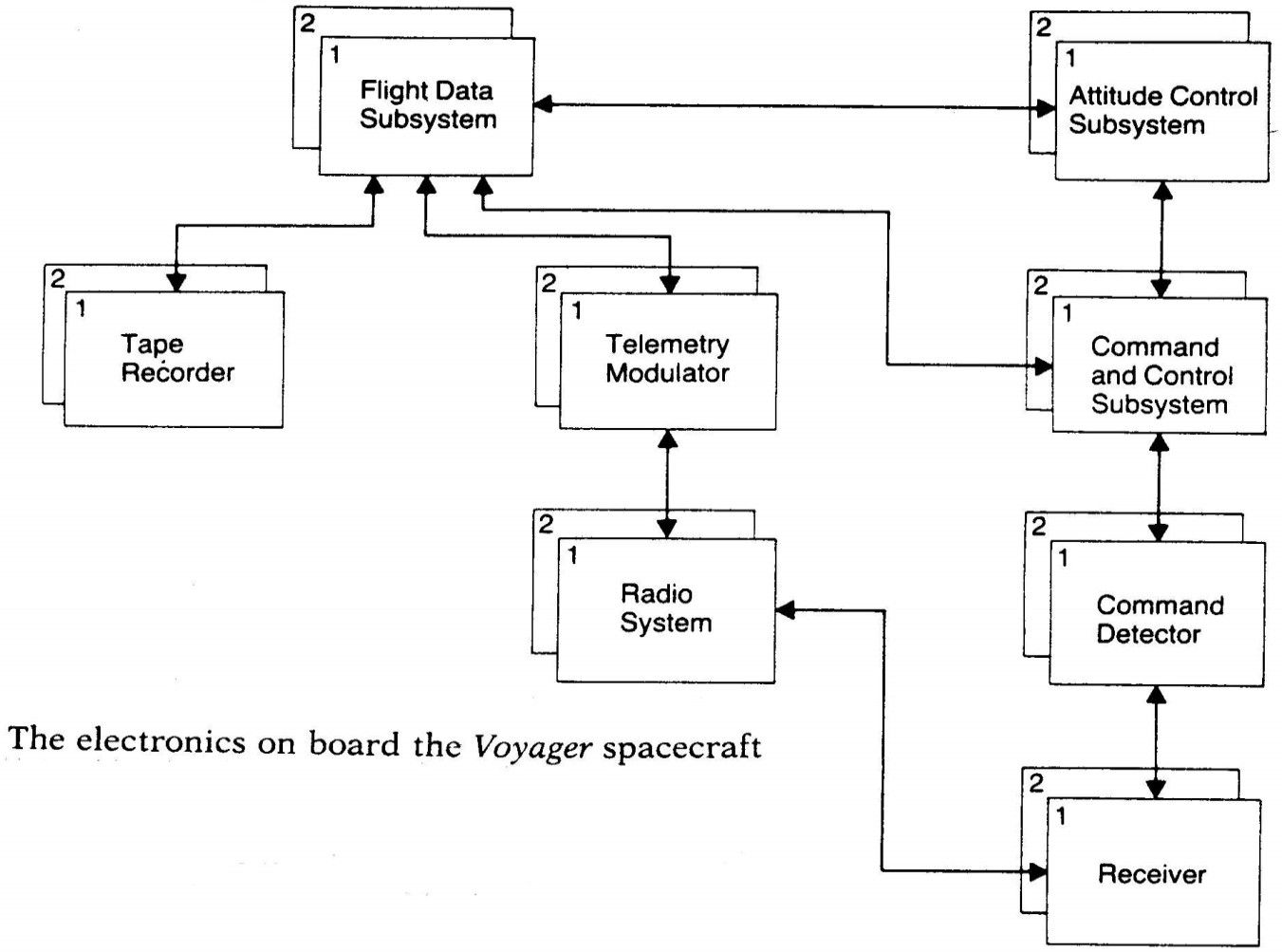
Permanent faults can be detected using time redundancy by modifying the way in which computations are performed the second time.

Software Redundancy

Consistent checks and N-version programming

Practical Fault-Tolerant System (1/4)

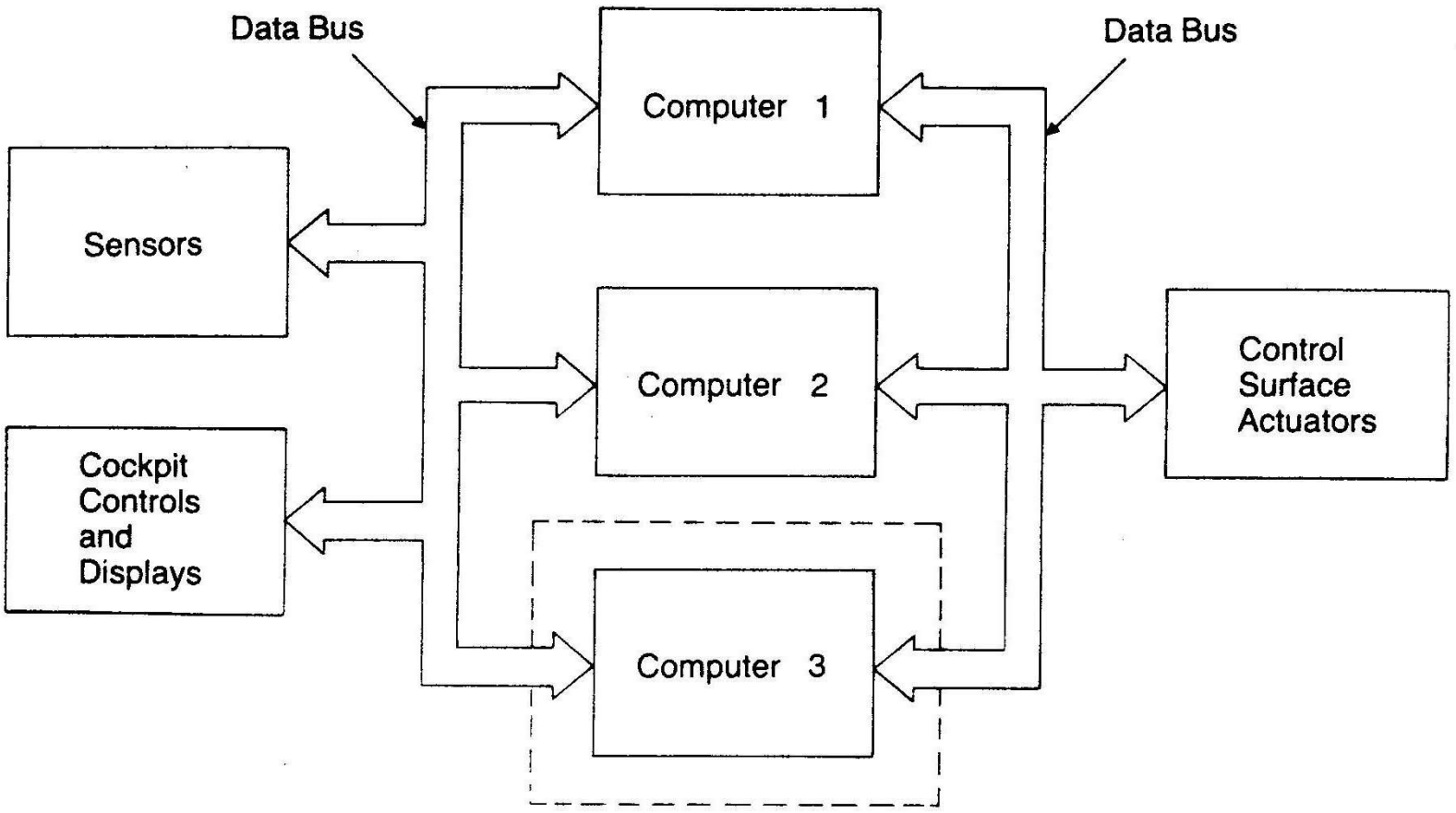
Long-Life Applications



The electronics on board the *Voyager* spacecraft

Practical Fault-Tolerant System (2/4)

Critical-Computation Applications



The X-29 flight control system contains three complete systems

Practical Fault-Tolerant System (3/4)

Maintenance Postponement Applications

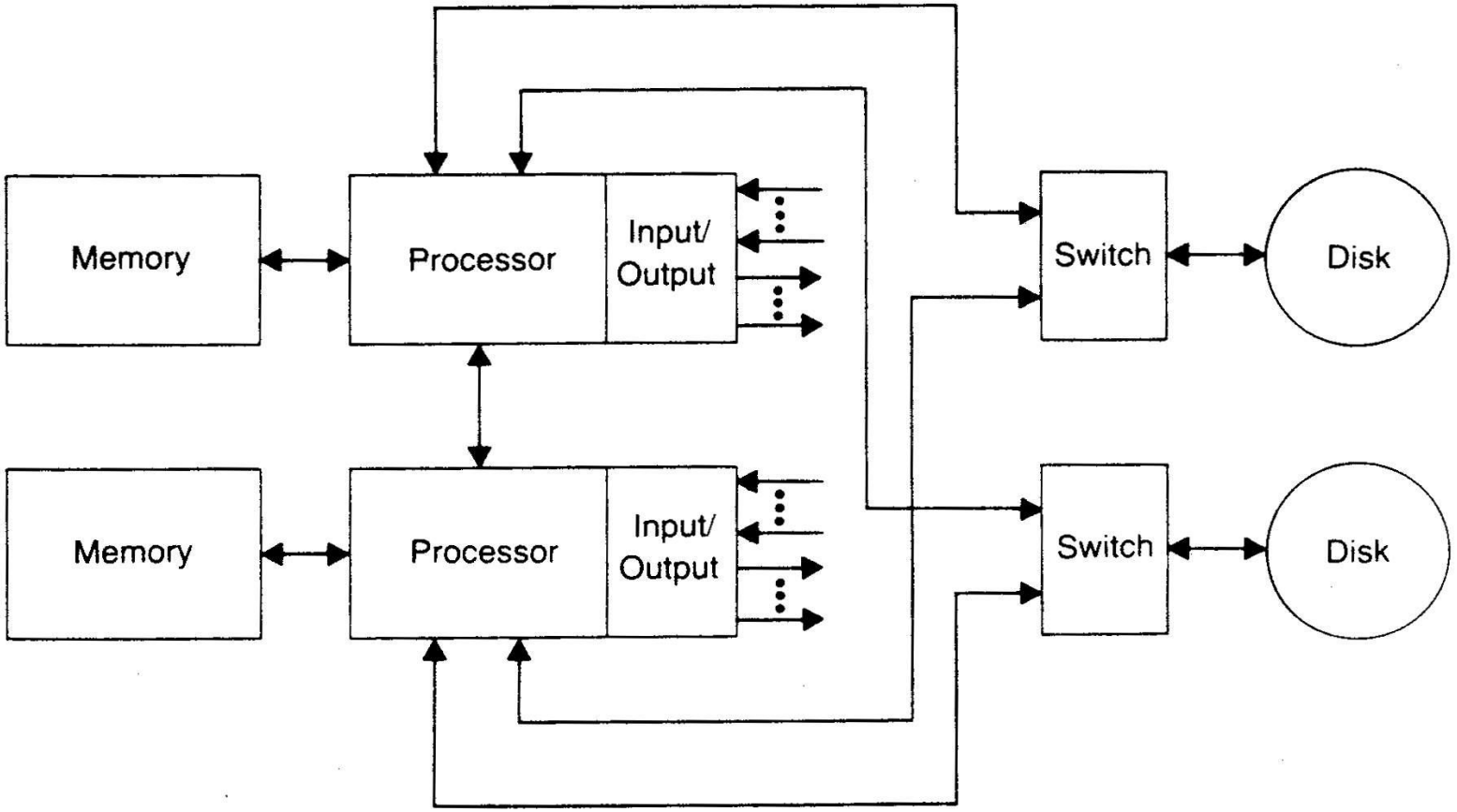
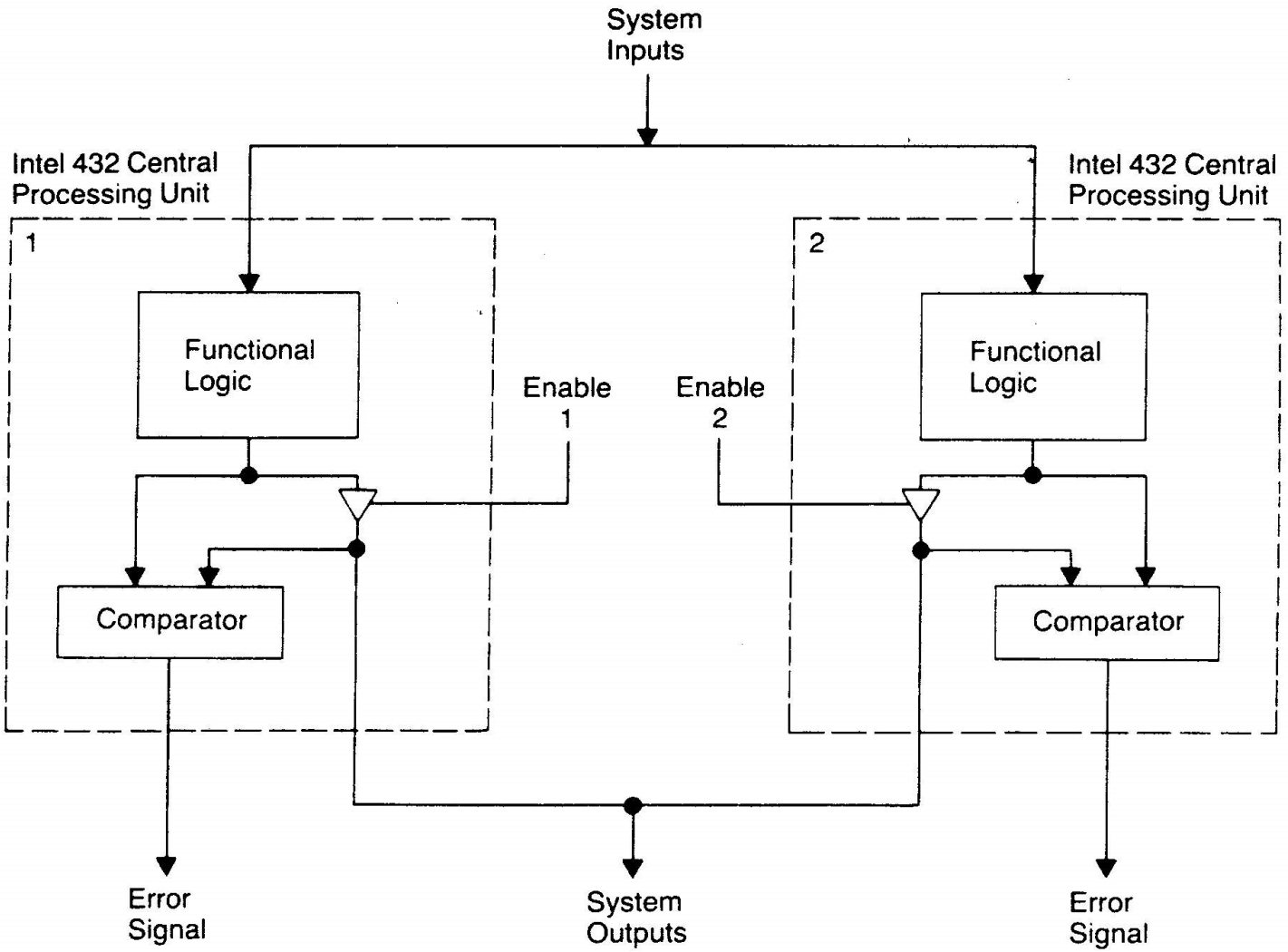


Fig. 1.3 Block diagram of the 3B20D processor used in the Bell Electronic Switching System. All critical components in this system are duplicated. (From [Serlin 1984] © 1984 IEEE)

Practical Fault-Tolerant System (4/4)

High-Availability Applications



Fault-Tolerant Design of VLSI Circuit (1/7)

Fault detection using duplication with complementary logic

Spare designs with comparison

(1) two identical modules (not good)

(2) two modules designed by different teams

(3) complementary logic (good)

Complementary Logic

$$f(x_1, x_2, x_3) = x_1 x_2' + x_3 \quad \text{postive logic (+5V for logic 1)}$$

Complementing f and replacing each variable with its complement

$$f' = (x_1 x_2' + x_3)' = (x_1' + x_2) x_3'$$

$$f_d(x_1, x_2, x_3) = (x_1 + x_2') x_3 \quad \text{negative logic (0 for logic 1)}$$

f and f_d are complementary

Fault-Tolerant Design of VLSI Circuit (2/7)

Three main advantages of using complementary logic:

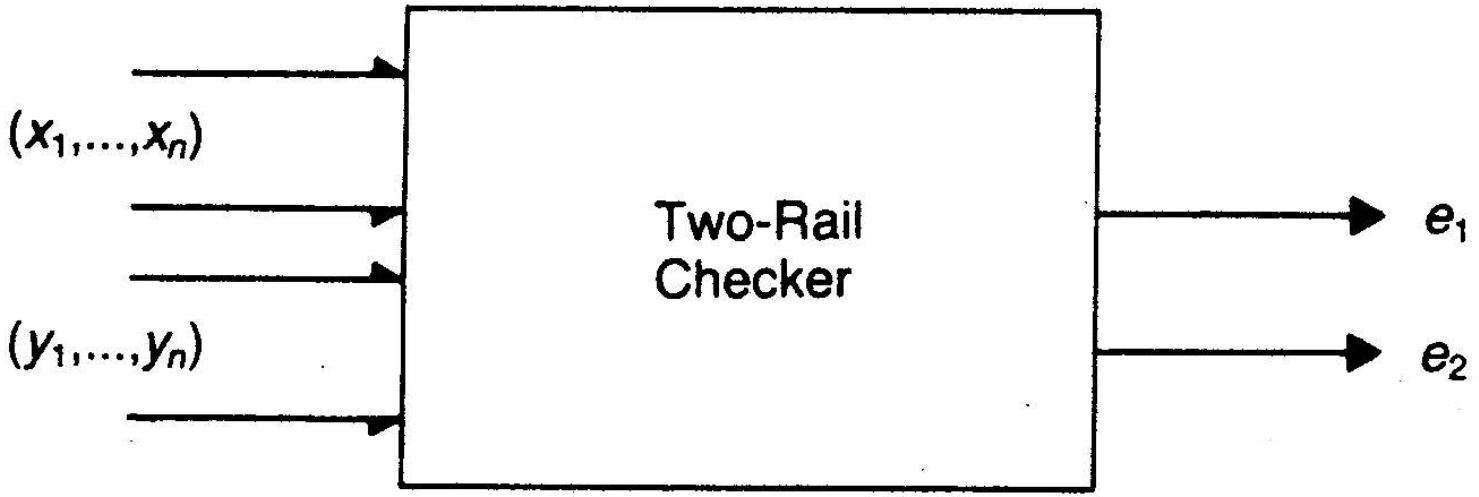
1. The use of separate masks to create the two modules
   (reduce the design mistakes and mask fault problems)
2. Voltage transitions on corresponding lines in the two modules are in opposite directions so that the possibility of faults that are sensitive to voltage transitions producing identical effects is reduced
3. The corresponding lines in the two modules are always at different voltage levels, so a short between two such lines always results in one of the two lines having an erroneous value and the other having the correct value

Fault-Tolerant Design of VLSI Circuit (3/7)

Checking the checker (or comparator)

Self-Checking Logic:

the circuit will either produce the correct output code word or an invalid code word when any single fault occurs



$e_1 = e_2'$, if and only if $x_i = y_i'$
for all values of i

Fault-Tolerant Design of VLSI Circuit (4/7)

$e_1 = x_0 y_1 + y_0 x_1$

$e_2 = x_0 x_1 + y_0 y_1$

$x_0 = y_0'$ and $x_1 = y_1'$ (fault free)

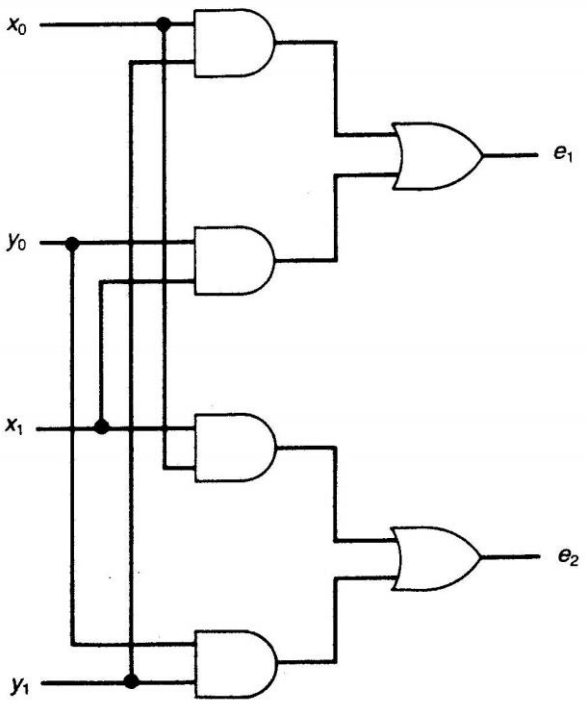
$e_1 = x_0 x_1' + x_0' x_1 = x_0 \oplus x_1$

$e_2 = x_0 x_1 + x_0' x_1' = (x_0 \oplus x_1)'$

$x_0 = y_0$ and $x_1 = y_1'$ (fault free)

$e_1 = x_0 x_1' + x_0 x_1$

$e_2 = x_0 x_1 + x_0 x_1' = e_1$ .....



Work well if the checker is fault free

If $x_0 x_1 = 00$ and $y_0 y_1 = 11$, x_0 s-a-1 or x_1 s-a-1, simulate the circuit

If $x_0 x_1 = 11$ and $y_0 y_1 = 00$, x_0 s-a-0 or x_1 s-a-0, simulate the circuit

Faults in checker can be detected.

If the two inputs are not complementary or the checker contains a fault, the outputs should not be complementary

Fault-Tolerant Design of VLSI Circuit (5/7)

Reconfigurable array structures

Three types of reconfiguration:

a. Fabrication-time reconfiguration:

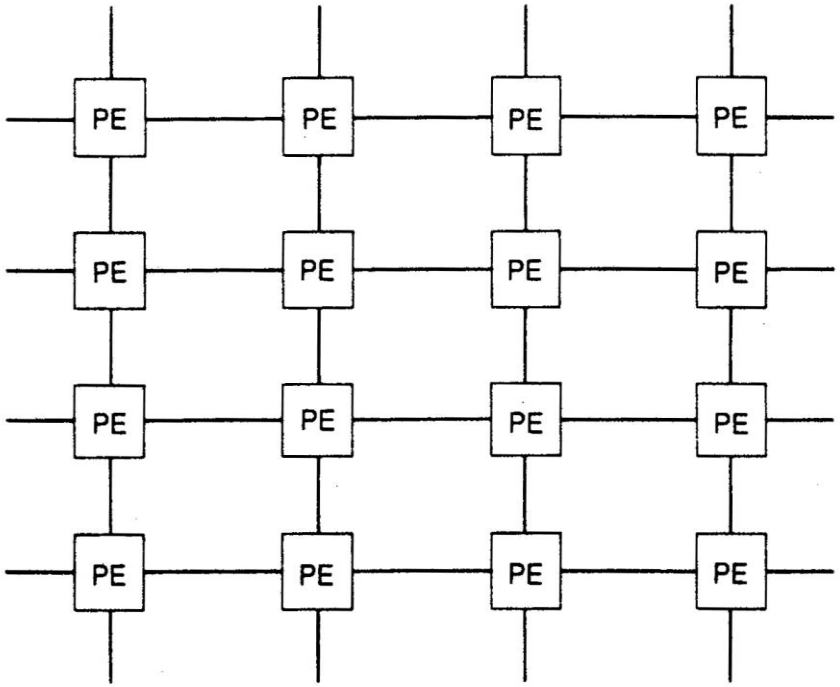
Performed immediately after manufacturing to produce an operational processing array.

b. Compile-time reconfiguration:

Performed before each use of the array

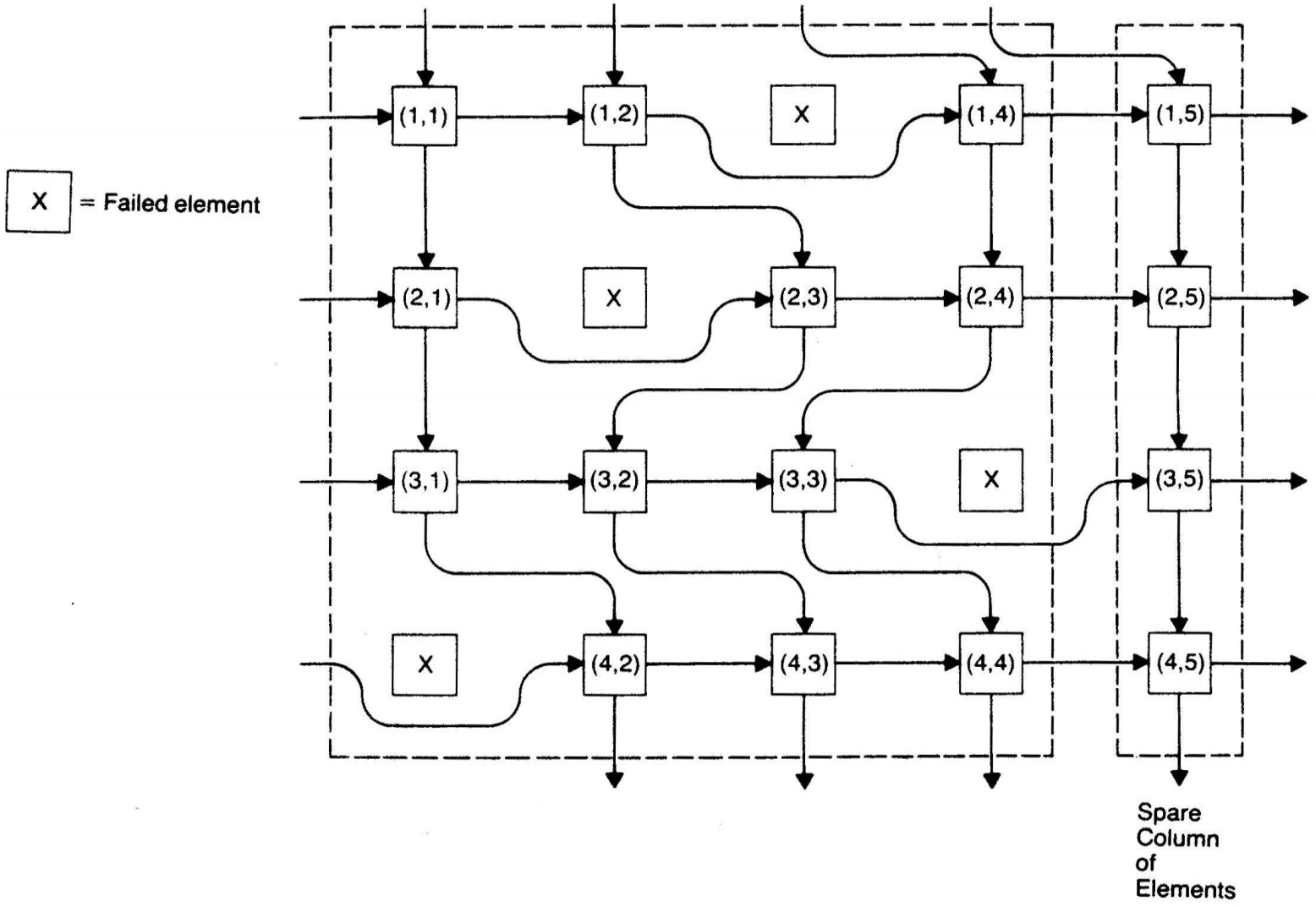
c. Real-time reconfiguration:

Performed while the array is in operation and continues to provide uninterrupted performance of its normal operations



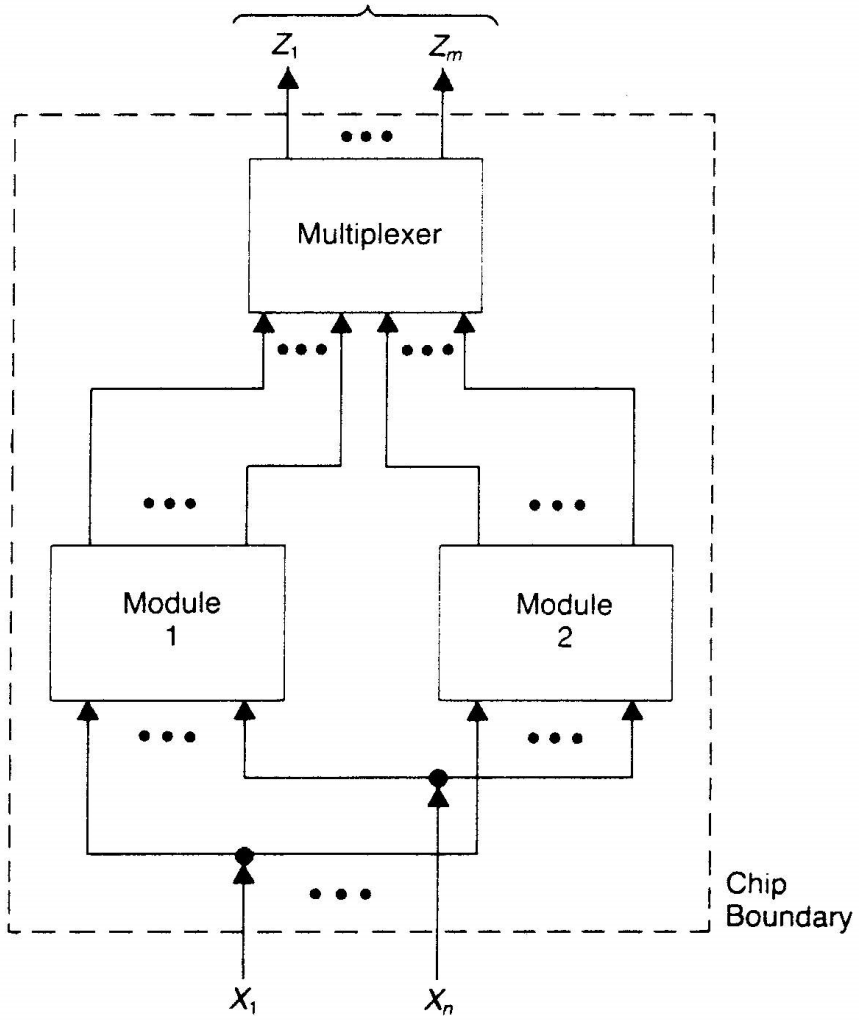
Fault-Tolerant Design of VLSI Circuit (6/7)

A reconfigurable array



Fault-Tolerant Design of VLSI Circuit (7/7)

Redundancy for yield enhancement



Most contents of this chapter referred to “digital and analysis of fault tolerant digital systems” Addison Wesley, Barry W. Johnson