Introduction to VLSI Testing

李昆忠
Kuen-Jong Lee
Dept. of Electrical Engineering
National Cheng-Kung University
Tainan, Taiwan, R.O.C.

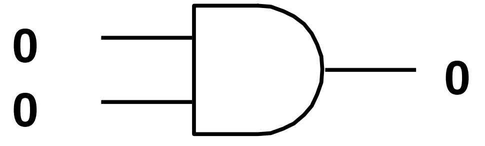
Problems to Think

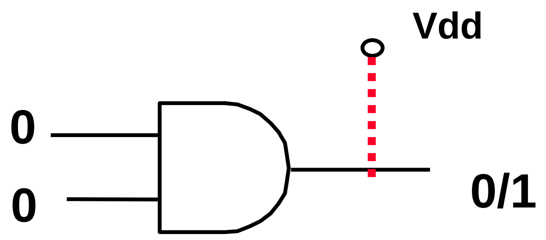
- A 32 bit adder
- A 32 bit counter with RESET function
- A 1MB cache memory
- A 10^7-transistor CPU

OUTLINE

- Introduction
- Fault modeling
- Fault simulation
- Test generation
- Automatic test pattern generation (ATPG)
- Design for testability
- Built-in self test
- Synthesis for testability
- An example

Testing: To tell whether a system is good or bad





Related fields

- **Verification:** To verify the correctness of a design
- **Diagnosis:** To tell the faulty site
- **Reliability:** To tell whether a good system will work after some time.

Importance of testing

N = # transistors in a chip

p = prob. (a transistor is faulty)

Pf = prob. (the chip is faulty)

$\Longrightarrow$ $Pf = 1- (1- p)^{N}$

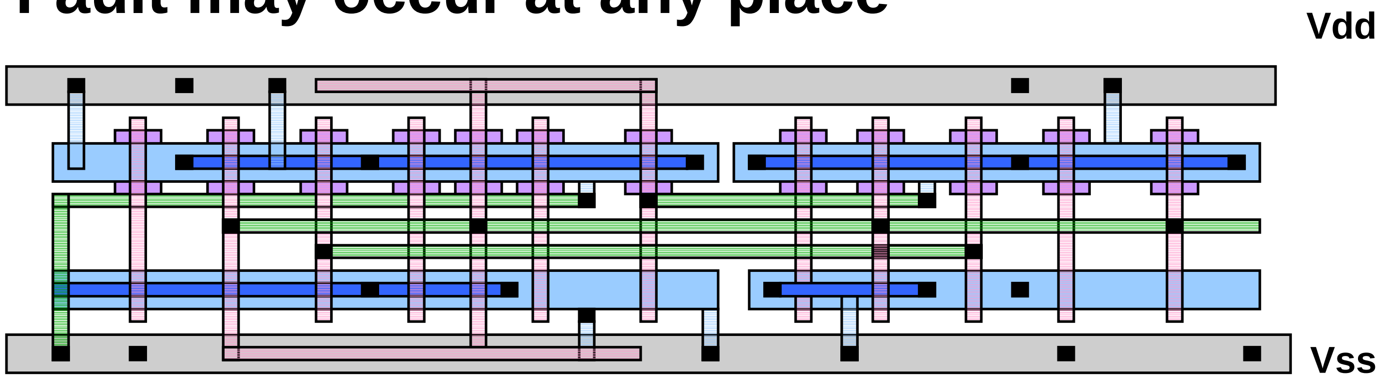
If $p = 10^{-6}$

$N = 10^{6}$

$\Longrightarrow$ $Pf = 63.2\%$

Difficulties in Testing

- **Fault may occur anytime**
  - Design
  - Process
  - Package
  - Field
- **Fault may occur at any place**



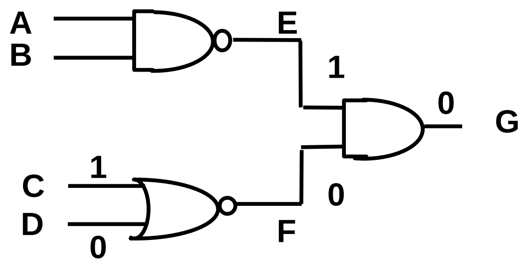
- **VLSI circuit are large**
  - Most problems encountered in testing are NP-complete
- **I/O access is limited**

How to do testing from Designer's points of view

• Circuit modeling • Fault modeling	Modeling
• Logic simulation • Fault simulation • Test generation	ATPG
• Design for test • Built-in self test	Testable design
• Synthesis for testability	

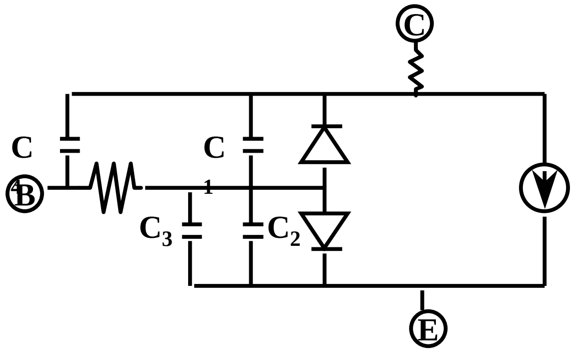
Circuit Modeling

- **Functional model---** logic function
  - `f(x1,x2,...)=...`
  - `Truth table`
- **Behavioral model---** functional + timing
  - `f(x1,x2,...)=... , Delay = 10`
- **Structural model---** collection of interconnected components or elements

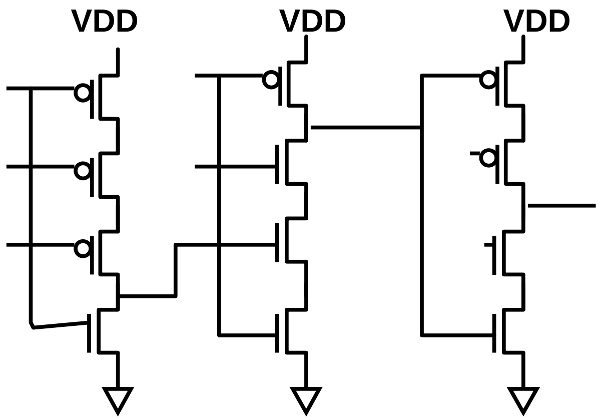


Levels of Description

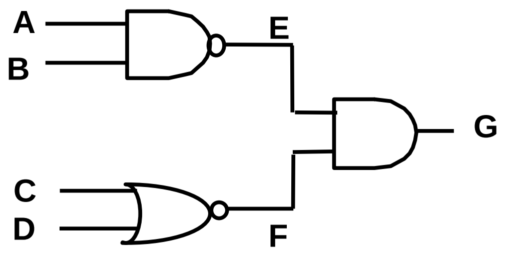
- Circuit level



- Switch level



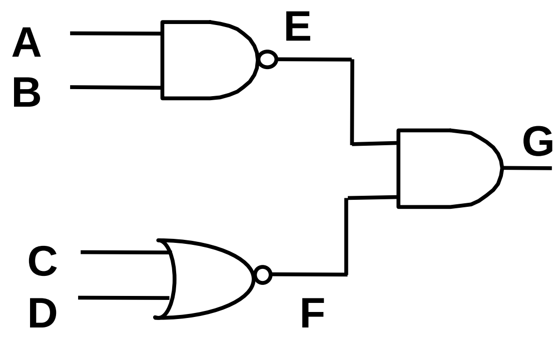
- Gate level



- Higher/ System level

Fault Modeling

- The effects of physical defects
- Most commonly used fault model: Single stuck-at fault



A s-a-1	B s-a-1	C s-a-1	D s-a-1
A s-a-0	B s-a-0	C s-a-0	D s-a-0
E s-a-1	F s-a-1	G s-a-1	
E s-a-0	F s-a-0	G s-a-0	

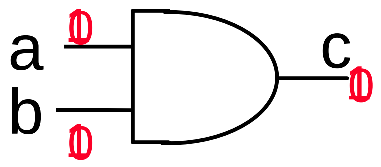
14 faults

- Other fault models:
  - Break faults, Bridging faults, Transistor stuck-open faults, Transistor stuck-on faults, Delay faults

Fault Coverage (FC)

$$FC = \frac{\#\ \text{faults detected}}{\#\ \text{faults in fault list}}$$

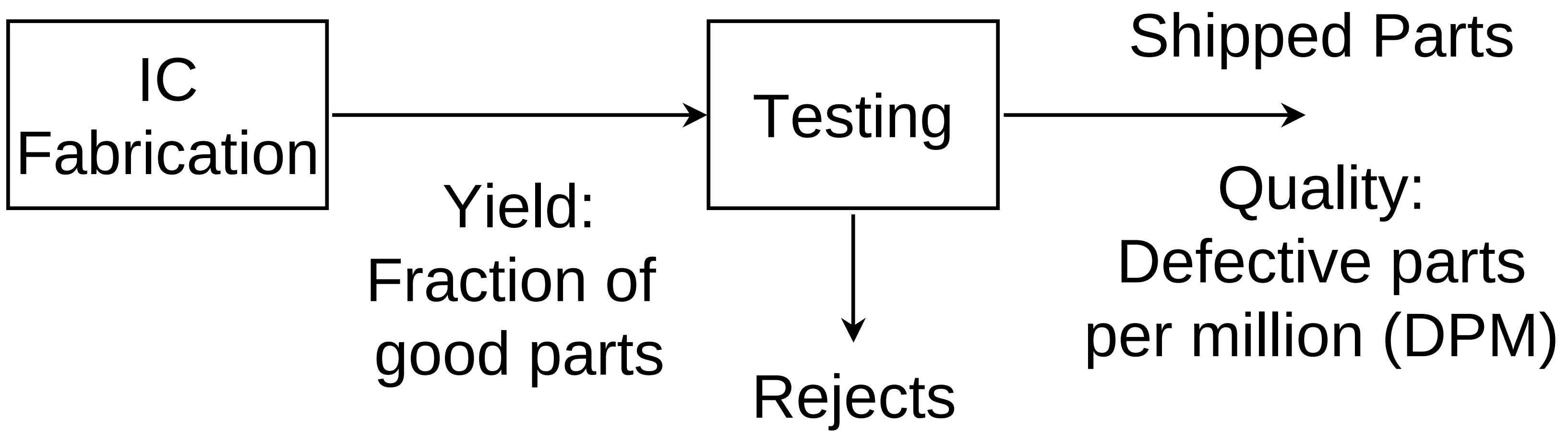
Example:



6 stuck-at faults
(a_0,a_1,b_0,b_1,c_0,c_1)

Test	faults detected	FC
{(0,0)}	c_1	16.67%
{(0,1)}	a_1,c_1	33.33%
{(1,1)}	a_0,b_0,c_0	50.00%
{(0,0),(1,1)}	a_0,b_0,c_0,c_1	66.67%
{(1,0),(0,1),(1,1)}	all	100.00%

Testing and Quality


IC Fabrication
Yield: Fraction of good parts
Testing
Rejects
Shipped Parts
Quality: Defective parts per million (DPM)

- **Quality of shipped parts is a function of yield Y and the test (fault) coverage T**
- **Defect level (DL) : fraction of shipped parts that are defective**

Defect Level, Yield and Fault Coverage

$$DL = 1 - Y^{(1-T)}$$

DL: defect level

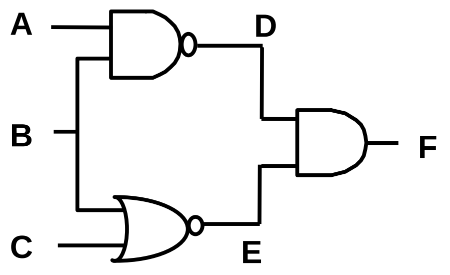
Y: yield

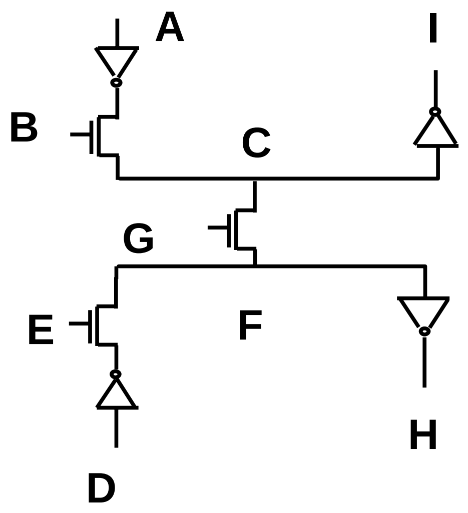
T: fault coverage

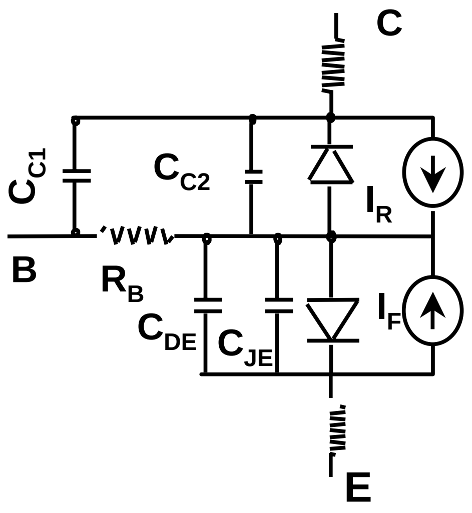
Yield (Y)	Fault Coverage (T)	DPM (DL)
50%	90%	67,000
75%	90%	28,000
90%	90%	10,000
95%	90%	5,000
99%	90%	1,000
90%	90%	10,000
90%	95%	5,000
90%	99%	1,000
90%	99.9%	100

Logic simulation

- To determine how a good circuit should work
- Given input vectors, determine the normal circuit response

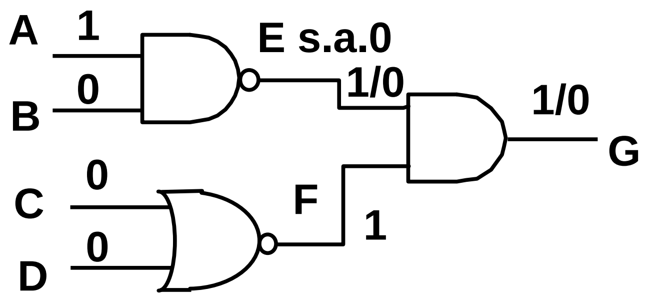






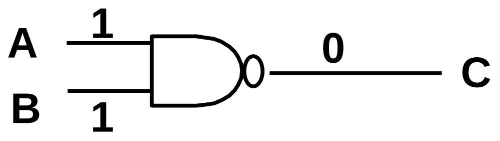
Fault simulation

- **To determine the behavior of faulty circuits**



- **Given a test vector, determine all faults that are detected by this test vector.**

  Example:

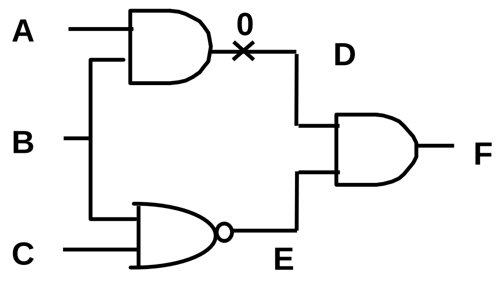


Test vector (1 1) detects
{ a_0, b_0, c_1}

Test generation

- Given a fault, identify a test to detect this fault

Example:



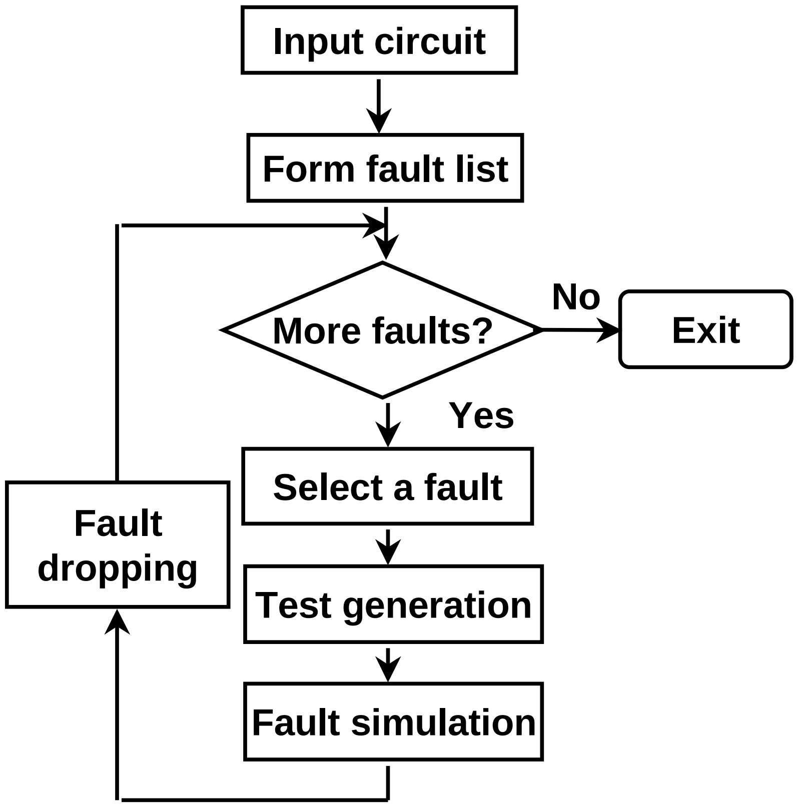
To detect D s-a-0, D must be set to 1.
Thus A=B=1.

To propagate fault effect to the primary output
E must be 1. Thus C must be 0.

Test vector: A=1, B=1, C=0

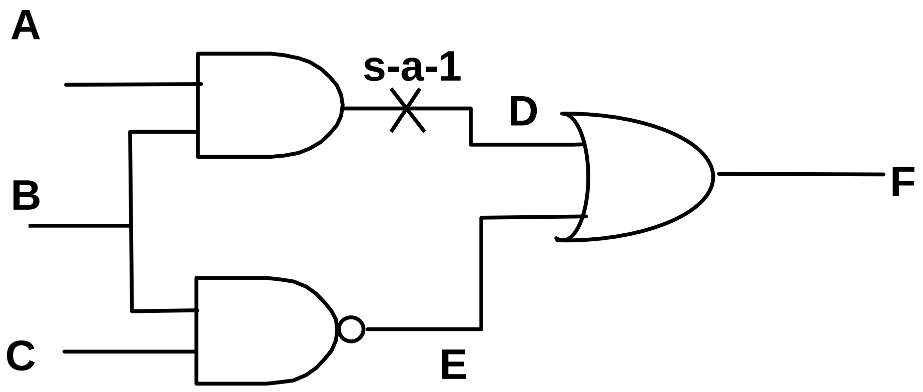
Automatic Test Pattern Generation (ATPG)

- Given a circuit, identify a set of test vectors to detect all faults under consideration.



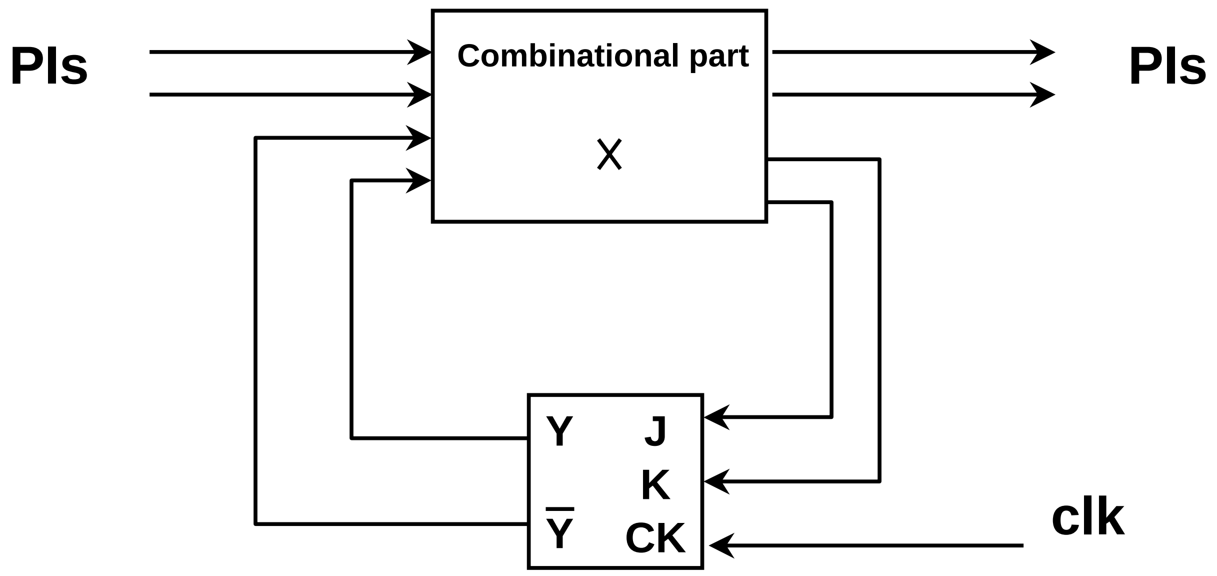
Difficulties in test generation

1. Reconvergent fanout



Difficulties in test generation (cont.)

2. Sequential test generation

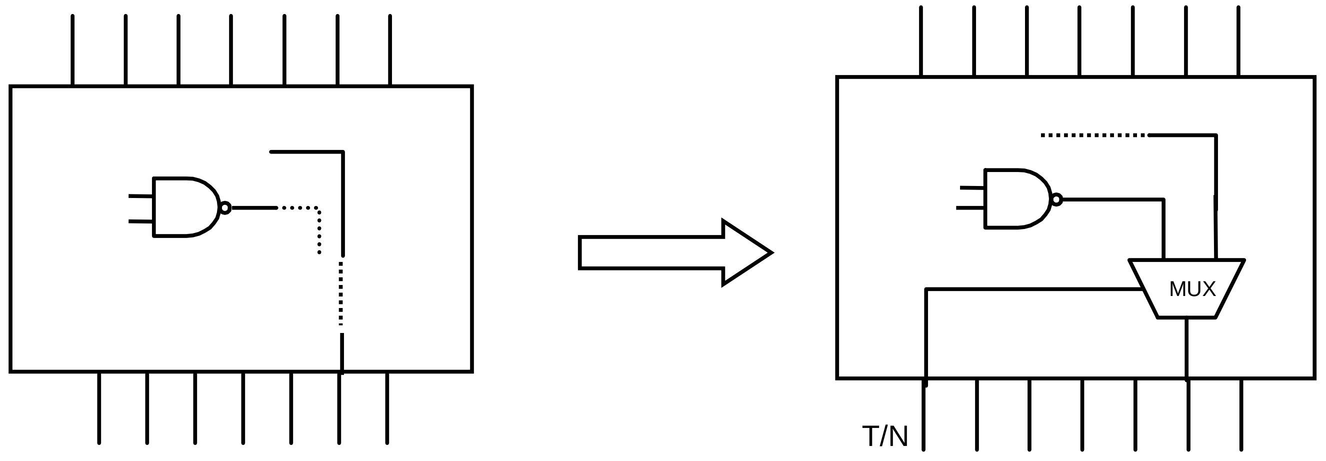


Testable Design

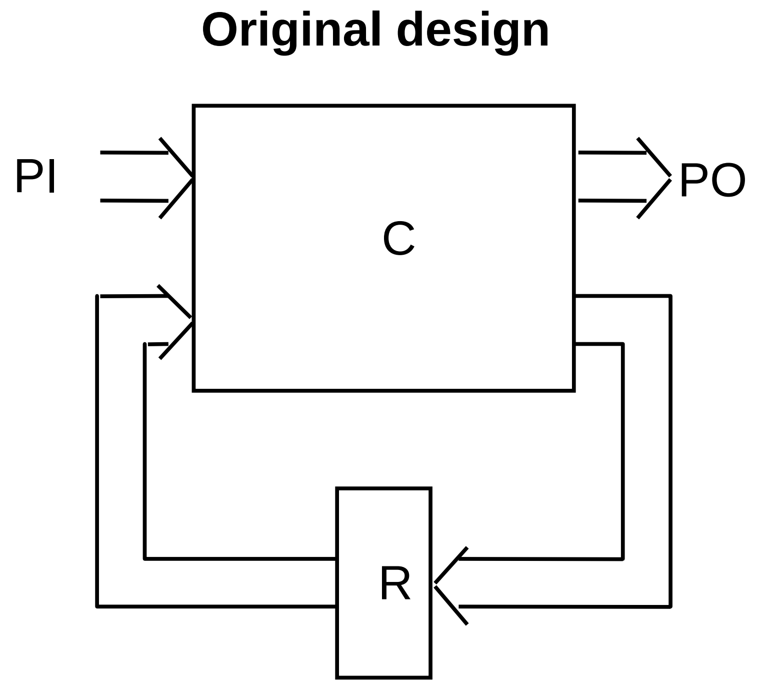
- Design for testability (DFT)
  - *ad hoc* techniques
  - Scan design
  - Boundary Scan
- Built-In Self Test (BIST)
  - Random number generator (RNG)
  - Signature Analyzer (SA)
- Synthesis for Testability

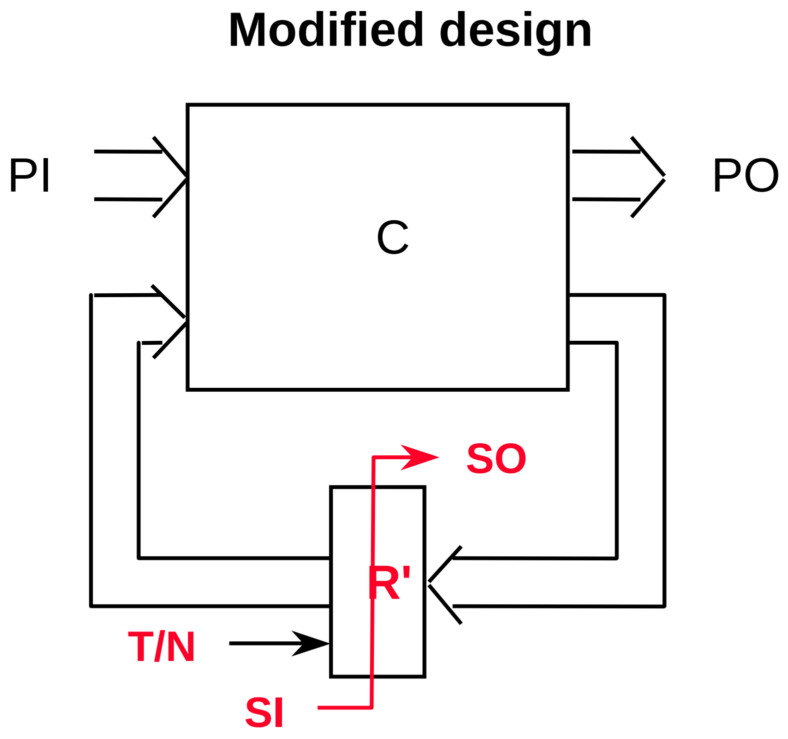
Example of *ad hoc* techniques

Insert test point

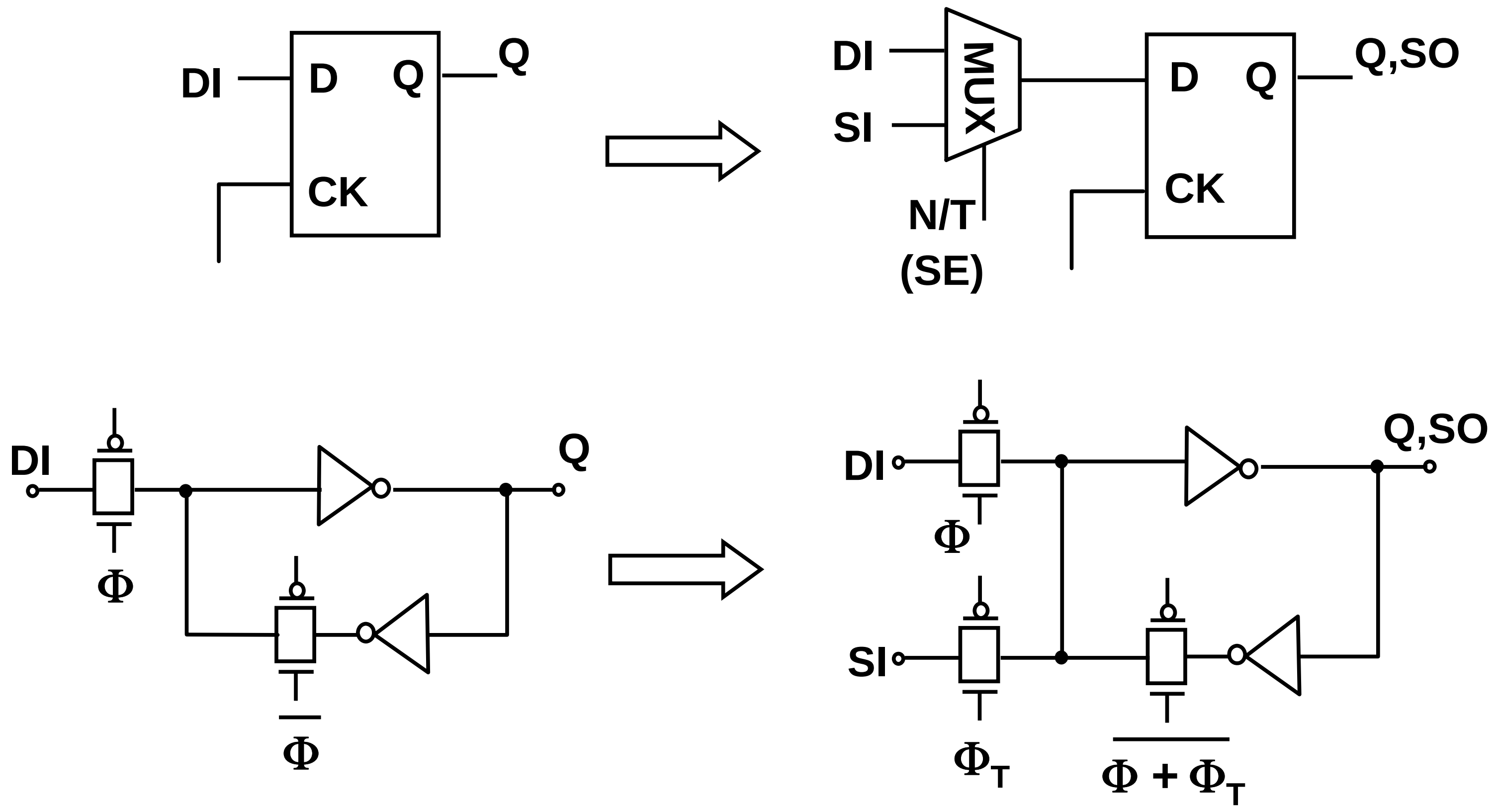


Scan System

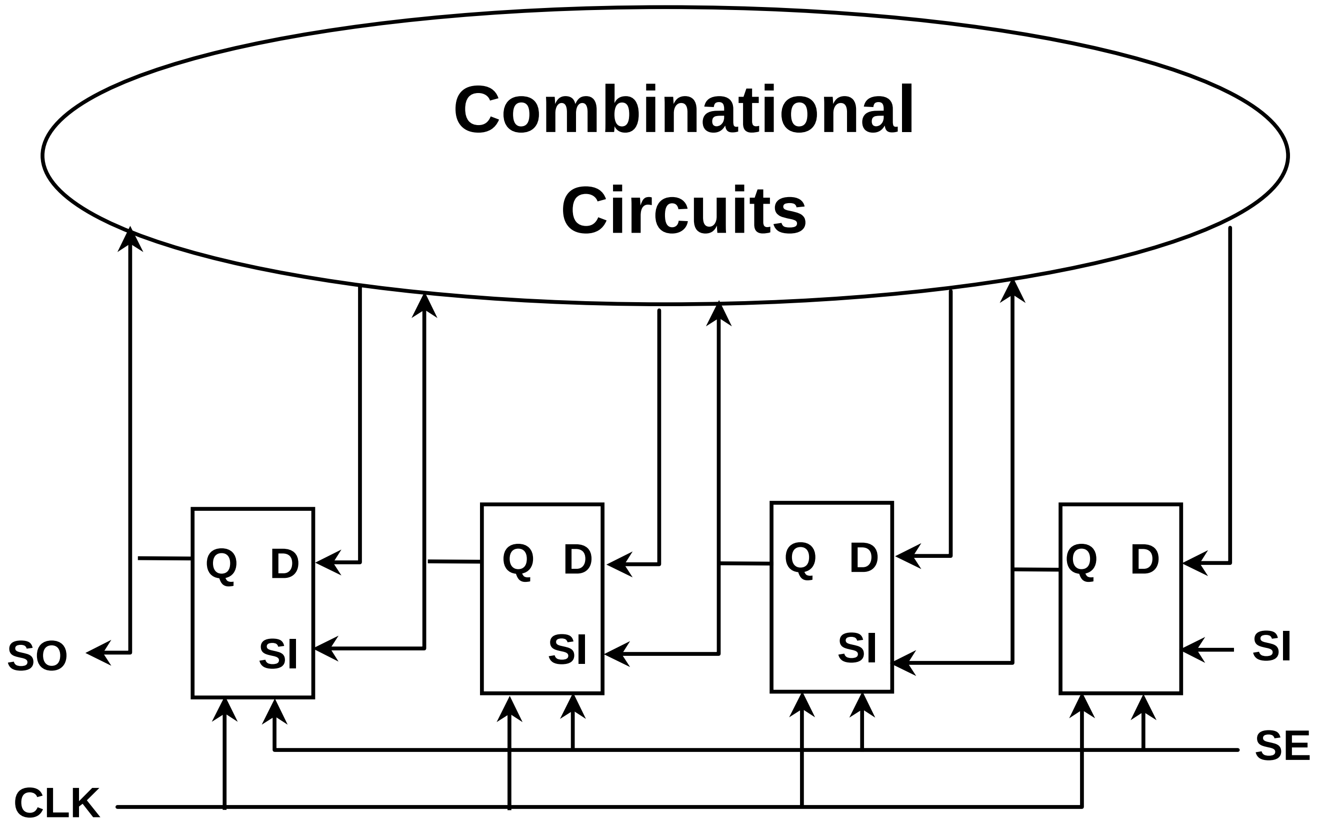




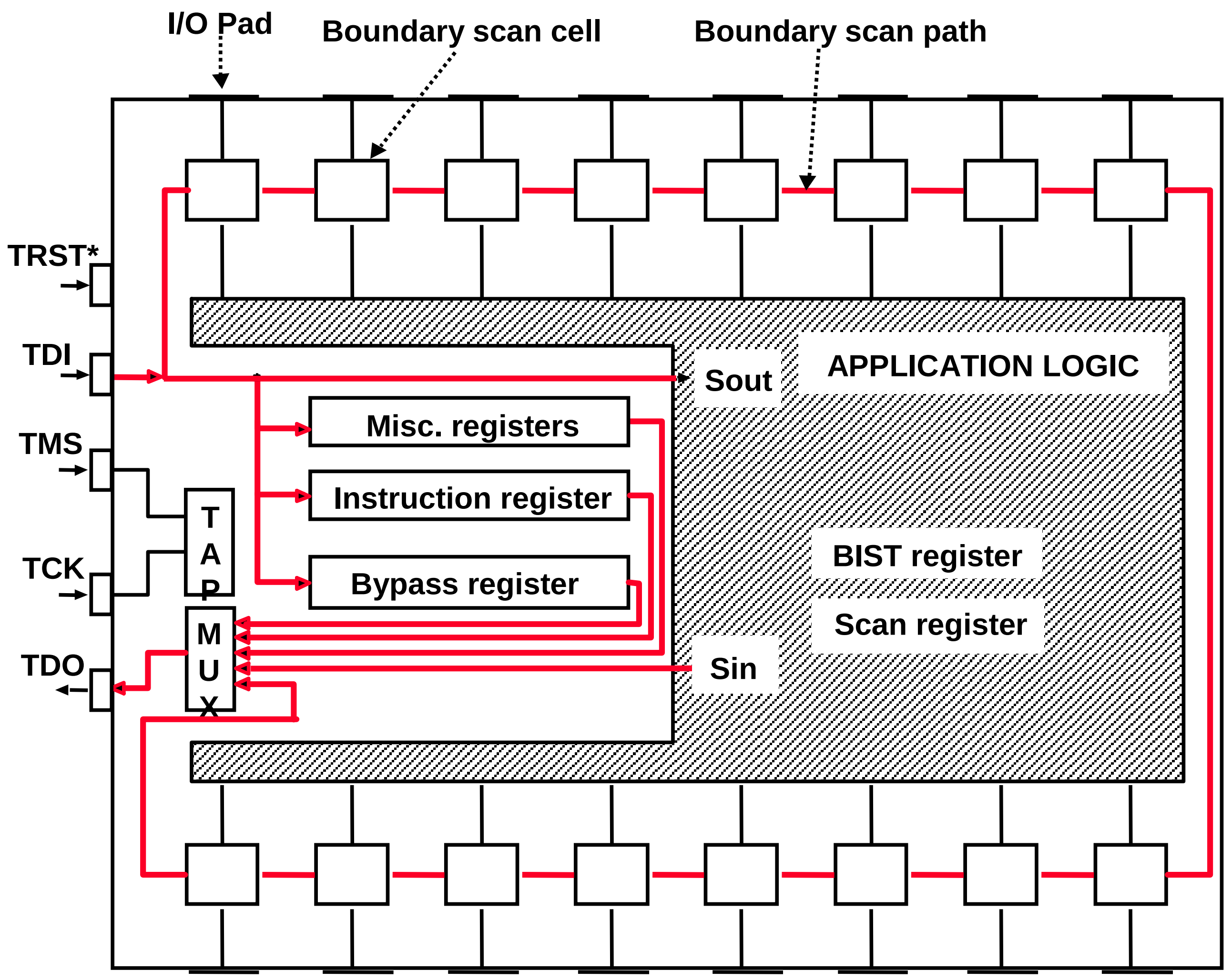
Scan Cell Design



Scan Register

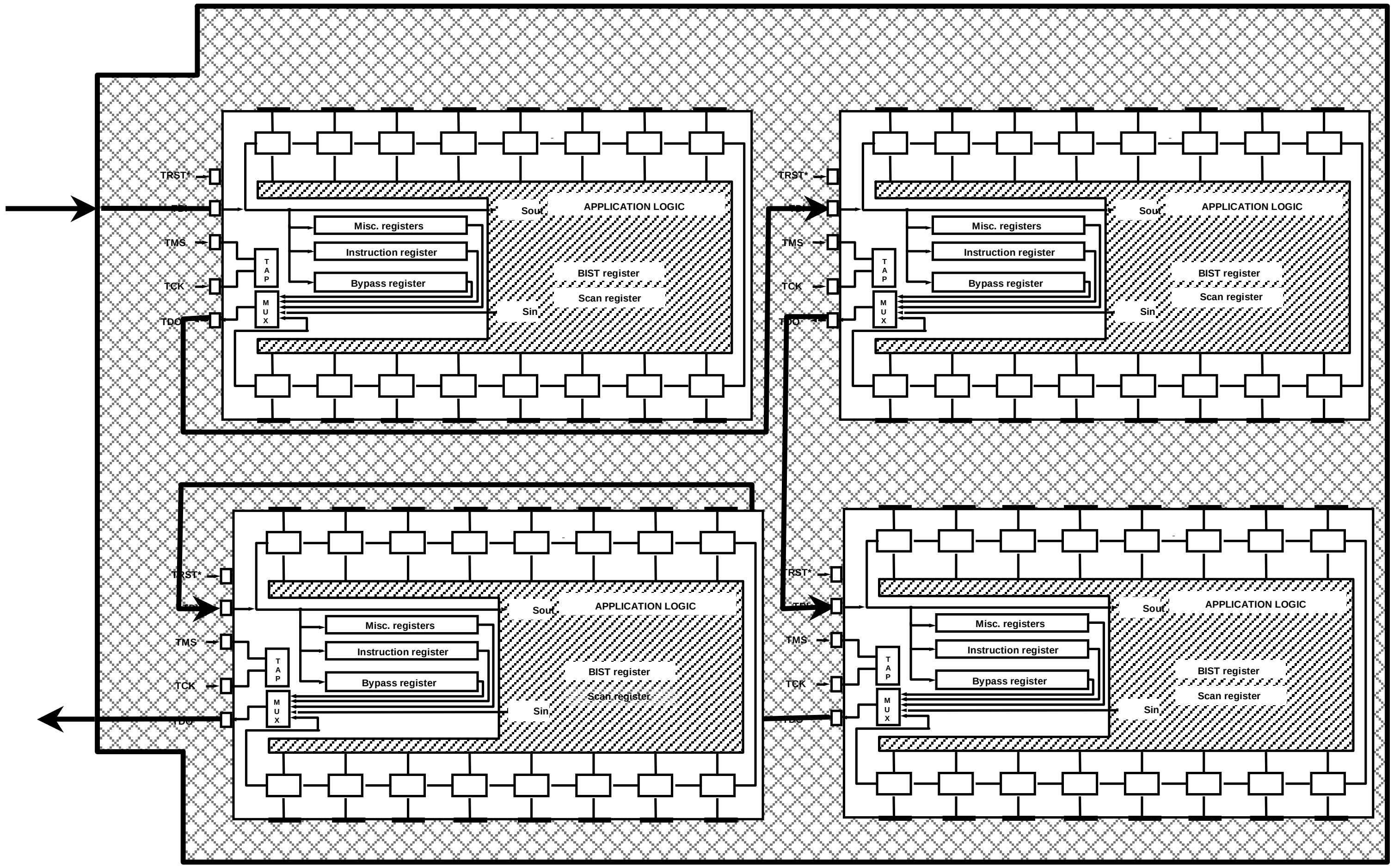


Boundary Scan



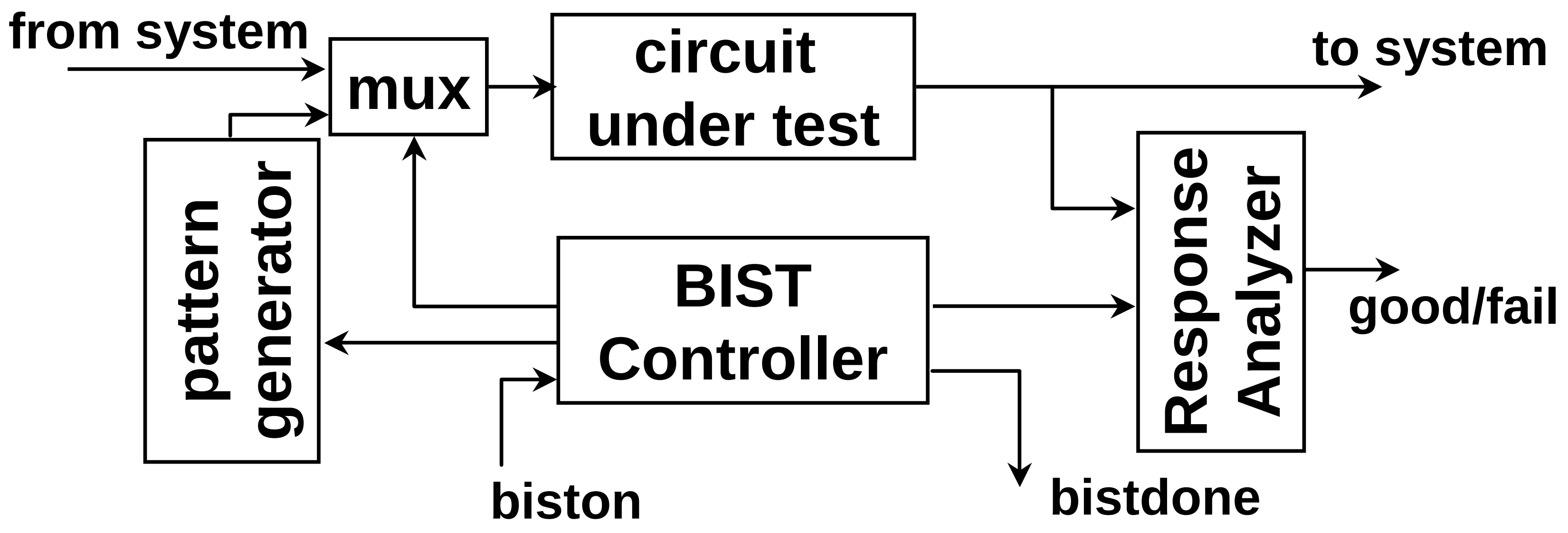
TRST*:Test rest (Optional)
TDI: Test data input
TD0: Test data output
TCK: Test clock
TMS: Test mode select

Boundary Scan (Cont.)



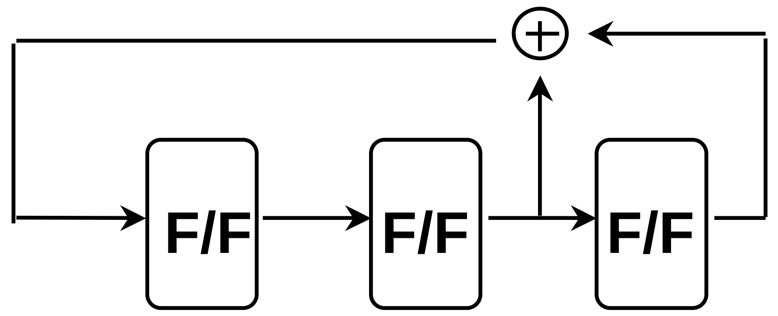
Built-In-Self Test (BIST)

- Places the job of device testing inside the device itself
- Generates its own stimulus and analyzes its own response

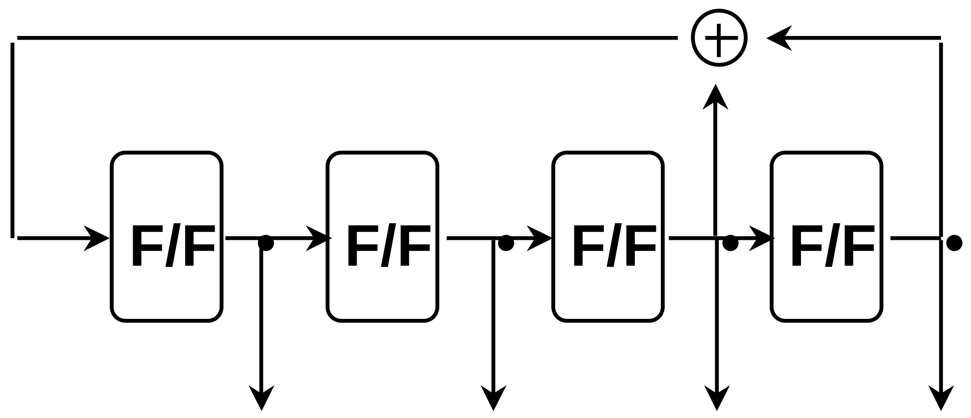


Built-In-Self Test (BIST) (Cont.)

- Two major tasks
  - Test pattern generation
  - Test result compaction
- Usually implemented by linear feedback shift register



Random Number Generator (RNG)

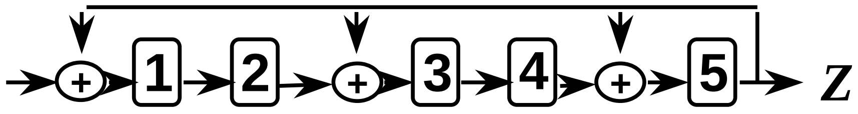


0001	0110	1111
1000	1011	0111
0100	0101	0011
0010	1010	0001
1001	1101	(repeat)
1100	1110	

1. Generate “pseudo” random patterns
2. Period is $2^n - 1$

Signature Analyzer (SA)

Input sequence 10101111 (8 bits)



$G(x) = 1 + x^2 + x^4 + x^5 + x^6 + x^7$ $\quad$ $P(x) = 1 + x^2 + x^4 + x^5$

Time	Input stream	Register contents	Output stream
0	1 0 1 0 1 1 1 1	0 0 0 0 0 ⟵	Initial state
1	1 0 1 0 1 1 1	1 0 0 0 0	
.	.	.	
.	.	.	
5	1 0 1	0 1 1 1 1	
6	1 0	0 0 0 1 0	1
7	1	0 0 0 0 1	0 1
8		$\underbrace{0\ 0\ 1\ 0\ 1}$	$\underbrace{1\ 0\ 1}$

Remainder
R(x) = x²+x⁴

Quotient
1+x²

Signature Analyzer (SA) (Cont.)

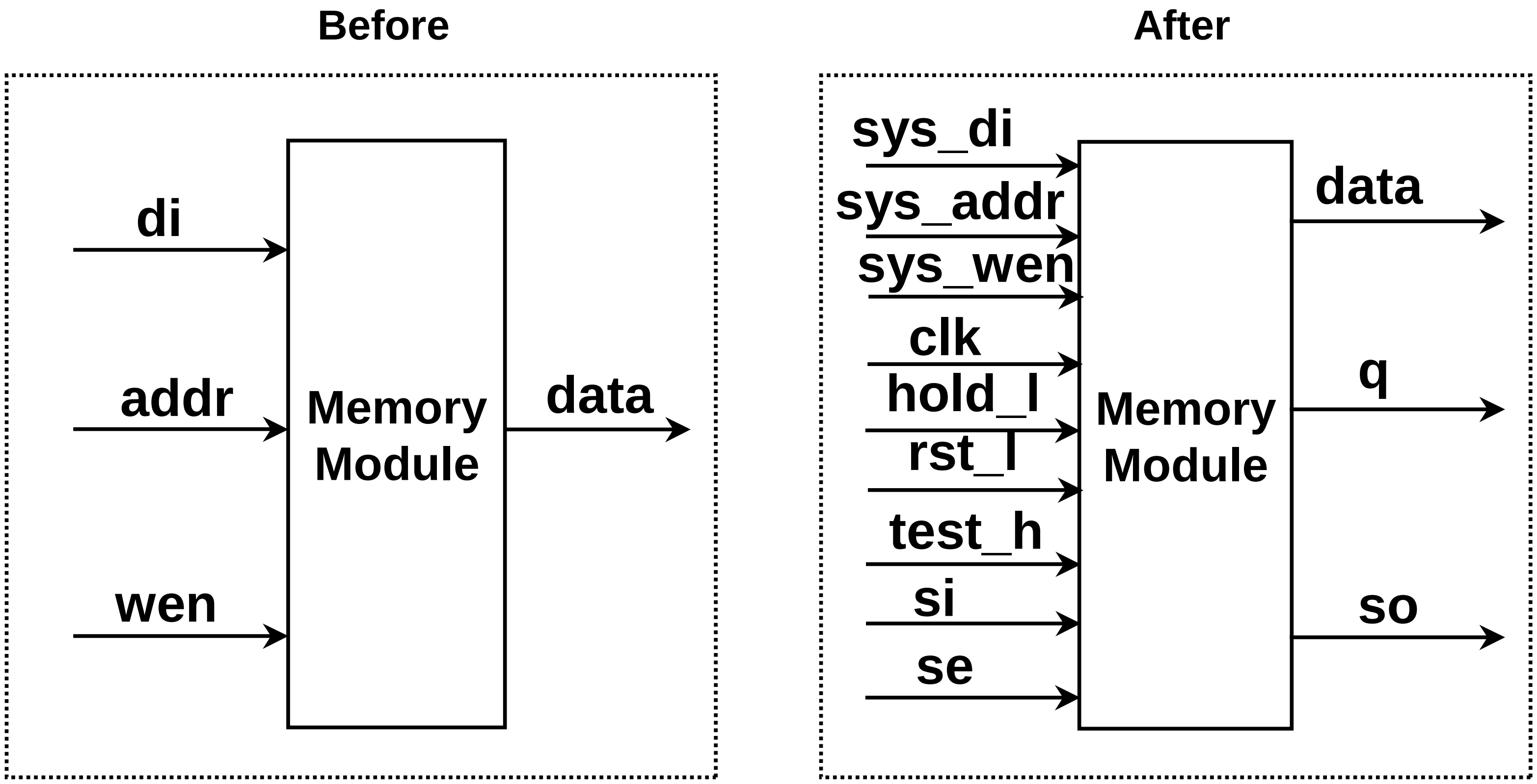
$$\begin{array}{rl} & P(x): x^5 + x^4 + x^2 + 1 \\ \times & Q(x): x^2 + 1 \\ \hline & x^7 + x^6 + x^4 + x^2 + x^5 + x^4 + x^2 + 1 \\ & = x^7 + x^6 + x^5 + 1 \end{array}$$

$$P(x)Q(x) + R(x) = x^7 + x^6 + x^5 + x^4 + x^2 + 1 = G(x)$$

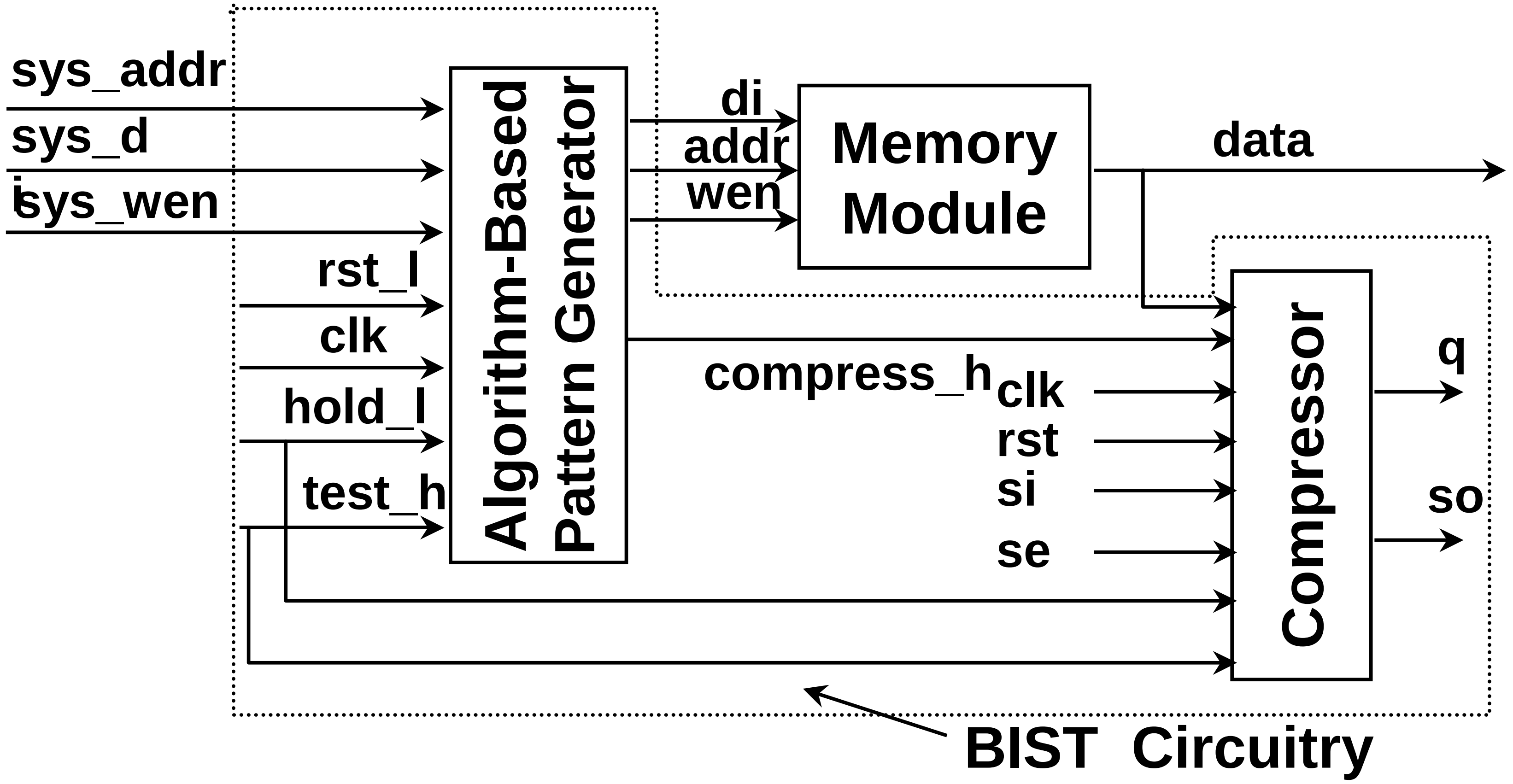
Prob. of aliasing error = 1/2^n

where n is # of FFs

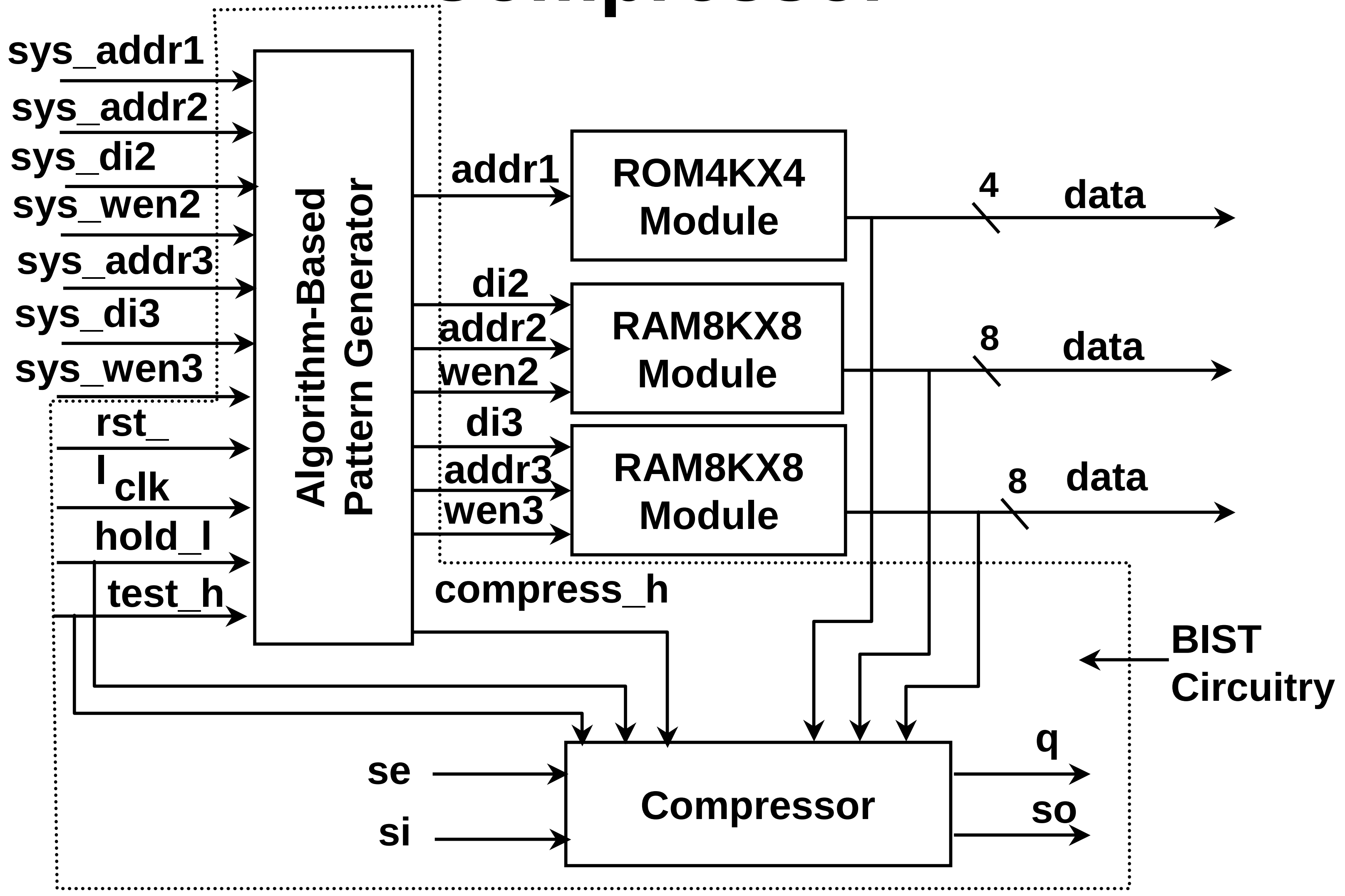
Memory BIST Architecture with a Compressor



Memory BIST Architecture with a Compressor (Cont.)



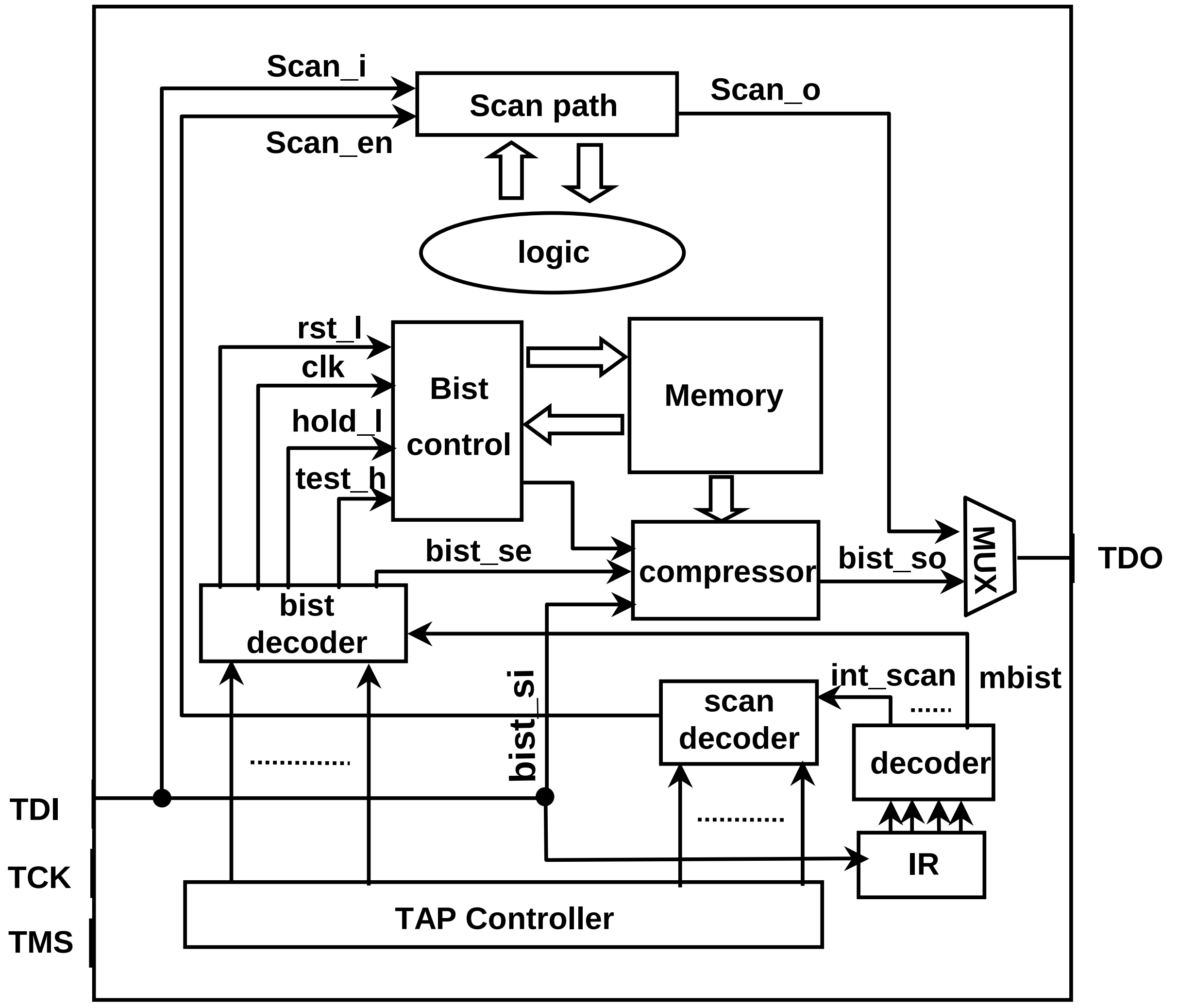
Three Memories and One Compressor



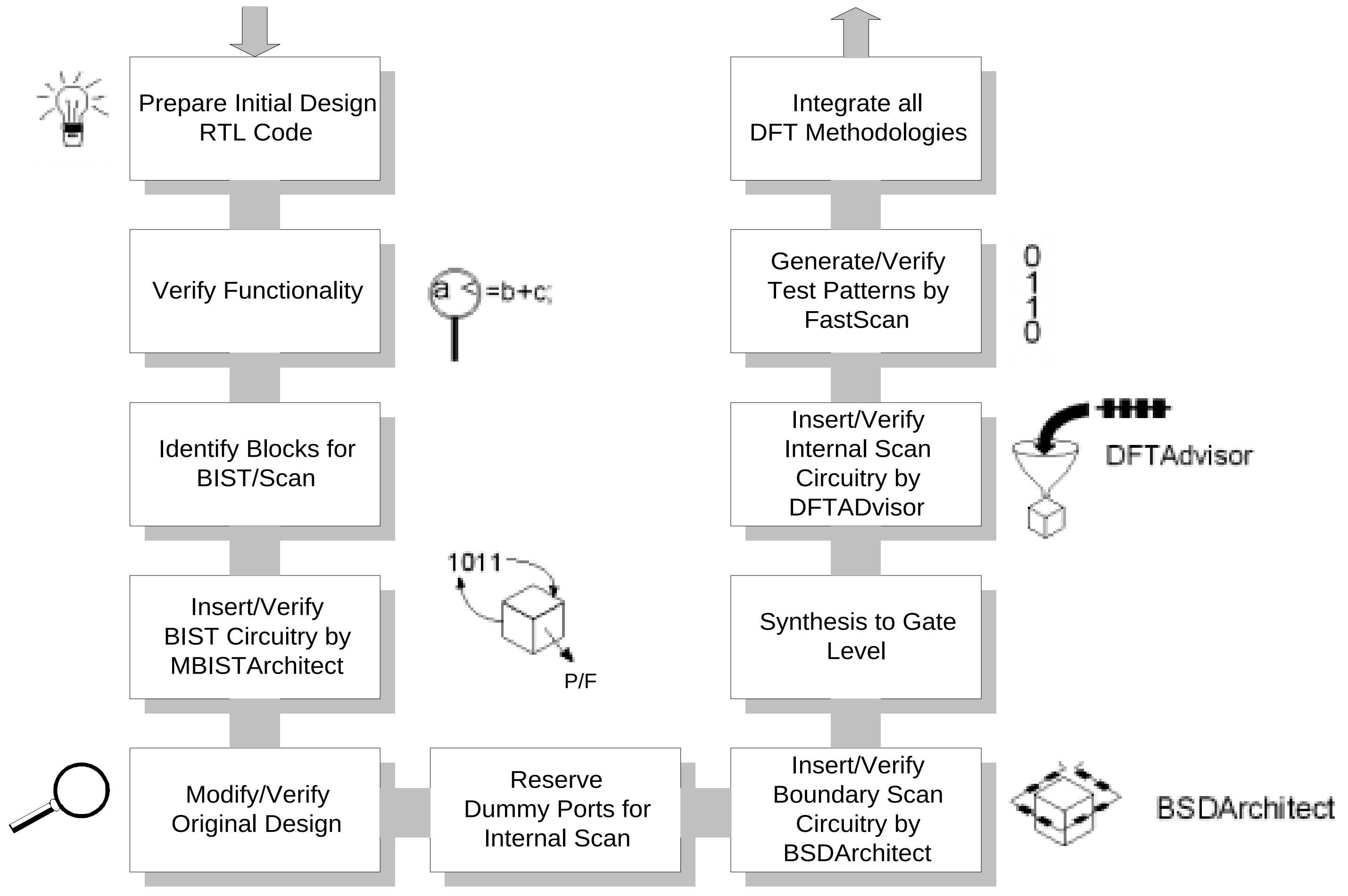
Synthesis for Testability

- Automatic v.s Semi-automatic
- Commercial products
    - Testability analysis tools
    - Full / partial scan insertion
    - BIST insertion
    - Boundary scan insertion
- Research
    - RTL synthesis
    - FSM synthesis
    - Gate level synthesis
    - Boolean equation synthesis

CPU Test Control Architecture



Testable Design Flow



Prepare Initial Design RTL Code
Verify Functionality
a =b+c;
Identify Blocks for BIST/Scan
Insert/Verify BIST Circuitry by MBISTArchitect
1011
P/F
Modify/Verify Original Design
Reserve Dummy Ports for Internal Scan
Insert/Verify Boundary Scan Circuitry by BSDArchitect
BSDArchitect
Synthesis to Gate Level
Insert/Verify Internal Scan Circuitry by DFTADvisor
DFTAdvisor
Generate/Verify Test Patterns by FastScan
0 1 1 0
Integrate all DFT Methodologies

Problems re-thinking

- A 32-bit adder --- ATPG
- A 32-bit counter --- Design for testability + ATPG
- A 1MB Cache memory --- BIST
- A 10^7-transistor CPU --- All test techniques

Conclusions

Two major fields in testing

- **ATPG**
  - --- Fault simulation
  - --- Test generation
- **Testable design**
  - --- Design for testability
  - --- Built-in self-test
  - --- Synthesis for testability