

Design for Test

Scan Test

Smith Text: Chapter 14.6

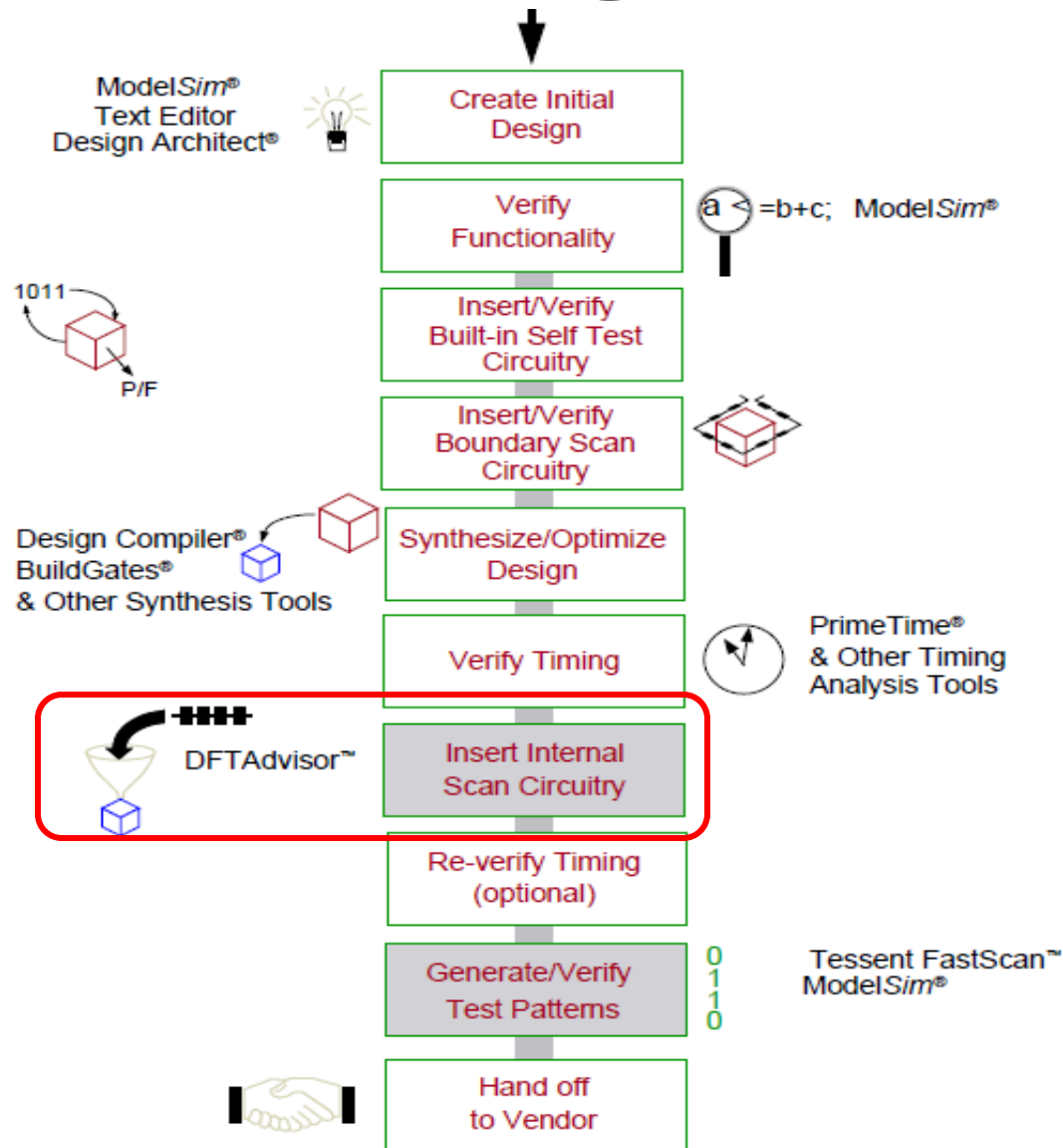
Mentor Graphics Documents:

“Scan and ATPG Process Guide”

“DFTAdvisor Reference Manual”

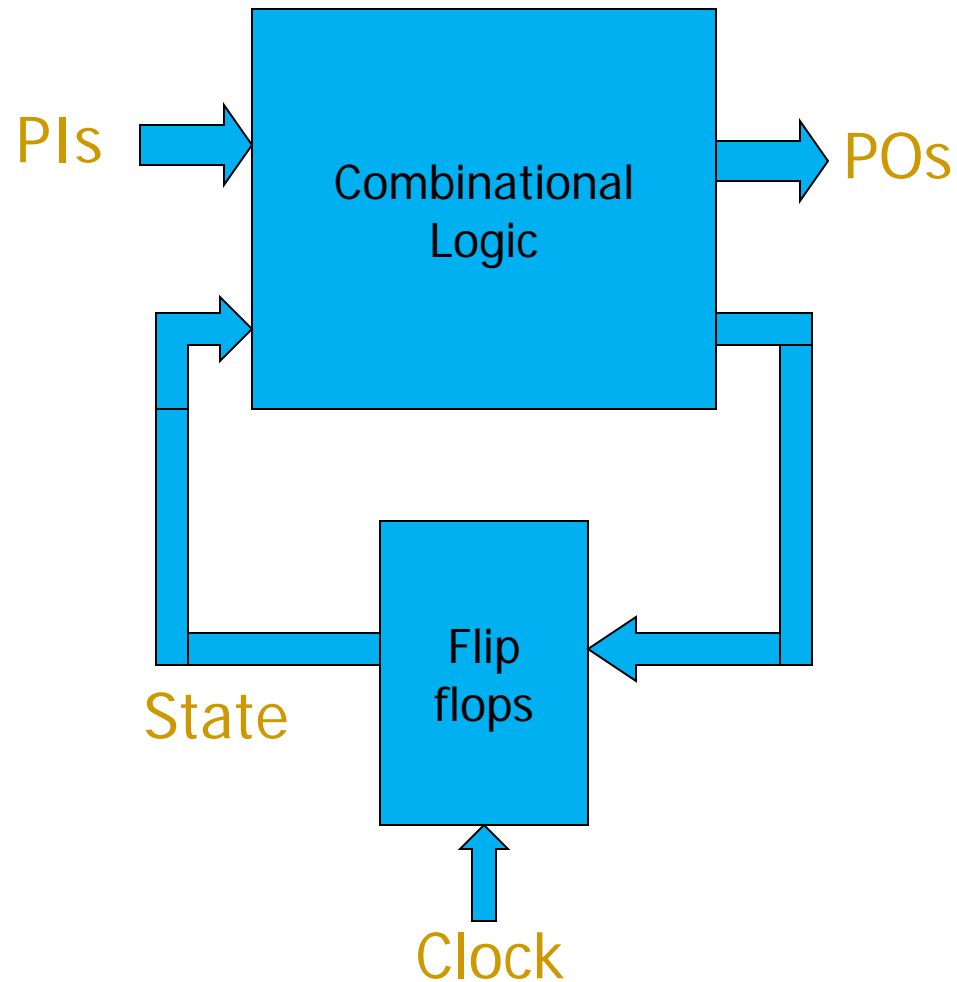
“Tessent Common Resources Manual for ATPG Products

Top-down test design flow

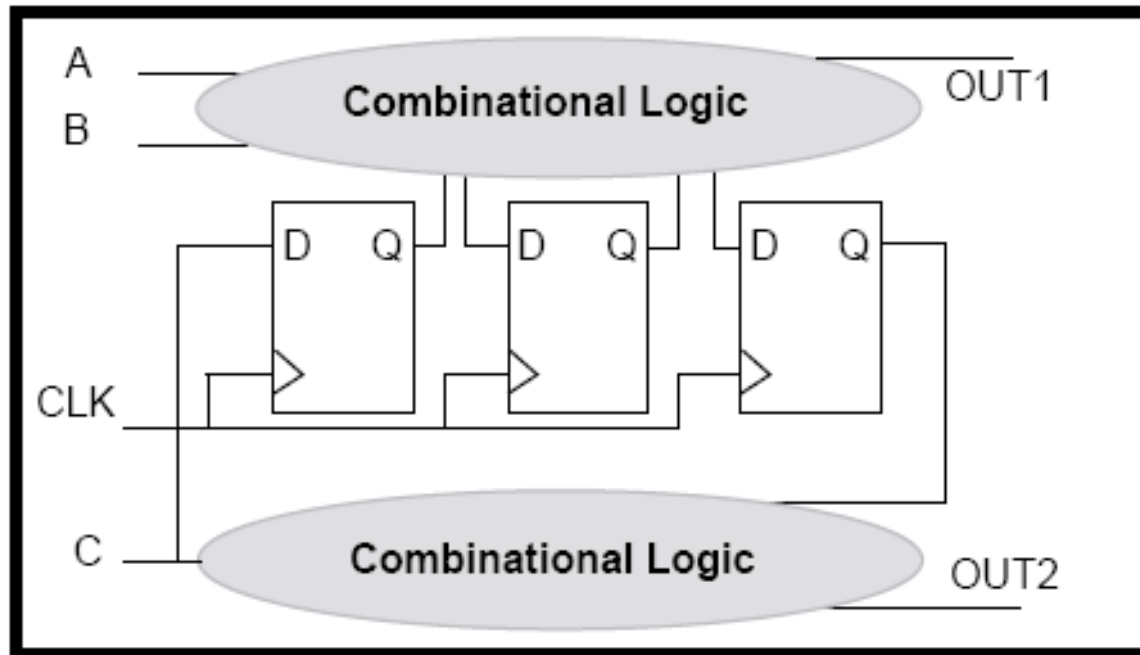


Sequential circuit testing problem

- External access only to PIs and POs
- Internal state is changed **indirectly**
- For N PIs and K state variables, must test 2^{N+K} combinations
- Some states difficult to reach, so even more test vectors are needed

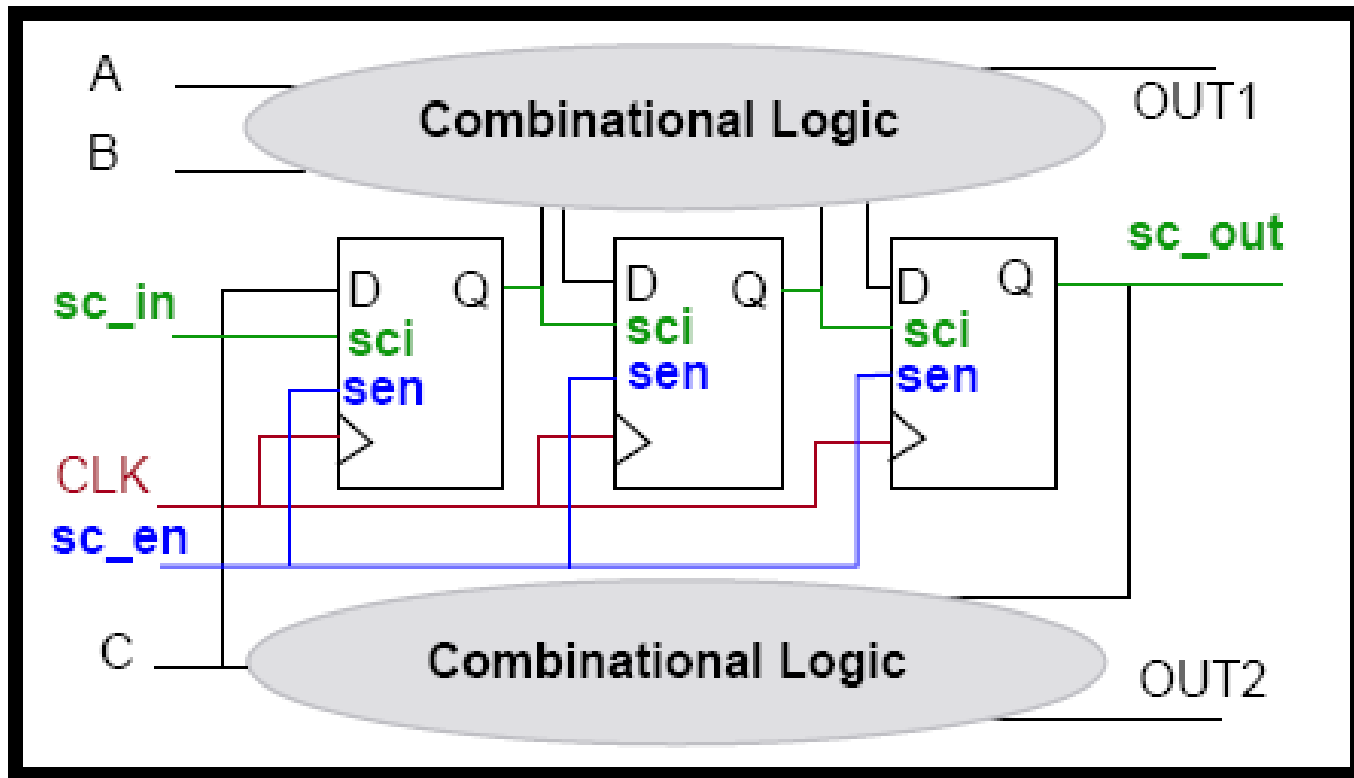


Design for Test (DFT)



Flip flop states are difficult to set from PIs A & B

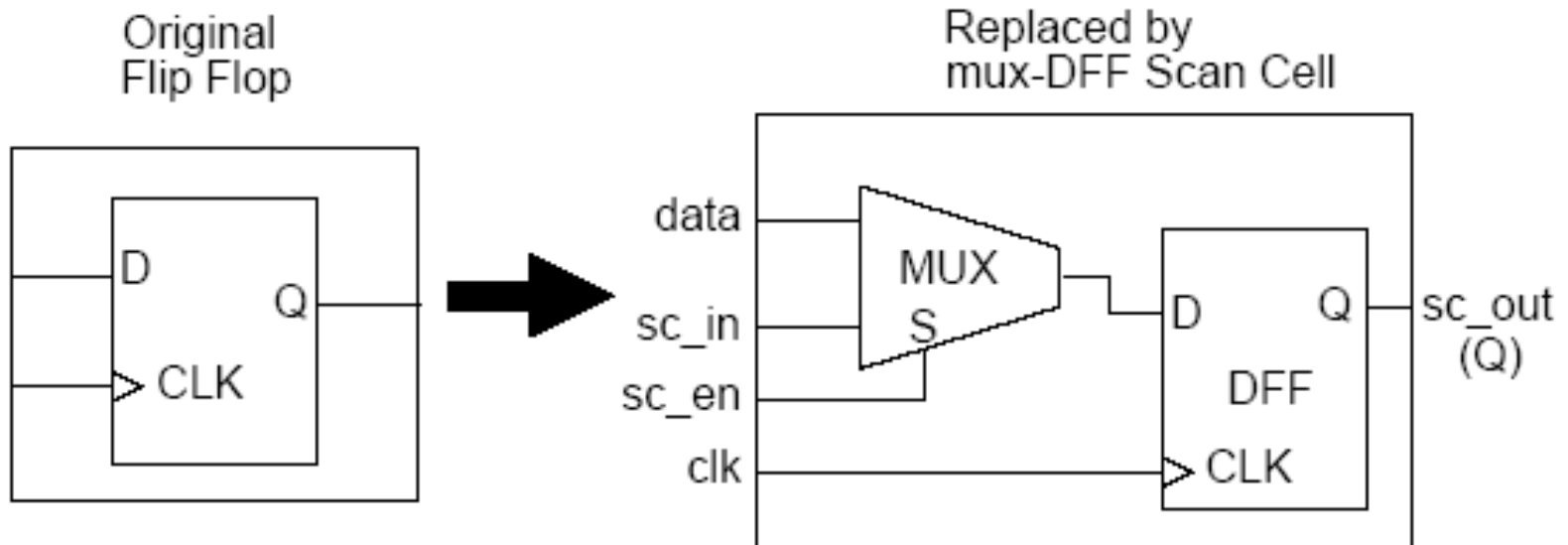
DFT: Scan Design



- Flip flops replaced with “scan” flip flops
- Scan flip flops form a **shift register** in “scan mode”
- Flip flop states set via “scan input” sc_in

Scan type: mux_scan

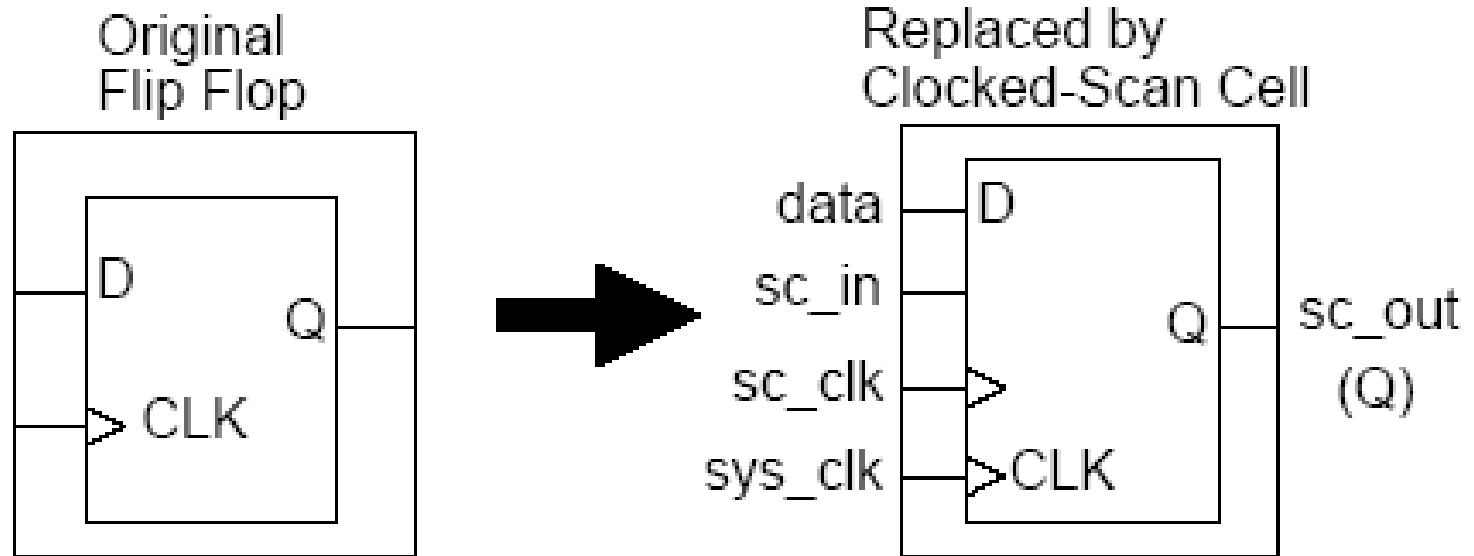
Standard D flip flop with a mux to select system data vs scan data



ADK components: sff, sffr_ni, sffs, sffs_macro, sffsr_ni, sffsr_ni_macro

Scan type: clocked_scan

Separate clocks to load system data and scan data

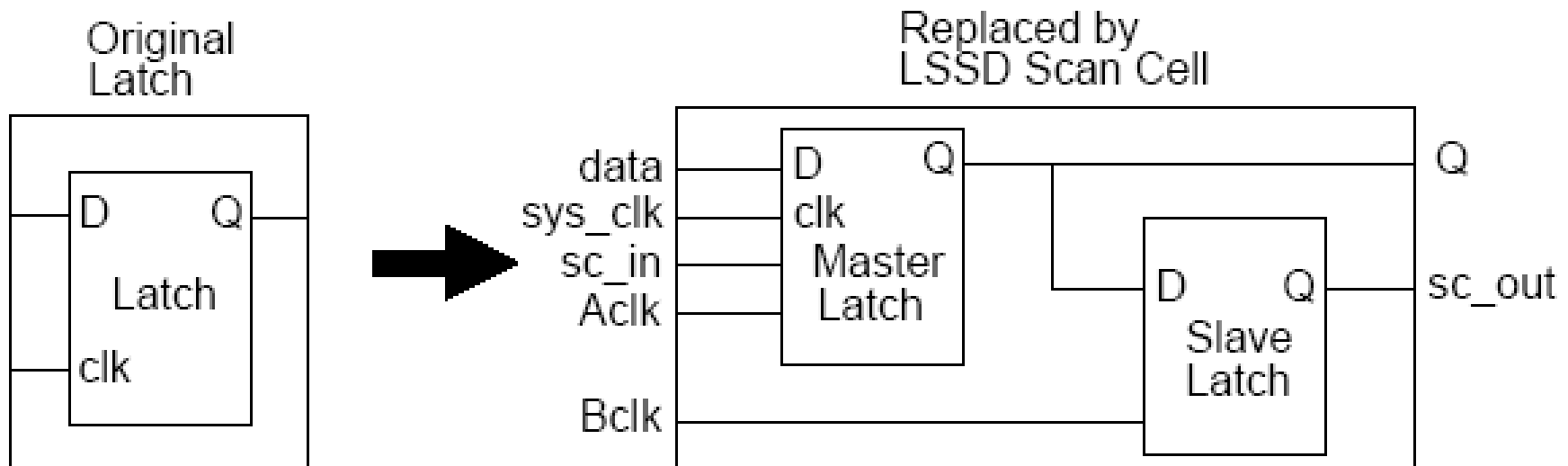


Scan type: LSSD

(Level-sensitive scan design – IBM)

Three clocks:

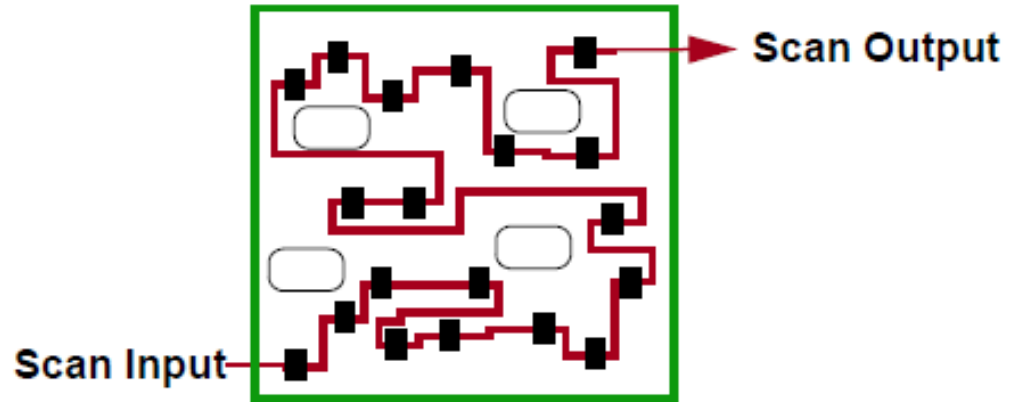
1. sys_clock loads system data into the master latch (normal mode)
2. Aclk loads scan data into the master latch
3. Bclk captures master data in the slave latch to drive scan output



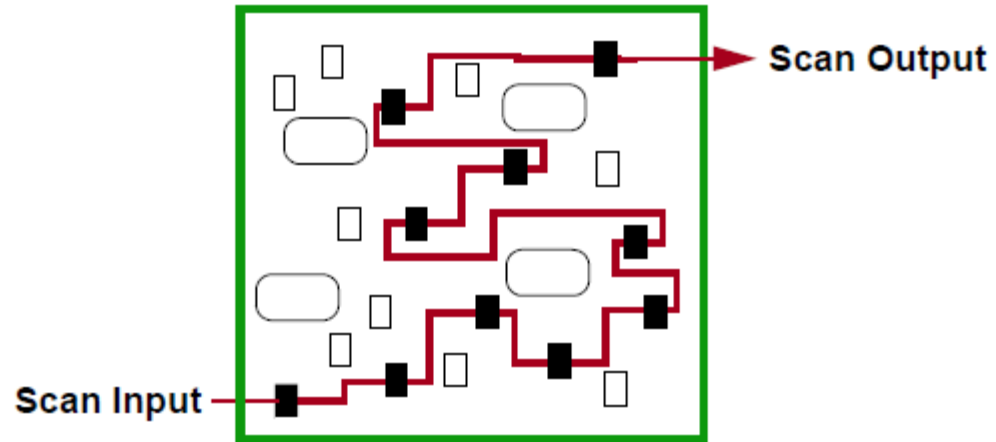
ADK components: [lssd_latch/latchsr/latchr/latches/latches_ni/latchsr_ni](#)

Full vs. partial scan

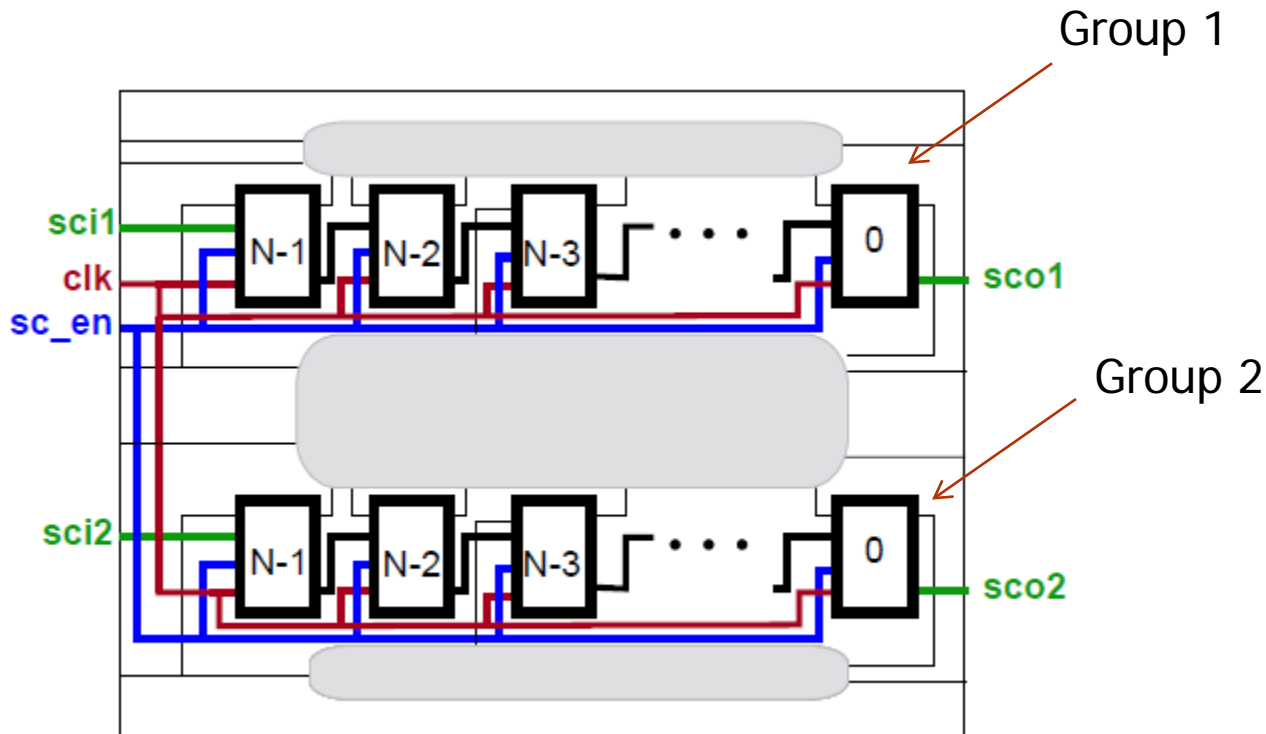
Full Scan
Design



Partial Scan:
Increase testability,
without affecting
critical timing/areas

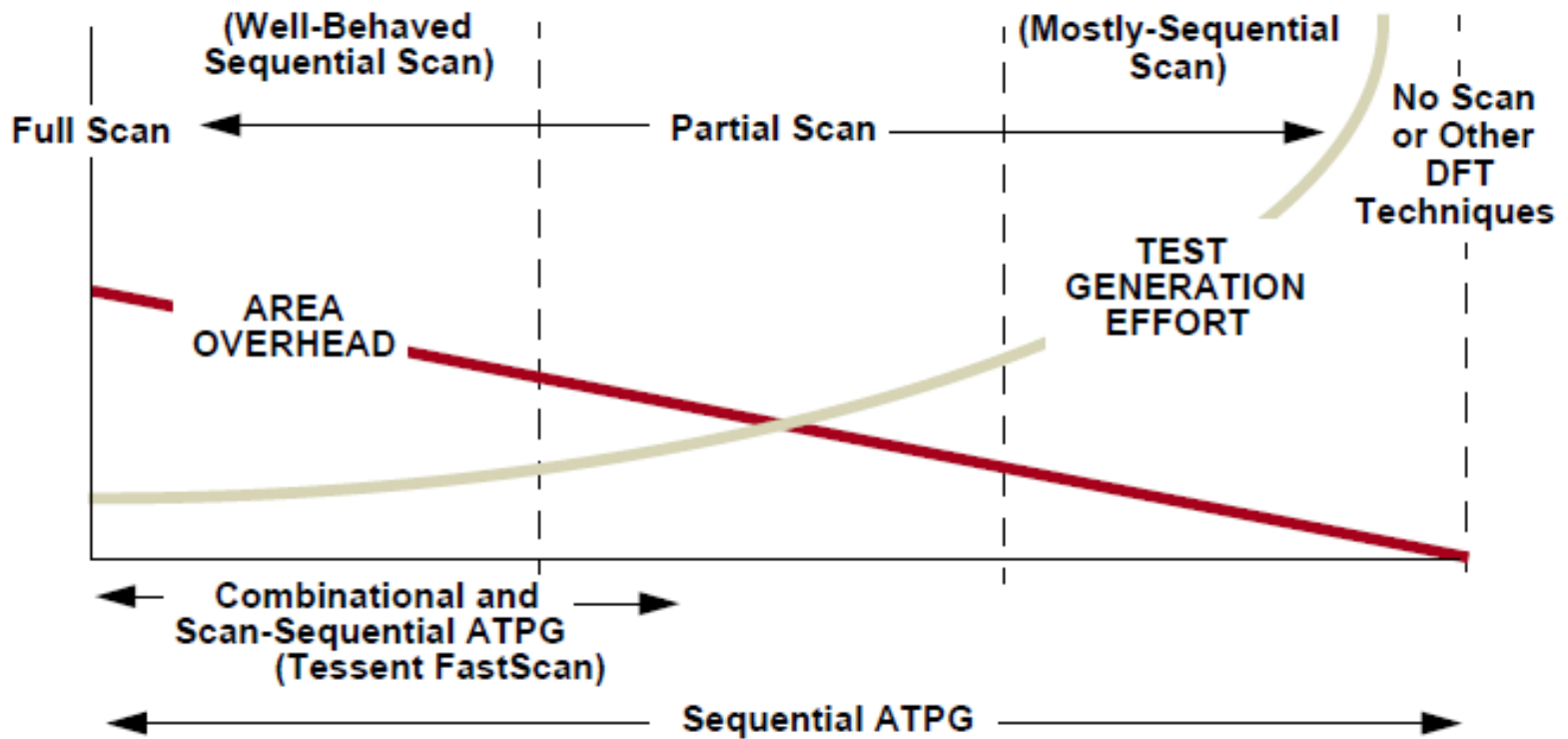


Scan chain groups

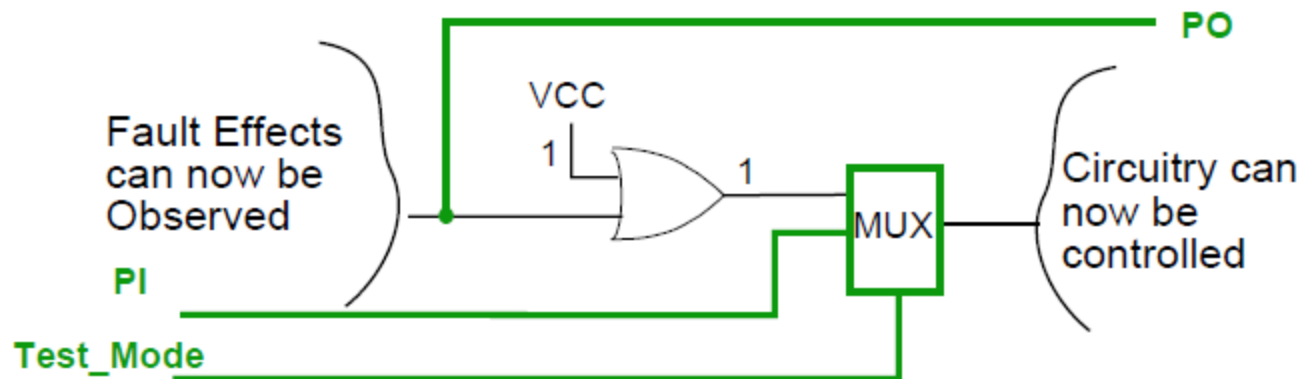
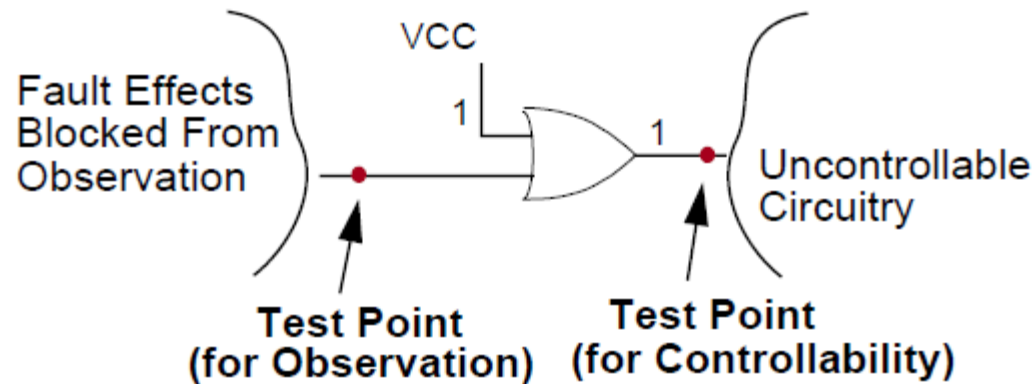


- Scan chains operate in parallel from separate scan inputs to reduce the number of clock cycles to load/unload the chain
- Control from one procedure file
- Can use separate clocks or a common clock

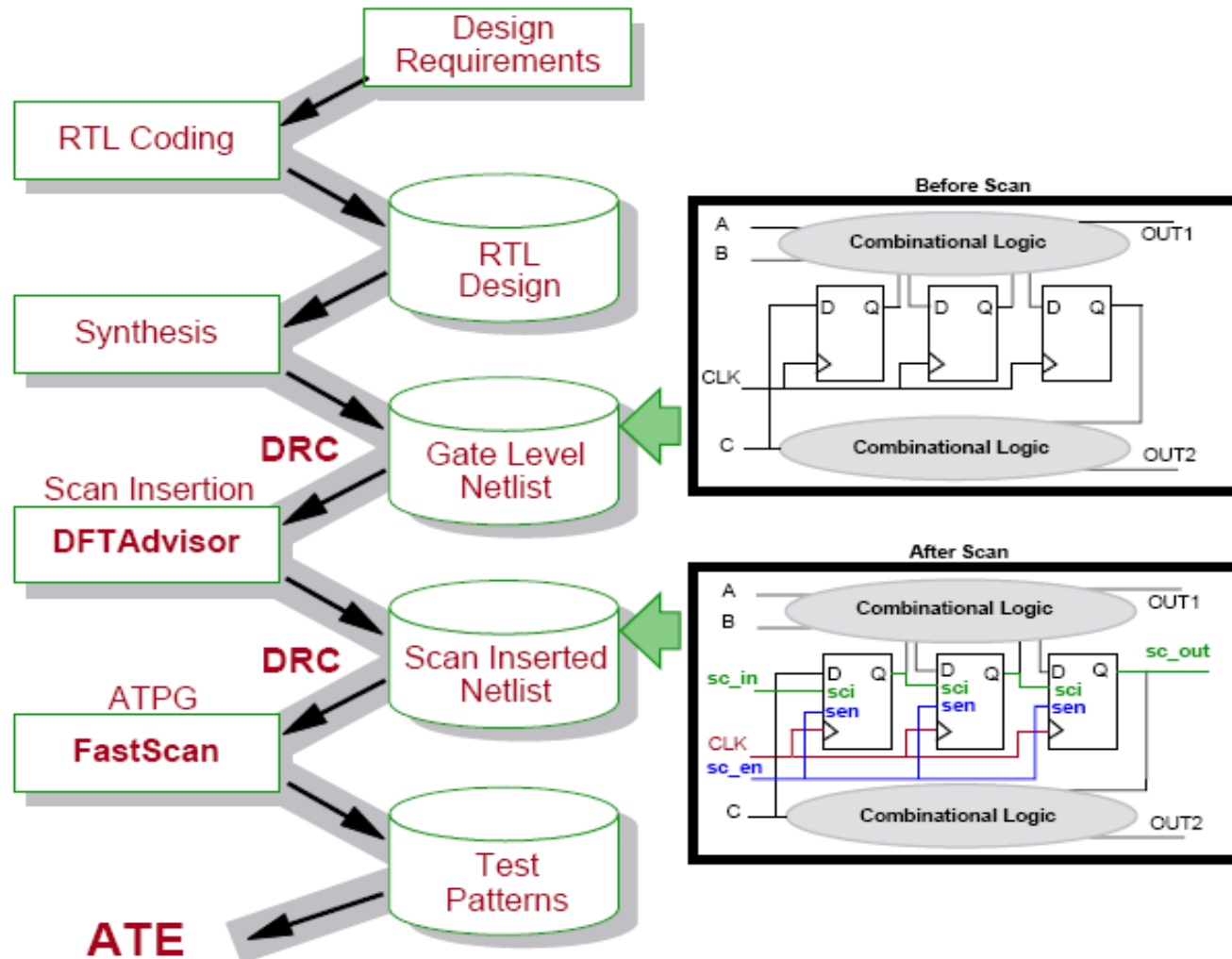
Choosing a DFT solution



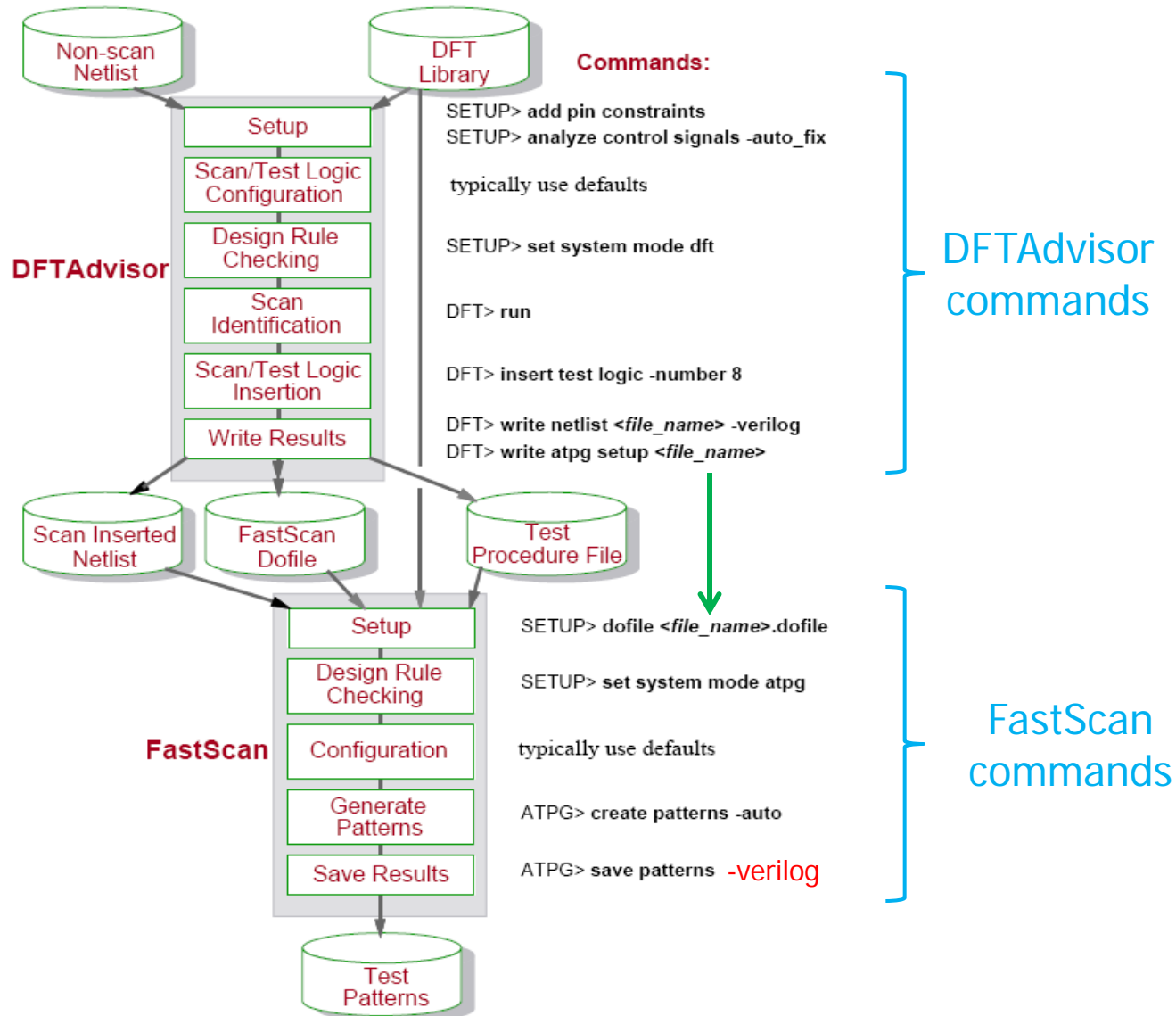
DFT = test point insertion



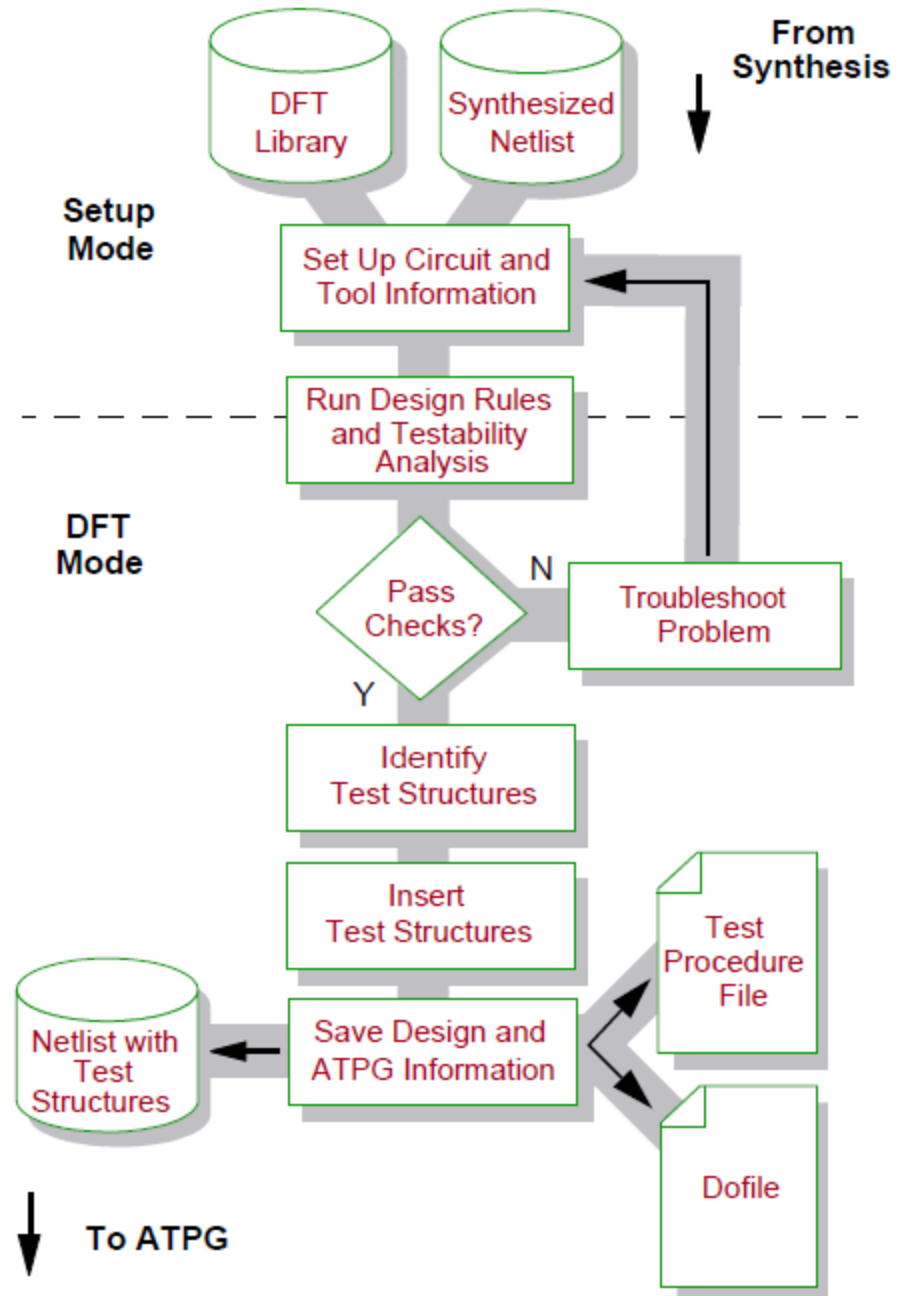
DFTadvisor/FastScan Design Flow



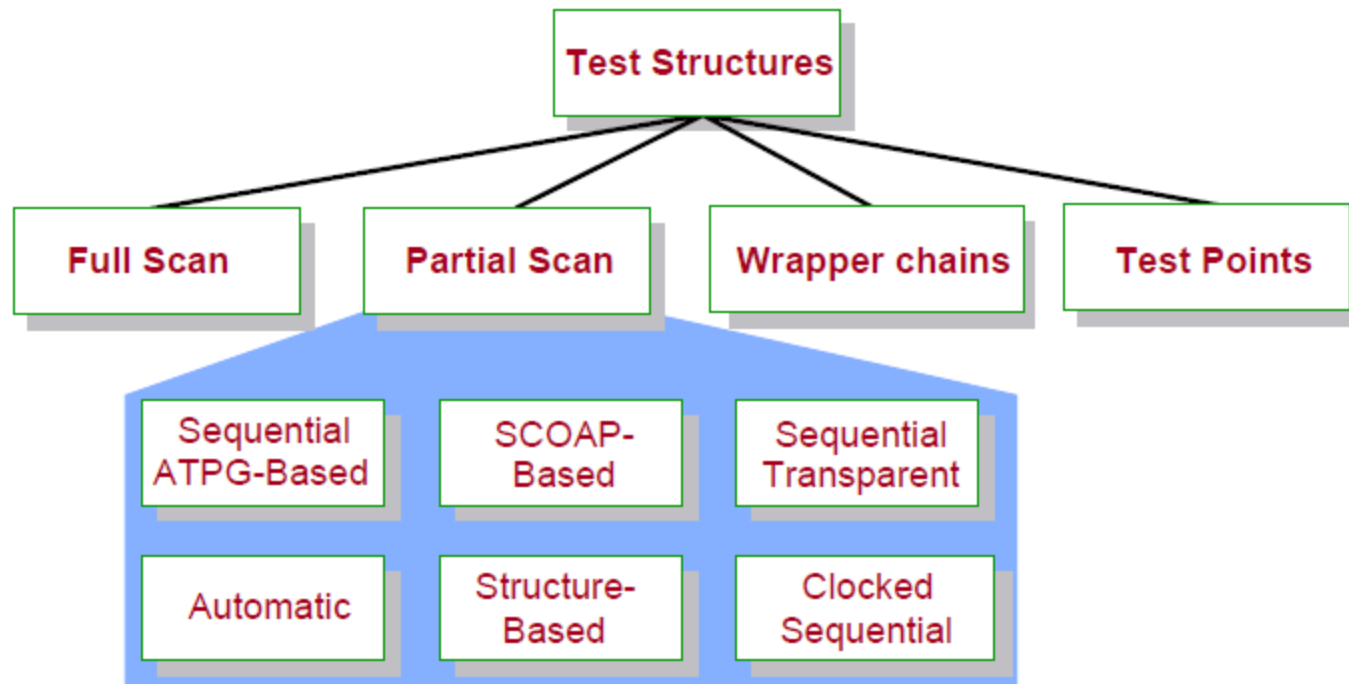
DFT test flow and commands



Basic scan insertion flow



DFTAdvisor supported test structures



Sequential ATPG-based: choose cells with a sequential ATPG algorithm

SCOAP: Sandia Controllability Observability Analysis Program (#'s for each ff)

Automatic: combine scan selection methods using several techniques

Structure-based: look at loop breaking, limiting sequential depth, etc.

Sequential Transparent: cut all sequential loops and evaluate

Clocked Sequential: cut sequential loops and limit sequential depth

Example DFTadvisor session

- Invoke:
 - `dftadvisor count4.v –lib $ADK/technology/adk.atpg`
- Implement scan **with defaults** (full scan, mux-DFF elements):
 - set system mode setup
 - analyze control signals `–auto` (find clocks, resets, etc.)
 - set system mode dft
 - run (run the identification process)
 - insert test logic (modify the netlist)
 - write netlist `count4_scan.v` (Verilog netlist)
 - write atpg setup `count4_scan` (dofile & test procedure for FastScan)

DFT options

- set scan type mux_scan
 - Others: lssd, clocked_scan
 - Find indicated scan flip flop type in the ATPG library
- setup scan identification “type”, where “type” =
 - full_scan (default)
 - sequential atpg –percent 50
 - clock_sequential [-depth integer]
 - etc.

(Show Chao Han examples of multiple scan chains)

Example FastScan ATPG session for a circuit containing scan chains

- Invoke:

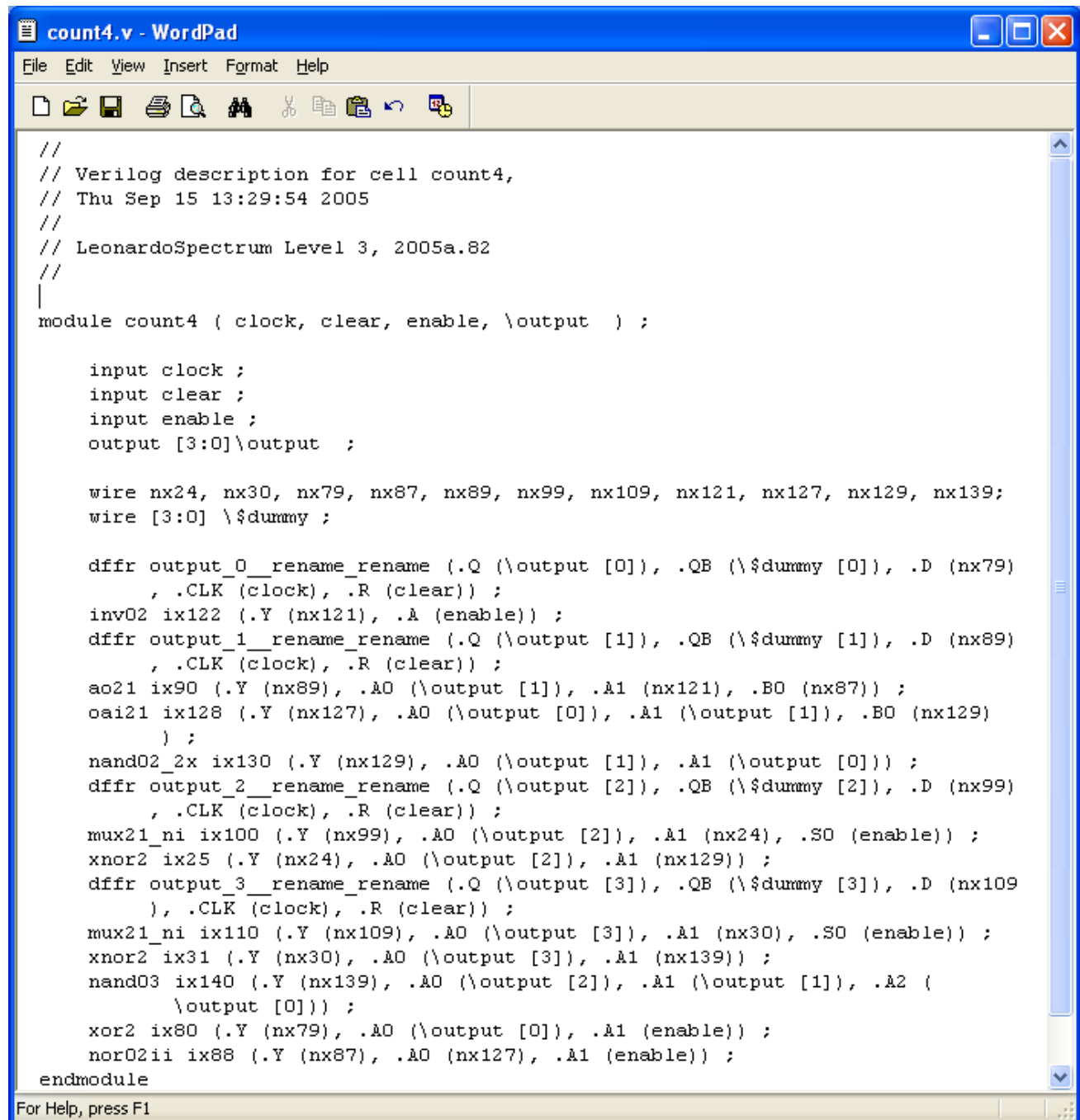
```
fastscan count4_scan.v -lib $ADK/technology/adk.atpg
```

- Generate test pattern file:

- `dofile count4_scan.dofile` (defines scan path & procedure)
- `set system mode atpg`
- `create patterns -auto`
- `save patterns -verilog` (write patterns & verilog test bench)
- `write faults count4_faults.txt` (write fault information to file)
- `write procfile count4.proc` (write test procedure & timing data)

Binary counter (4-bit)

Synthesized by
Leonardo



```
count4.v - WordPad
File Edit View Insert Format Help
[Icons]
//
// Verilog description for cell count4,
// Thu Sep 15 13:29:54 2005
//
// LeonardoSpectrum Level 3, 2005a.82
//
|
module count4 ( clock, clear, enable, \output ) ;

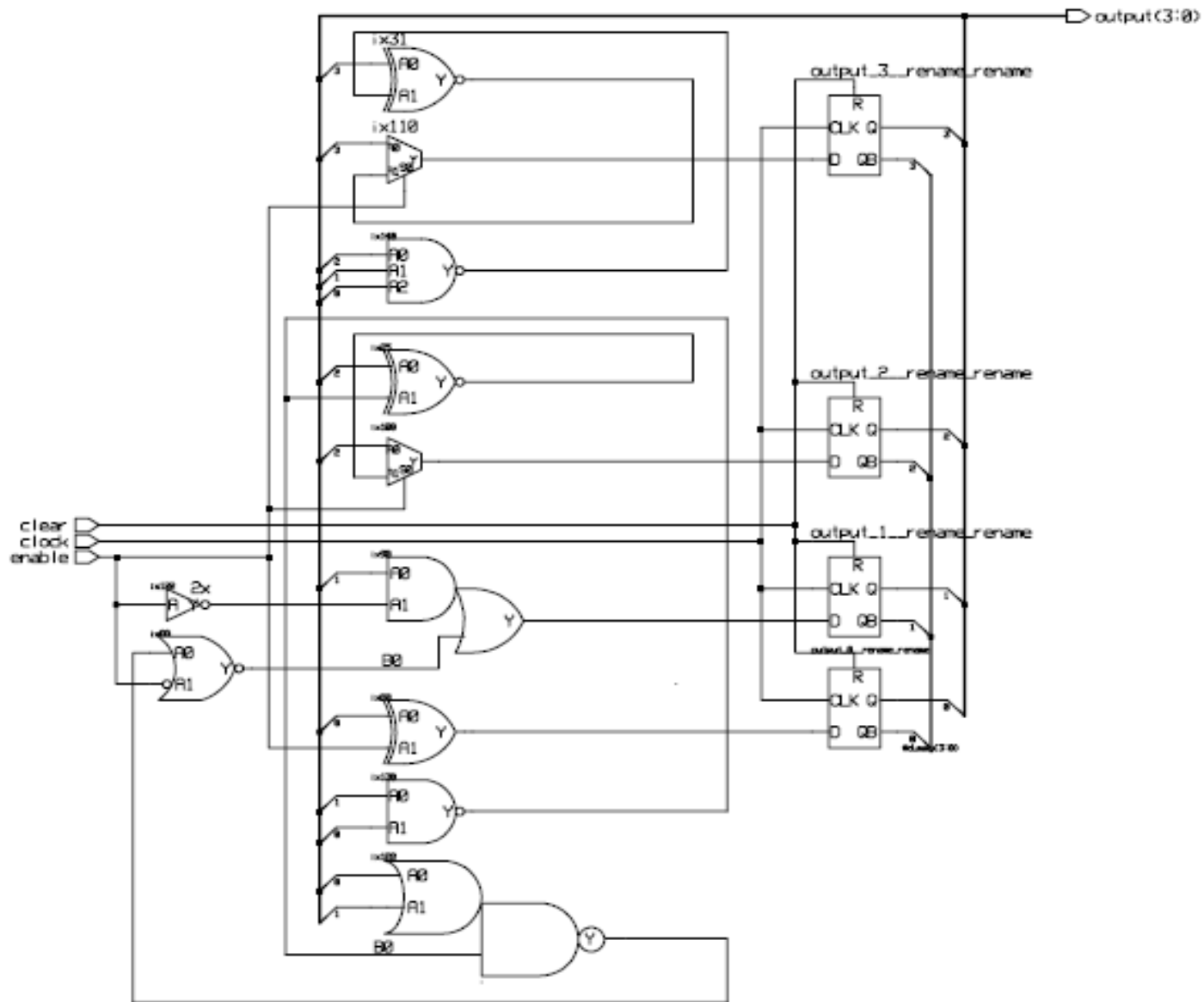
    input clock ;
    input clear ;
    input enable ;
    output [3:0]\output ;

    wire nx24, nx30, nx79, nx87, nx89, nx99, nx109, nx121, nx127, nx129, nx139;
    wire [3:0] \$dummy ;

    dffr output_0_rename_rename (.Q (\output [0]), .QB (\$dummy [0]), .D (nx79)
        , .CLK (clock), .R (clear)) ;
    inv02 ix122 (.Y (nx121), .A (enable)) ;
    dffr output_1_rename_rename (.Q (\output [1]), .QB (\$dummy [1]), .D (nx89)
        , .CLK (clock), .R (clear)) ;
    ao21 ix90 (.Y (nx89), .AO (\output [1]), .A1 (nx121), .BO (nx87)) ;
    oai21 ix128 (.Y (nx127), .AO (\output [0]), .A1 (\output [1]), .BO (nx129)
        ) ;
    nand02_2x ix130 (.Y (nx129), .AO (\output [1]), .A1 (\output [0])) ;
    dffr output_2_rename_rename (.Q (\output [2]), .QB (\$dummy [2]), .D (nx99)
        , .CLK (clock), .R (clear)) ;
    mux21_ni ix100 (.Y (nx99), .AO (\output [2]), .A1 (nx24), .SO (enable)) ;
    xnor2 ix25 (.Y (nx24), .AO (\output [2]), .A1 (nx129)) ;
    dffr output_3_rename_rename (.Q (\output [3]), .QB (\$dummy [3]), .D (nx109)
        , .CLK (clock), .R (clear)) ;
    mux21_ni ix110 (.Y (nx109), .AO (\output [3]), .A1 (nx30), .SO (enable)) ;
    xnor2 ix31 (.Y (nx30), .AO (\output [3]), .A1 (nx139)) ;
    nand03 ix140 (.Y (nx139), .AO (\output [2]), .A1 (\output [1]), .A2 (
        \output [0])) ;
    xor2 ix80 (.Y (nx79), .AO (\output [0]), .A1 (enable)) ;
    nor02ii ix88 (.Y (nx87), .AO (nx127), .A1 (enable)) ;

endmodule
For Help, press F1
```

count4 - without scan design



Binary counter (4-bit)

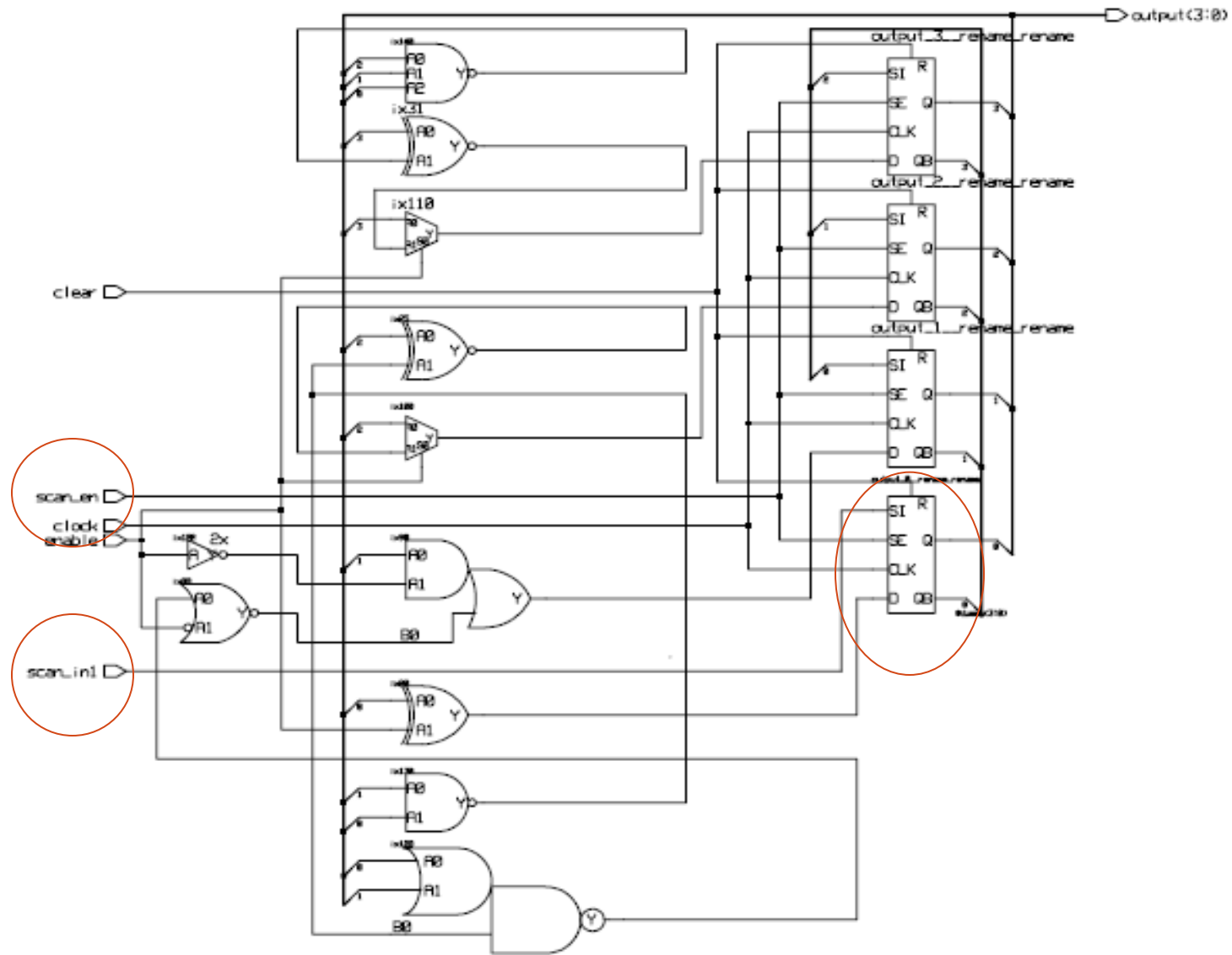
Synthesized by Leonardo

DFTAdvisor Changed to Scan Design

```
count4_scan.v - WordPad
File Edit View Insert Format Help
[Icons]
/*
 *   DESC: Generated by DFTAdvisor at Wed Nov 30 17:01:15 2005
 */
module count4 ( clock , clear , enable , \output , scan_in1 , scan_en );
input  clock , clear , enable , scan_in1 , scan_en ;
output [3:0] \output ;
  wire nx139 , nx30 , nx109 , nx24 , nx99 , nx127 , nx129 , nx87 , nx89 , nx121 , nx79 ;
  wire [3:0] \$dummy ;

  sffr_ni output_0_rename_rename (.D ( nx79 ) , .SI ( scan_in1 ) , .SE
    ( scan_en ) , .CLK ( clock ) , .R ( clear ) , .Q ( \output [0] ) ,
    .QB ( \$dummy [0] ));
  inv02 ix122 (.A ( enable ) , .Y ( nx121 ));
  sffr_ni output_1_rename_rename (.D ( nx89 ) , .SI ( \$dummy [0] ) , .SE
    ( scan_en ) , .CLK ( clock ) , .R ( clear ) , .Q ( \output [1] ) ,
    .QB ( \$dummy [1] ));
  ao21 ix90 (.AO ( \output [1] ) , .A1 ( nx121 ) , .B0 ( nx87 ) , .Y ( nx89 ));
  oai21 ix128 (.AO ( \output [0] ) , .A1 ( \output [1] ) , .B0 ( nx129 ) ,
    .Y ( nx127 ));
  nand02 2x ix130 (.AO ( \output [1] ) , .A1 ( \output [0] ) , .Y ( nx129 ));
  sffr_ni output_2_rename_rename (.D ( nx99 ) , .SI ( \$dummy [1] ) , .SE
    ( scan_en ) , .CLK ( clock ) , .R ( clear ) , .Q ( \output [2] ) ,
    .QB ( \$dummy [2] ));
  mux21_ni ix100 (.AO ( \output [2] ) , .A1 ( nx24 ) , .S0 ( enable ) , .Y
    ( nx99 ));
  xnor2 ix25 (.AO ( \output [2] ) , .A1 ( nx129 ) , .Y ( nx24 ));
  sffr_ni output_3_rename_rename (.D ( nx109 ) , .SI ( \$dummy [2] ) , .SE
    ( scan_en ) , .CLK ( clock ) , .R ( clear ) , .Q ( \output [3] ) ,
    .QB ( \$dummy [3] ));
  mux21_ni ix110 (.AO ( \output [3] ) , .A1 ( nx30 ) , .S0 ( enable ) , .Y
    ( nx109 ));
  xnor2 ix31 (.AO ( \output [3] ) , .A1 ( nx139 ) , .Y ( nx30 ));
  nand03 ix140 (.AO ( \output [2] ) , .A1 ( \output [1] ) , .A2
    ( \output [0] ) , .Y ( nx139 ));
  xor2 ix80 (.AO ( \output [0] ) , .A1 ( enable ) , .Y ( nx79 ));
  nor02ii ix88 (.AO ( nx127 ) , .A1 ( enable ) , .Y ( nx87 ));
endmodule
For Help, press F1
```

count4 – scan inserted by DFTadvisor

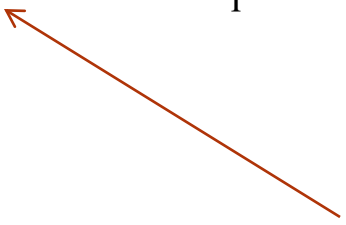


Test file: scan chain definition and load/unload procedures

```
scan_group "grp1" =  
  scan_chain "chain1" =  
    scan_in = "/scan_in1";  
    scan_out = "/output[3]";  
    length = 4;  
end;
```

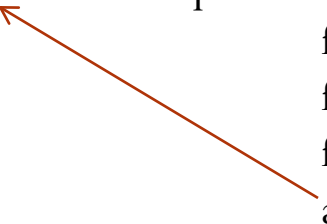
```
procedure shift "grp1_load_shift" =  
  force_sci "chain1" 0;  
  force "/clock" 1 20;  
  force "/clock" 0 30;  
  period 40;  
end;
```

```
procedure load "grp1_load" =  
  force "/clear" 0 0;  
  force "/clock" 0 0;  
  force "/scan_en" 1 0;  
  apply "grp1_load_shift" 4 40;  
end;
```



```
procedure shift "grp1_unload_shift" =  
  measure_sco "chain1" 10;  
  force "/clock" 1 20;  
  force "/clock" 0 30;  
  period 40;  
end;
```

```
procedure unload "grp1_unload" =  
  force "/clear" 0 0;  
  force "/clock" 0 0;  
  force "/scan_en" 1 0;  
  apply "grp1_unload_shift" 4 40;  
end;
```



end;

Test file: scan chain test

// send a pattern through the scan chain

CHAIN_TEST =

pattern = 0;

(pattern #)

apply "grp1_load" 0 =

(use grp1_load proc.)

chain "chain1" = "0011";

(pattern to scan in)

end;

apply "grp1_unload" 1 =

(use grp1_unload proc.)

chain "chain1" = "1100";

(expected pattern scanned out)

end;

end;

Test file: sample test pattern

// one of 14 patterns for the counter circuit

```
pattern = 0;                (pattern #)
  apply "grp1_load" 0 =      (load scan chain)
    chain "chain1" = "1000"; (scan-in pattern)
  end;
  force "PI" "00110" 1;      (apply PI pattern)
  measure "PO" "0010" 2;     (expected POs)
  pulse "/clock" 3;          (normal op. cycle)
  apply "grp1_unload" 4 =    (read scan chain)
    chain "chain1" = "0110"; (expected pattern)
  end;
```

DFTAdvisor example (Chao Han)

```
//dofile for dftadvisor
```

```
analyze control signals -auto_fix
```

```
set scan type mux_scan
```

```
set system mode dft
```

```
setup scan identification full_scan
```

```
run
```

```
//specify # scan chains to create
```

```
insert test logic -scan on -number 3
```

```
//alternative – specify maximum scan chain length
```

```
//insert test logic -scan on -max_length 30
```

```
write netlist s1423_scan.v -verilog -replace
```

```
//write dofile and procedure file for fastscan
```

```
write atpg setup s1423_scan -procfile -replace
```

```
exit
```