TEST 14

Key terms and concepts: production test • wafer test or wafer sort • probe card • production tester • test program • test response • test vector • final test • goods-inward test • printed-circuit board (PCB or board) • failure analysis • field repair

14.1 The Importance of Test

Key terms and concepts: product quality • defect level • average quality level (AQL)

Defect levels in printed-circuit boards (PCB)

ASIC defect level	Defective ASICs	Total PCB repair cost
5%	5000	$1million
1%	1000	$200,000
0.1%	100	$20,000
0.01%	10	$2,000

Defect levels in systems

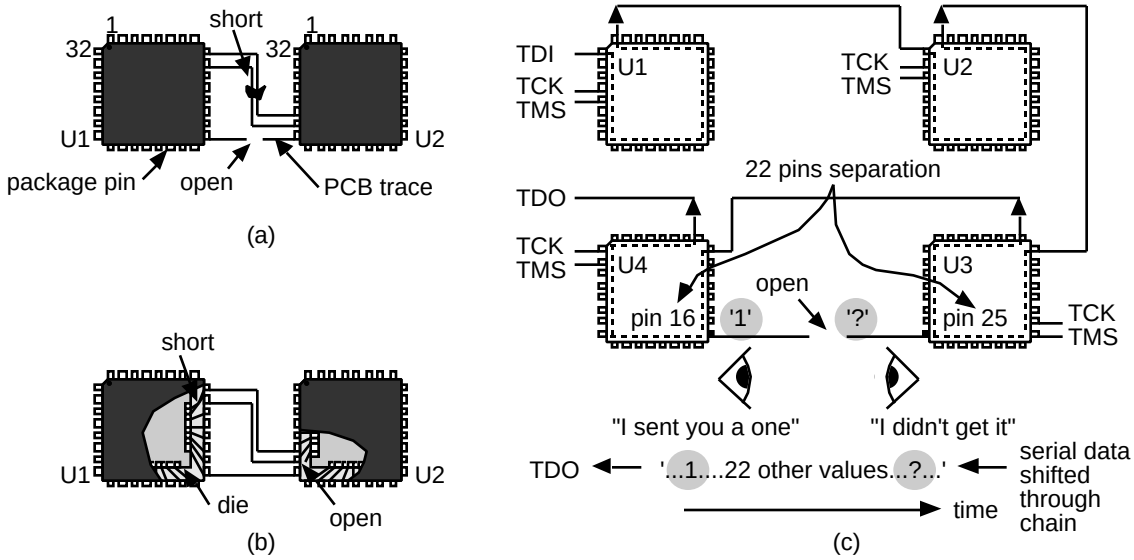
ASIC defect level	Defective ASICs	Defective boards	Total repair cost at system level
5%	5000	500	$5million
1%	1000	100	$1million
0.1%	100	10	$100,000
0.01%	10	1	$10,000

14.2 Boundary-Scan Test

Key terms and concepts: 4/5-wire interface for board-level test • Joint Test Action Group (JTAG) • **IEEE Standard 1149.1 Test Port and Boundary-Scan Architecture** • boundary-scan test (BST) • test-data output (TDO) • test-data registers (TDR) • test clock (TCK) • test-mode select (TMS) • test-reset input signal (TRST*) • test-access port (TAP)

Boundary-scan terminology

Acronym	Meaning	Explanation
BR	Bypass register	A TDR, directly connects TDI and TDO, bypassing BSR
BSC	Boundary-scan cell	Each I/O pad has a BSC to monitor signals
BSR	Boundary-scan register	A TDR, a shift register formed from a chain of BSCs
BST	Boundary-scan test	Not to be confused with BIST (built-in self-test)
IDCODE	Device-identification register	Optional TDR, contains manufacturer and part number
IR	Instruction register	Holds a BST instruction, provides control signals
JTAG	Joint Test Action Group	The organization that developed boundary scan
TAP	Test-access port	Four- (or five-)wire test interface to an ASIC
TCK	Test clock	A TAP wire, the clock that controls BST operation
TDI	Test-data input	A TAP wire, the input to the IR and TDRs
TDO	Test-data output	A TAP wire, the output from the IR and TDRs
TDR	Test-data register	Group of BST registers: IDCODE, BR, BSR
TMS	Test-mode select	A TAP wire, together with TCK controls the BST state
TRST* or nTRST	Test-reset input signal	Optional TAP wire, resets the TAP controller (active-low)



IEEE 1149.1 boundary scan

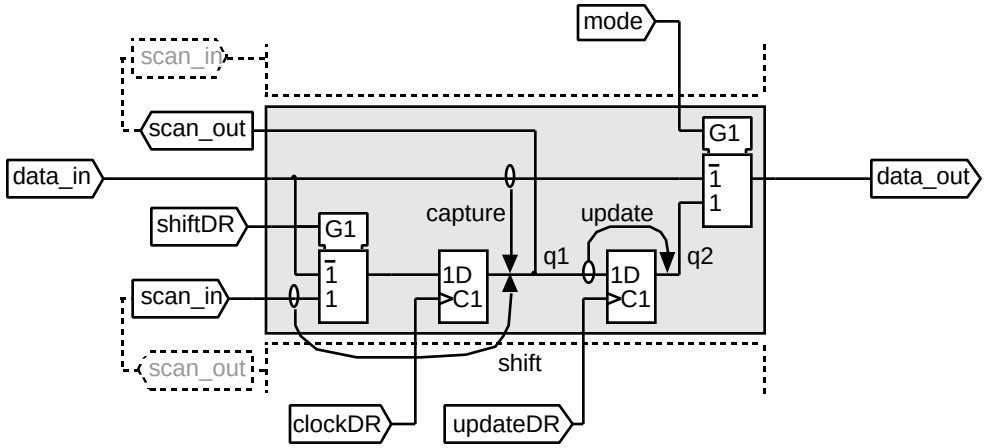
(a) Boundary scan is intended to check for shorts or opens between ICs mounted on a board

(b) Shorts and opens may also occur inside the IC package

(c) The boundary-scan architecture is a long chain of shift registers allowing data to be sent over all the connections between the ICs on a board

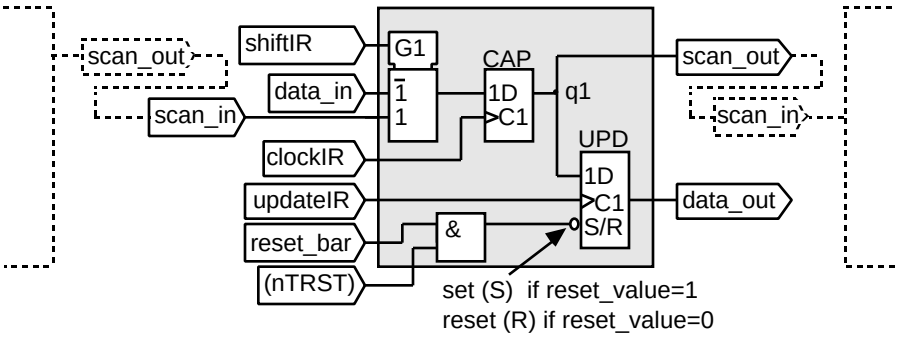
14.2.1 BST Cells

Key terms and concepts: **data-register cell** (DR cell) • **boundary-scan cell** (BS cell, or BSC) • capture flip-flop or capture register • update flip-flop, or update latch • scan in (serial in or SI) • data in (parallel in or PI) • mode (also called test/normal) • scan out (serial out or SO) • data out (parallel out or PO) • reversible • **bypass-register cell** (**BR cell**) • **instruction-register cell** (**IR cell**)



A DR (data register) cell

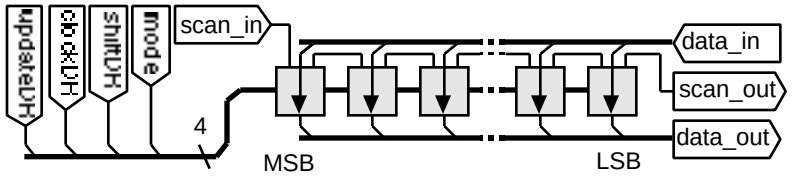
The most common use of this cell is as a boundary-scan cell (BSC)



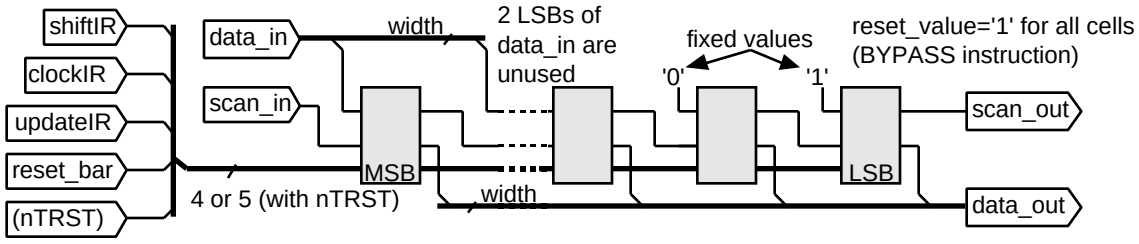
An IR (instruction register) cell

14.2.2 BST Registers

Key terms and concepts: **boundary-scan register** (BSR) • **instruction register** (IR)



A BSR (boundary-scan register)



An IR (instruction register)

14.2.3 Instruction Decoder

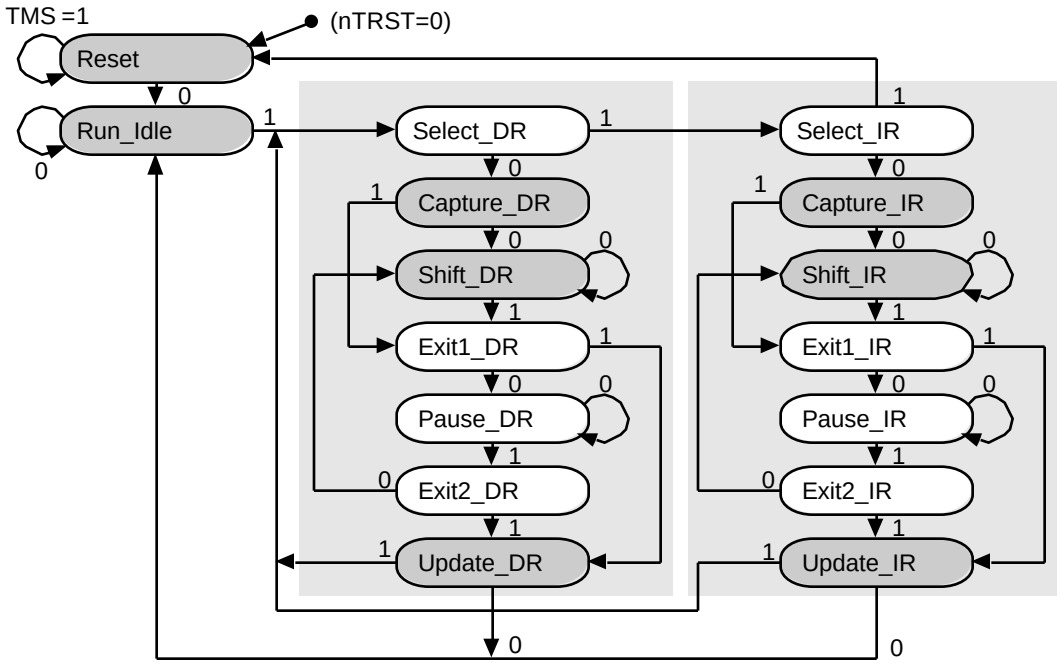
instruction decoder • device-identification register

An IR (instruction register) decoder

```
entity IR_decoder is generic (width : INTEGER := 4); port (
shiftDR, clockDR, updateDR : BIT; IR_PO : BIT_VECTOR (width-1 downto 0) ;
test_mode, selectBR, shiftBR, clockBR, shiftBSR, clockBSR, updateBSR : out BIT );
end IR_decoder;
architecture behave of IR_decoder is
type INSTRUCTION is (EXTEST, SAMPLE_PRELOAD, IDCODE, BYPASS);
signal I : INSTRUCTION;
begin process (IR_PO) begin case BIT_VECTOR'( IR_PO(1), IR_PO(0) ) is
  when "00" => I <= EXTEST; when "01" => I <= SAMPLE_PRELOAD;
  when "10" => I <= IDCODE; when "11" => I <= BYPASS;
end case; end process;
test_mode <= '1' when I = EXTEST else '0';
selectBR  <= '1' when (I = BYPASS or I = IDCODE) else '0';
shiftBR <= shiftDR;
clockBR <= clockDR when (I = BYPASS or I = IDCODE) else '1';
shiftBSR <= shiftDR;
clockBSR <= clockDR when (I = EXTEST or I = SAMPLE_PRELOAD) else '1';
updateBSR <= updateDR when (I = EXTEST or I = SAMPLE_PRELOAD) else '0';
end behave;
```

14.2.4 TAP Controller

Key terms and concepts: JTAG “brain” • four-button digital watch • **clean** signal • dirty gated clocks



The TAP (test-access port) controller state machine

14.2.5 Boundary-Scan Controller

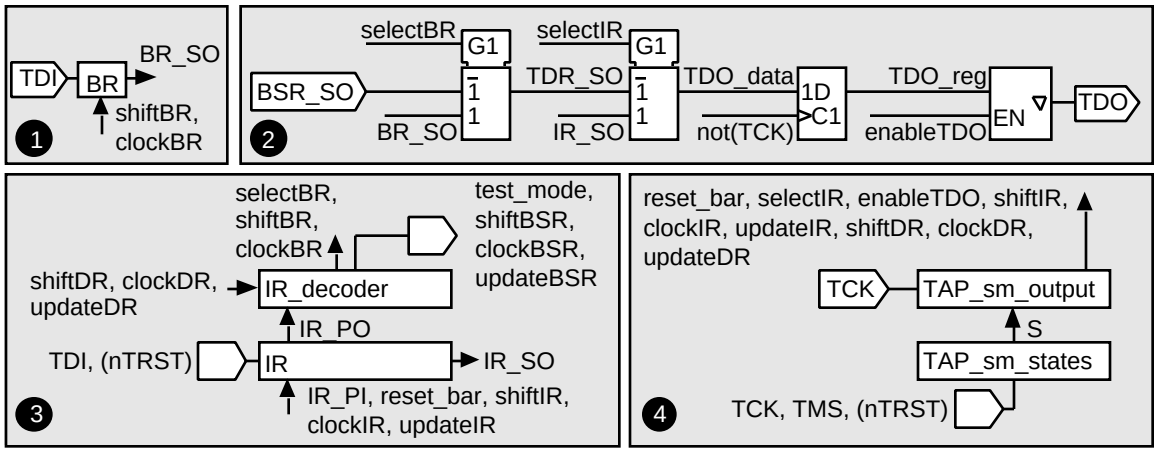
Key terms and concepts: bypass register • TDO output circuit. • instruction register and instruction decoder • TAP controller

14.2.6 A Simple Boundary-Scan Example

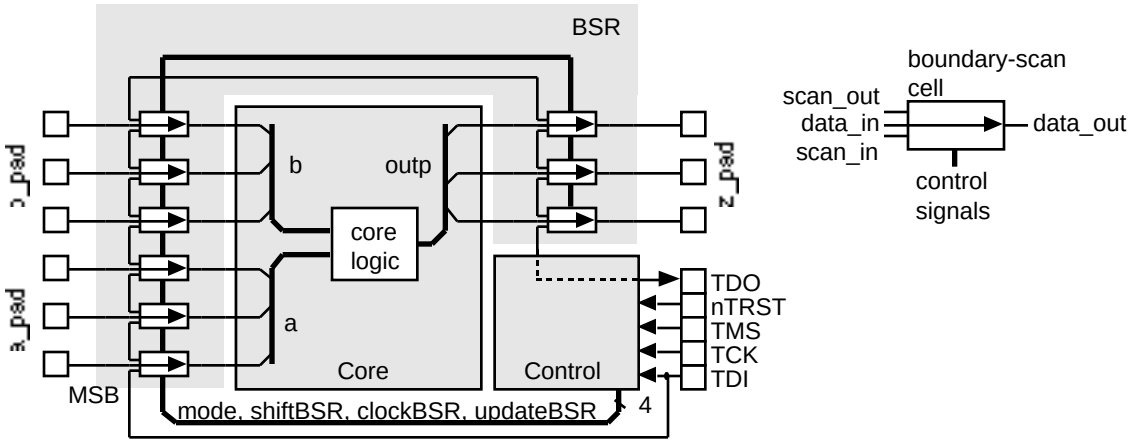
Key terms and concepts: Example: comparator/MUX containing boundary scan

14.2.7 BSDL

Key terms and concepts: **boundary-scan description language** (**BSDL**)



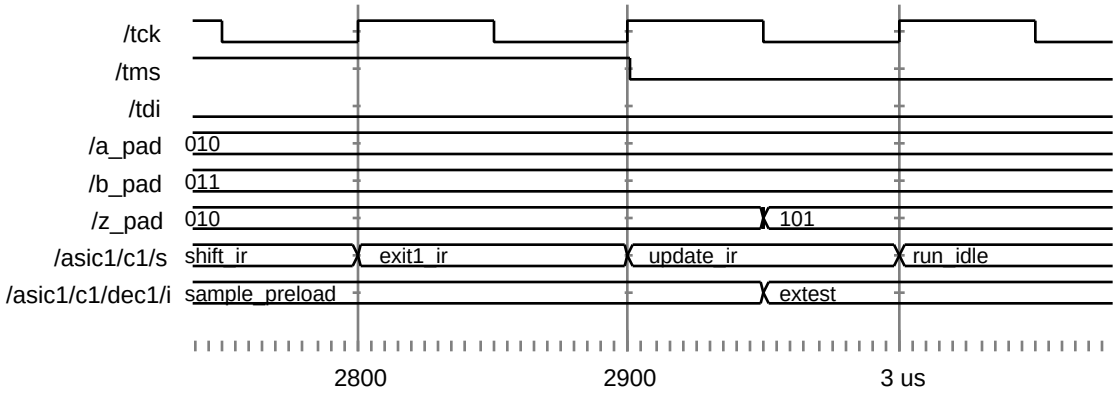
A boundary-scan controller



A boundary-scan example

14.3 Faults

Key terms and concepts: defect • **fault** • defect mechanisms• bridge or short circuit (shorts)• breaks or open circuits (opens)• rework



Results from the MTI simulator for the boundary-scan testbench

14.3.1 Reliability

Key terms and concepts: infant mortality• bathtub curve• wearout mechanisms • burn-in• $\exp(-E_a/kT)$ • Arrhenius equation • activation energy• reliability• mean time between failures (MTBF) • mean time to failure (MTTF) • failures in time (FITs)

14.3.2 Fault Models

Key terms and concepts: fault level • physical fault • fault model • logical fault • degradation fault • parametric fault • delay fault (timing fault)• open-circuit fault • short-circuit fault • bridging faults• metal coverage • feedback bridging faults and nonfeedback bridging faults

Mapping physical faults to logical faults

		Logical fault		
Fault level	**Physical fault**	**Degradation fault**	**Open-circuit fault**	**Short-circuit fault**
Chip				
	Leakage or short between package leads	•		•
	Broken, misaligned, or poor wire bonding		•	
	Surface contamination, moisture	•		
	Metal migration, stress, peeling		•	•
	Metallization (open or short)		•	•
Gate				
	Contact opens		•	
	Gate to S/D junction short	•		•
	Field-oxide parasitic device	•		•
	Gate-oxide imperfection, spiking	•		•
	Mask misalignment	•		•

14.3.3 Physical Faults

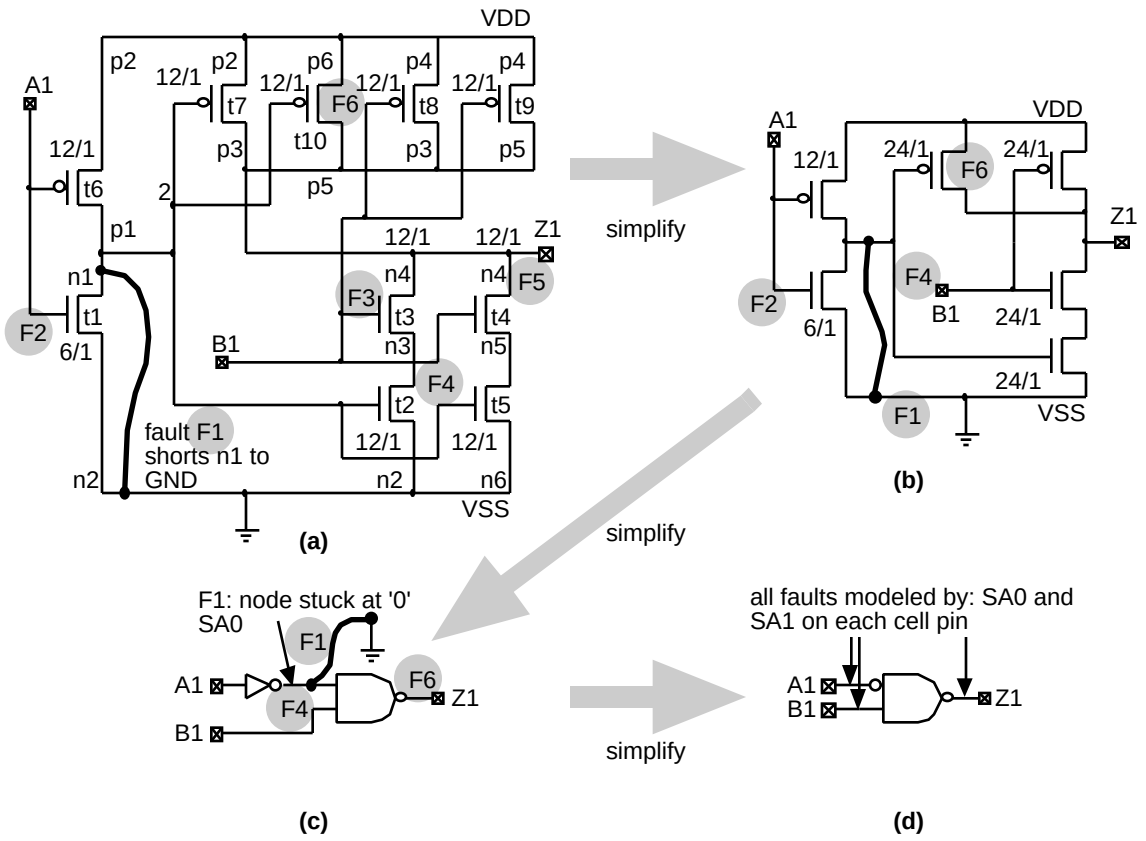
Key terms and concepts: **stuck-at fault model**

14.3.4 Stuck-at Fault Model

Key terms and concepts: single stuck-at fault (SSF) • multiple stuck-at fault model • stuck-on fault and stuck-open fault (or stuck-off fault) • stuck-at faults are: a stuck-at-1 fault (abbreviated to SA1 or s@1) and a stuck-at-0 fault (SA0 or s@0) • place faults (inject faults, seed faults, or apply faults) • fault origin • net fault • input fault • output fault • supply-strength fault (or rail-strength fault) • output-fault strength • node fault • pin-fault model • structural level, gate level, or cell level • transistor level or switch level • fault effect • fault propagation • structural fault propagation • behavioral fault propagation • mixed-level fault simulation

14.3.5 Logical Faults

Key terms and concepts: not all physical faults translate to logical faults—most do not



Fault models

(a) Physical faults at the layout level (problems during fabrication) translate to electrical problems on the detailed circuit schematic. The location and effect of fault F1 is shown. The locations of the other faults are shown, but not their effect

(b) We can translate some of these faults to the simplified transistor schematic

(c) Only a few of the physical faults still remain in a gate-level fault model of the logic cell

(d) Finally at the functional-level fault model of a logic cell, we abandon the connection between physical and logical faults and model all faults by stuck-at faults. This is a very poor model of the physical reality, but it works well in practice.

14.3.6 IDDQ Test

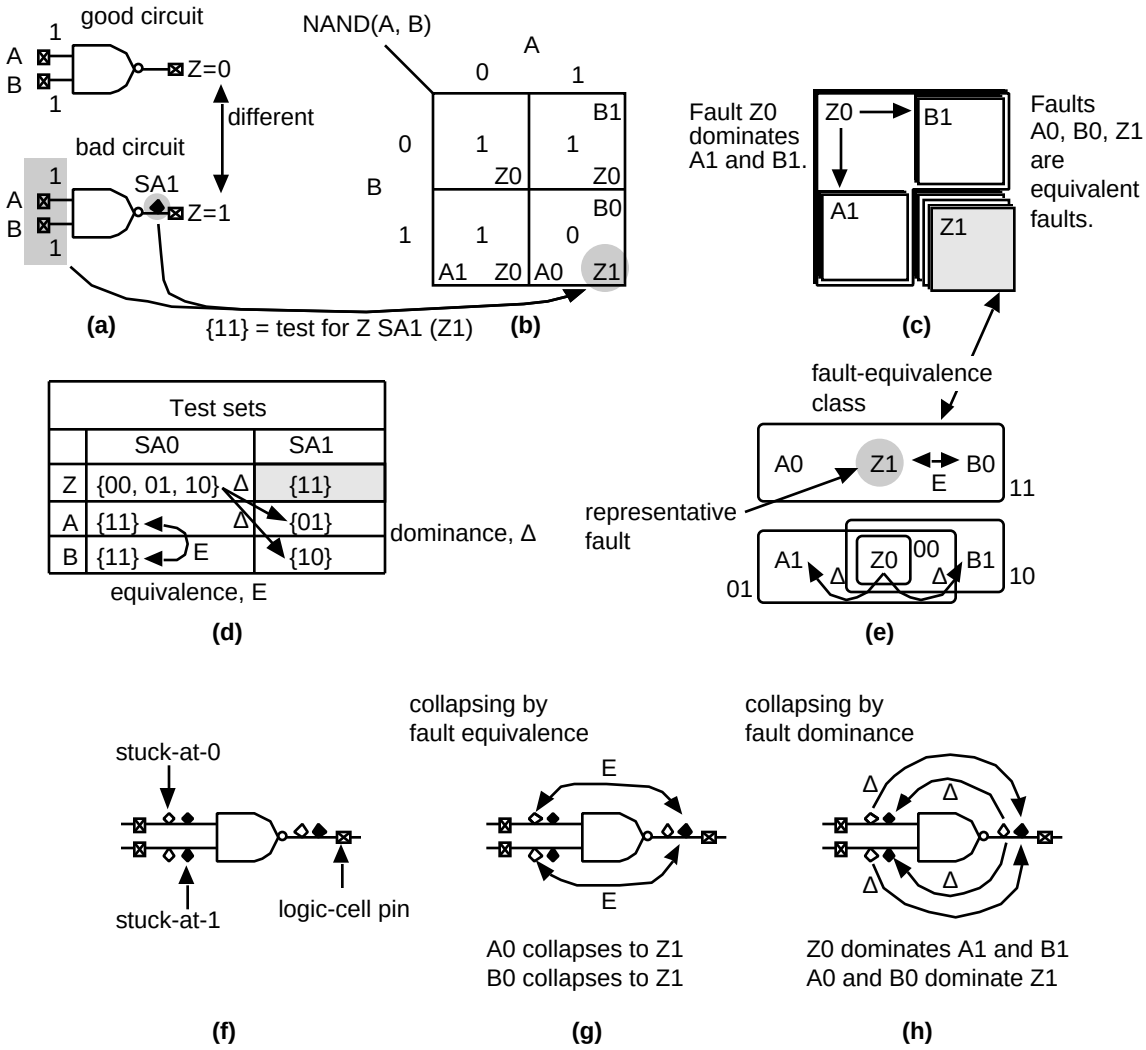
Key terms and concepts: **IDDQ** • high supply current can result from bridging faults

14.3.7 Fault Collapsing

Key terms and concepts: bad circuit (also called the faulty circuit or faulty machine) • fault collapsing • equivalent faults (or indistinguishable faults) • fault-equivalence class• prime fault or representative fault• dominant fault• dominant fault collapsing

14.3.8 Fault-Collapsing Example

Key terms and concepts: gate collapsing• node collapsing



Fault dominance and fault equivalence

(a) A test for fault Z0 (Z stuck at 0) makes the bad circuit differ from the good circuit

(b) Some test vectors provide tests for more than one fault

(c) A test for A1 also tests for Z0, Z0 dominates A1. A0, B0, Z1 are the same (equivalent)

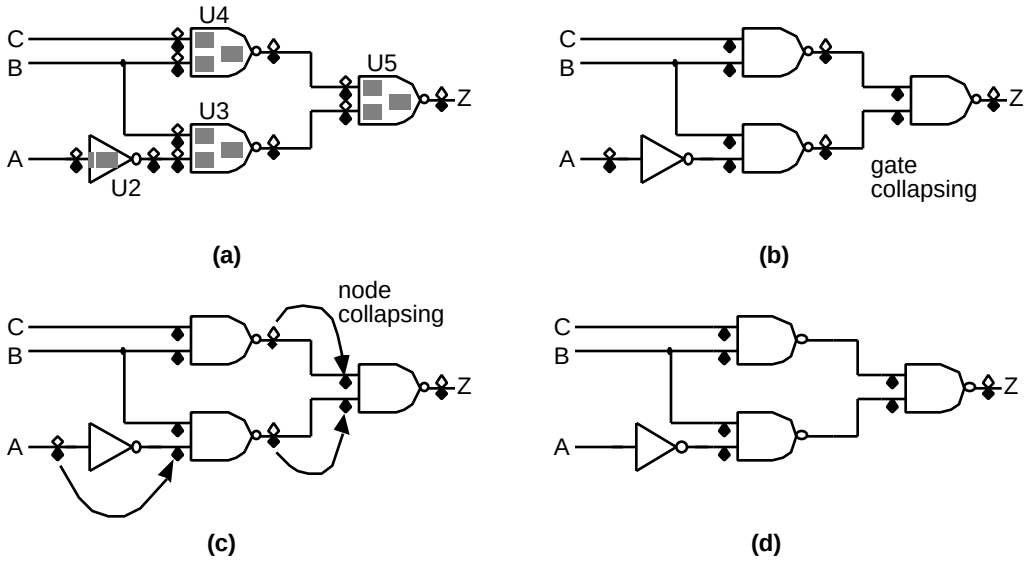
(d) There are six sets of input vectors that test for the six stuck-at faults

(e) We only need to choose a subset of all test vectors that test for all faults

(f) The six stuck-at faults for a two-input NAND logic cell

(g) Using fault equivalence we can collapse six faults to four

(h) Using fault dominance we can collapse six faults to three.



Fault collapsing for A'B+BC

(a) A pin-fault model. Each pin has stuck-at-0 and stuck-at-1 faults

(b) Using fault equivalence the pin faults at the input pins and output pins of logic cells are collapsed. This is gate collapsing

(c) We can reduce the number of faults we need to consider further by collapsing equivalent faults on nodes and between logic cells. This is node collapsing

(d) The final circuit has eight stuck-at faults (reduced from the 22 original faults). If we wished to use fault dominance we could also eliminate the stuck-at-0 fault on Z. Notice that in a pin-fault model we cannot collapse the faults U4.A1.SA1 and U3.A2.SA1 even though they are on the same net.

14.4 Fault Simulation

Key terms and concepts: **fault simulation** • primary inputs (PIs) and primary outputs (POs) • stimulus • test vector • test program • test-cycle time • sense (or strobe) • detected fault • undetected fault • fault origins • fault coverage

Average quality level as a function of single stuck-at fault coverage		
Fault coverage	**Average defect level**	**Average quality level (AQL)**
50%	7%	93%
90%	3%	97%
95%	1%	99%
99%	0.1%	99.9%
99.9%	0.01%	99.99%

14.4.1 Serial Fault Simulation

Key terms and concepts: serial fault simulation • machines • good machine • faulty machine

14.4.2 Parallel Fault Simulation

Key terms and concepts: **parallel fault simulation** uses multiple bits per word • a bit is either a `'1'` or `'0'` for each node in the circuit• a 32-bit word can simulate 32 circuits at once

14.4.3 Concurrent Fault Simulation

Key terms and concepts: **concurrent fault simulation** takes advantage of the fact that a fault does not affect the whole circuit • diverged circuit • fault-activity signature • faults per pass

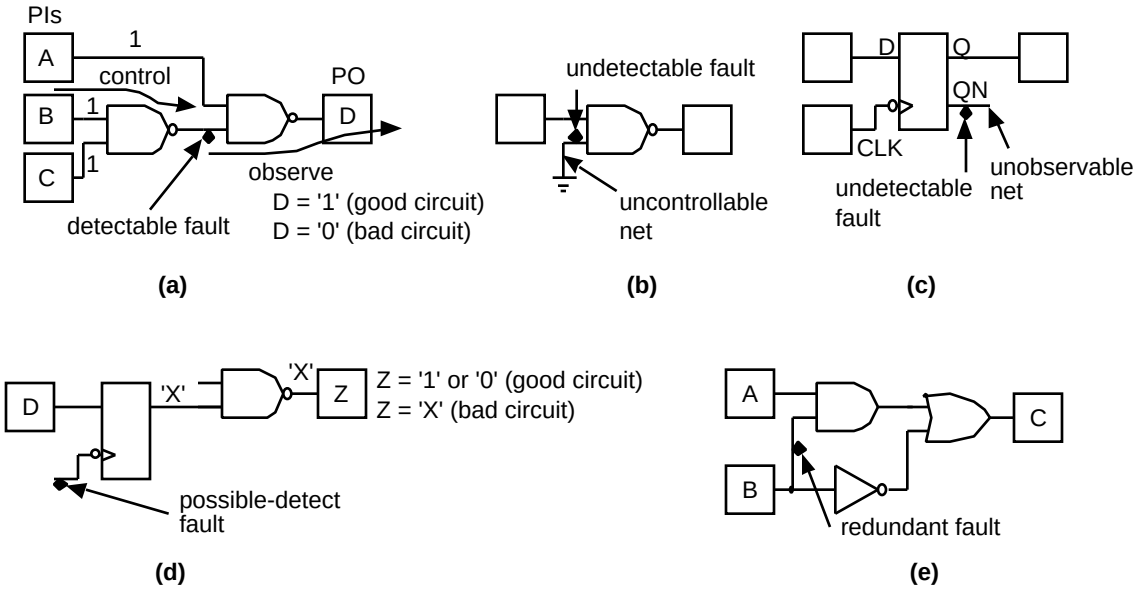
14.4.4 Nondeterministic Fault Simulation

Key terms and concepts: serial, parallel, and concurrent fault-simulation algorithms are forms of deterministic fault simulation• **probabilistic fault simulation** simulates a subset or sample of the faults and extrapolates coverage • statistical fault simulation performs a fault-free simulation and use the results to predict fault coverage • toggle test • vector quality • toggle coverage

14.4.5 Fault-Simulation Results

Key terms and concepts: fault categories• testable fault • controllable net • observable net • uncontrollable net and unobservable net • untested fault • **hard-detected fault** • undetected fault

• possibly detected fault • soft-detected fault • fault-drop threshold • fault dropping • redundant fault • irredundant • oscillatory fault • hyperactive fault



Fault categories

(a) A detectable fault requires the ability to control and observe the fault origin

(b) A net that is fixed in value is uncontrollable and therefore will produce one undetected fault

(c) Any net that is unconnected is unobservable and will produce undetected faults

(d) A net that produces an unknown 'X' in the faulty circuit and a '1' or a '0' in the good circuit may be detected (depending on whether the 'X' is in fact a '0' or '1'), but we cannot say for sure. At some point this type of fault is likely to produce a discrepancy between good and bad circuits and will eventually be detected

(e) A redundant fault does not affect the operation of the good circuit. In this case the AND gate is redundant since AB+B'=A+B'

14.4.6 Fault-Simulator Logic Systems

Key terms and concepts: fault grading • dead test cycles • fault list • faulty output vector • fault signature

The VeriFault concurrent fault simulator logic system

Good circuit \ Faulty circuit	0	1	Z	L	H	X
0	U	D	P	P	P	P
1	D	U	P	P	P	P
Z	U	U	U	U	U	U
L	U	U	U	U	U	U
H	U	U	U	U	U	U
X	U	U	U	U	U	U

14.4.7 Hardware Acceleration

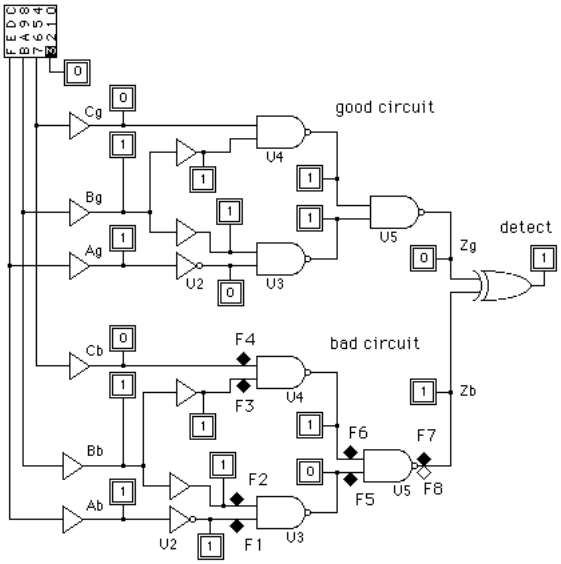
Key terms and concepts: simulation engines or hardware accelerators • distributed fault simulation

14.4.8 A Fault-Simulation Example

Key terms and concepts: test-vector compression or test-vector compaction • structurally equivalent

14.4.9 Fault Simulation in an ASIC Design Flow

Key terms and concepts: canned test vectors



Fault	Type	Vectors (hex)	Good output	Bad output
F1	SA1	**3**	0	1
F2	SA1	0, 4	0, 0	1, 1
F3	SA1	4, 5	0, 0	1, 1
F4	SA1	3	0	1
F5	SA1	2	1	0
F6	SA1	7	1	0
F7	SA1	0, 1, 3, 4, 5	0, 0, 0, 0, 0	1, 1, 1, 1, 1
F8	SA0	2, 6, 7	1, 1, 1	0, 0, 0

[1]Test vector format:

3 = 011, so that CBA = 011: C = '0', B = '1', A = '1'

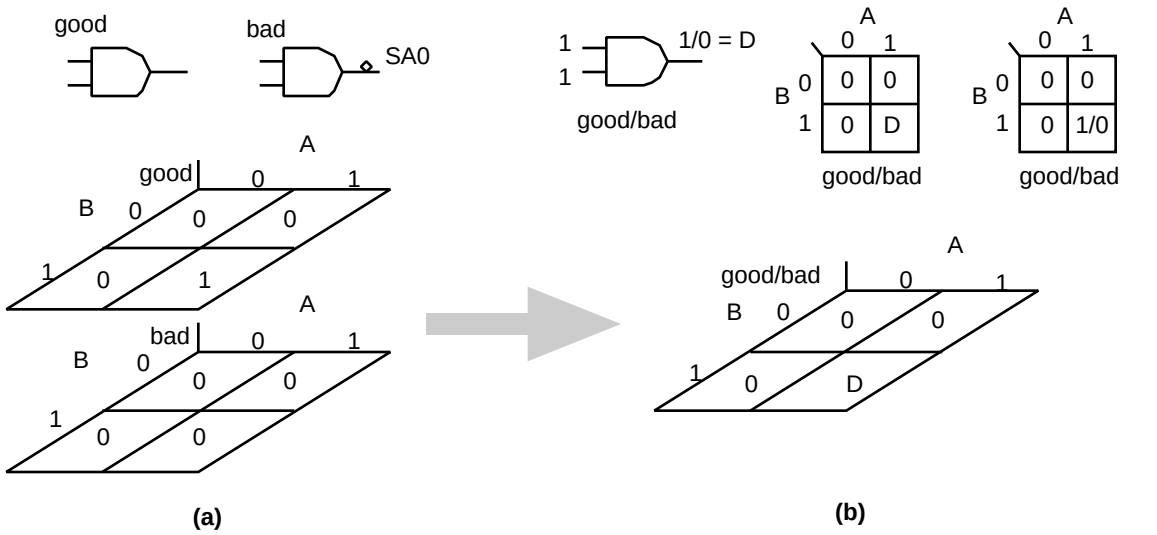
Fault simulation of A'B+BC

The simulation results for fault F1 (U2 output stuck at 1) with test vector value hex **3** (shown in bold in the table) are shown on the LogicWorks schematic

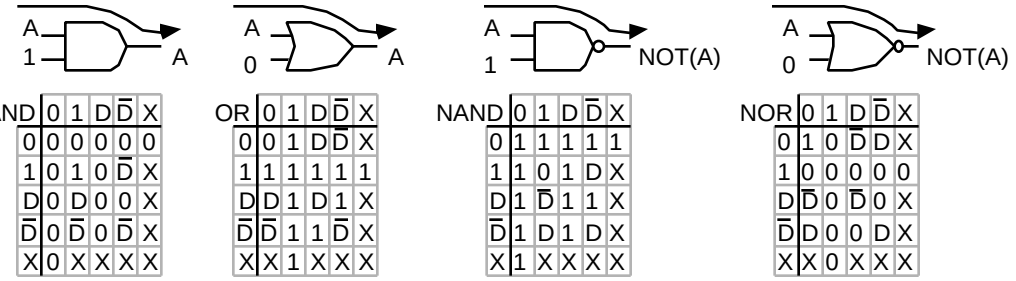
Notice that the output of U2 is 0 in the good circuit and stuck at 1 in the bad circuit.

14.5 Automatic Test-Pattern Generation

Key terms and concepts: PODEM, for **automatic test-pattern generation** (ATPG) or automatic test-vector generation (ATVG)



(c)



AND	0	1	D	$\bar{D}$	X
0	0	0	0	0	0
1	0	1	0	$\bar{D}$	X
D	0	D	0	0	X
$\bar{D}$	0	$\bar{D}$	0	$\bar{D}$	X
X	0	X	X	X	X

OR	0	1	D	$\bar{D}$	X
0	0	1	D	$\bar{D}$	X
1	1	1	1	1	1
D	D	1	D	1	X
$\bar{D}$	$\bar{D}$	1	1	$\bar{D}$	X
X	X	1	X	X	X

NAND	0	1	D	$\bar{D}$	X
0	1	1	1	1	1
1	1	0	1	D	X
D	1	$\bar{D}$	1	1	X
$\bar{D}$	1	D	1	D	X
X	1	X	X	X	X

NOR	0	1	D	$\bar{D}$	X
0	1	0	$\bar{D}$	D	X
1	0	0	0	0	0
D	$\bar{D}$	0	$\bar{D}$	0	X
$\bar{D}$	D	0	0	D	X
X	X	0	X	X	X

The D-calculus

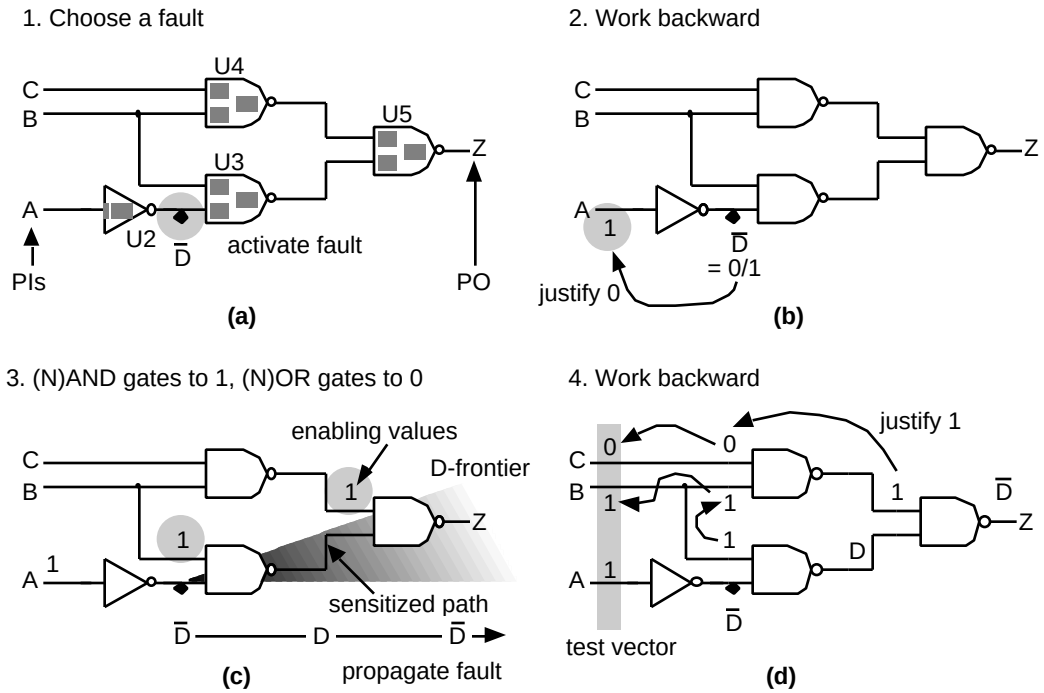
(a) We need a way to represent the behavior of the good circuit and the bad circuit at the same time

(b) The composite logic value D (for detect) represents a logic '1' in the good circuit and a logic '0' in the bad circuit. We can also write this as D=1/0

(c) The logic behavior of simple logic cells using the D-calculus. Composite logic values can propagate through simple logic gates if the other inputs are set to their enabling values.

14.5.1 The D-Calculus

Key terms and concepts: D-calculus • D-algorithm • D (for detect) • D=0/1 • g/b, a composite logic value • propagate • enabling value • controlling value • justifies



A basic ATPG (automatic test-pattern generation) algorithm for A'B+BC

(a) We activate a fault, U2.ZN stuck at 1, by setting the pin or node to '0', the opposite value of the fault

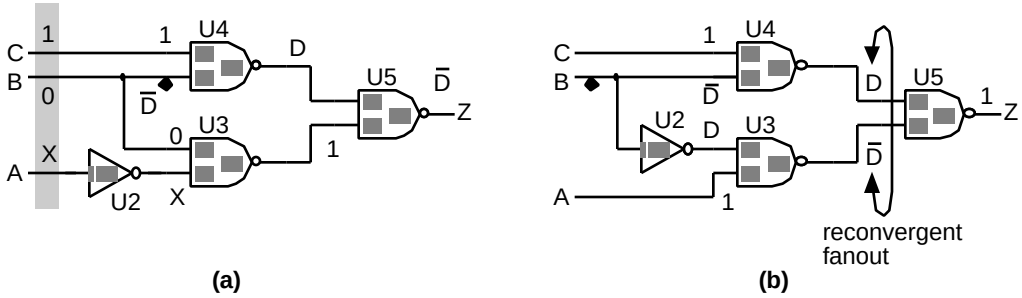
(b) We work backward from the fault origin to the PIs (primary inputs) by recursively justifying signals at the output of logic cells

(c) We then work forward from the fault origin to a PO (primary output), setting inputs to gates on a sensitized path to their enabling values. We propagate the fault until the D-frontier reaches a PO

(d) We then work backward from the PO to the PIs recursively justifying outputs to generate the sensitized path. This simple algorithm always works, providing signals do not branch out and then rejoin again.

14.5.2 A Basic ATPG Algorithm

Key terms and concepts: activating (or exciting the fault)• sensitize • observed • D-frontier, • reconvergent fanout • multipath sensitization



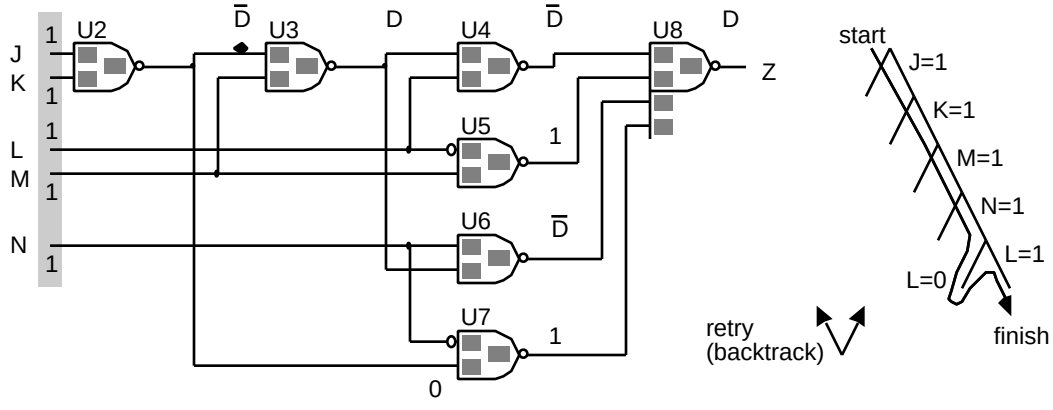
Reconvergent fanout

(a) Signal B branches and then reconverges at logic gate U5, but the fault U4.A1 stuck at 1 can still be excited and a path sensitized using the basic algorithm

(b) Fault B stuck at 1 branches and then reconverges at gate U5. When we enable the inputs to both gates U3 and U4 we create two sensitized paths that prevent the fault from propagating to the PO (primary output). We can solve this problem by changing A to '0', but this breaks the rules of the algorithm. The PODEM algorithm solves this problem.

14.5.3 The PODEM Algorithm

Key terms and concepts: path-oriented decision making (PODEM) • objective • backtrace • implication • D-frontier • X-path check• backtrack• FAN (fanout-oriented test generation)



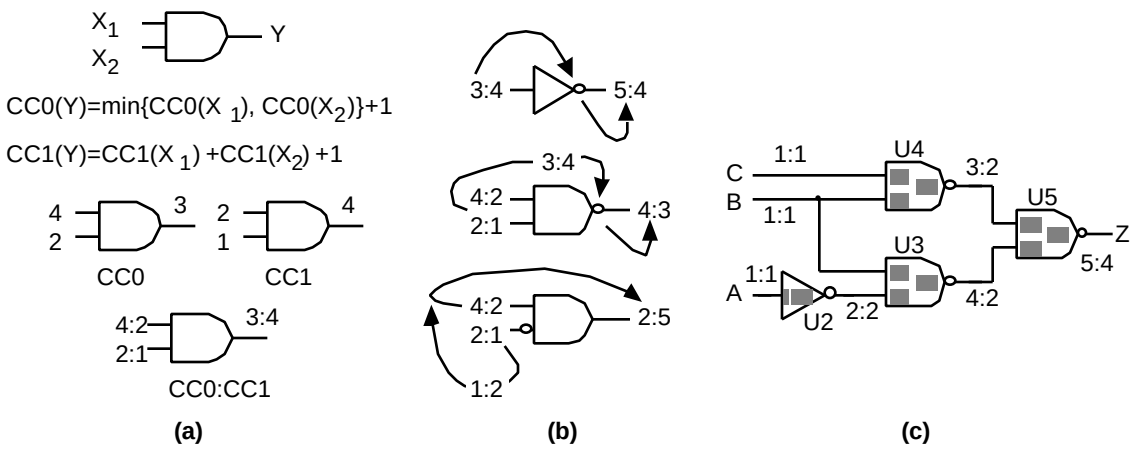
Iteration **Objective** **Backtrace** **Implication** **D-frontier**

1	U3.A2=0	J=1		
2	U3.A2=0	K=1	U7.ZN=1	
3	U3.A1=1	M=1	U3.ZN=D	U4, U6
4	U6.A2=1	N=1	U6.ZN=$\overline{D}$	U4, U8
5a	U8.A1=1	L=0	U8.ZN=1	U4, U8
5b	Retry	L=1	U8.ZN=D	A

The PODEM (path-oriented decision making) algorithm.

14.5.4 Controllability and Observability

Key terms and concepts: controllability (three l's) • observability • SCOAP (Sandia controllability/observability analysis program)• combinational controllability• sequential controllability• zero-controllability and one-controllability• combinational zero-controllability • logic distance • combinational one-controllability • **combinational observability**

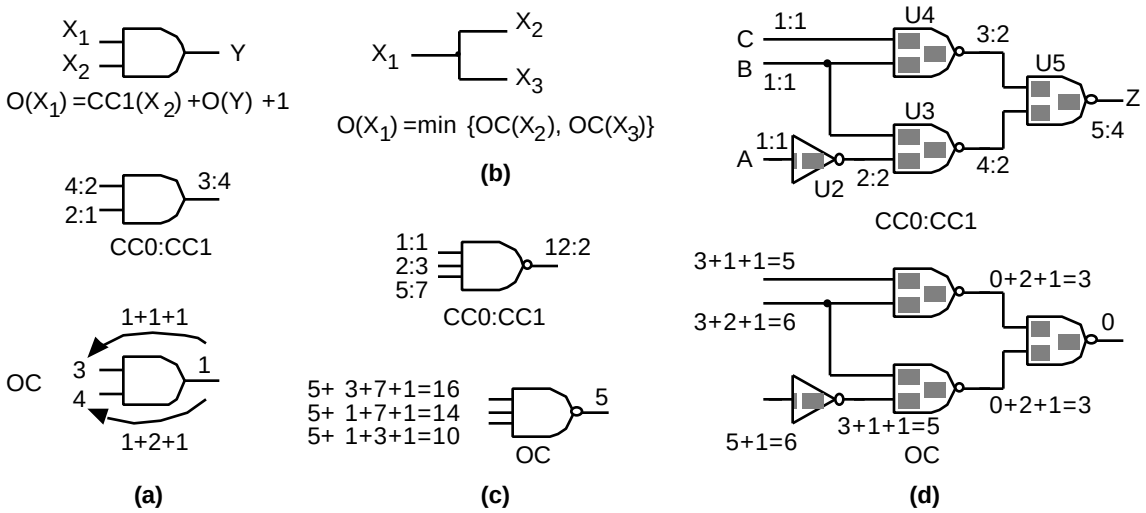


Controllability measures

(a) Definition of combinational zero-controllability, CC0, and combinational one-controllability, CC1, for a two-input AND gate

(b) Examples of controllability calculations for simple gates, showing intermediate steps

(c) Controllability in a combinational circuit



Observability measures

(a) The combinational observability, $OC(X_1)$, of an input, X_1, to a two-input AND gate defined in terms of the controllability of the other input and the observability of the output

(b) The observability of a fanout node is equal to the observability of the most observable branch

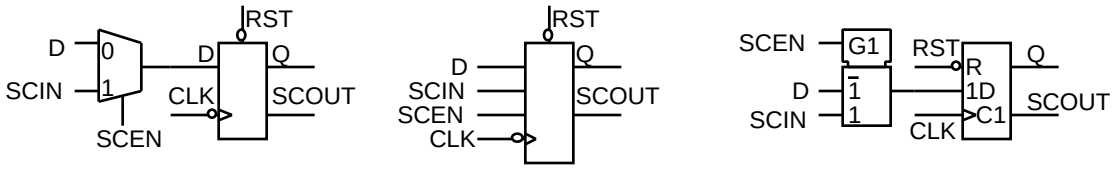
(c) Example of an observability calculation at a three-input NAND gate

(d) The observability of a combinational network can be calculated from the controllability measures, CC0:CC1. The observability of a PO (primary output) is defined to be zero.

14.6 Scan Test

Key terms and concepts: structured test • design for test • test compiler • scan insertion• pseudoprimary input • pseudoprimary output • partial scan • destructive scan • nondestructive scan • level-sensitive scan design (LSSD)

Scan flip-flop



14.7 Built-in Self-test

Key terms and concepts: **built-in self-test** (BIST) • circuit under test (CUT) or device under test (DUT)

14.7.1 LFSR

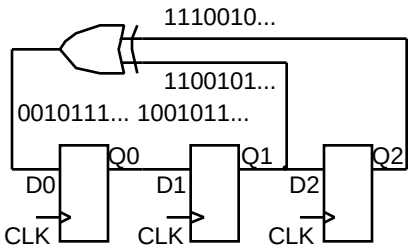
Key terms and concepts: **linear feedback shift register** (LFSR) • pseudorandom binary sequence (PRBS) • maximal-length sequence

LFSR example

Clock tick, t=	$Q0_{t+1}=Q1_t$ $Q2_t$	$Q1_{t+1}=Q0_t$	$Q2_{t+1}=Q1_t$	Q0Q1Q2
1	1	1	1	7
2	0	1	1	3
3	0	0	1	1
4	1	0	0	4
5	0	1	0	2
6	1	0	1	5
7	1	1	0	6
8	1	1	1	7

A linear feedback shift register (LFSR).

A 3-bit maximal-length LFSR produces a repeating string of seven pseudorandom binary numbers: 7, 3, 1, 4, 2, 5, 6.



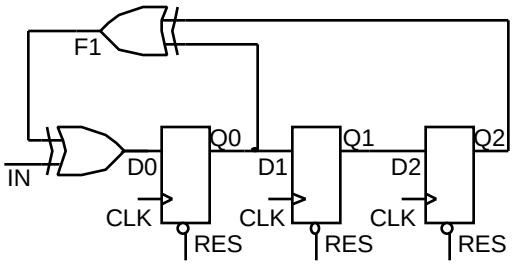
14.7.2 Signature Analysis

Key terms and concepts: data compaction • signature • serial-input signature register (SISR) • signature analysis• Hewlett-Packard

A 3-bit serial-input signature register (SISR) using an LFSR (linear feedback shift register)

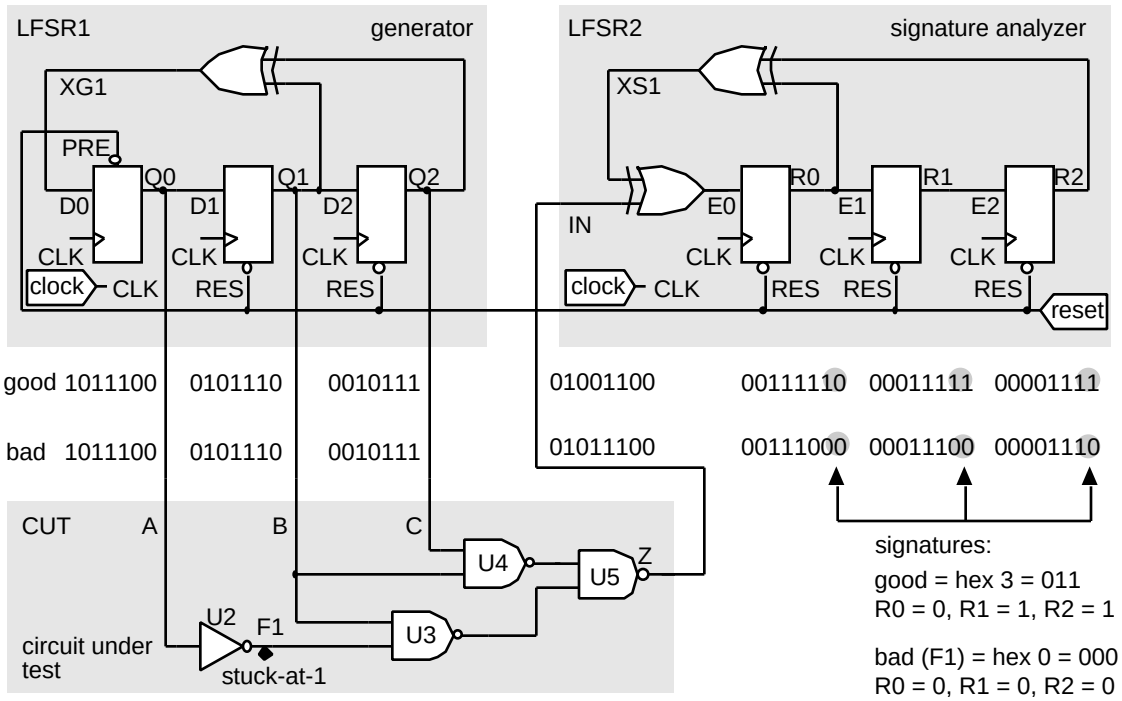
The LFSR is initialized to Q1Q2Q3='000' using the common RES (reset) signal

The signature, Q1Q2Q3, is formed from shift-and-add operations on the sequence of input bits (IN)



14.7.3 A Simple BIST Example

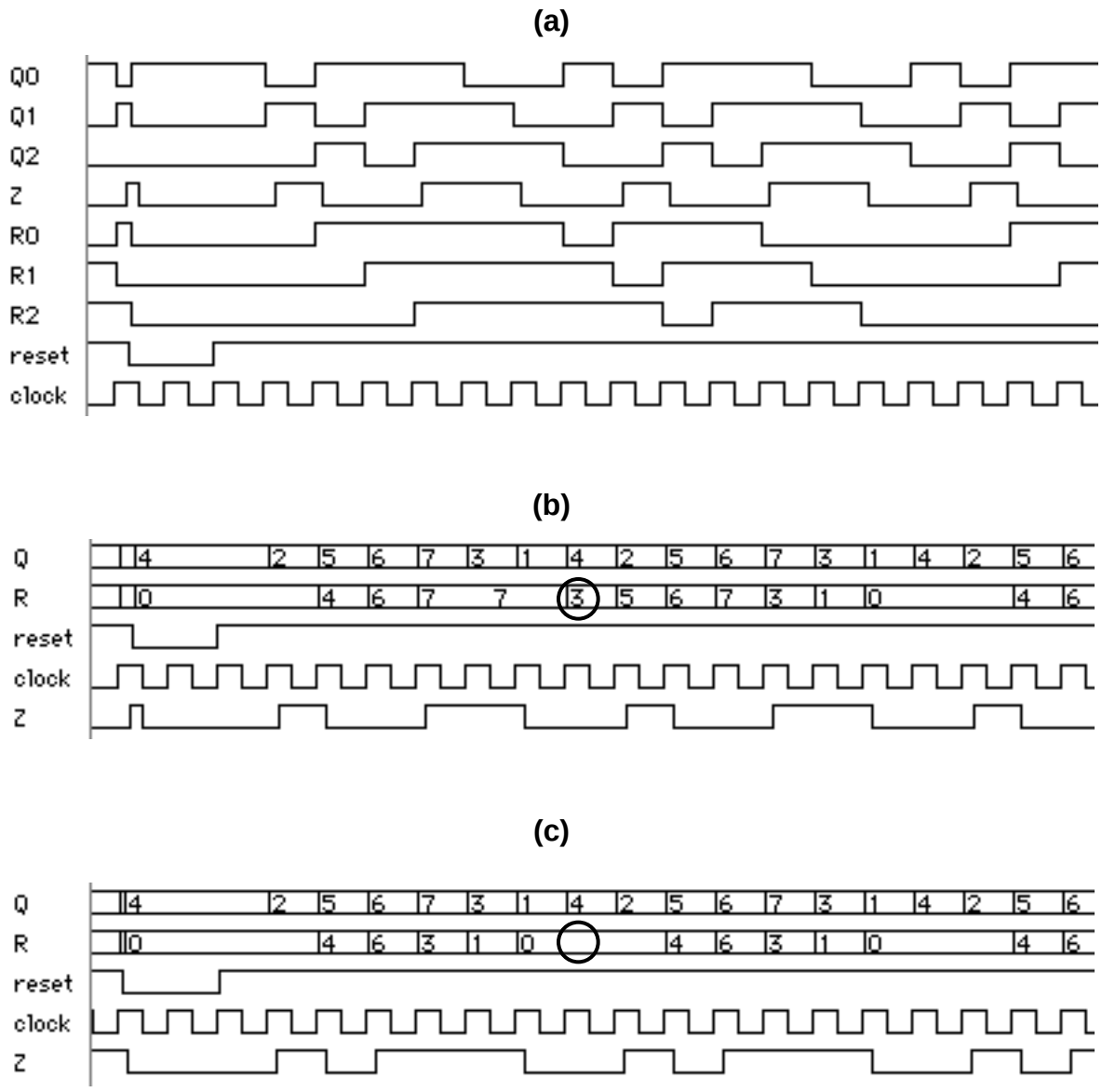
(a)



(b)

$Q0_{t+1}=Q1_t$ $Q2_t$	$Q1_{t+1}=Q0_t$	$Q2_{t+1}=Q1_t$	$Z=Q0'.Q1+Q1.Q2$	$R0_{t+1}=Z_t$ $R0_t$ $R2_t$	$R1_{t+1}=R0_t$	$R2_{t+1}=R1_t$
1	0	0	0	0	0	0
0	1	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	1	1	0
1	1	1	1	1	1	1
0	1	1	1	1	1	1
0	0	1	0	1	1	1
1	0	0	0	0	1	1

BIST example. **(a)** A simple BIST structure showing bit sequences for both good and bad circuits. **(b)** Bit sequence calculations for the good circuit. The signature appears on the eighth clock cycle (after seven positive clock edges) and is R0='0', R1='1', R2='1'; with R2 as the MSB this is '011' or hex 3.



The waveforms of the BIST example

(a) The good-circuit response. The waveforms Q1 and Q2, as well as R1 and R2, are delayed by one clock cycle as they move through each stage of the shift registers

(b) The same good-circuit response with the register outputs Q0–Q2 and R0–R2 grouped and their values displayed in hexadecimal (Q0 and R0 are the MSBs). The signature hex 3 or '011' (R0=0, R1=1, R2=1) in R appears seven positive clock edges after the reset signal is taken high. This is one clock cycle after the generator completes its first sequence (hex pattern 4, 2, 5, 6, 7, 3, 1)

(c) The response of the bad circuit with fault F1 and fault signature hex 0 (circled).

14.7.4 Aliasing

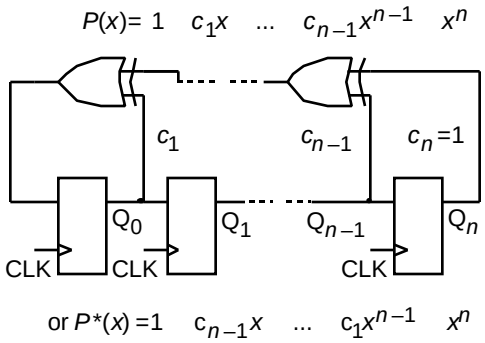
Key terms and concepts: **aliasing** • error coverage

14.7.5 LFSR Theory

Key terms and concepts: polynomials and Galois-field theory • characteristic polynomial • primitive polynomials • external-XOR LFSR • type 1 LFSR • internal-XOR LFSR • type 2 LFSR

n	s	Octal	Binary
1	0, 1	3	11
2	0, 1, 2	7	111
3	0, 1, 3	13	1011
4	0, 1, 4	3	10011
5	0, 2, 5	45	100101
6	0, 1, 6	103	1000011
7	0, 1, 7	211	10001001
8	0, 1, 5, 6, 8	435	100011101
9	0, 4, 9	1021	1000010001
10	0, 3, 10	2011	10000001001

For n=3 and s=0, 1, 3: c_0=1, c_1=1, c_2=0, c_3=1



Primitive polynomial coefficients for LFSRs (linear feedback shift registers) that generate a maximal-length PRBS (pseudorandom binary sequence)

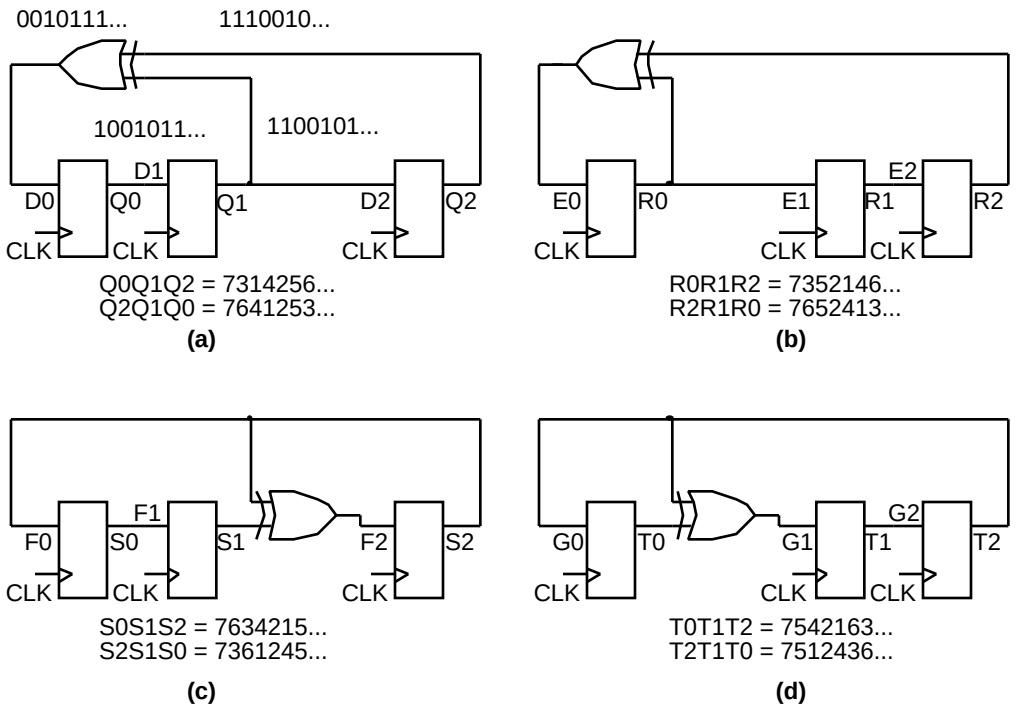
A schematic for a type 1 LFSR is shown.

14.7.6 LFSR Example

Key terms and concepts: automatic generation of LFSR and SISR structures

14.7.7 MISR

Key terms and concepts: multiple-input signature register (MISR) • built-in logic block observer (BILBO)• circular self-test path (CSTP) • complete LFSR • **scanBIST**



For every primitive polynomial there are four linear feedback shift registers (LFSRs).

There are two types of LFSR; one type uses external XOR gates (type 1) and the other type uses internal XOR gates (type 2).

For each type the feedback taps can be constructed either from the polynomial P(x) or from its reciprocal, P*(x). The LFSRs in this figure correspond to P(x)=1 x x^3 and P*(x)= 1 x^2 x^3.

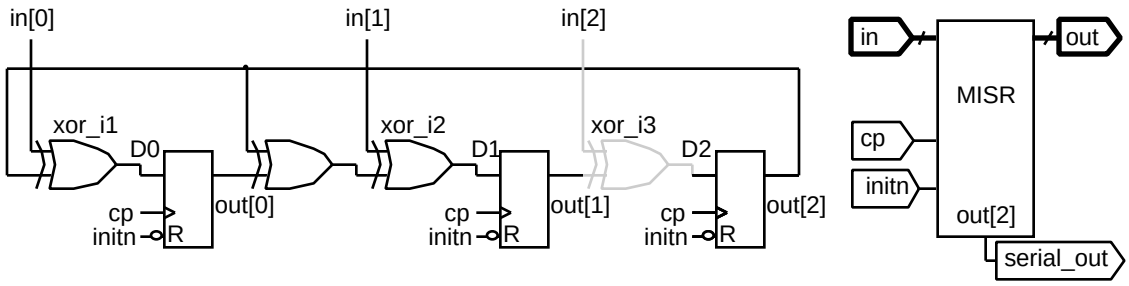
Each LFSR produces a different pseudorandom sequence, as shown. The binary values of the LFSR seen as a register, with the bit labeled as zero being the MSB, are shown in hexadecimal.

The sequences shown are for each register initialized to '111', hex 7.

(a) Type 1, P*(x). **(b)** Type 1, P(x). **(c)** Type 2, P(x). **(d)** Type 1, P*(x).

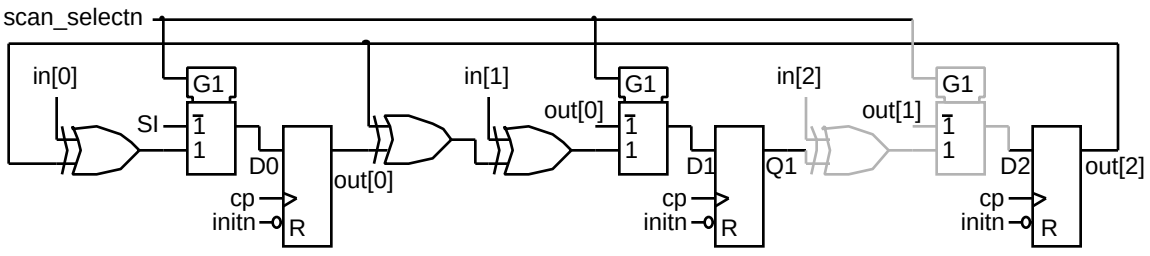
Compiled LFSR generator, using P*(x)=1 x^2 x^3

```
module lfsr_generator (OUT, SERIAL_OUT, INITN, CP);
output [2:0] OUT; output SERIAL_OUT; input  INITN, CP;
  dfptnb FF2 (.D(FF0_Q), .CP(u4_Z), .SDN(u2_Z), .Q(FF2_Q), .QN(FF2_QN));
  dfctnb FF1 (.D(XOR0_Z), .CP(u4_Z), .CDN(u2_Z), .Q(FF1_Q), .QN(FF1_QN));
  dfctnb FF0 (.D(FF1_Q), .CP(u4_Z), .CDN(u2_Z), .Q(FF0_Q), .QN(FF0_QN));
  ni01d1 u2 (.I(u3_Z), .Z(u2_Z)); ni01d1 u3 (.I(INITN), .Z(u3_Z));
  ni01d1 u4 (.I(u5_Z), .Z(u4_Z)); ni01d1 u5 (.I(CP), .Z(u5_Z));
  xo02d1 XOR0 (.A1(FF2_Q), .A2(FF0_Q), .Z(XOR0_Z));
  in02d1 INV2X0 (.I(FF0_QN), .ZN(OUT[0]));
  in02d1 INV2X1 (.I(FF1_QN), .ZN(OUT[1]));
  in02d1 INV2X2 (.I(FF2_QN), .ZN(OUT[2]));
  in02d1 INV2X3 (.I(FF0_QN), .ZN(SERIAL_OUT));
endmodule
```



Multiple-input signature register (MISR).

This MISR is formed from the type 2 LFSR (with P*(x)=1 x^2 x^3) by adding XOR gates xor_i1, xor_i2, and xor_i3. This 3-bit MISR can form a signature from logic with three outputs. If we only need to test two outputs then we do not need XOR gate, xor_i3, corresponding to input in[2].

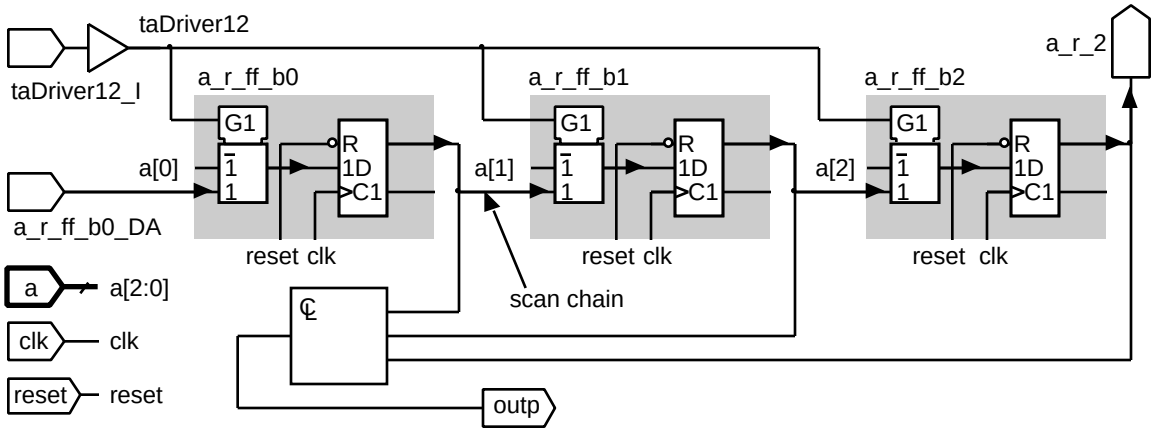


Multiple-input signature register (MISR) with scan

14.8 A Simple Test Example

14.8.1 Test-Logic Insertion

Key terms and concepts: `outp = a_r[0]'.a_r[1] + a_r[1].a_r[2]` • **test-logic insertion**



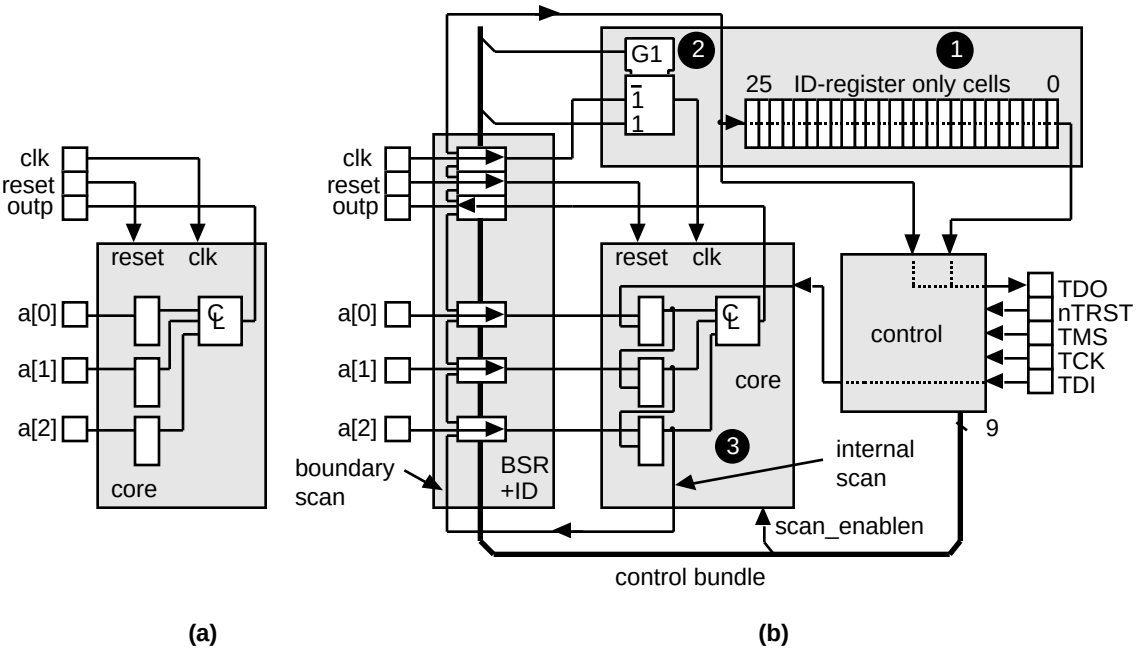
The core of the Threegates ASIC after test-logic insertion.

14.8.2 How the Test Software Works

Key terms and concepts: **polarity-hold flip-flop**

14.8.3 ATVG and Fault Simulation

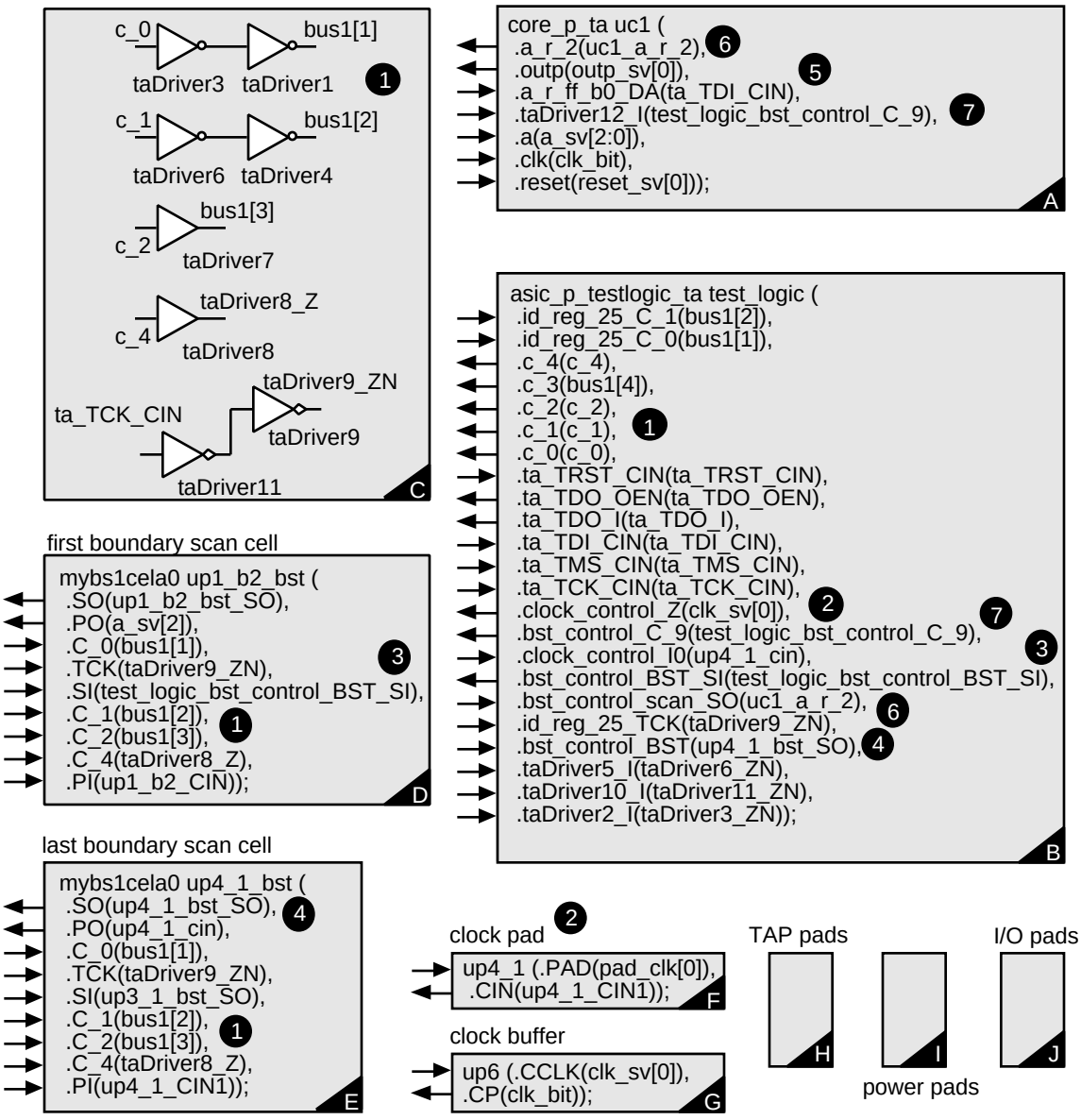
Key terms and concepts: **flush test**



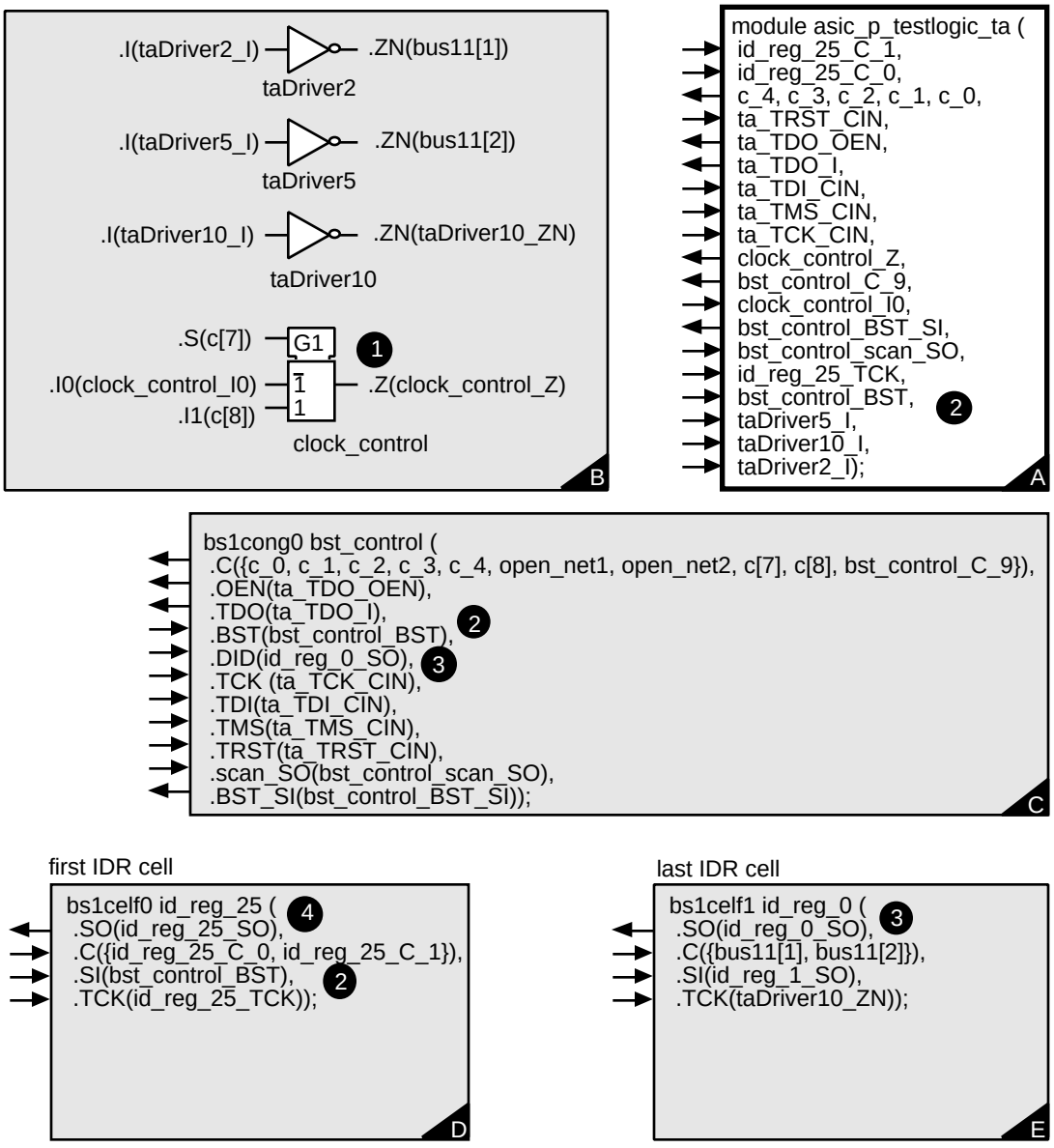
The Threegates ASIC.

(a) Before test-logic insertion.

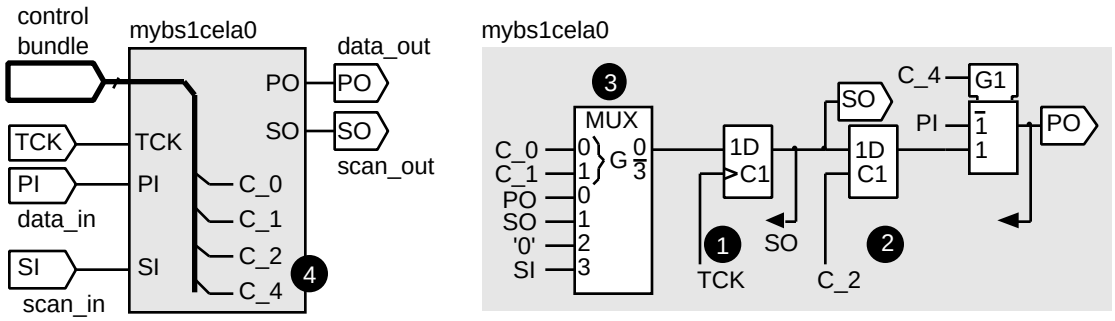
(b) After test-logic insertion.



The top level of the Threegates ASIC after test-logic insertion.



Test logic inserted in the Threegate ASIC.



Input boundary-scan cell (BSC) for the Threegates ASIC.

Compare this to a generic data-register (DR) cell (used as a BSC).

ATVG (automatic test-vector generation) report for the Threegates ASIC

```
CREATE: Output vector database cell defaulted to [svf]asic_p_ta
CREATE: Backtrack limit defaulted to 30
CREATE: Minimal compression effort: 10 (default)
Fault list generation/collapsing
Total number of faults: 184
Number of faults in collapsed fault list: 80
Vector generation
#
# VECTORS  FAULTS   FAULT COVER
#         processed
#
#      5      184     60.54%
#
# Total number of backtracks: 0
# Highest backtrack         : 0
# Total number of vectors   : 5
#
# STAR RESULTS summary
#                             Noncollapsed       Collapsed
# Fault counts:
#   Aborted                      0                  0
#   Detected                     89                 43
#   Untested                     58                 20
#                               ------             ------
#   Total of detectable          147                63
#
#   Redundant                    6                  2
#   Tied                         31                 15
#
# FAULT COVERAGE                60.54 %            68.25 %
#
# Fault coverage = nb of detected faults / nb of detectable faults
Vector/fault list database [svf]asic_p_ta created.
```

14.8.4 Test Vectors

Key terms and concepts: serial vectors • parallel vectors • broadside vectors

14.8.5 Production Tester Vector Formats

Key terms and concepts: Sentry tester file format

```
# Pin declaration: pin names are separated by semi-colons (all pins
# on a bus must be listed and separated by commas)
pre_; clr_; d; clk; q; q_;
# Pin declarations are separated from test vectors by $
$
# The first number on each line is the time since start in ns,
# followed by space or a tab.
# The symbols following the time are the test vectors
# (in the same order as the pin declaration)
# an "=" means don't do anything
# an "s" means sense the pin at the beginning of this time point
# (before the input changes at this time point have any effect)
#
#  pcdcqq
#  rlal _
#  ertk
#  __a
00 1010== # clear the flip-flop
10 1110ss # d=1, clock=0
20 1111ss # d=1, clock=1
30 1110ss # d=1, clock=0
40 1100ss # d=0, clock=0
50 1101ss # d=0, clock=1
60 1100ss # d=0, clock=0
70 ====ss
```

14.8.6 Test Flow

Key terms and concepts: test-vector generation and the production-test program generation is the last step in ASIC design after physical design is complete

Timing effects of test-logic insertion for the Viterbi decoder

Timing of critical paths before test-logic insertion

```
#  Slack(ns)  Num Paths
#   -3.3826     1       *
#   -1.7536     18      *******
#    -.1245     4       **
#    1.5045     1       *
#    3.1336     0       *
#    4.7626     0       *
#    6.3916     134     ****************************************
#    8.0207     6       ***
#    9.6497     3       **
#   11.2787     0       *
#   12.9078     24      ********
```

```
# instance name
# inPin --> outPin      incr    arrival  trs  rampDel  cap    cell
#                       (ns)     (ns)          (ns)    (pf)

# v_1.u100.u1.subout6.Q_ff_b0
# CP --> QN              1.73     1.73    R    .20     .10    dfctnb
...
# v_1.u100.u2.metric0.Q_ff_b4
# setup: D --> CP         .16    21.75    F    .00     .00    dfctnh
```

After test-logic insertion

```
#   -4.0034     1       *
#   -1.9835     18      *****
#     .0365     4       **
#    2.0565     1       *
#    4.0764     0       *
#    6.0964     138     ******************************
#    8.1164     2       *
#   10.1363     3       **
#   12.1563     24      ******
#   14.1763     0       *
#   16.1963     187     ****************************************
```

```
# v_1.u100.u1.subout7.Q_ff_b1
# CP --> Q               1.40     1.40    R    .28     .13    mfctnb
...
# v_1.u100.u2.metric0.Q_ff_b4
# setup: DB --> CP        .39    21.98    F    .00     .00    mfctnh
```

14.9 The Viterbi Decoder Example

Fault coverage for the Viterbi decoder

```
Fault list generation/collapsing
Total number of faults: 8846
Number of faults in collapsed fault list: 3869
Vector generation
#
# VECTORS  FAULTS   FAULT COVER
#         processed
#
#     20     7515     82.92%
#     40     8087     89.39%
#     60     8313     91.74%
#     80     8632     95.29%
#     87     8846     96.06%

# Total number of backtracks: 3000
# Highest backtrack         : 30
# Total number of vectors   : 87

# STAR RESULTS summary
#                             Noncollapsed       Collapsed
# Fault counts:
#   Aborted                      178               85
#   Detected                     8427              3680
#   Untested                     168               60
#                               ------            ------
#   Total of detectable          8773              3825
#
#   Redundant                    10                6
#   Tied                         63                38
#
# FAULT COVERAGE                96.06 %           96.21 %
```

14.10 Summary

Key terms and concepts: Consider test early during ASIC design otherwise it can become very expensive • Boundary scan • Single stuck-at fault model • Controllability and observability • ATPG using test vectors • BIST with no test vectors