Faults, Testing & Test Generation

Smith Text: Chapter 14.1,14.3, 14.4

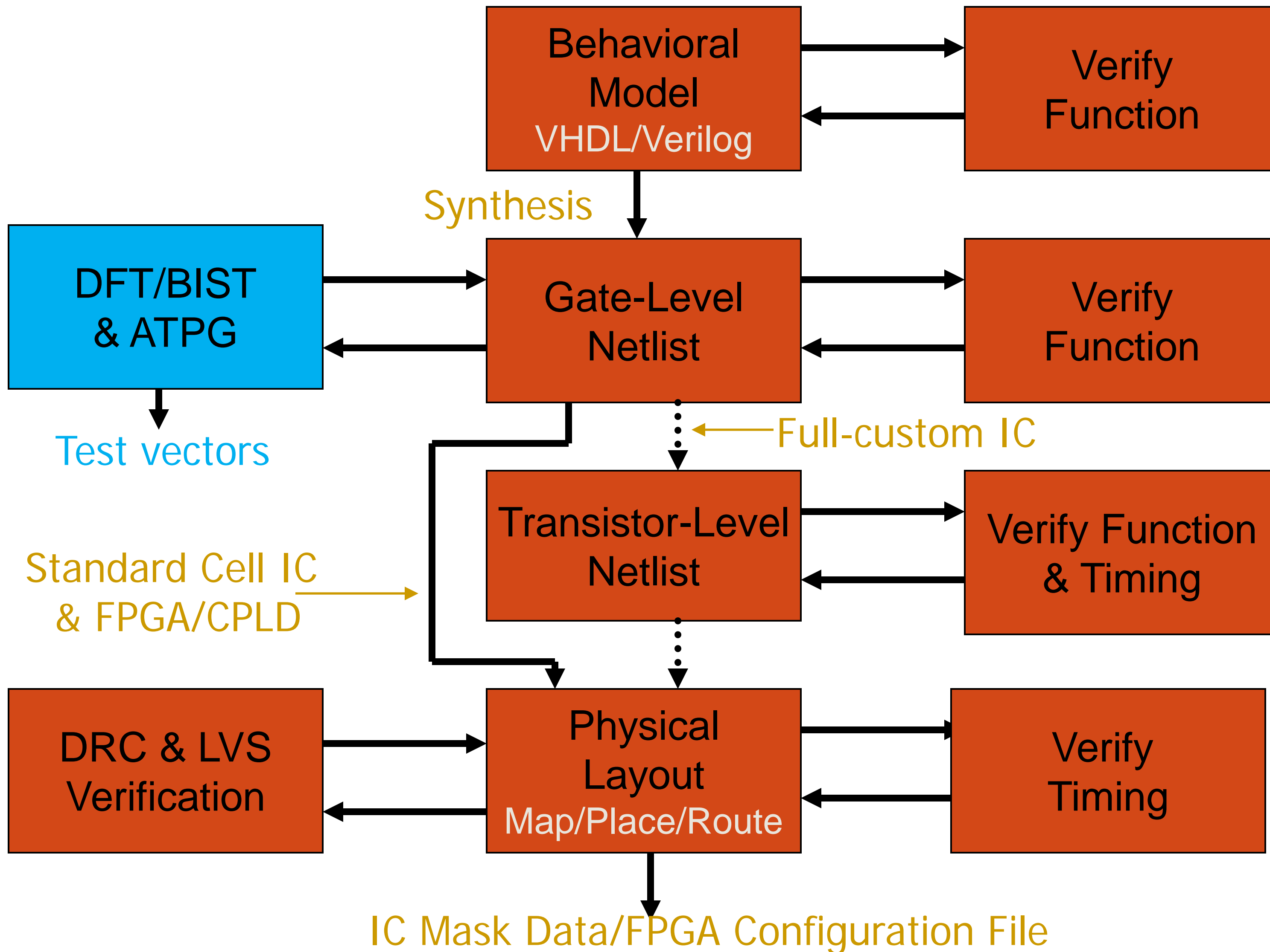
Mentor Graphics/Tessent:

"Scan and ATPG Process Guide"

"ATPG and Failure Diagnosis Tools Reference Manual"

(access via "mgcdocs")

ASIC Design Flow



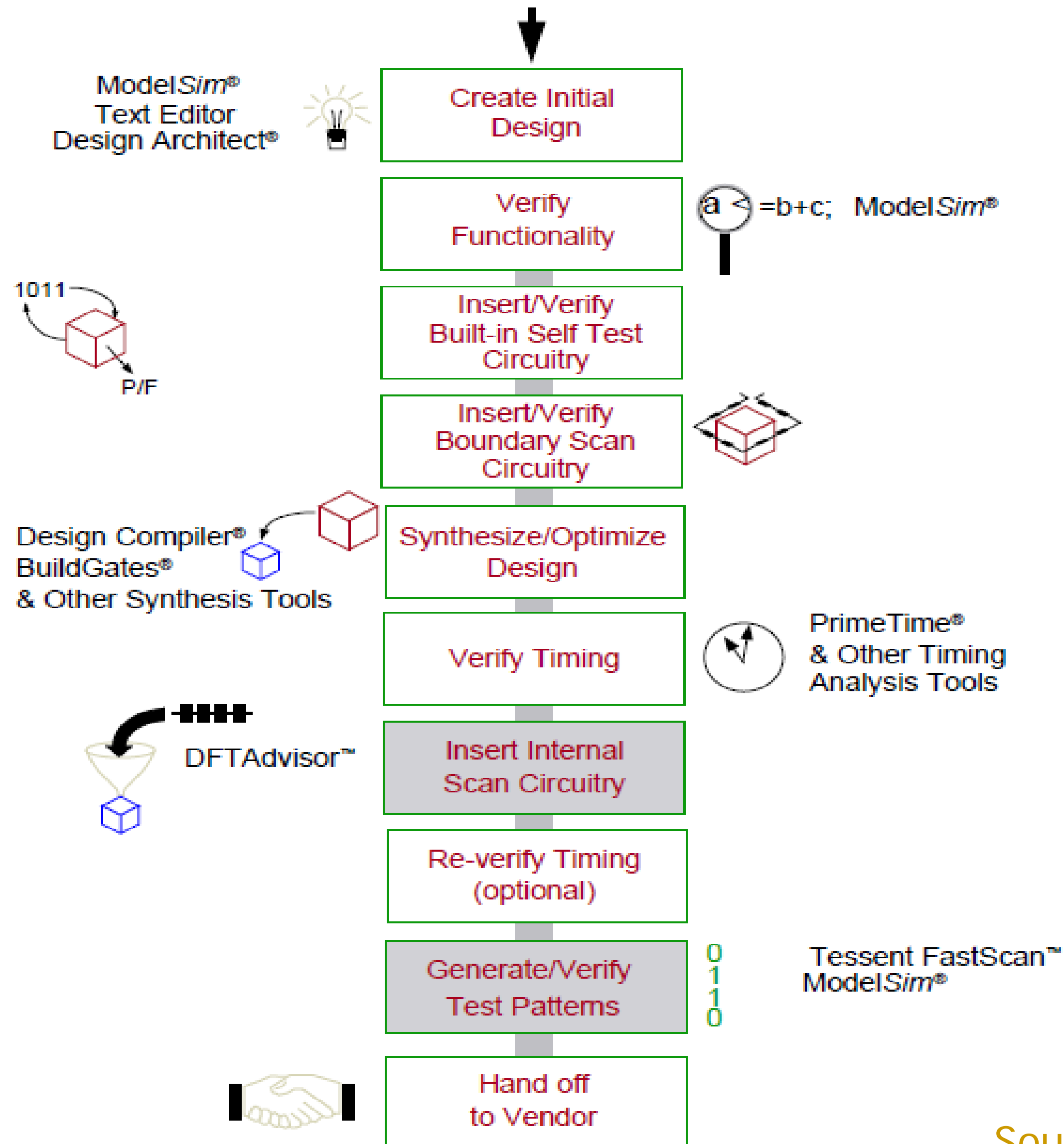
Importance of test (Ch. 14.1)

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

Parts shipped: 100,000 @ part cost = $10
PCB cost: $200, System cost: $5,000, 100,000 systems shipped
System repair/replace cost: $10,000

Q: What would it cost to reduce defect density from 5% to 1% ?

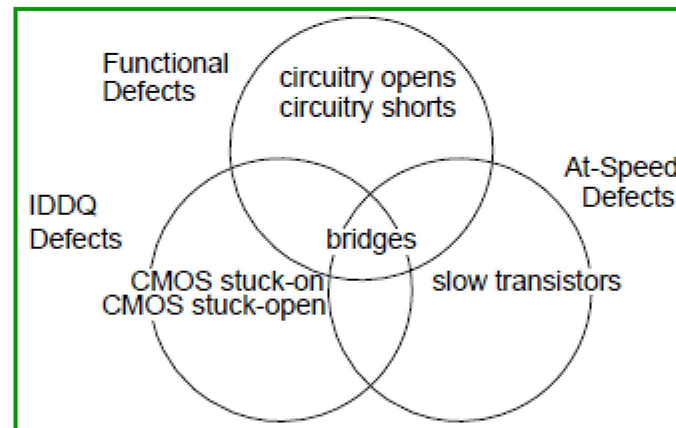
Top-down test design flow



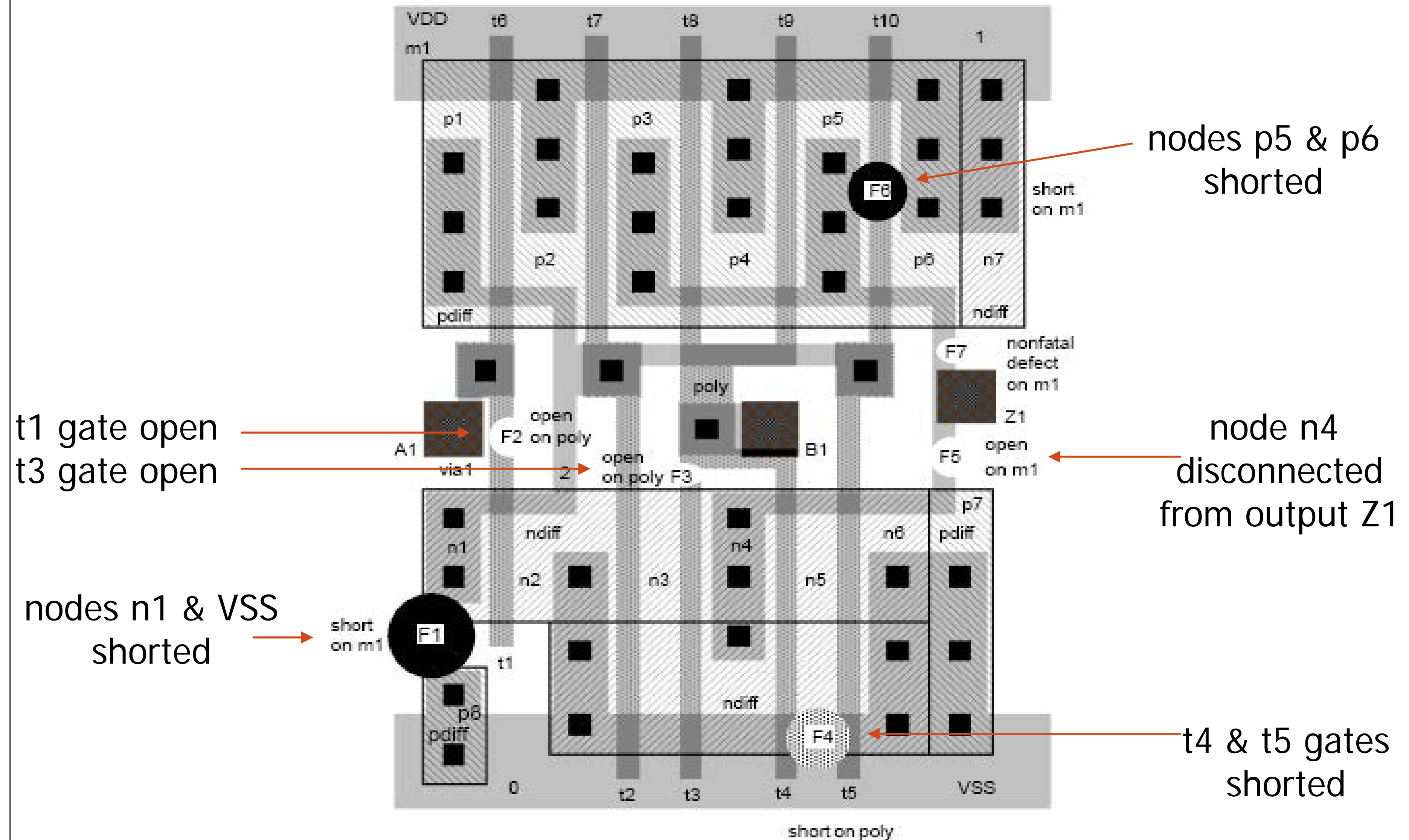
Source: ATPG Manual

Fault models *(supported by FastScan)*

- Map physical fault to a "logical" fault, i.e. faults that change the logic function of the circuit
  - Stuck-at: net effectively stuck at a logic 0/1
  - Bridging: value on one net affects another net
- "Parametric" faults alter physical properties other than logic function
  - Delay
  - Transition
  - IDDQ
  - Voltage level
  - Signal strength



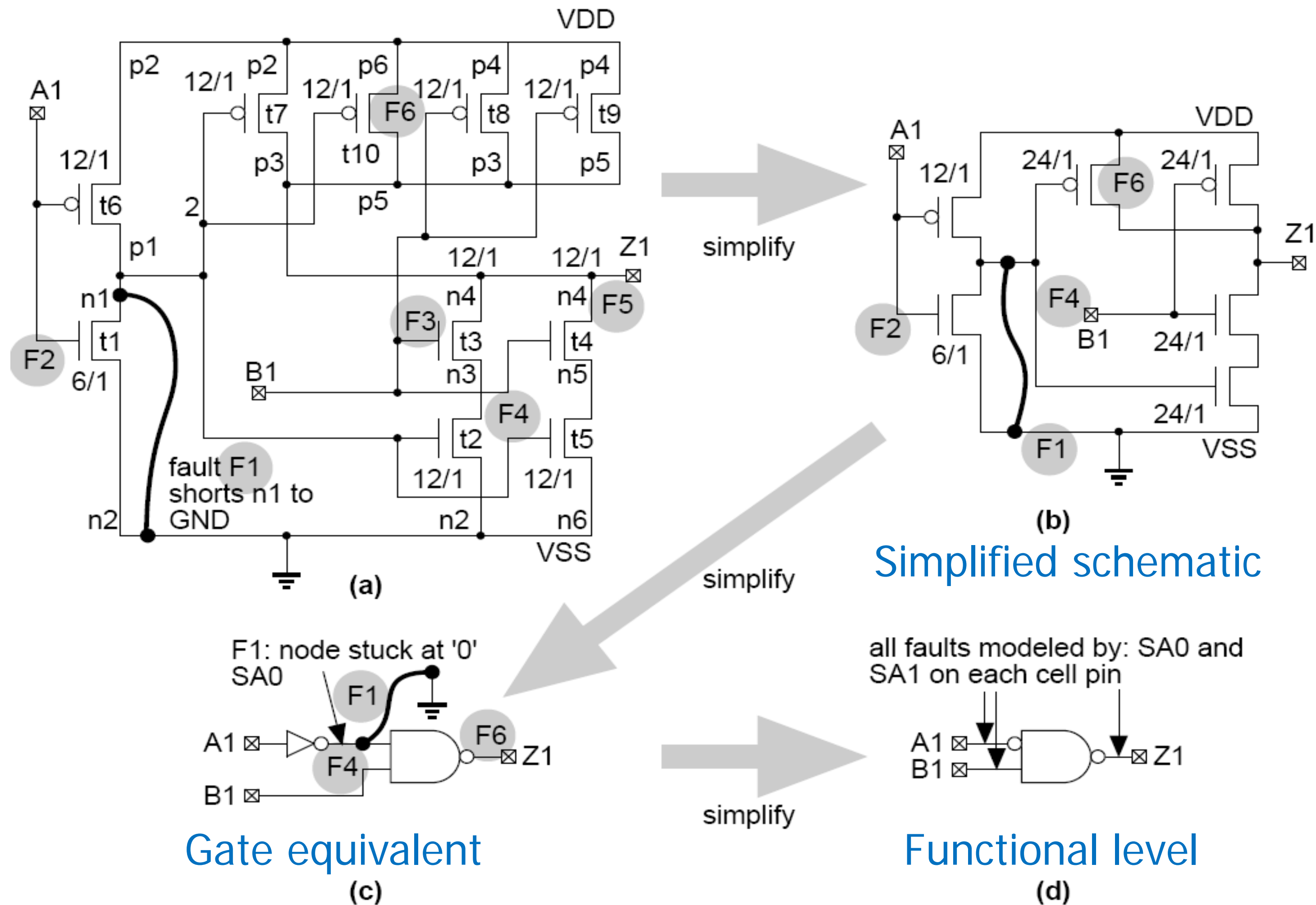
Defects and physical faults



Smith Text: Figure 14.11

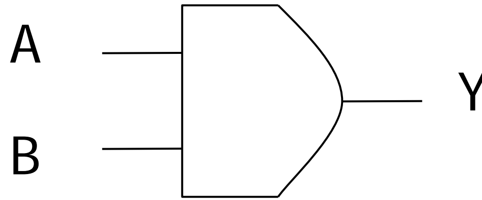
Defects translated to logical faults

(defects from previous slide)



Smith Text: Figure 14.12

Stuck-at faults



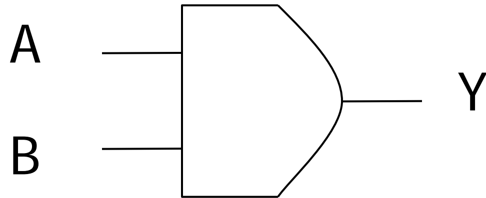
Fault	Test (A,B)	Y(good)	Y(faulty)	Resulting function
A/0	11	1	0	Y=0
A/1	01	0	1	Y=B
B/0	11	1	0	Y=0
B/1	10	0	1	Y=A
Y/0	11	1	0	Y=0
Y/1	01,10,00	0	1	Y=1

Fault collapsing

- Fault A equivalent to fault B if every test for A is also a test for B and vice versa
  - Faults are indistinguishable
  - Group equivalent faults into a fault-equivalence class (FEC)
  - ***Only one fault from a FEC needs to be tested***
- Fault A dominates fault B if any test for B also detects A, but only some of A tests detect B
  - ***Need only test dominated fault B***

    (guarantees A tested also)

Fault collapsing example

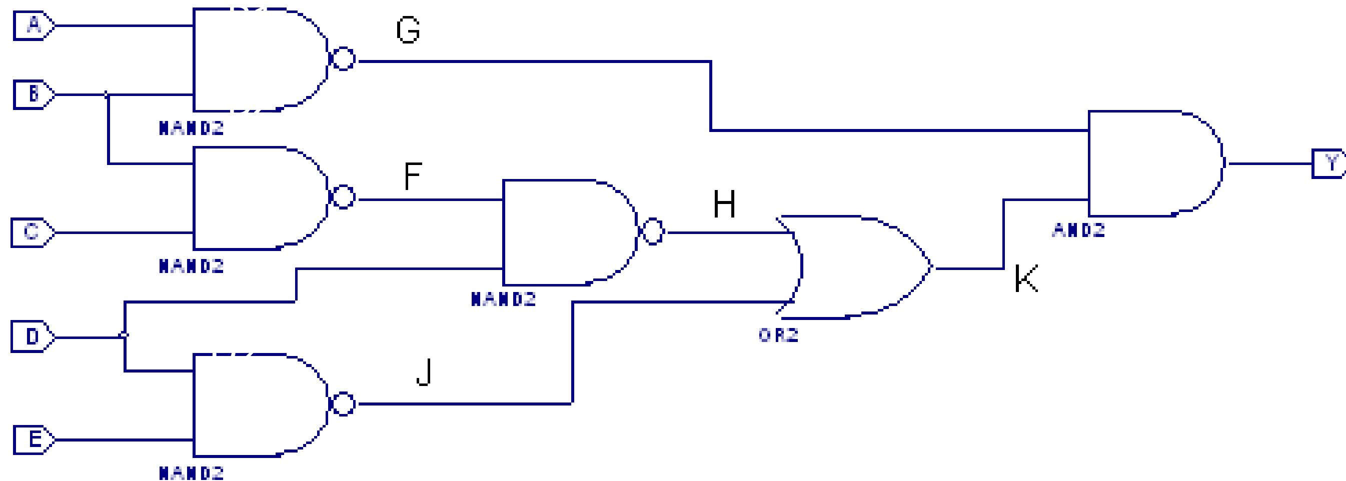


A/0 = B/0 = Y/0 (remove B/0, Y/0)
A/1 dominates Y/1 (remove Y/1)
B/1 dominates Y/1 (remove Y/1)

Collapsed fault set = {A/0, A/1, B/1}
Min test set = {11, 01, 10}

Fault	Test (A,B)	Y(good)	Y(faulty)	Collapsing
A/0	11	1	0	A/0 equiv to B/0, Y/0
A/1	01	0	1	A/1 dom by Y/1
B/0	11	1	0	B/0 equiv to A/0, Y/0
B/1	10	0	1	B/1 dom by Y/1
Y/0	11	1	0	Y/0 equiv to A/0, B/0
Y/1	01,10,00	0	1	Y/1 dom A/1, B/1

Fault collapsing example



Equivalence classes:

{A/0, B1/0, G/1} – keep G/1
{B2/0, C/0, F/1} – keep F/1
{G/0, K/0, Y/0} – keep G/0
{D2/0, E/0, J/1, H/1, K/1, F/0, D1/0} – keep F/0

Fault dominance: (remove dominated)

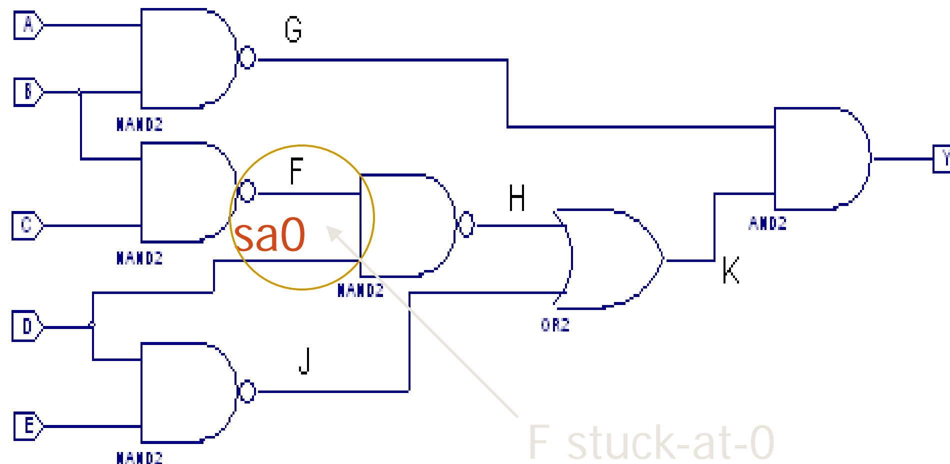
A/1 -> G/0	B1/1 -> G/0
B2/1 -> F/0	C/1 -> H/0
D2/1 -> J/0	E/1 -> J/0
F/1 -> H/0	D1/1 -> H/0
H/0 -> K/0	J/0 -> K/0
G/1 -> Y/1	K/1 -> Y/1

Collapsed fault set = {A/1, B1/1, B2/1, C/1, D1/1, D2/1, E/1, G/1, F/1, K/1}

Test generation: D Algorithm

- Select a fault on net N (ex. sa0)
- Produce an error at N (ex. N=1)
  - "D" denotes expected 1, actual 0 values
  - "D*" denotes expected 0, actual 1 values
- Propagate D condition to primary output
  - Set gate inputs to propage D
- "Justify" D condition from primary inputs
  - Set gate inputs to create D condition

D Algorithm – tabular method



Step	A	B	C	D	E	F	G	H	J	K	Y
control						D					
observe				1		**D**		**D***			
observe				1		D		**D***	**0**	**D***	
observe				1		D	**1**	D*	0	**D***	**D***
Consistency - J				**1**	**1**	D	1	D*	**0**	D*	D*
Consistency - G	**x**	**0**		1	1	D	**1**	D*	0	D*	D*
Consistency - F	x	**0**	**x**	1	1	**D**	1	D*	0	D*	D*

Test vectors: ABCDE = {-0-11, 0-011} = {00011,10011,00111,10111,01011}

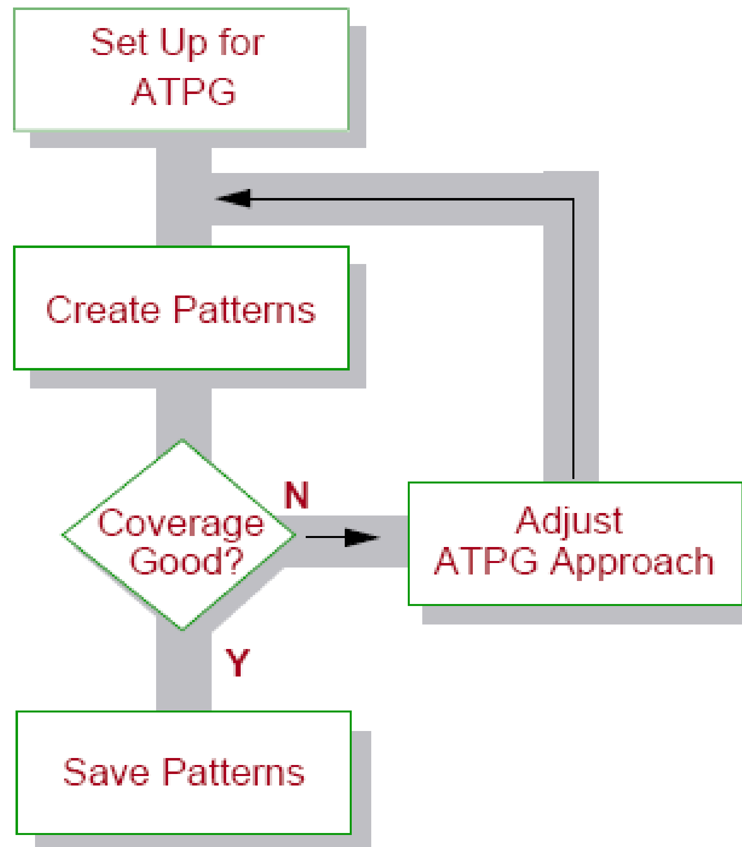
Generate and verify a test set

- Collapse the fault set
  - minimize fault set required for 100% fault coverage

    "coverage" = (# faults detected)/(# possible faults)
- Automatic test pattern generation (ATPG)
  - apply D algorithm or other method to derive test patterns for all faults in the collapsed fault set
  - "random patterns" detect many faults – use deterministic method to detect the others (*FastSscan*)
- Fault simulation
  - verify fault coverage of test patterns
  - simulate fault, apply test pattern, and observe output

    fault detected if output different from expected value
  - repeat for each fault & test pattern combination

ATPG flow



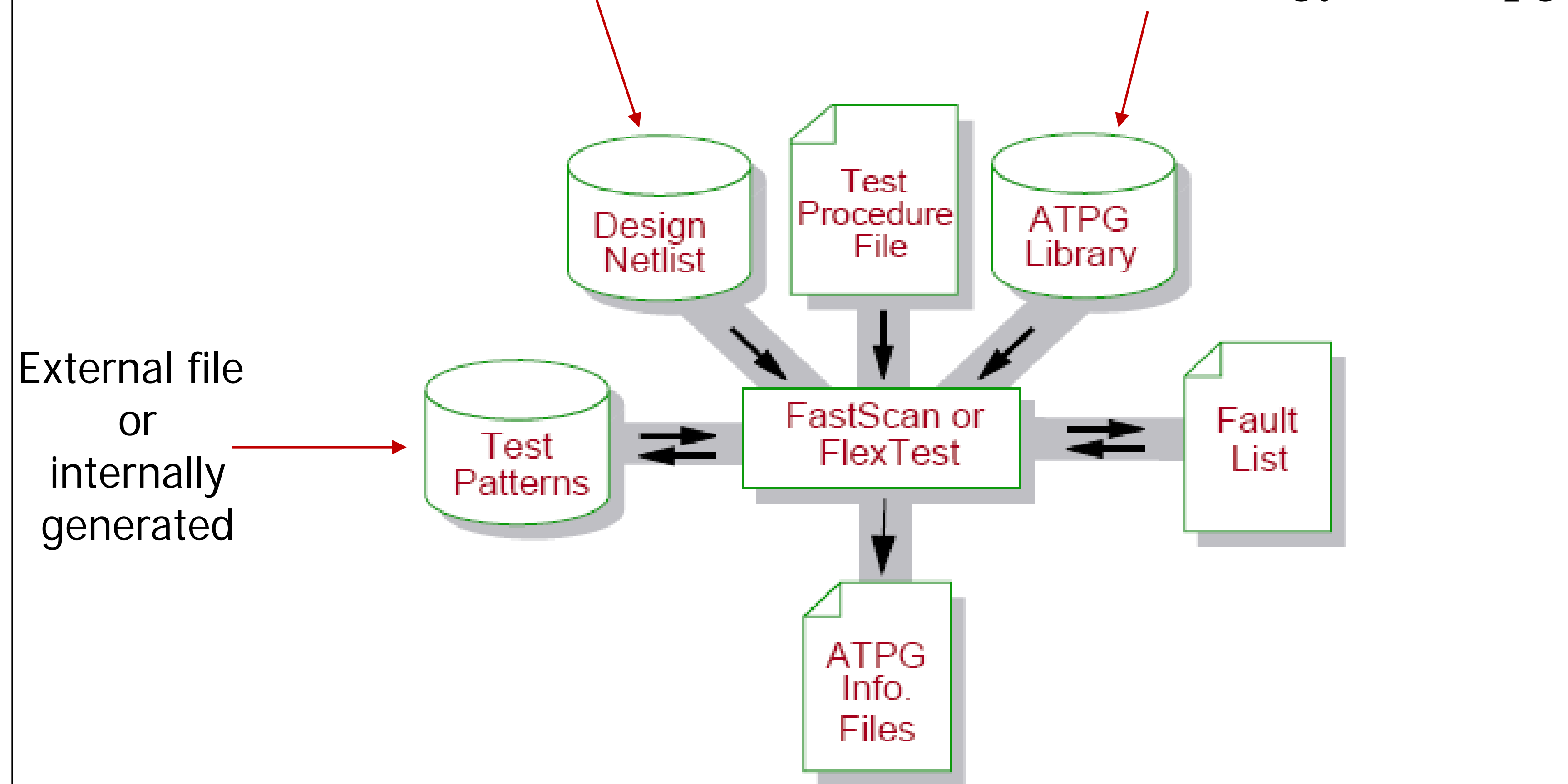
Source: ATPG Manual

Mentor Graphics “Tessent” *FastScan*

- Perform design for testability (DFT), ATPG, and fault simulation
  - FastScan: full-scan designs

    Legacy: FlexTest: non-scan through full-scan designs
- Typical flow:
  1. Implement DFT
  2. Generate test patterns (ATPG)
  3. Verify fault coverage of patterns through fault simulation

FastScan inputs & outputs



Source: ATPG Manual

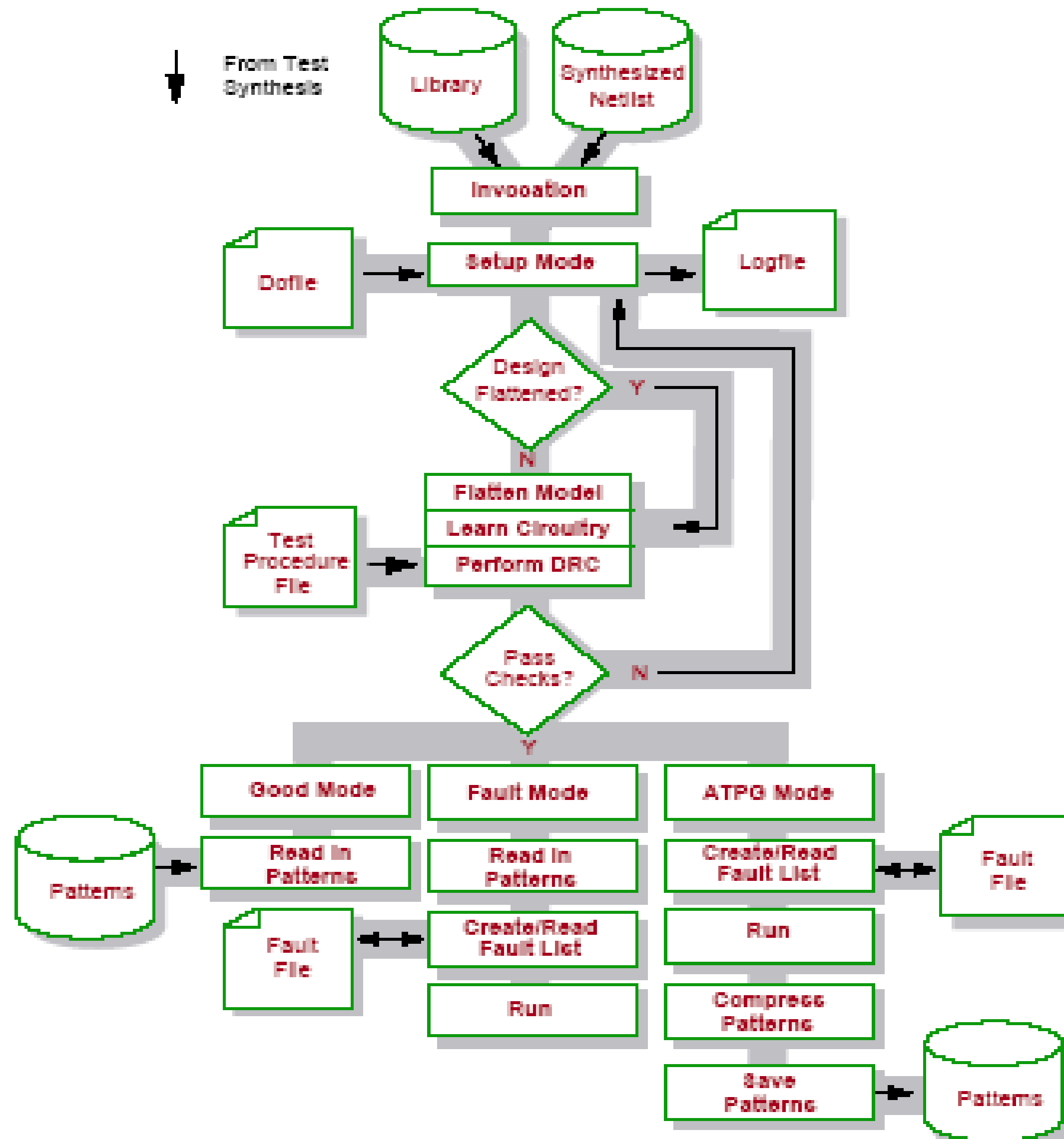
FastScan Flow

Setup mode

Flatten model, Study circuit, Perform DRC

Enter: Good/ATPG/Fault mode

Save patterns & fault lists



Source: ATPG Manual

Invoking FastScan

Command> fastscan testckt.v –lib $ADK/technology/adk.atpg

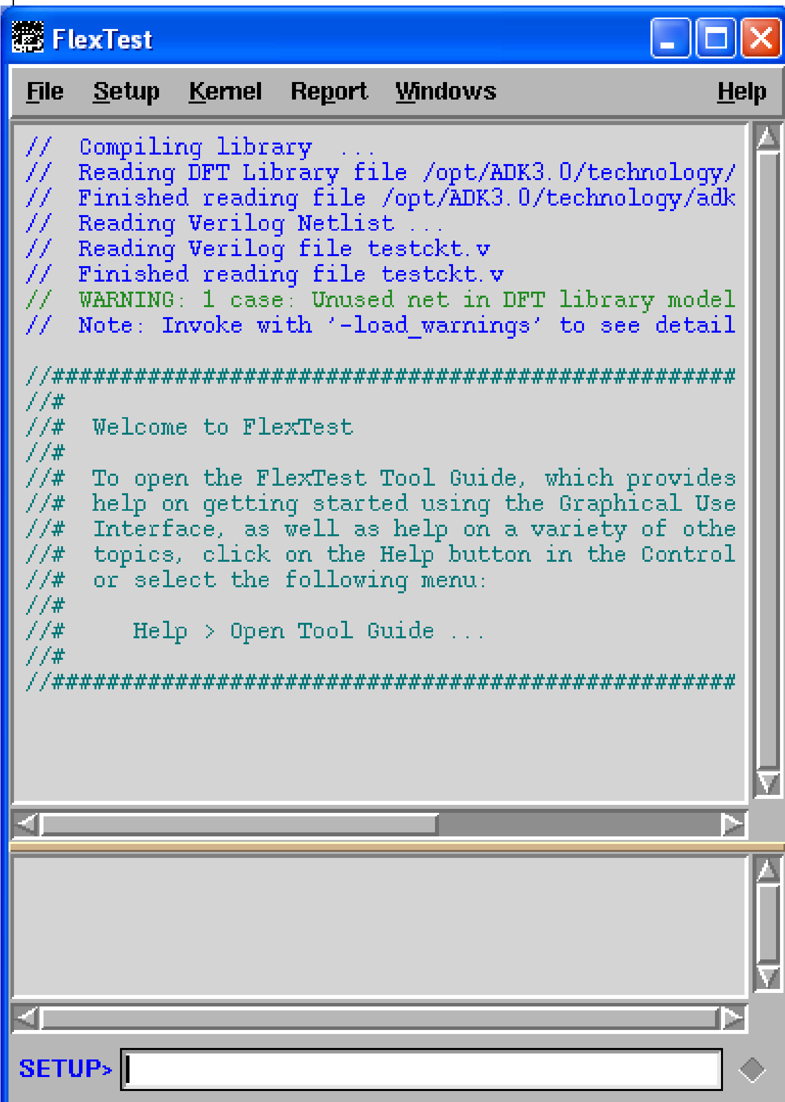
Verilog Netlist

ATPG Library

-DOFile dofile_name

Execute commands From "do file"

“Legacy” FlexTest/FastScan control panel



FlexTest Control Panel

Circuit Setup

Internal Circuitry

Clocks

Scan Circuitry

Primary Outputs

Primary Inputs

RAM

RD

WR

Dout

Done With Setup

Session Transcripting...

Modeling/DRC Setup...

ATPG & Fault Sim Setup...

Cycle Timing...

Open DFTInsight

Dofile...

Exit...

Help...

Setup

DRC and Circuit Learning

DRC Violation Debugging

ATPG or Simulation

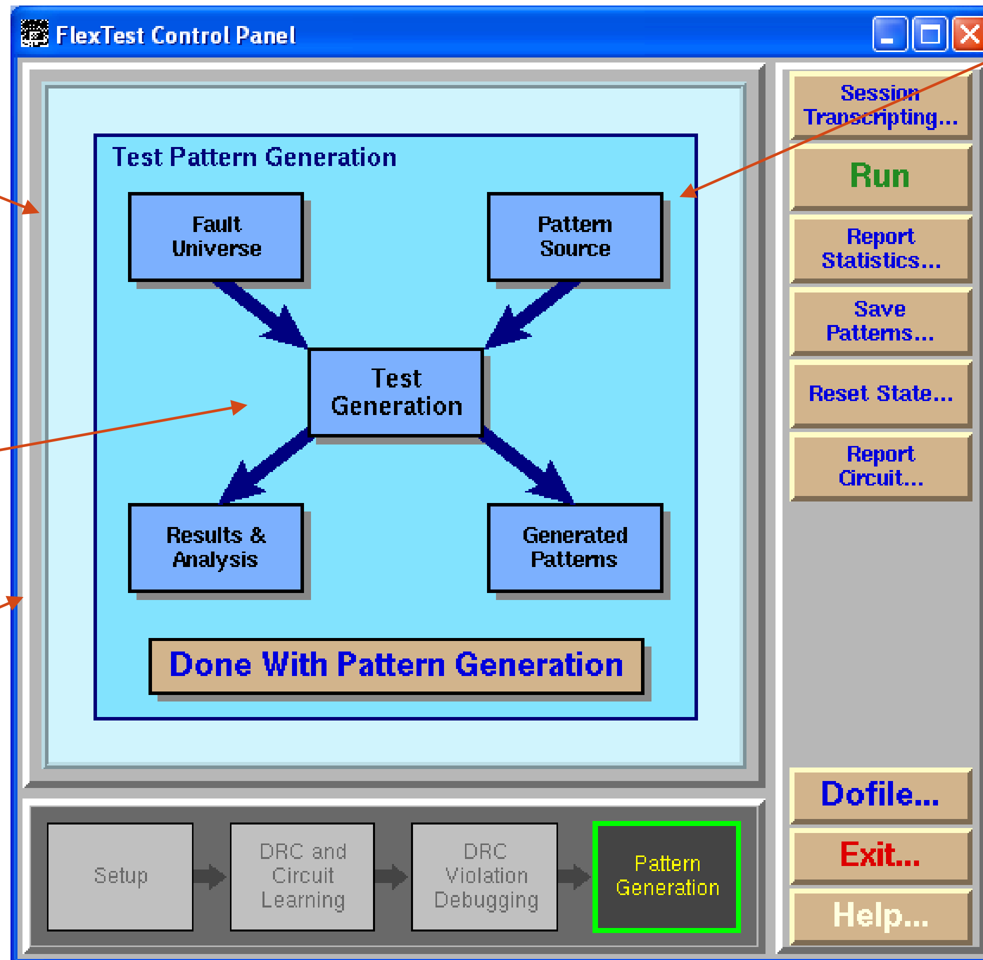
“Legacy” FlexTest ATPG control panel

1. Select faults to be tested

2. Select auto test patterns or external test file

3. Run the ATPG and fault simulation

4. Report results



Design rule checks

Check for design issues that affect testability

Example: synchronous circuit DRC:

- System has a minimum number of clocks—optimally only one.
- Register all design inputs and account for metastability.
  - Treat metastability time as another delay in the path.
  - If the propagation delay plus the metastability time is less than the clock period, the system is synchronous.
  - If it is greater than or equal to the clock period, add an extra flip-flop to ensure the proper data enters the circuit.
- No combinational logic drives flip flop set/reset/clock inputs.
- No asynchronous signals set or reset the flip-flops.
- Buffers or other delay elements do not delay clock signals.
- Do not use logic to delay signals.
- Do not assume logic delays are longer than routing delays.

Select fault types/models

- *set fault type Stuck* (default)
  - Fault model to be used for ATPG
  - Also: Iddq, Toggle, Transition, Path_delay, Bridge
  - Can specify "multiple detection" # to require multiple detections of each fault
- *add faults –all*
  - add faults to current fault list, discarding all patterns and setting all faults to undetected
  - options for each fault model (ex. Stuck_at 01 or 1 or 0)
- *load faults file* - load fault list from file
- *write faults file* – write fault list to file
- *report faults* – print list of modeled faults

Select the test pattern source

- ***set pattern source Internal***
  - Generate and use ATPG patterns
- ***set pattern source External filename***
  - Load and use patterns from file
- ***set pattern source Random***
  - Generate and use random patterns
- ***set pattern source Bist***
  - BIST circuit generates patterns

FastScan simulation modes

- Good – verify the simulation model
  - Use ATPG algorithm to generate test patterns
  - Apply patterns and capture outputs without simulating faults
  - Produces expected output for each test pattern
- Fault – determine fault coverage of a set of patterns
  - User provides a set of test patterns and fault list
  - Perform fault simulation using these patterns and faults to determine coverage
- ATPG -
  - Use ATPG algorithms to generate test patterns for given faults
  - Perform fault simulation using generated patterns to determine coverage of the ATPG-produced test set

Fault simulation with external file selected as "Pattern Source" ("Table Pattern" option)

// FastScan test pattern file – define inputs

PI A

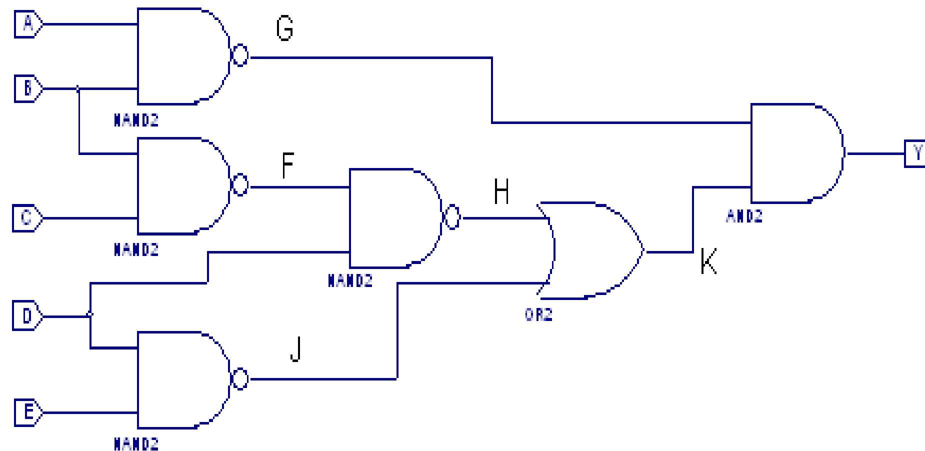
PI B

PI C

PI D

PI E

PO Y



// test patterns – bits in above order

000100

010000

011111

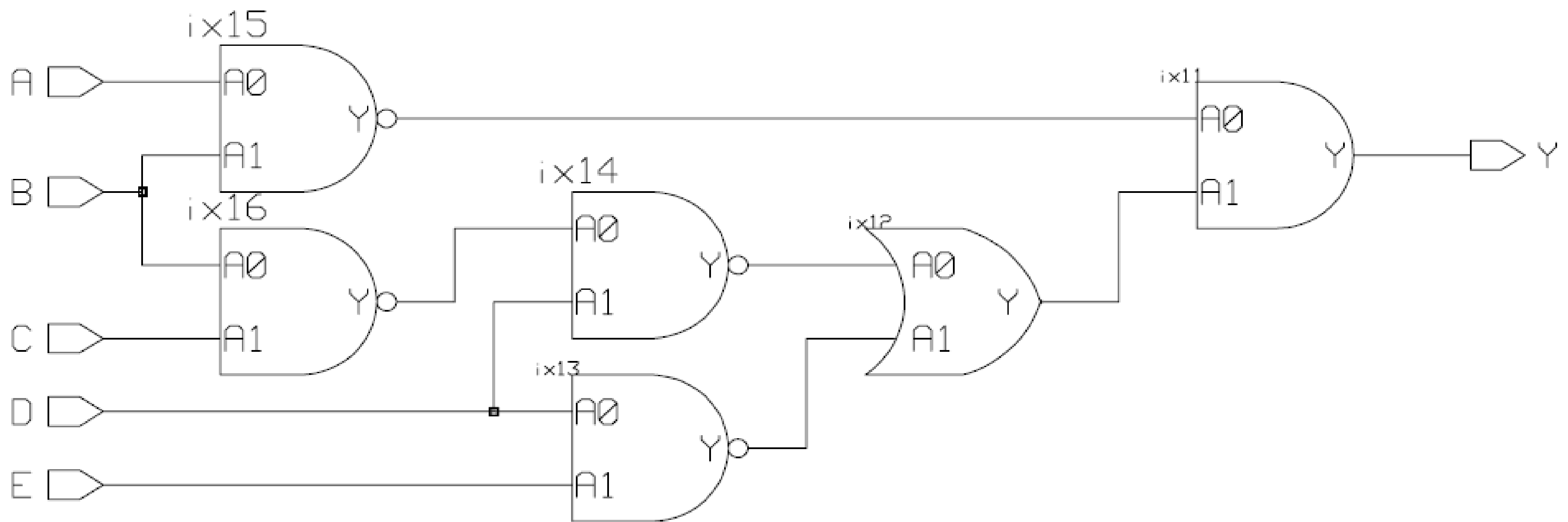
100111

100010

Note: These were "random" patterns

Example circuit (Design Architect)

Net names appear in FastScan report on next page



FastScan fault simulation results

1	RE	/ix14/A1
1	RE	/ix13/A0
1	DS	/ix15/A1
1	DS	/B
1	DS	/D
0	DS	/D
1	DS	/ix11/A1
1	DS	/ix12/Y
1	DS	/ix12/A1
1	DS	/ix13/Y
0	DS	/ix13/A1
0	DS	/E
0	DS	/ix13/A0
1	DS	/ix12/A0
1	DS	/ix14/Y
0	DS	/ix14/A0
0	DS	/ix16/Y
0	DS	/ix14/A1
1	DS	/Y
1	DS	/ix11/Y
0	DS	/B
1	DS	/ix14/A0
1	DS	/ix16/Y
0	DS	/ix16/A1
0	DS	/C
0	DS	/ix16/A0
0	DS	/ix12/A0
0	DS	/ix14/Y
1	DS	/ix15/A0
1	DS	/A
1	DS	/ix13/A1
1	DS	/E
0	DS	/ix12/A1
0	DS	/ix13/Y
0	DS	/Y
0	DS	/ix11/Y
0	DS	/ix11/A0
0	DS	/ix15/Y
0	DS	/ix11/A1
0	DS	/ix12/Y
1	UO	/ix16/A1
1	UO	/C
1	UO	/ix16/A0
1	UC	/ix11/A0
1	UC	/ix15/Y
0	UC	/ix15/A0
0	UC	/A
0	UC	/ix15/A1

Test coverage = 38 detected/48 faults = 79%

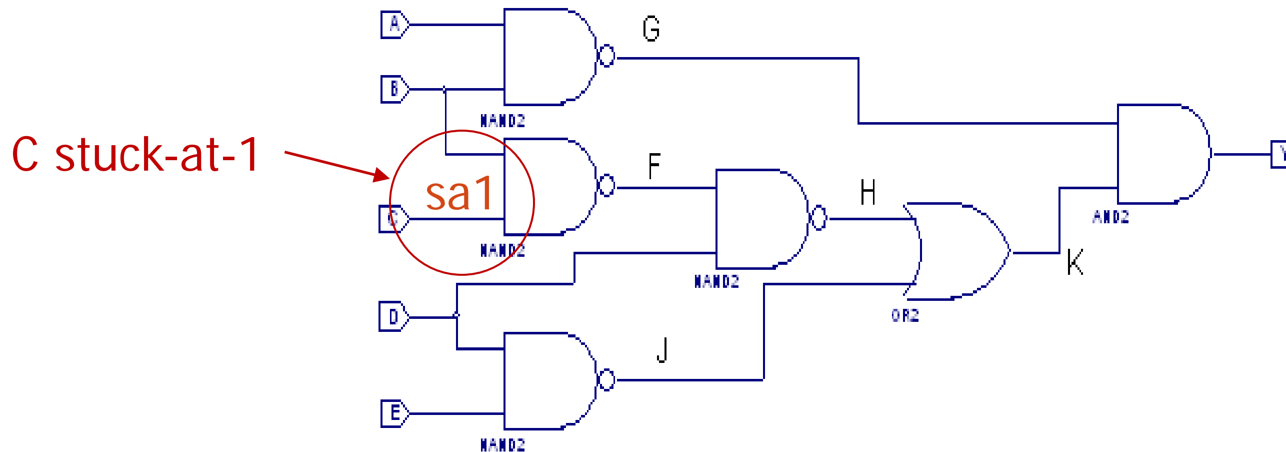
DS – fault detected in simulation

RE – redundant fault

UO – unobserved fault

UC – uncontrolled fault

Design individual tests for UC/UO faults



Step	A	B	C	D	E	F	G	H	J	K	Y
control			D*								
observe		1	D*			D					
observe				1		D		D*			
observe								D*	0	D*	
observe							1			D*	D*
Consistency - J				1	1				0		
Consistency - G	0	x					1				
Final values	0	x	D*	1	1	D	1	D*	0	D*	D*

Test vector: ABCDE = {01011} – Raised fault coverage to 80%

Fault Simulation (Chap. 14.4)

- Deliberately induce faults to determine what happens to circuit operation
- Access limited to primary inputs (PIs) & primary outputs (POs)
- Apply pattern to PIs at start of test cycle
- At end of test cycle, compare POs to expected values
- Fault detected if POs differ from correct values
- Fault coverage = detected faults/modeled faults

Fault simulation types

- Serial – simulate two copies of circuit
  - One good circuit
  - One with a single fault injected
  - Repeat for each possible faulty circuit (slow)
- Parallel – simulate N copies of the circuit in parallel
  - N-bit data word holds values of a given signal line in each copy
- Concurrent – Simulate the good circuit, and then only those parts affected by each fault
  - Different faults affect different parts of the circuit
  - Simulate those faults concurrently

Fault simulation types (continued)

- Deterministic fault simulation:
  - Determine which faults are detected by a test set (as in serial, parallel & concurrent)
- Probabilistic fault simulation:
  - Simulate a subset of faults and extrapolate coverage from the sample
- Statistical fault simulation:
  - Predict coverage from fault-free simulation, based on ability to control/observe nodes