MIC05: Teste de Circuitos Integrados

Marcelo Lubaszewski

PPGMicro - UFRGS

2007/II

Costly Test Problems

- Increasing chip logic-to-pin ratio – harder observability
- Increasingly dense devices and faster clocks
- Increasing test generation and application times
- Increasing size of test vectors stored in ATE
- Expensive ATE needed for 1 GHz clocking chips
- Hard testability insertion – designers unfamiliar with gate-level logic, since they design at behavioral level
- *In-circuit testing* no longer technically feasible
- Circuit testing cannot be easily partitioned

Typical Quality Requirements

- **98% single stuck-at fault coverage**
- **100% interconnect fault coverage**
- **Reject ratio – 1 in 100,000**

Auto Teste Integrado

Requisitos para Equipamentos de Teste:

altas freqüências
precisão
memória
imunidade ao ruído

Alto Custo

Uso de Auto-teste integrado (*BIST*)

Funções do testador transferidas ao chip ou placa

BIST Motivation

- **Useful for field test and diagnosis (less expensive than a local automatic test equipment)**

- **Hardware BIST benefits:**
  - **Lower system test effort**
  - **Improved system maintenance and repair**
  - **Improved component repair**
  - **Better diagnosis**

BIST Benefits

- Faults tested:
  - Single combinational / sequential stuck-at faults
  - Delay faults
  - Single stuck-at faults in BIST hardware
- BIST benefits
  - Reduced testing and maintenance cost
  - Lower test generation cost
  - Reduced storage / maintenance of test patterns
  - Simpler and less expensive ATE
  - Can test many units in parallel
  - Shorter test application times
  - Can test at functional system speed

Economics – BIST Costs

- Chip area overhead for:
  - Test controller
  - Hardware pattern generator
  - Hardware response compacter
  - Testing of BIST hardware
- Pin overhead -- At least 1 pin needed to activate BIST operation
- Performance overhead – extra path delays due to BIST
- *Yield loss* – due to increased chip area or more chips In system because of BIST
- Reliability reduction – due to increased area
- Increased BIST hardware complexity – happens when BIST hardware is made testable

Benefits and Costs of BIST with DFT

Level	Design and test	Fabri-cation	Manuf. Test	Maintenance test	Diagnosis and repair	Service interruption
Chips	+ / -	+	-			
Boards	+ / -	+	-		-	
System	+ / -	+	-	-	-	-

+ Cost increase
- Cost saving
+/- Cost increase may balance cost reduction

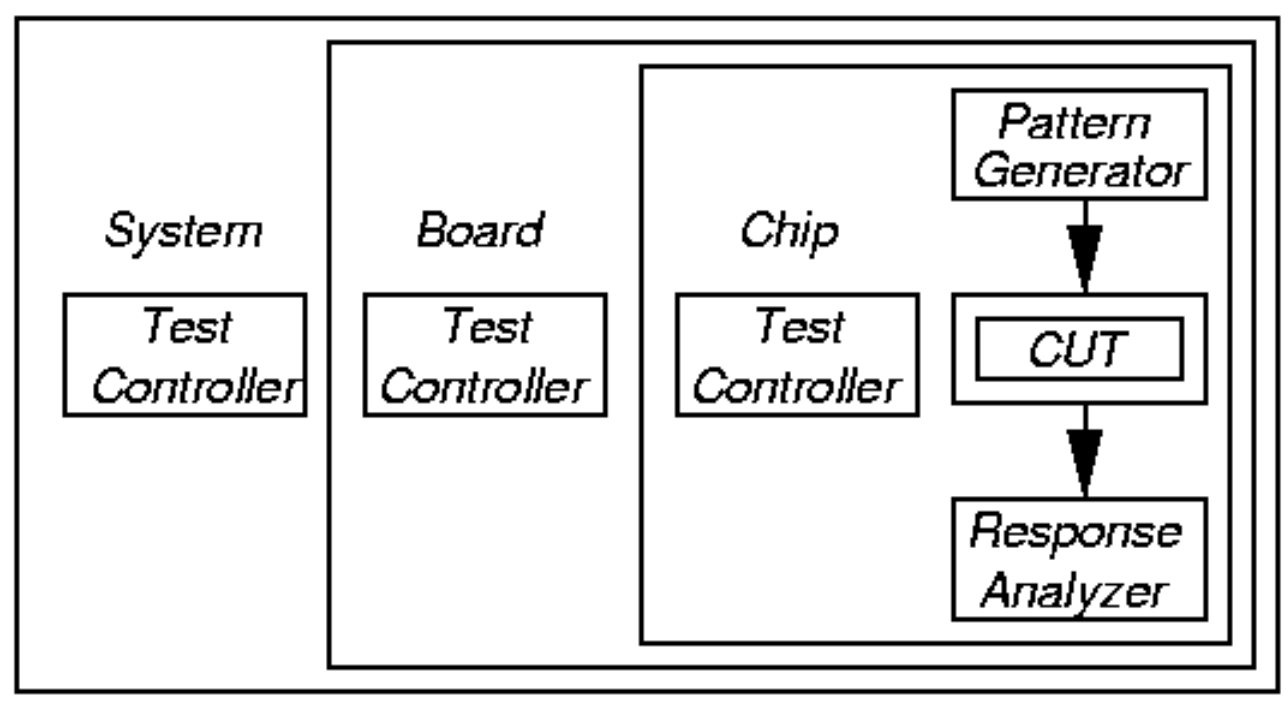
Definitions

- CUT – *Circuit-under-test*
- *Off-line testing:* circuit stops normal operation to be tested
- *Concurrent or on-line* testing – Testing process that detects faults during normal system operation

More Definitions

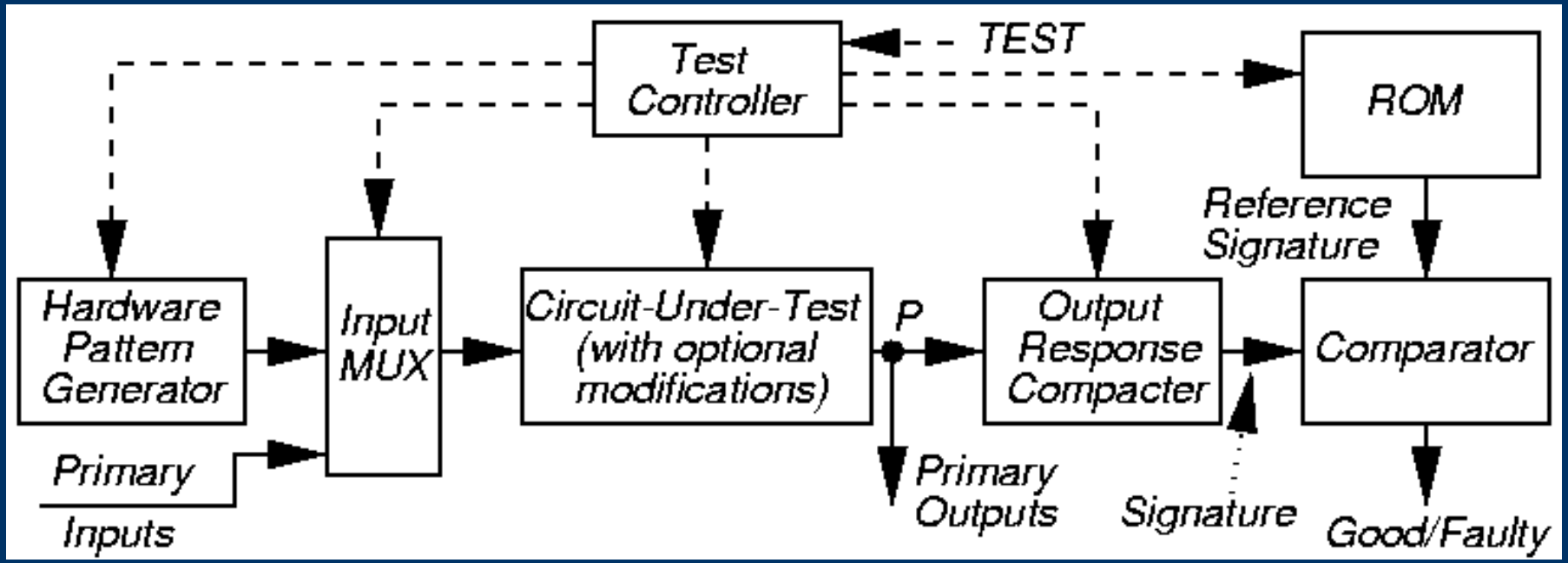
- *Exhaustive testing* – Apply all possible 2^n patterns to a circuit with n inputs
- *Pseudo-exhaustive testing* – Break circuit into small, overlapping blocks and test each exhaustively
- *Pseudo-random testing* – Algorithmic pattern generator that produces a subset of all possible tests with most of the properties of randomly-generated patterns
- *Signature* – Any statistical circuit property distinguishing between bad and good circuits
- TPG – Hardware *test pattern generator*
- ORA – Output Response Analyser

BIST Process



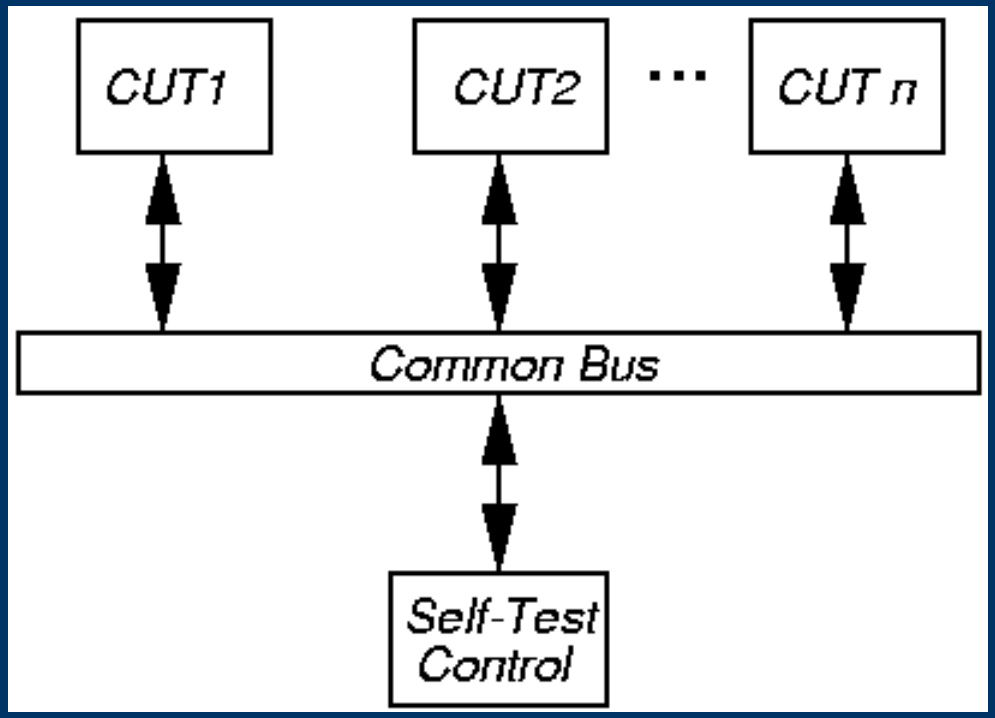
- *Test controller* – Hardware that activates self-test simultaneously on all PCBs
- Each board controller activates parallel chip BIST Diagnosis effective only if very high fault coverage

BIST Architecture



- Note: BIST cannot test wires and transistors:
  - From PI pins to Input MUX
  - From POs to output pins

Bus-Based BIST Architecture



- *Self-test control* broadcasts patterns to each CUT over bus – parallel pattern generation
- Awaits bus transactions showing CUT's responses to the patterns: serialized compaction

Diagnosis

- BIST must be implemented at all (or most) levels
  - The more the better
- Test is performed in one level at a time and reported to the higher level

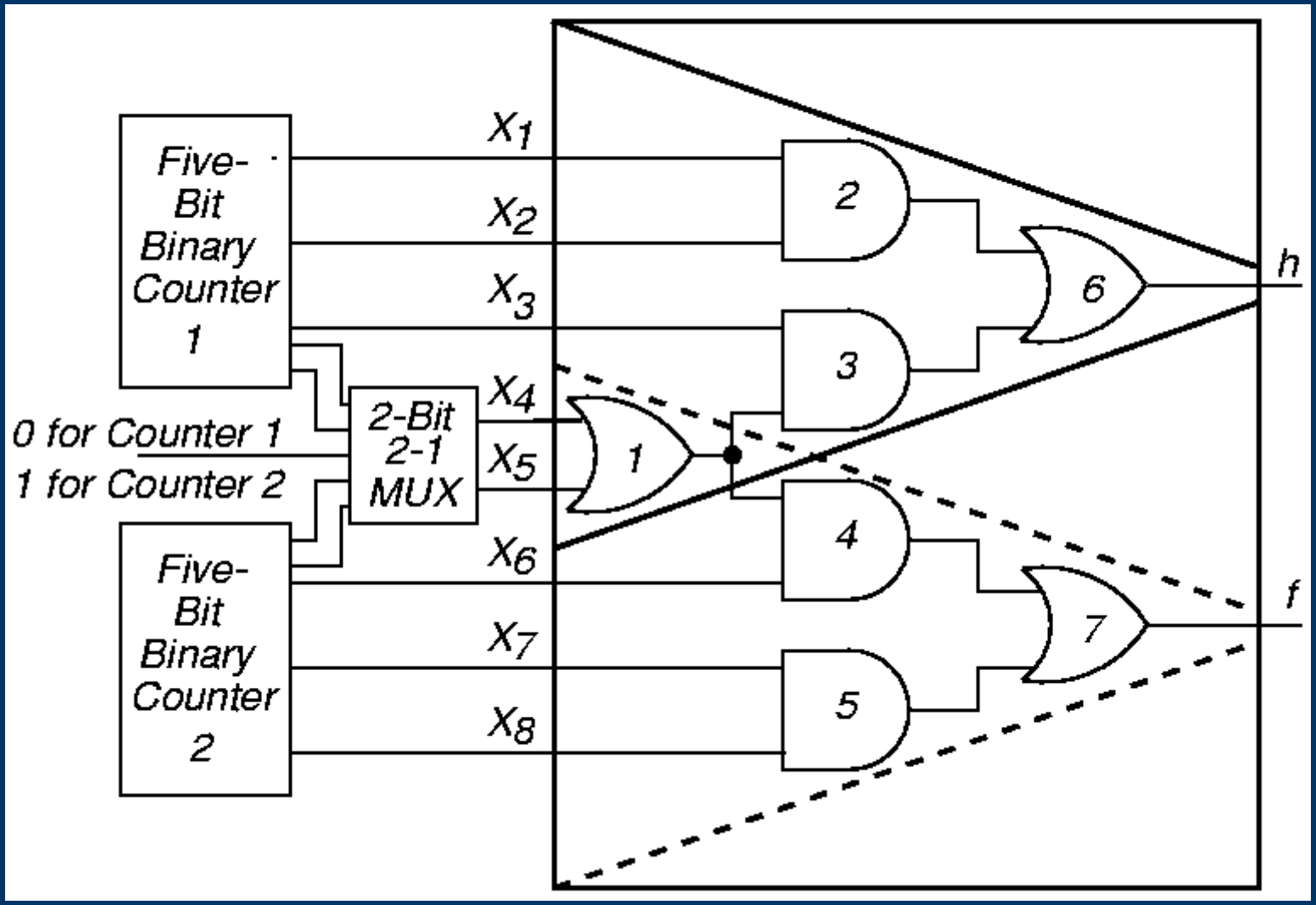
Pattern Generation

- Store in ROM – too expensive
- *Exhaustive – impractical for n >20*
- *Pseudo-exhaustive*
- *Pseudo-random* (LFSR) – Preferred method
- Test pattern *augmentation*
  - LFSR combined with a few patterns in ROM

Pseudo-Exhaustive Method

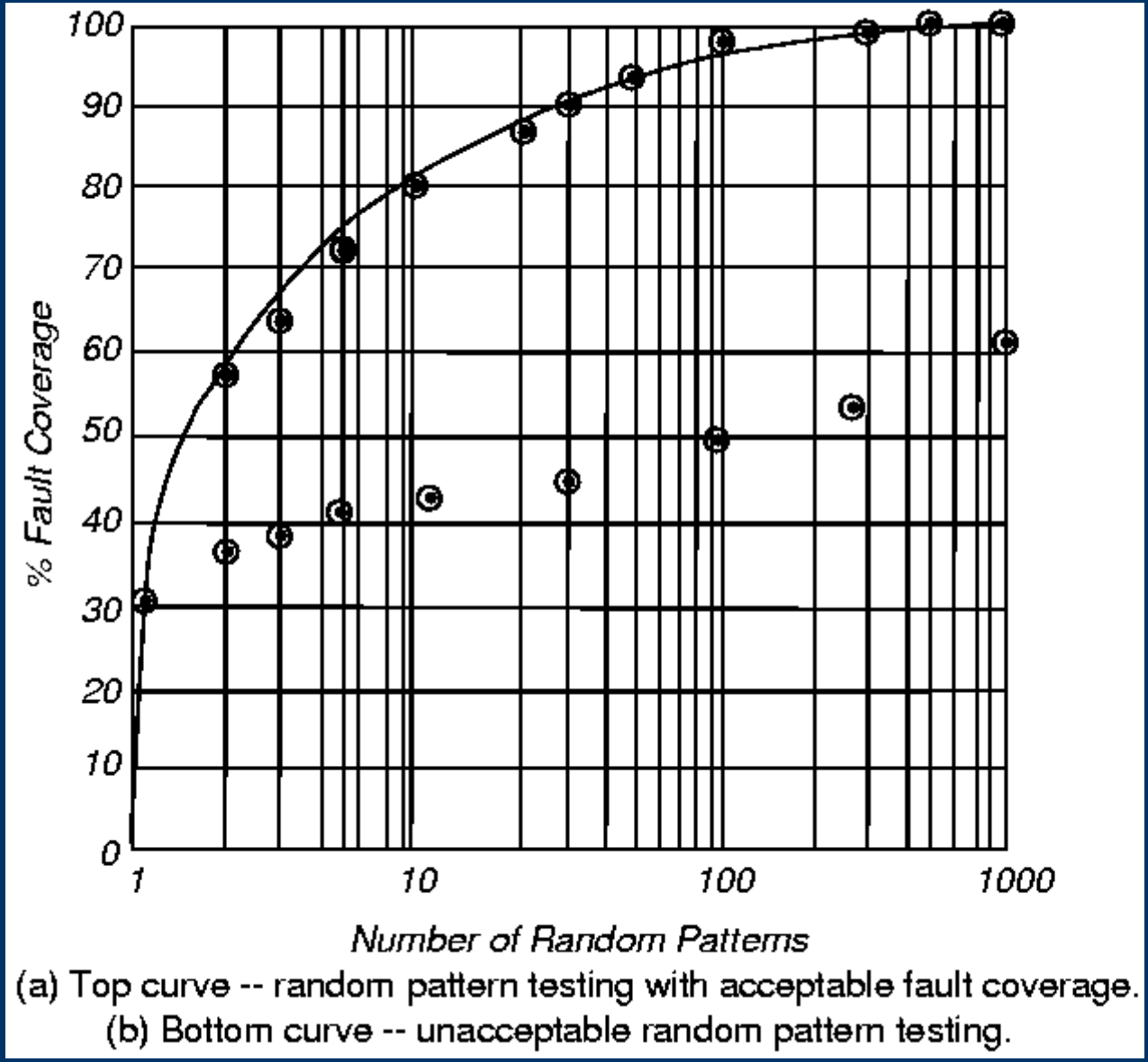
- Partition large circuit into *fanin cones*
  - Backtrace from each PO to PIs influencing it
  - Test fanin cones in parallel
- Reduced # of tests from $2^8 = 256$ to $2^5 \times 2 = 64$
  - Incomplete fault coverage

Pseudo-Exhaustive Pattern Generation



Random Pattern Testing

Bottom: Random-Pattern Resistant circuit



(a) Top curve -- random pattern testing with acceptable fault coverage.
(b) Bottom curve -- unacceptable random pattern testing.

Random Patterns Generation

- Just guess bits values
  - Not repeatable!
- Algorithmically
  - Define a seed
  - Operate over the seed
  - ULA
- Cheaper hardware?

Polynomials

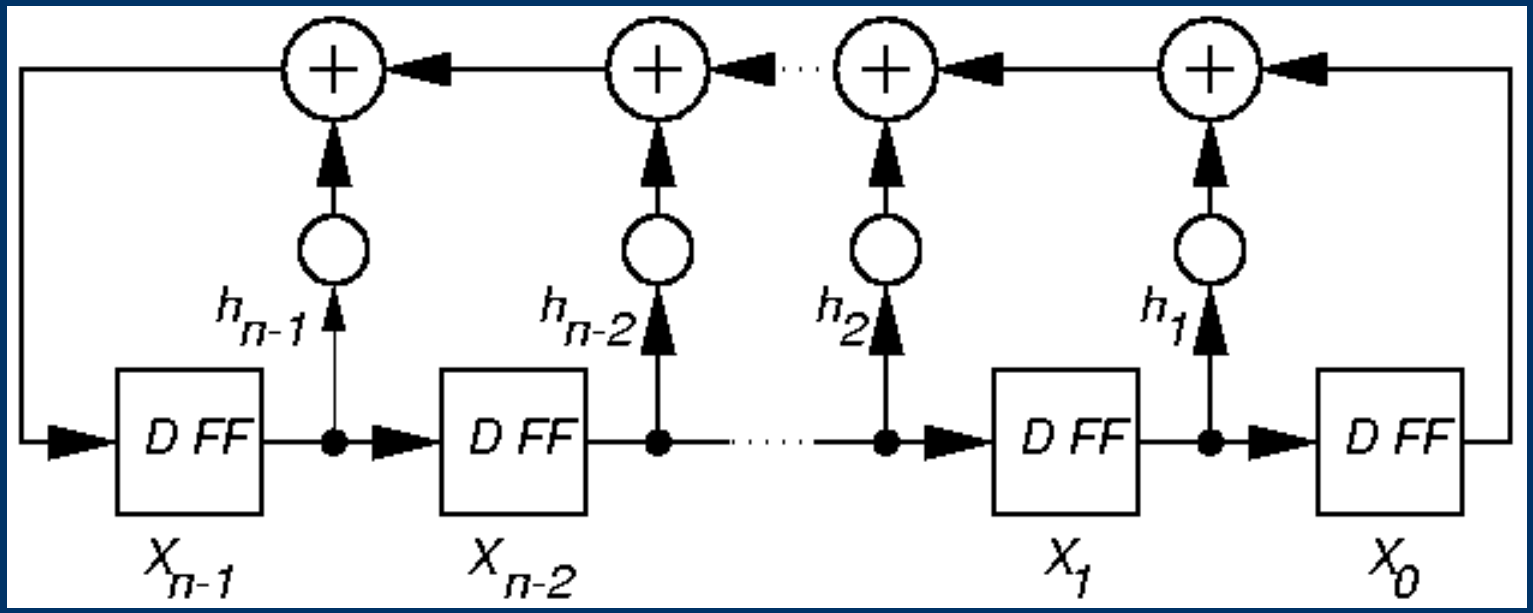
- **Vector = binary number**
- **Binary number can be read as a polinomial module 2**
- **$101001 = 1\,x^5 + 0x^4 + 1\,x^3 + 0x^2 + 0x^1 + 1x^0$**
- Module 2 sum = XOR!
- So, the binary number can be written as:

$$1\,x^5 \oplus 0x^4 \oplus 1\,x^3 \oplus 0x^2 \oplus 0x^1 \oplus 1x^0$$

Polynomials

- How can we represent or implement such a sum in HW?
- Coeficients must be stored: FFs
- Sums must be performed: XOR gates!
- So, a LFSR can represent a polynomial and vice-versa

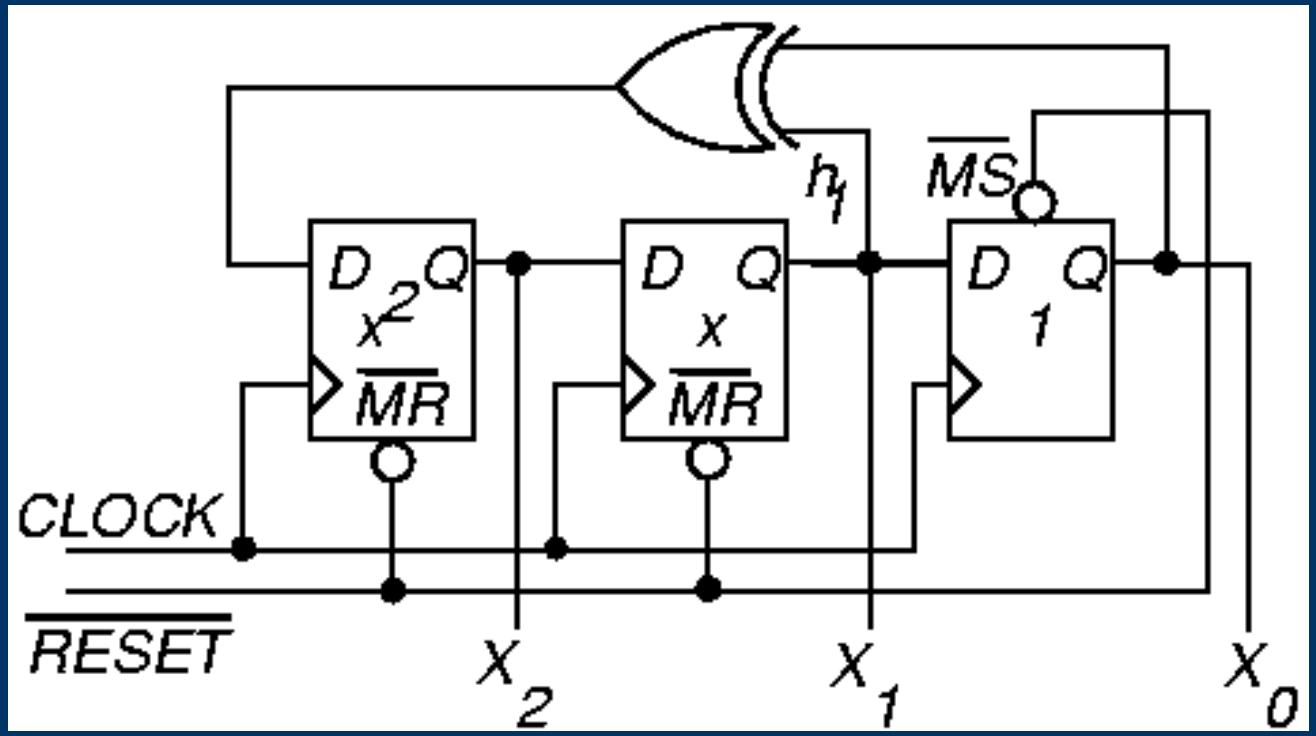
Generic LFSR



- *Standard Linear Feedback Shift Register* (LFSR) (Type 1)
- The location of the XOR gates represent the non-zero coefficients of the associated polynomial (characteristic polynomial)
- Ex: $x^3 + x^2 + x + 1$ or $x^3 + x + 1$

LFSR as TPG

- LFSR has cyclic and repetitive behavior
- Ex: characteristic polynomial = $x^3 + x + 1$

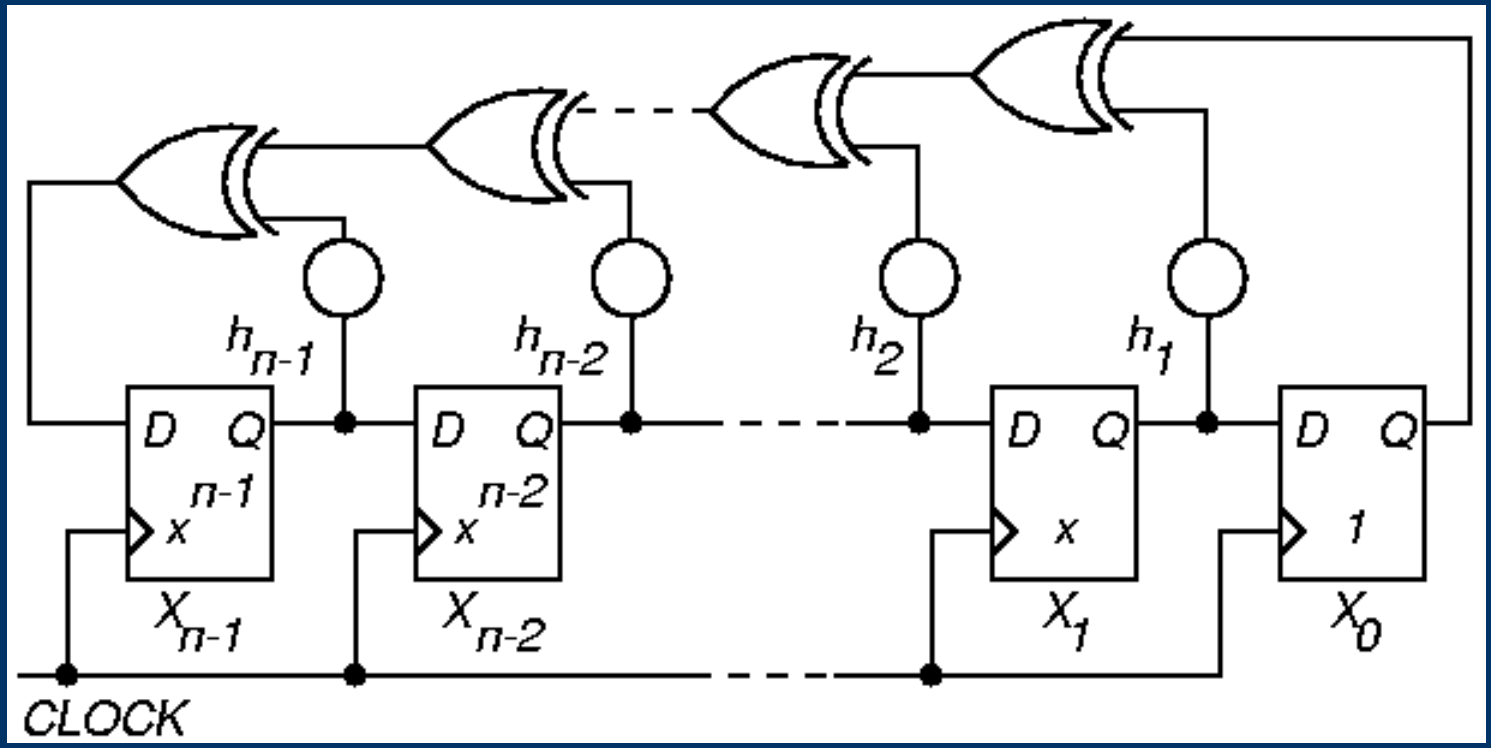


LFSR as TPG

- LFSR has cyclic and repetitive behavior
- Ex: characteristic polynomial = $x^3 + x + 1$
- Does it remind you of something?
  - Maybe patterns...

- What if initial state is 000????
- LFSR can be used as a pseudo-random TPG!

Standard *n*-Stage LFSR Implementation



- If $h_i = 0$, that XOR gate is deleted

Still LFSRs

- All LFSR is cyclic but the number of distinct patterns generated varies...
- Ex: For the same 3 bits, let´s try the characteristic polynomial: $x^3 + x^2 + x + 1$
- Start with 010
- Then with 001
- What do we want for testing?

Primitive Polynomial

- The more distinct patterns the better
- A LFSR can generate all $2^N - 1$ distinct values when the characteristic polynomial is a primitive one.
- Primitive polynomial: it divides the polynomial $1 + x^k$, for $k = 2^N - 1$
- $x^3 + x^2 + x + 1$?
- $x^3 + x + 1$?

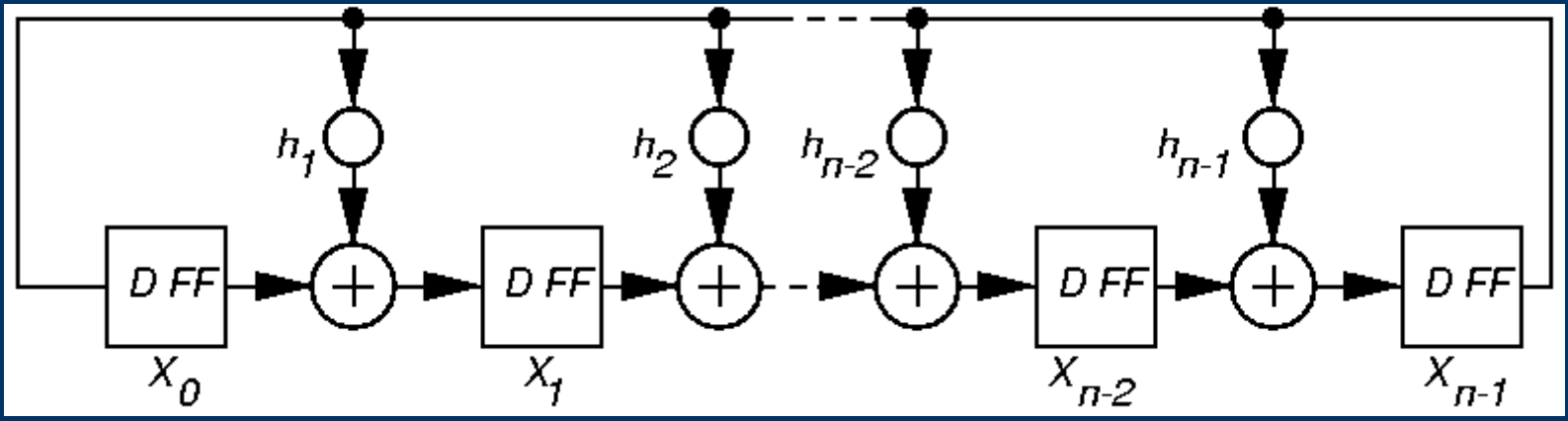
Primitive Polynomials

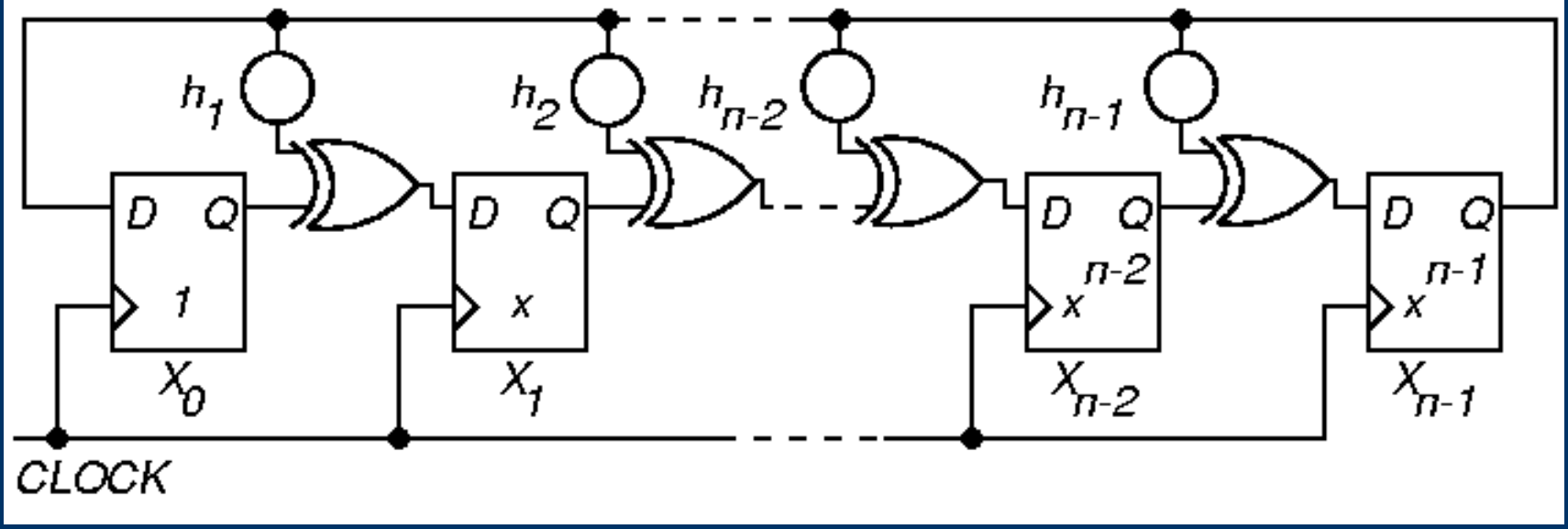
- To find a primitive polynomial is NP-complete problem!
- 8-bit LFSR has 16 primitive polyn.
- 16-bit LFSR has 2048
- 32-bit LFSR has 67108864
- Which polynomial must we use?????
- Look up for the tables!!!

Notation

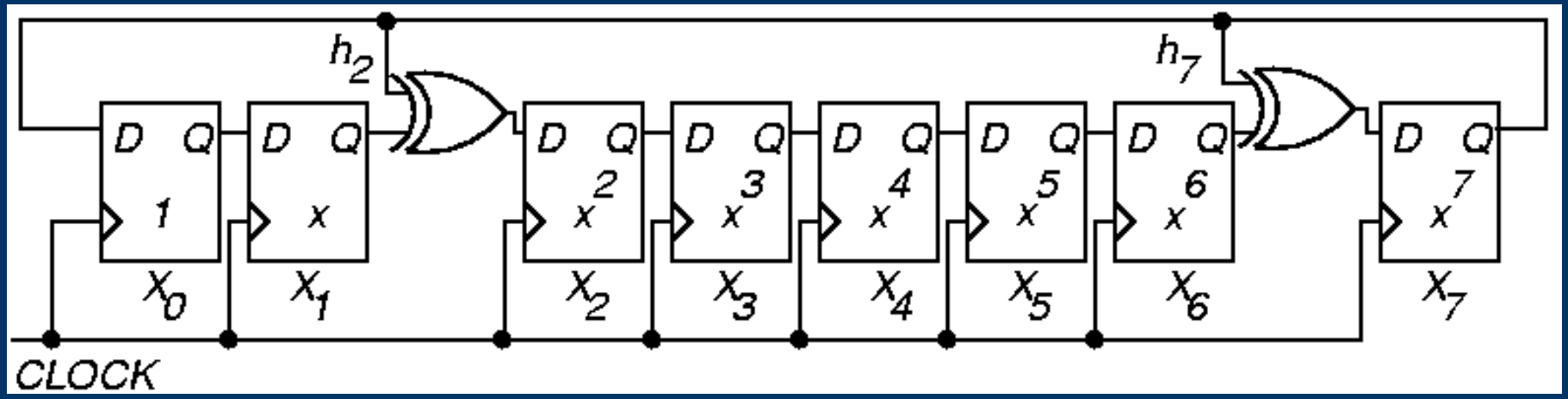
- 3: 1 0
  - $x^3 + x + 1$
- 12:7 4 3 0
  - $x^{12} + x^7 + x^4 + x^3 + 1$
- First and last coefficients are ALWAYS there!
- Now, define the LFSR representing the following polyonomial:
  - 5: 2 0
- What if we read the FFs from left to right?!?!?!?

Generic Modular LFSR (type 2)





Example Modular LFSR



- $f(x) = 1 + x^2 + x^7 + x^8$
- **Read LFSR tap coefficients from left to right**
- **Same theory applies!**

Test Pattern Augmentation

- **Secondary ROM – to get LFSR to 100% SAF coverage**
  - **Add a small ROM with missing test patterns**
  - **Add extra circuit mode to *Input MUX* – shift to ROM patterns after LFSR done**
  - **Important to compact extra test patterns**

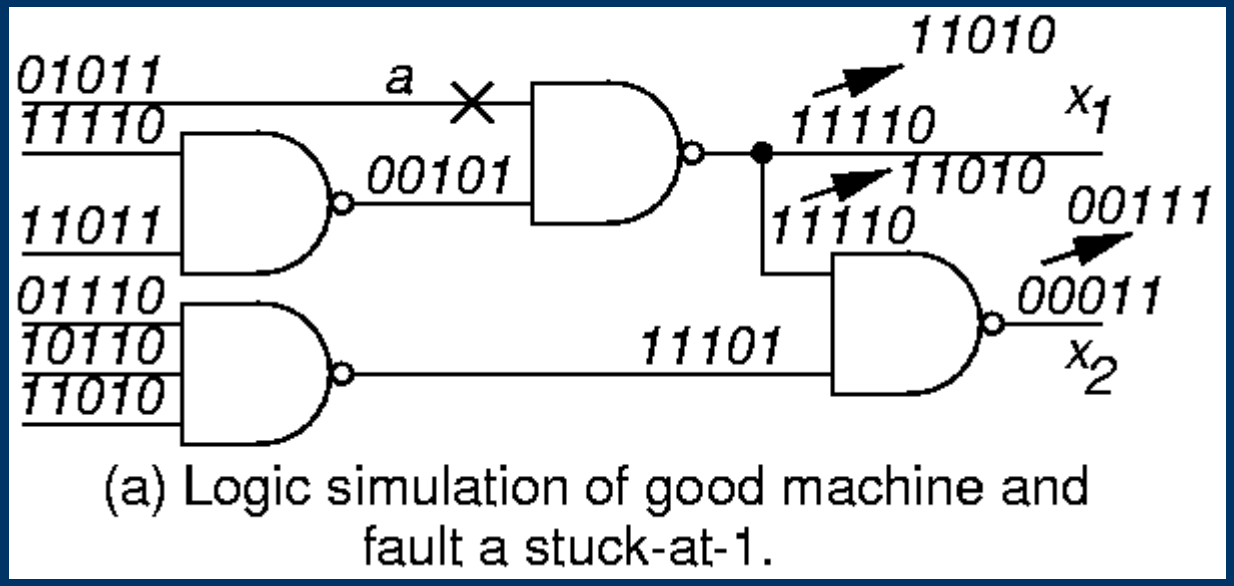
Response Compaction

- Severe amounts of data in CUT response to LFSR patterns – example:
  - Generate 5 million random patterns
  - CUT has 200 outputs
  - Leads to: 5 million x 200 = 1 billion bits response
- Uneconomical to store and check all of these responses on chip
- Responses must be compacted

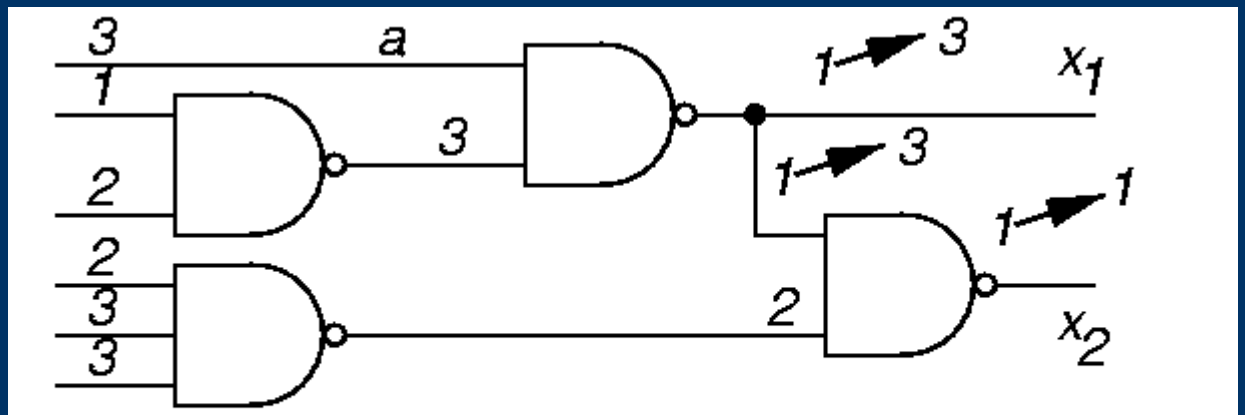
Definitions

- *Signature -* Any statistical circuit property distinguishing between bad and good circuits
- *Aliasing* – Due to information loss, signatures of good and some bad machines match
- *Compaction* – Drastically reduce # bits in original circuit response – loose information
- *Compression* – Reduce # bits in original *circuit response* – *no information loss – fully invertible* (can get back original response)
- *Signature analysis* – Compact good machine response into *good machine signature*. Actual signature generated during testing, and compared with good machine signature
- *Transition Count Response Compaction* – Count # transitions from 0 → 1 and 1 → 0 as a signature

Transition Counting



(a) Logic simulation of good machine and fault a stuck-at-1.



(b) Transition counts of good and failing machines.

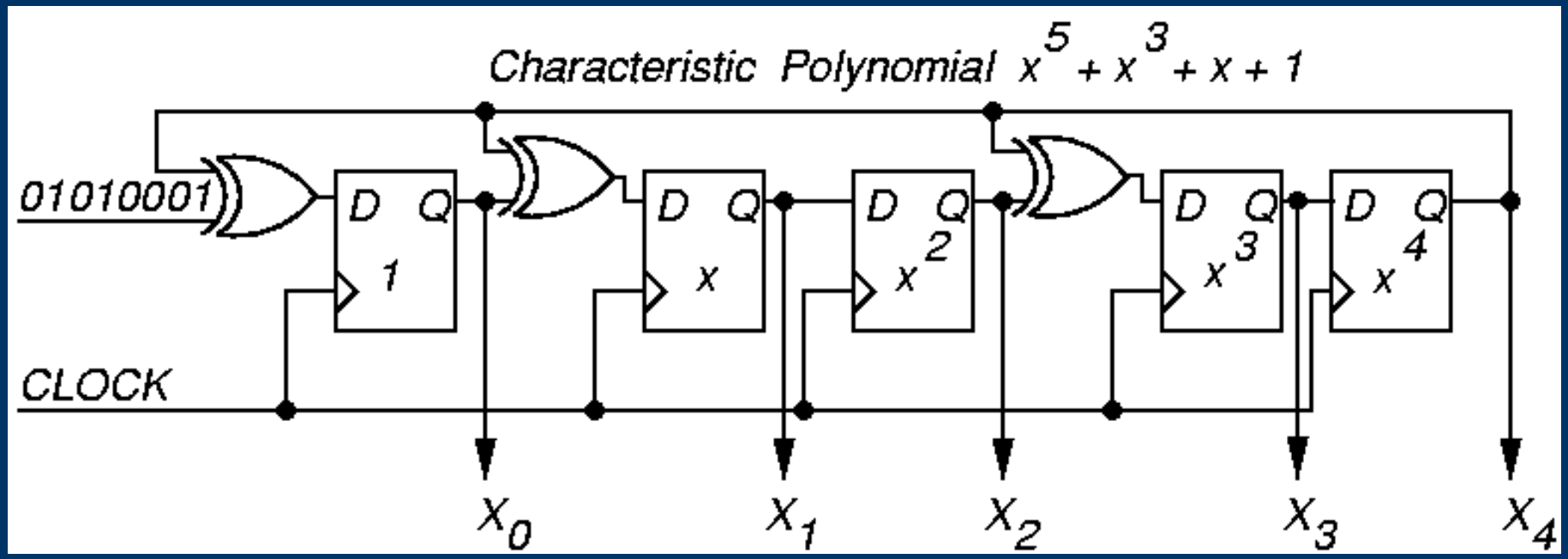
Digital Integration

- Add up all outputs

LFSR for Response Compaction

- Use *cyclic redundancy check code* (CRCC) generator (LFSR) for response compacter
- Treat data bits from circuit POs to be compacted as a decreasing order coefficient polynomial
- CRCC divides the PO polynomial by its characteristic polynomial
  - Leaves remainder of division in LFSR
  - Must initialize LFSR to *seed value* (usually 0) before testing
- After testing – compare signature in LFSR to known good machine signature
- Critical: Must compute good machine signature

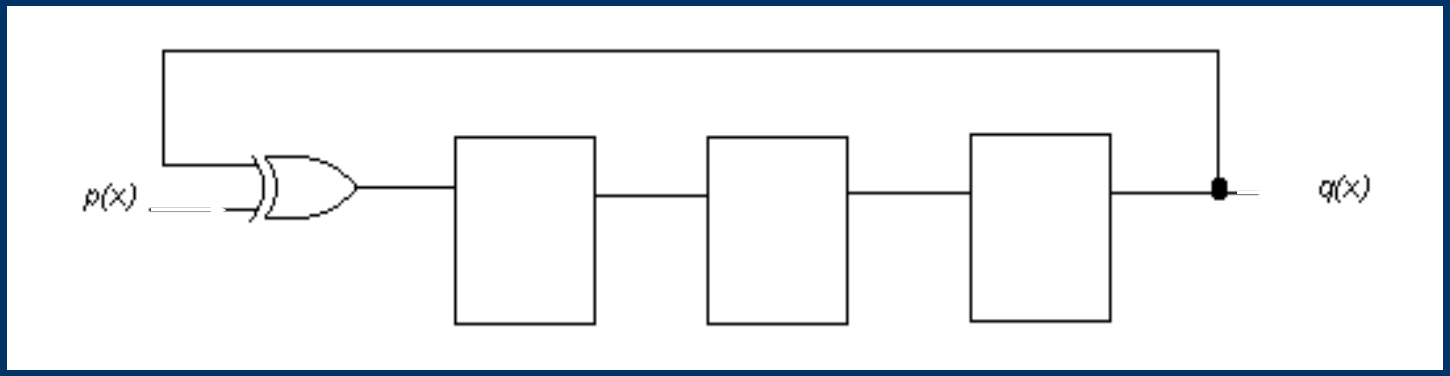
Example Modular LFSR Response Compacter



- LFSR seed value is “00000”
- Usually Type-2 LFSR is used

Auto Teste Integrado

Divisão Polinomial:

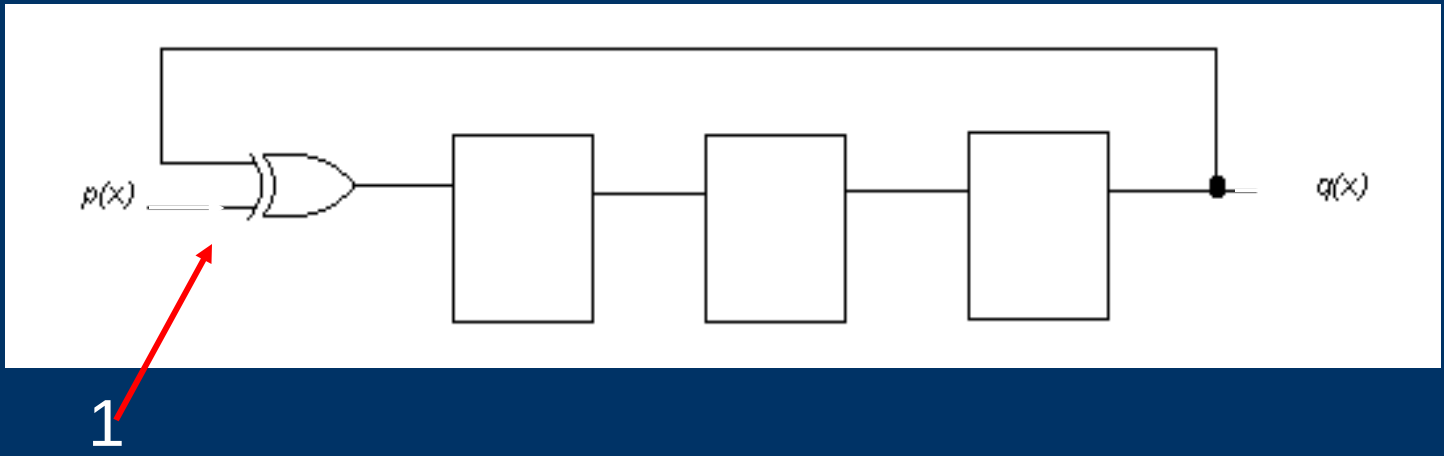


resposta

1
0
0
1
1
0

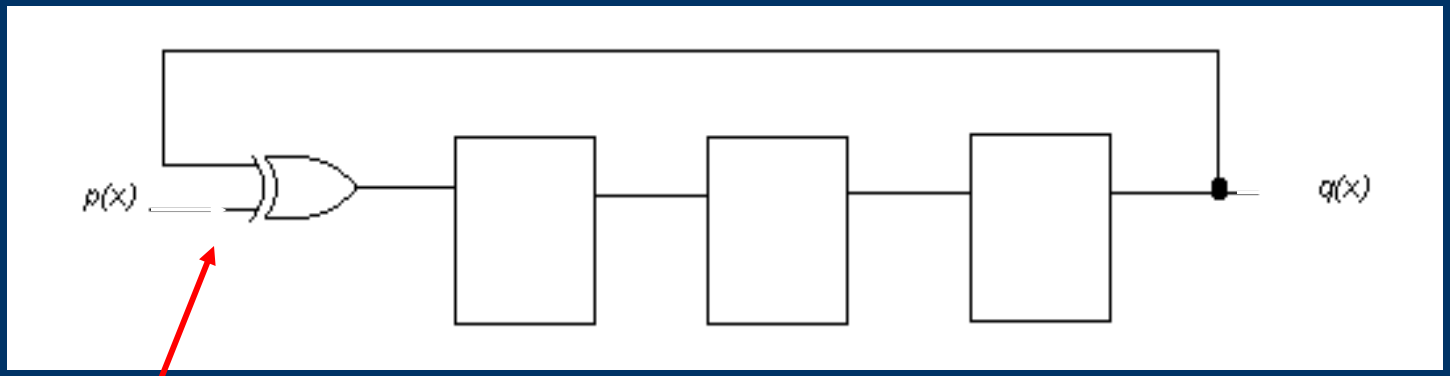
Auto Teste Integrado

Divisão Polinomial:



Auto Teste Integrado

Divisão Polinomial:



1

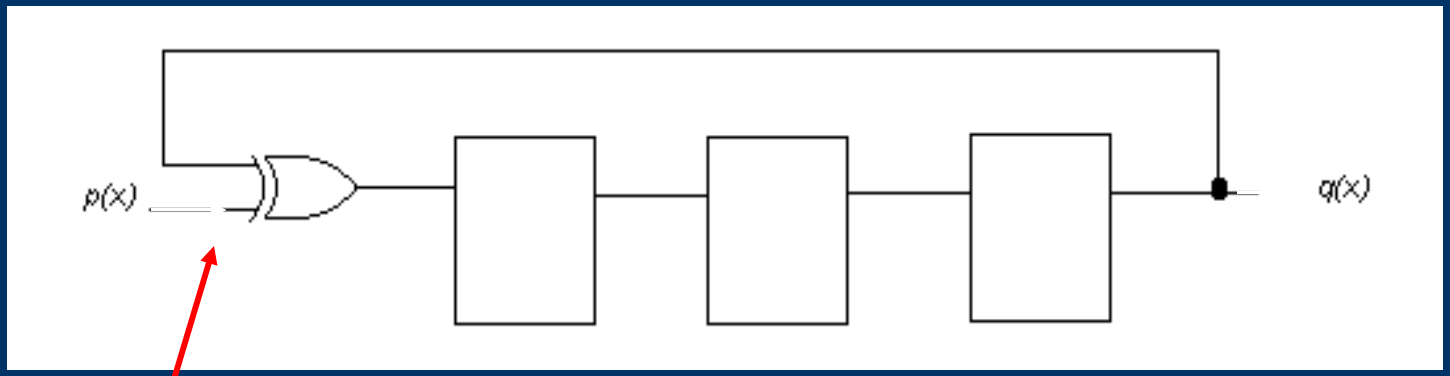
0

0

q(x)

Auto Teste Integrado

Divisão Polinomial:



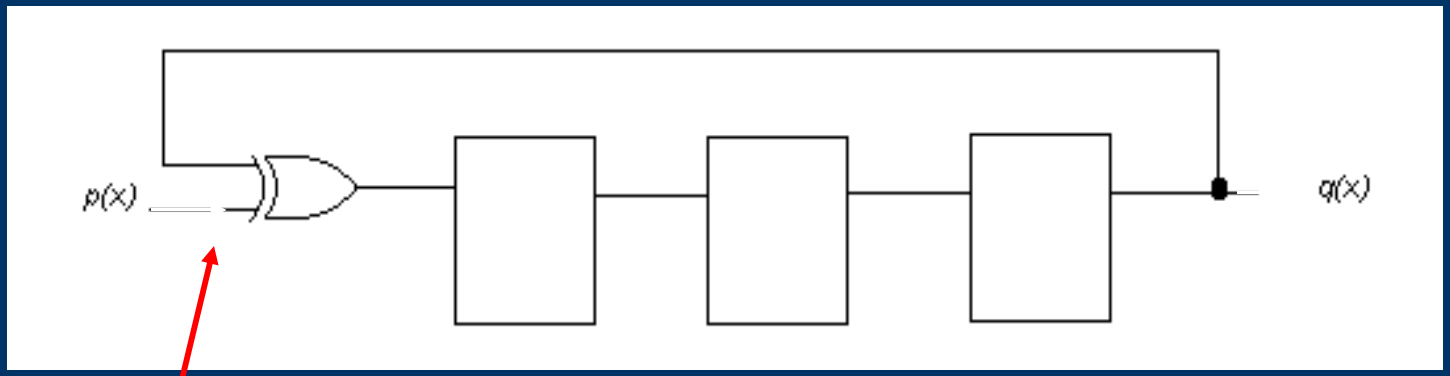
1
0
0

0
0

q(x)

Auto Teste Integrado

Divisão Polinomial:



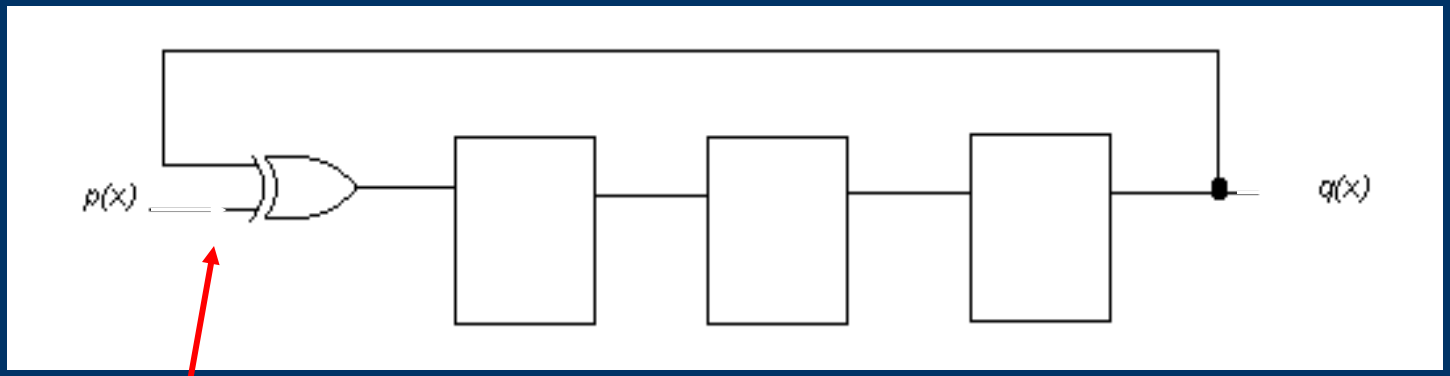
1
0
0
1

0
0
0

q(x)

Auto Teste Integrado

Divisão Polinomial:



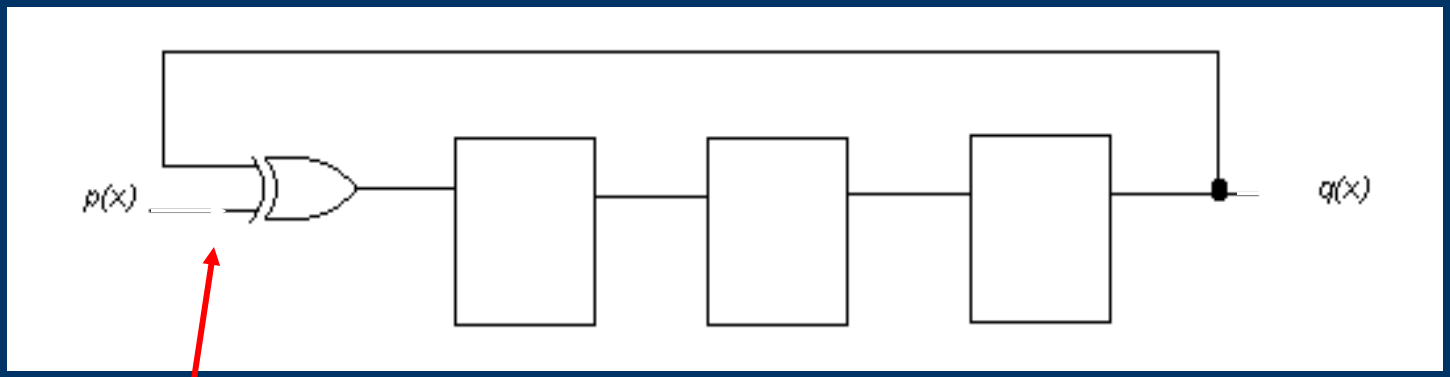
1
0
0
1
1

0
0
0
1

q(x)

Auto Teste Integrado

Divisão Polinomial:



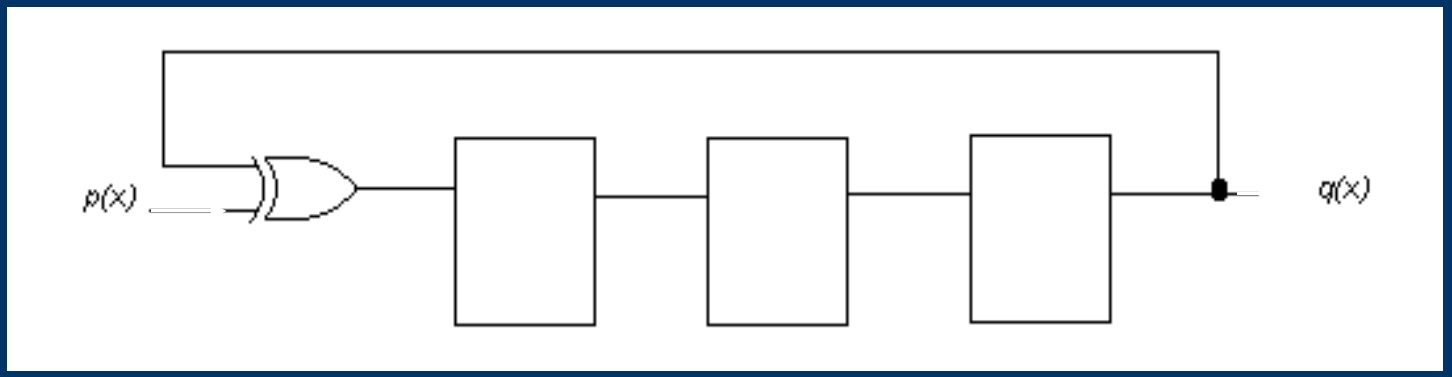
1
0
0
1
1
0

0
0
0
1
0

q(x)

Auto Teste Integrado

Divisão Polinomial:



1
0
0
1
1
0

r(x) 0 1 0

0
0
0
1
0
0

q(x)

Auto Teste Integrado

Divisão Polinomial:

$$q(x) = \frac{p(x)}{c(x)} + r(x)$$

$p(x) = x^5 + x^2 + x$ (100110) | $c(x) = x^3 + 1$ (001001)

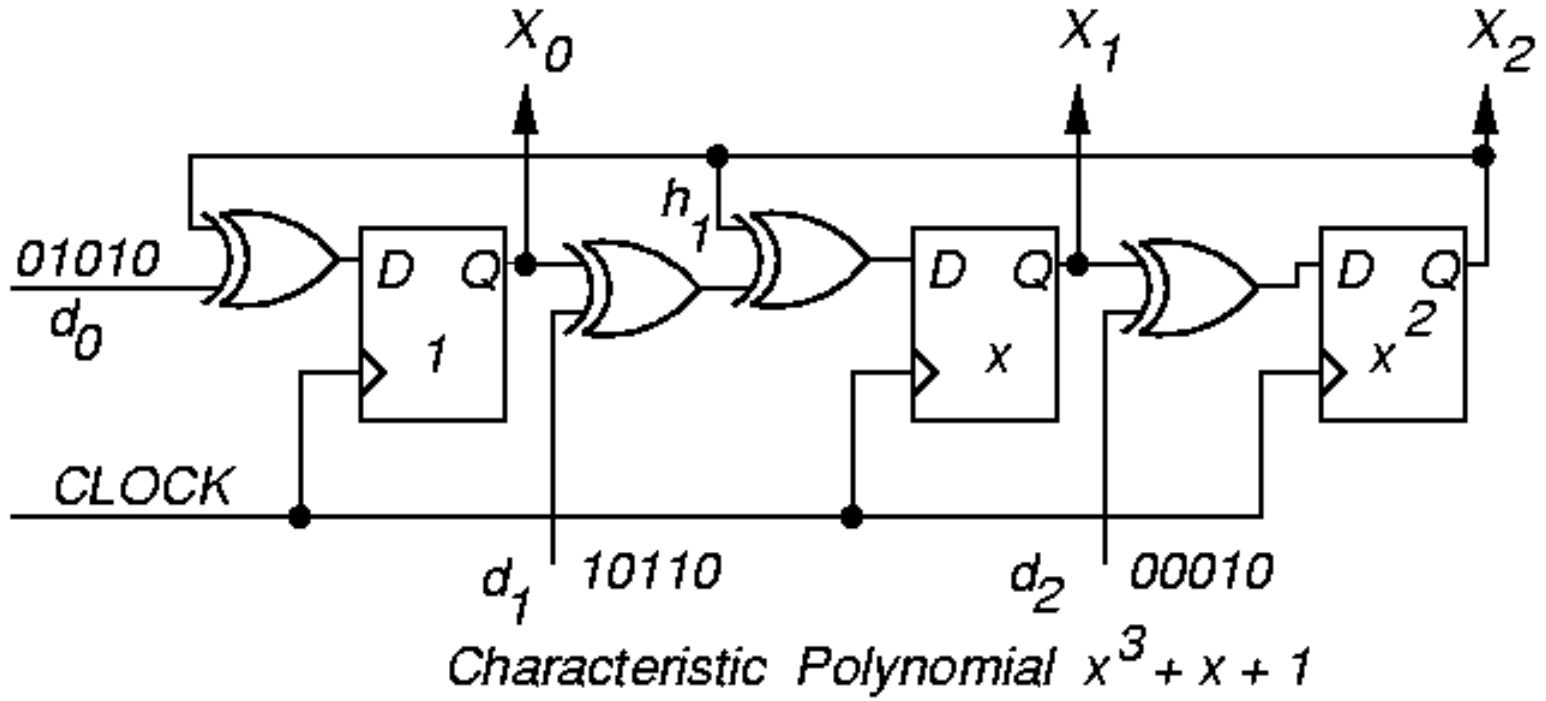
$-x^5 - x^2$ | $q(x) = x^2$ (000100)

$r(x) = x$ (000010)

Multiple-Input Signature Register (MISR)

- Problem with ordinary LFSR response compacter:
  - Too much hardware if one of these is put on each *primary output* (PO)
- Solution: MISR – compacts all outputs into one LFSR
  - Works because LFSR is linear – obeys *superposition principle*
  - Superimpose all responses in one LFSR – final remainder is XOR sum of remainders of polynomial divisions of each PO by the characteristic polynomial

Modular MISR Example



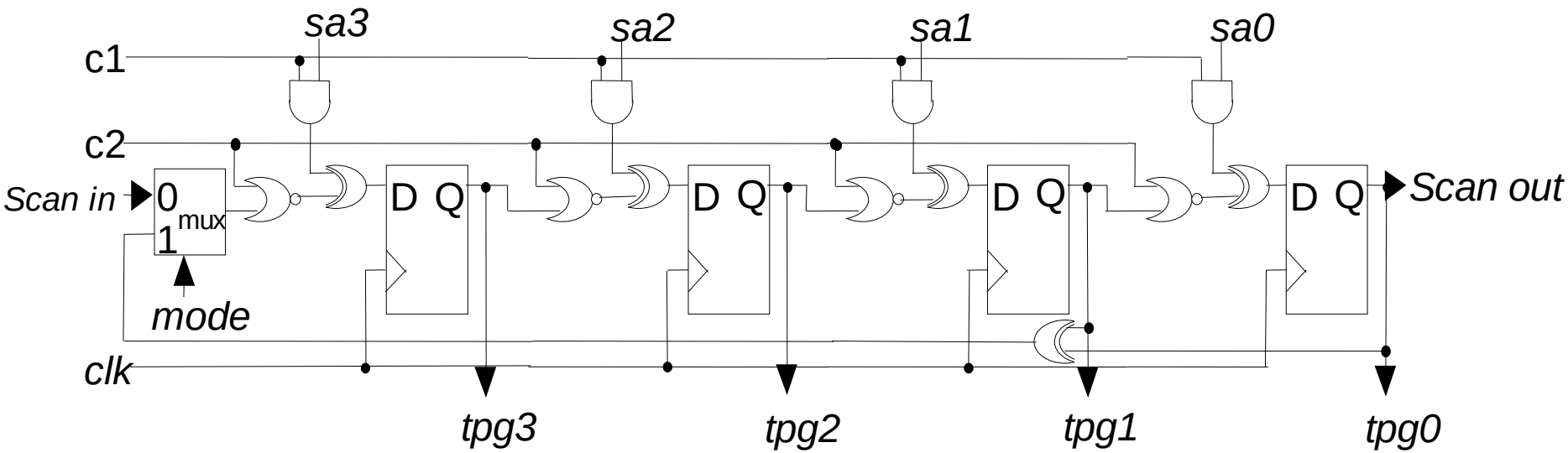
Some other possibilities

- Used with scan chains

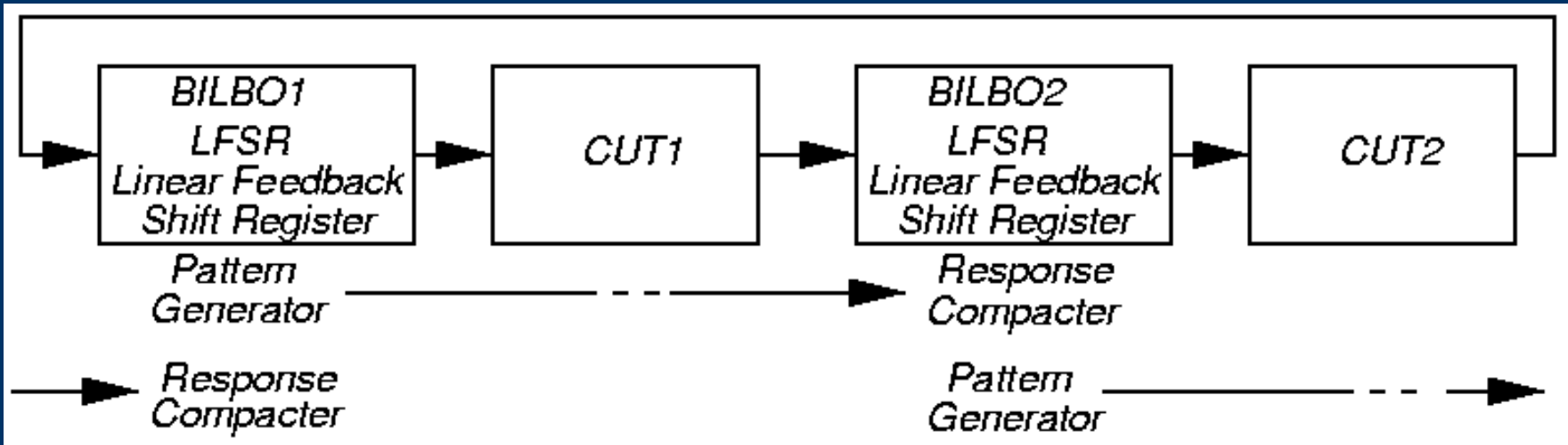
Auto Teste Integrado

Built-In Logic Block Observer (BILBO)

Estrutura multifuncional:

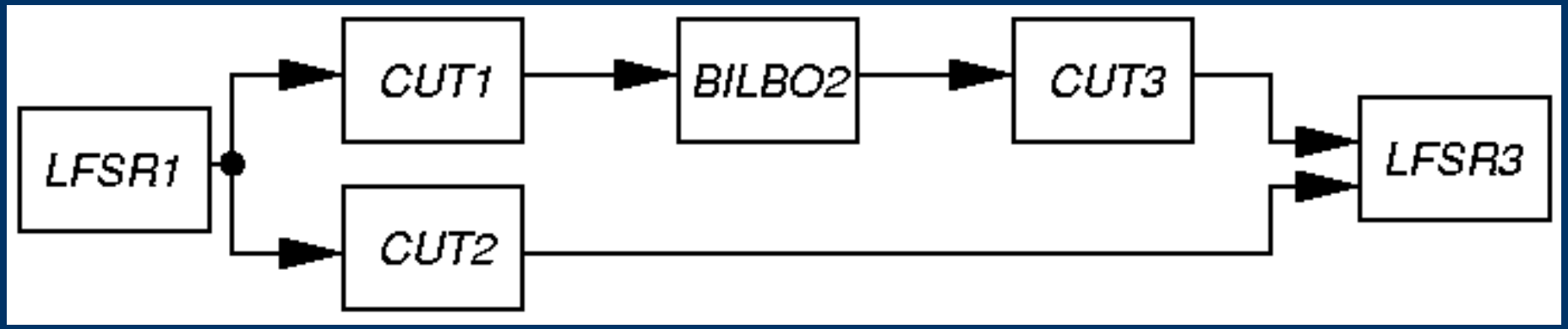


BILBO – Works as Both a PG and a RC



- *Built-in Logic Block Observer* (BILBO) -- 4 modes:
  1. Flip-flop
  2. LFSR pattern generator
  3. LFSR response compacter
  4. Scan chain for flip-flops

Complex BIST Architecture



- Testing epoch I:
  - LFSR1 generates tests for CUT1 and CUT2
  - BILBO2 (LFSR3) compacts CUT1 (CUT2)
- Testing epoch II:
  - BILBO2 generates test patterns for CUT3
  - LFSR3 compacts CUT3 response