Tessent: Scan and ATPG

Module 3

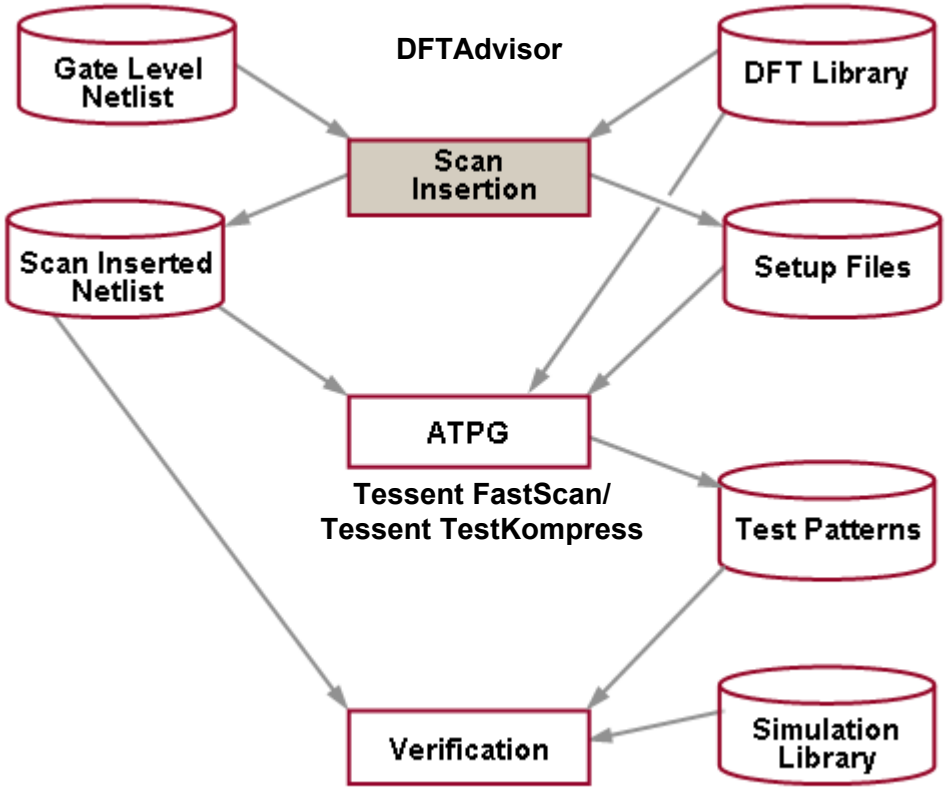
Scan Insertion and Configuration

Objectives

Upon completion of this module, you will be able to:

- **Use DFTAdvisor to insert full scan.**
- **Write a scan-inserted netlist file.**
- **Write ATPG setup files.**
- **Insert test logic.**
- **Create, configure, and balance scan chains.**
- **Edit a scan chain order file and change the order of the scan cells.**

DFTAdvisor Tool Flow: An Overview



Invoking DFTAdvisor

- **The DFTAdvisor executable is installed at:**

  <install_dir>/bin/dftadvisor

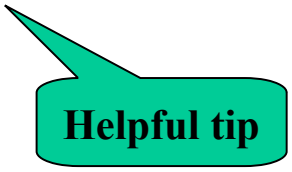
- **Invocation requirements:**
  - **A non-scan inserted design netlist in Verilog**
  - **DFT library**

Invoking DFTAdvisor (Cont.)

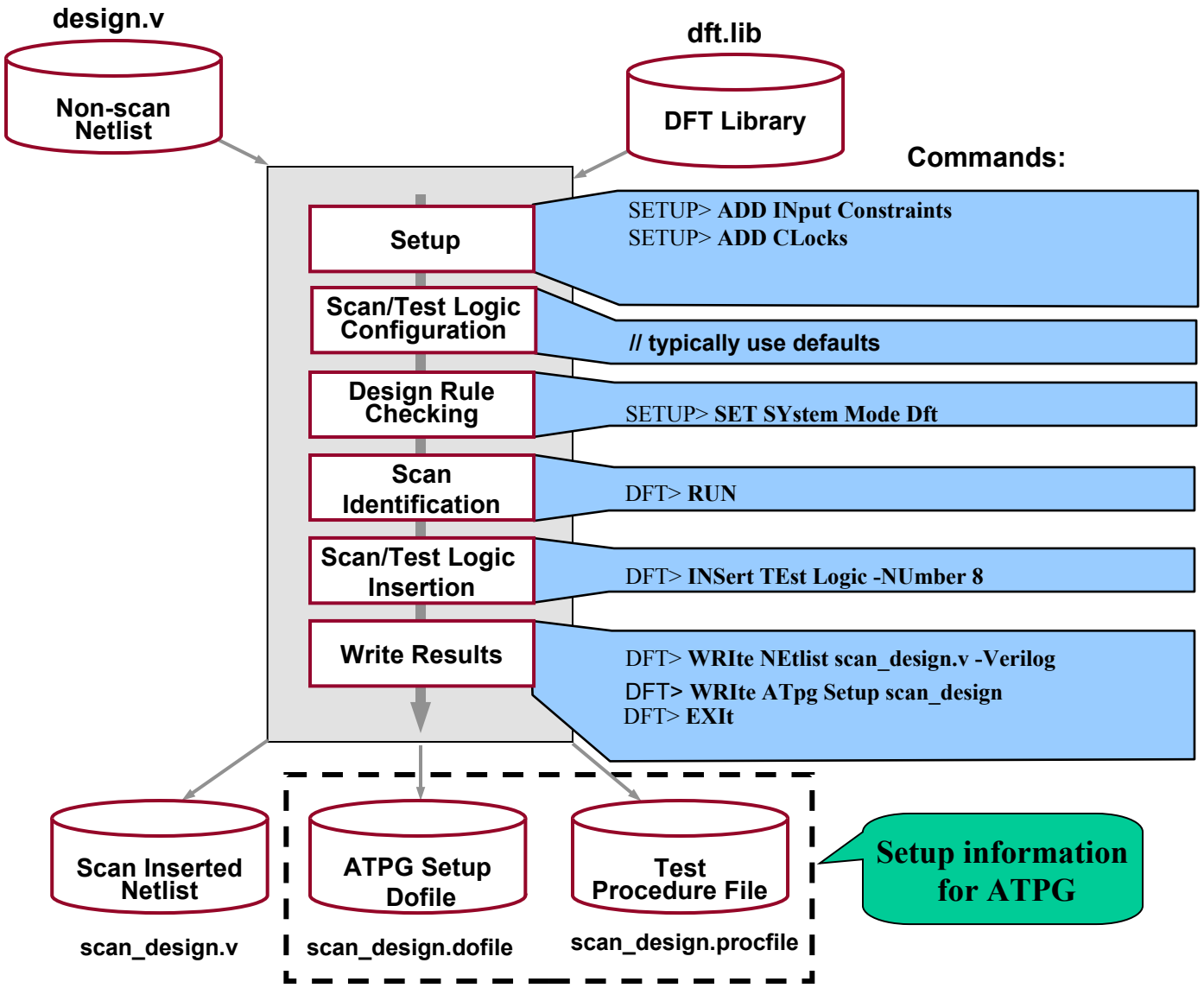
Invocation:

```
shell> dftadvisor design.v -verilog \
-lib dft.lib -log transcript.log -replace
```

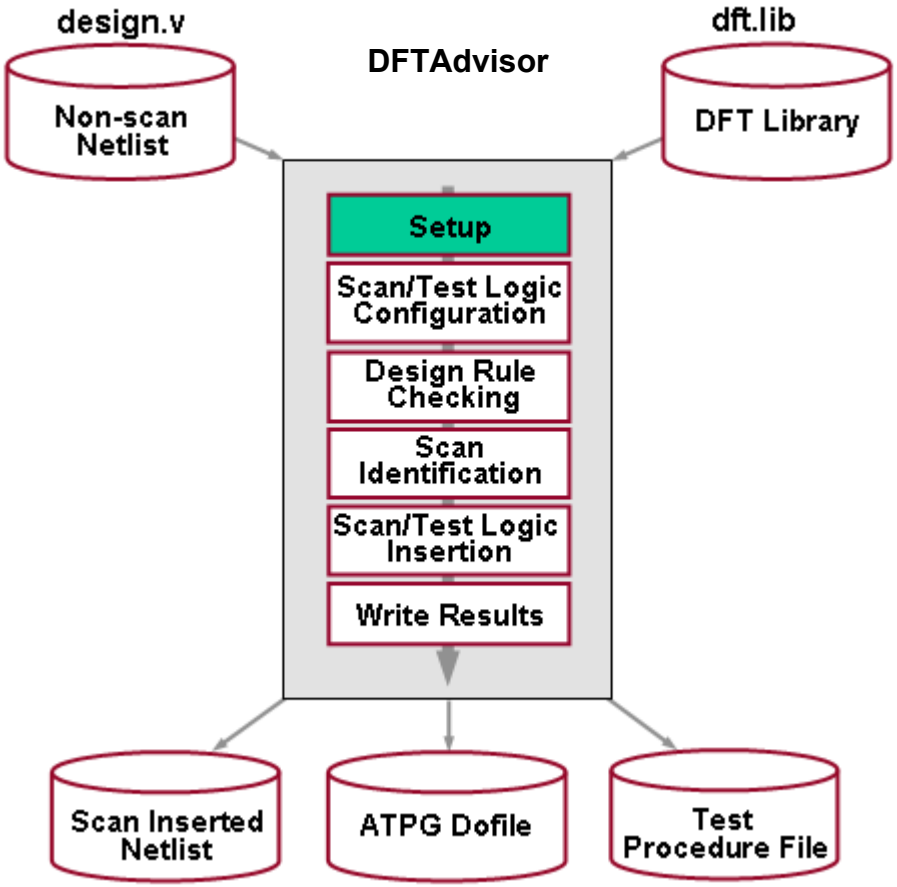
Use the `-log <filename>` option to write detailed session information to a file



DFTAdvisor Tool Flow With Commands



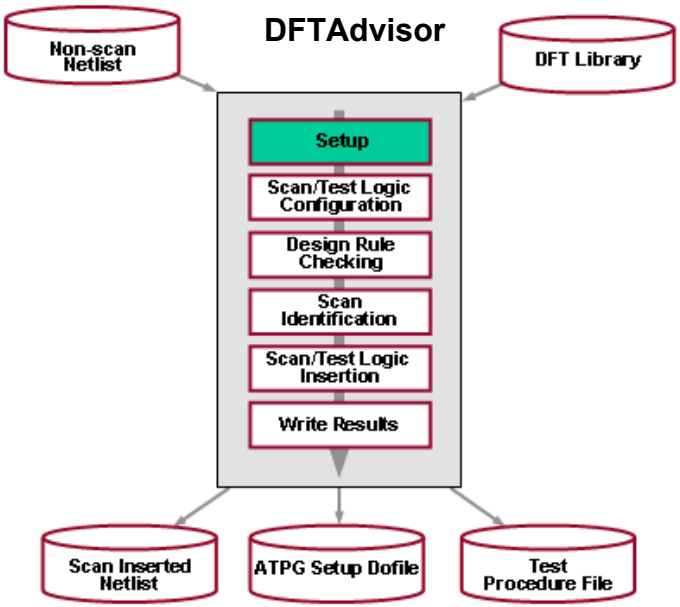
Scan Methodology: Scan Cells



`Mux scan` as scan cell type.

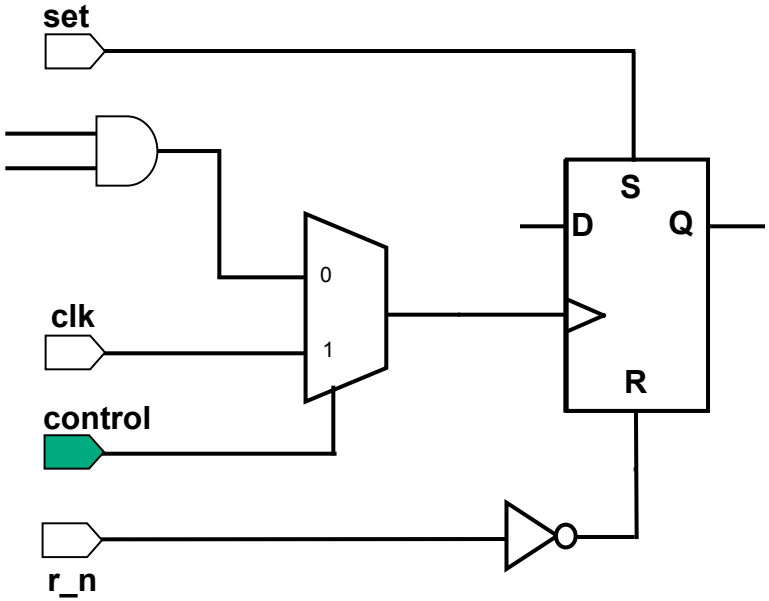
```
SET SCan Type Mux scan // default
```

SETUP

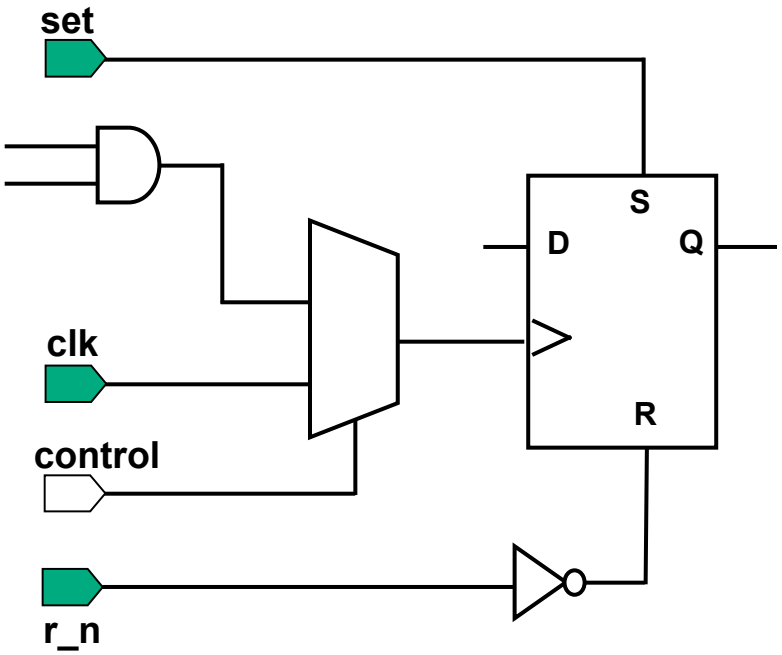


- **FIRST**: Define input constraints.
  - Define during setup mode.
  - Pin constraints are signals that are held at a constant value during test.

```
SETUP> ADD INput Constraints control -C1
                //constrain to constant 1
```



SETUP (Cont.)



- **SECOND: Define clocks.**
  - Clocks are primary input signals that asynchronously change the state of sequential logic elements.
    - clocks
    - sets
    - resets
    - RAM read/write clocks

SETUP> `ADD Clocks 0 clk set`

SETUP> `ADD Clocks 1 r_n`

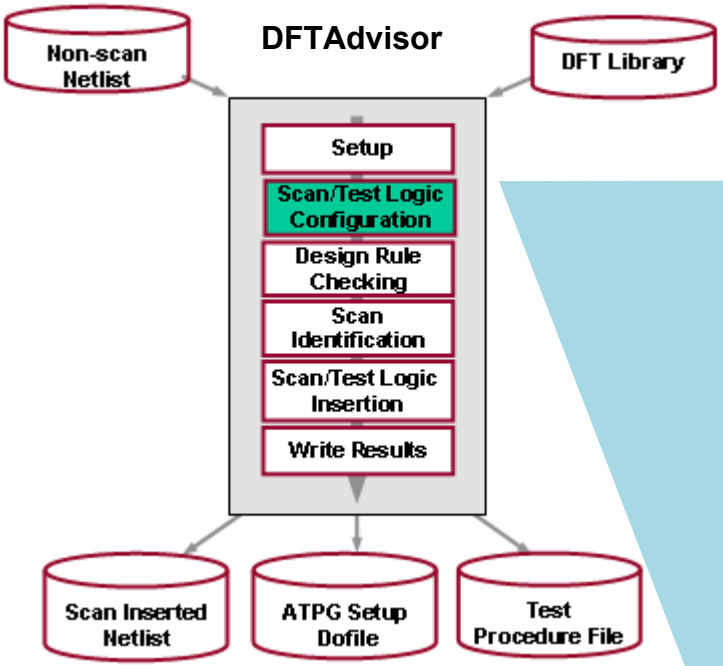
Off state

Primary input pin

SETUP (Cont.)

- The `ANAlyze COntrol Signals` command identifies and optionally defines primary inputs of control signals.
  - Clocks, set, reset, write-control, read-control, etc.
  - Performs the flattening process automatically.
  - Does not support gated clocks.
  - Default is `-report only`.
  - `-verbose` enables the tool to issue messages indicating why certain control signals are not reported as controllable.
  - `-Auto_fix` specifies the tool to define all identified primary inputs.
    - Limited capability.
    - Does not always correctly identify off states.
    - Hard to determine off states of every clock in the clock cone.
      - Important because you do not want scan cells to capture data during the clock's off state.
      - Use `ADD Clocks` command instead.

Scan/Test Logic Configuration



Perform Scan/Test Logic Configuration

- Default settings exist
- User can specify
  - Scan methodology
  - Test pin names
  - Areas not to scan
  - Test logic options
  - Existing scan
  - Circuit clocks

Set Test Logic Configuration

- **User-defined setting options in SETUP mode:**
  - **Scan Methodology**

    SET SCan Type {Mux_scan | Lssd | Clocked_scan}

    (Setup mode)

  - **Test Pin namings:**

    SETup SCan Insertion [-Ten *pathname*][-TClk *pathname*]
    [-SClk *pathname*][-SMclk *pathname*][-SSclk *pathname*]
    [-SET *pathname*][-RESet *pathname*]
    [-Write *pathname*][-REAd *pathname*][**-Muxed** | -Disabled |-Gated]
    [-Active {High | Low}]

    (Setup mode)

    SETup SCan Pins {**Input** | **Output**} [**-INDexed** |-Bused][-Prefix *base_name*][-INItial *index#*][-Modifier *incr_index#*]
    [-Suffix *suffix_name*]

    (All modes)

    ADD SCan Pins *chain_name scan_input_pin scan_output_pin* [-Clock *pin_name*][-CUt][-Registered]
    [-Top *primary_input_pin primary_out_pin*]

    (All modes)

Set Test Logic Configuration (Cont.)

- **Control bidirectional pins**

  SET BIdi Gating [**OFF**| **ON** |**Scan**][-Control {Sen | Ten}] [-Direction {Input | Output}][-Top {ALl | *primary_bidi_pin…*}] [-Force_gating]

  (Setup mode)

- **Control tri-state devices**

  SET Tristate Gating {**OFF** | **ON** | **Busdrivers** | **Scan** | *primary_input_or_output…* | **Decoded**} [-Control {Sen | Ten} [-Force_gating]

  (Setup mode)

- **Test logic options**

  SET TEst Logic{-Set {ON | ***OFf***} | -Reset {ON | **OFf**}| -Clock {ON | **OFf**} | -Ram {ON | Off}}…

  (Setup mode)

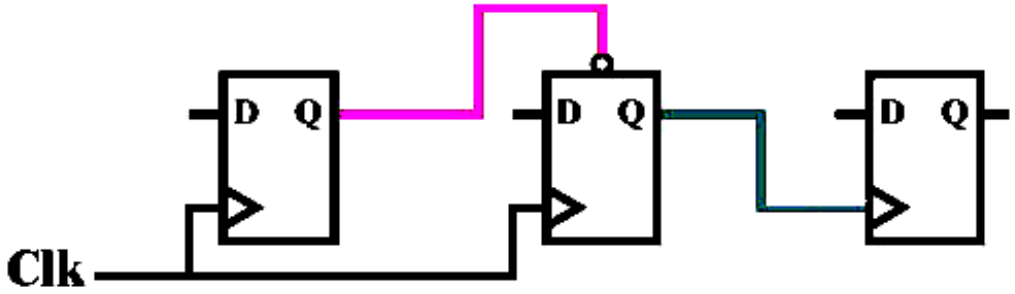
- **Areas not to scan**

  ADD NONscan Instances *pathname…*|*instance_expression* {**-INStance** | -Control_signal | -Module}

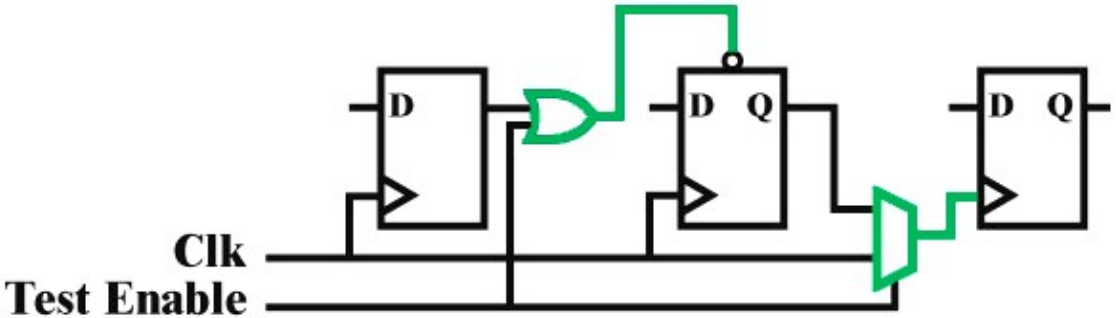
  (All modes except DFT mode only for –Control option)

Adding Test Logic

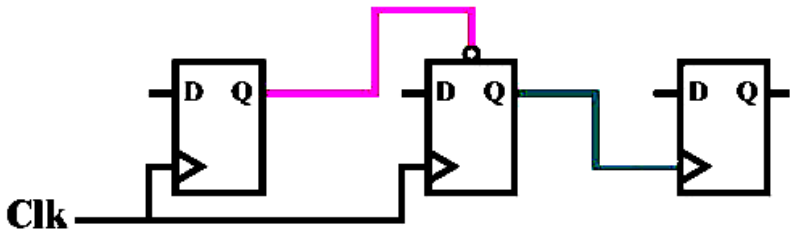
- Some designs contain uncontrollable clock circuitry.



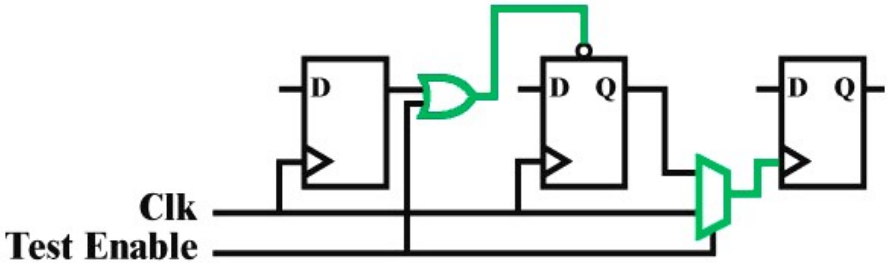
- Test logic is added to make the circuit scannable.



Adding Test Logic (Cont.)



Non-Scannable



Scannable

- **Required modifications:**
  - **Specify which types of control lines are controllable.**

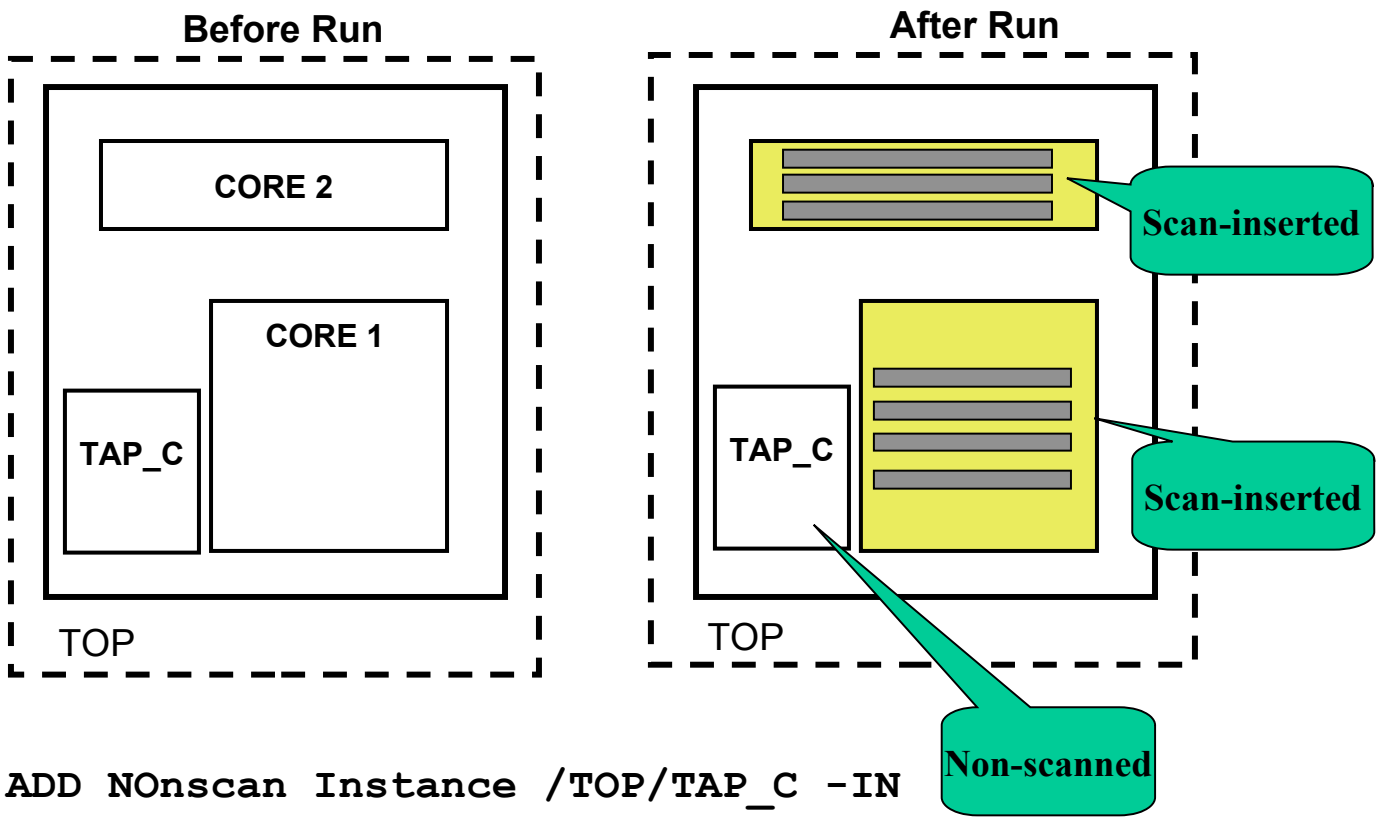
    SETUP> **SET Test Logic -Set ON -Clock on**
  - **Disable set/reset.**

    SETUP> **ADD CEll Models -Type OR <cell name>**
  - **Activate test mode with "test enable" signal.**

    SETUP> **ADD CEll Models -Type MUX selector data0 data <cell name>**
    - **Or: gates for test logic can be defined by "`cell-type`" in the library model.**

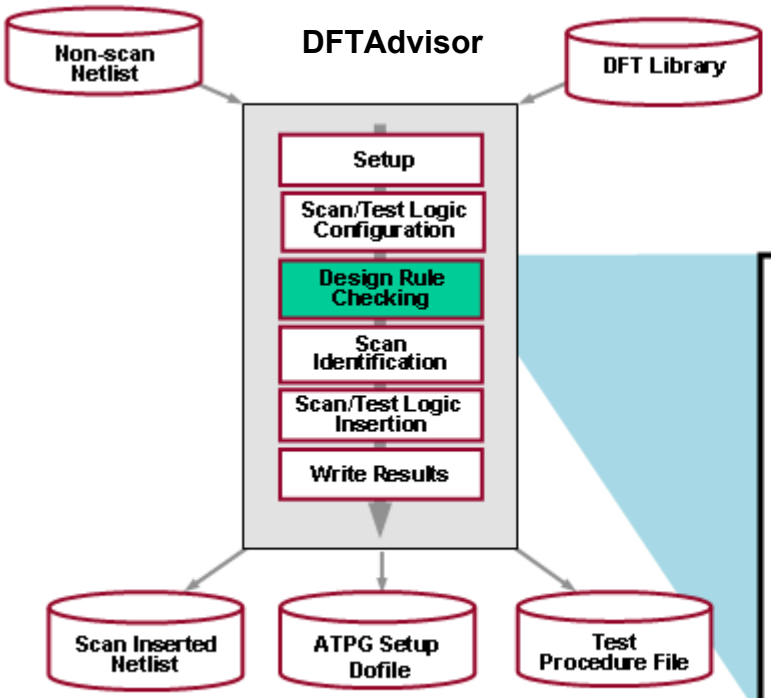
Set Test Logic Configuration (Defining Non-Scan Areas)

- **Excluding the TAP controller from scan**




```
SETUP> ADD NOnscan Instance /TOP/TAP_C -IN
```

Design Rule Checking (DRC)



Perform DRC

- DFT tools perform DRC to ensure the following:
  - Correct scan operation
  - Proper handling of special design situations
  - Correct syntax and operation of test procedure file

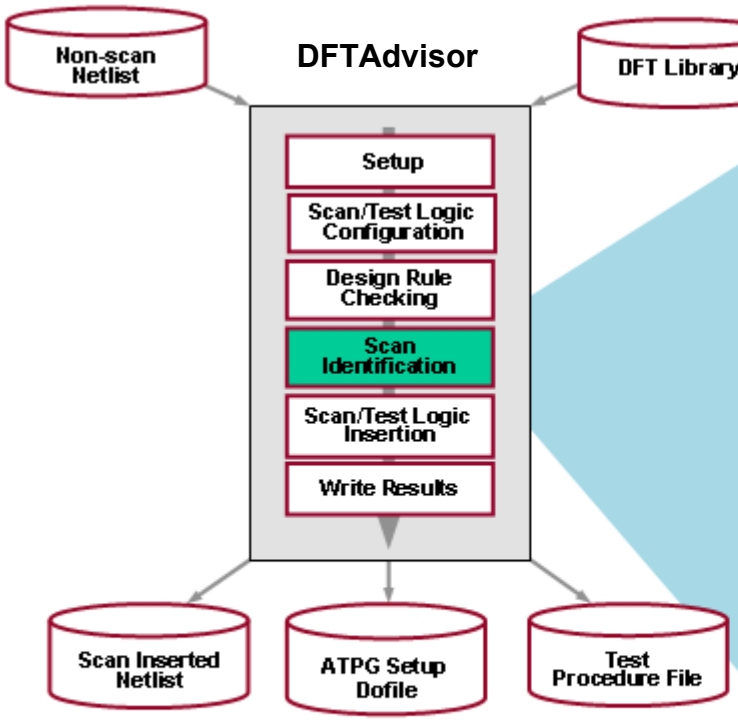
Scan Specific DRCs

- General rules (G rules)
  - Checks for gross scan definition errors
- Trace rules (T rules)
  - Uses test procedure files to trace scan chains
    - Bus contention or data shifted through scan chains
- Scannability rules (S rules)
  - Ensures that DFTAdvisor can safely convert a sequential element into a scan element
  - Checks scannability during DRC
    - S1 rule checking:
      - Ensures that all clocks off- sequential elements are stable and inactive
    - S2 rule checking
      - Ensure that defined clocks capture data when all other clocks are off

```
DFT> REPort DRc Rules
```

Displays all failing DRC violations

Scan Identification



Perform Scan Identification

- DFTAdvisor does the following:
  - Identifies which instances to convert to scan
  - Determines netlist changes
  - Does not alter netlist

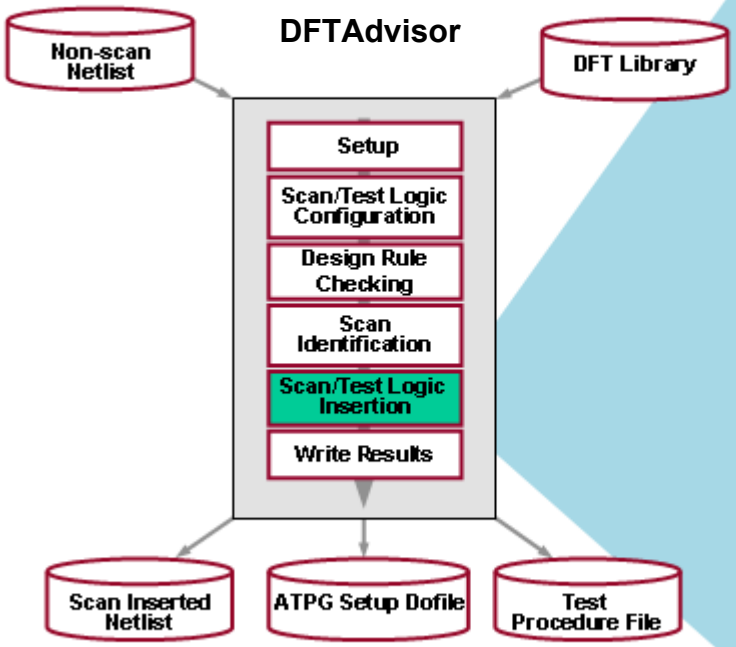
Display results:

- Sequential instances
- Non-scan and scan instances
- Instances scannable with test logic

Commands:

```
DFT> RUN
DFT> REPort STatistics
```

Scan/Test Logic Insertion



Perform Scan/Test Logic Insertion.

- User-identifiable `INSert TEst Logic` arguments determines:
  - Instance order in the scan chain (via a file).
  - Maximum length of scan chain.
  - Number of scan chains.
  - Clock domain handling within scan chains.

DFTAdvisor creates a scan-inserted netlist.

- `INSert TEst Logic` command changes the netlist.

Command:

```
DFT> INSert TEst Logic -NUmber 4
```

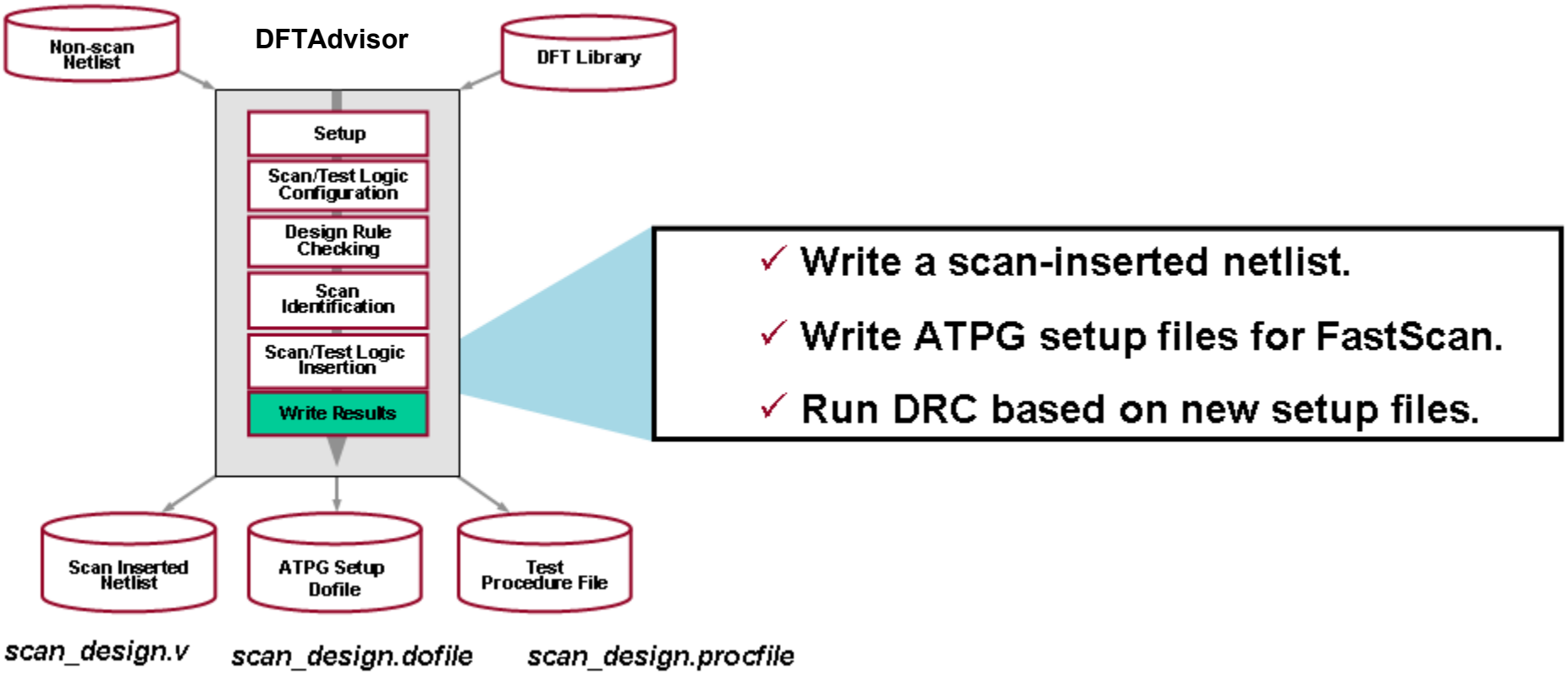
Scan/Test Logic Insertion (Cont.)

- **To display results:**

```
DFT> REPort SCan Chains
DFT> REPort TEst Logic
```

```
DFT> INSert TEst Logic -NUmber 2
//  WARNING: Flattened model has been freed
DFT> REPort SCan Chains
chain = chain1   group = dummy  input = /scan_in1  output = /scan_out1  length = 4
  scan_enable = /scan_en  clock = /clk
  reset = /rst
chain = chain2   group = dummy  input = /scan_in2  output = /scan_out2  length = 3
  scan_enable = /scan_en  clock = /clk
  reset = /rst
DFT> REPort TEst Logic
New pins added in top module: example_ckt
/scan_in1
/scan_out1
/scan_in2
/scan_out2
/scan_en
Number of new pins inserted = 5
```

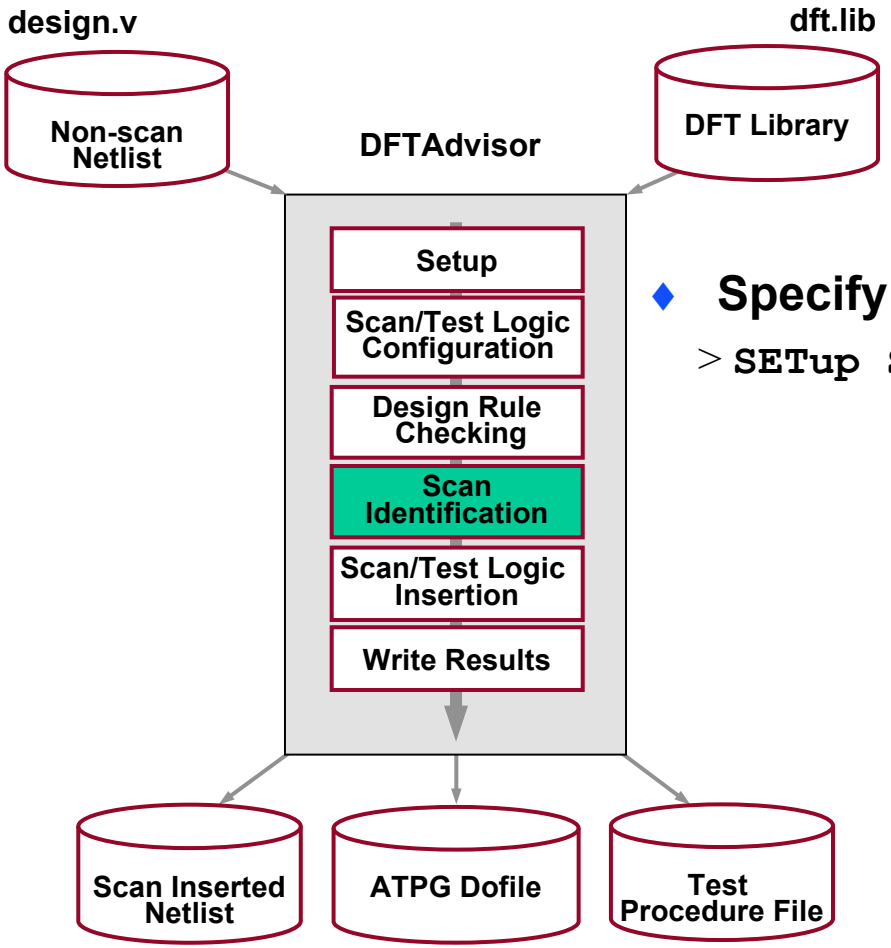
Write Results



Commands:

```
DFT> WRIte NEtlist scan_design.v
DFT> WRIte ATpg Setup scan_design
```

Scan Methodology: Full Scan

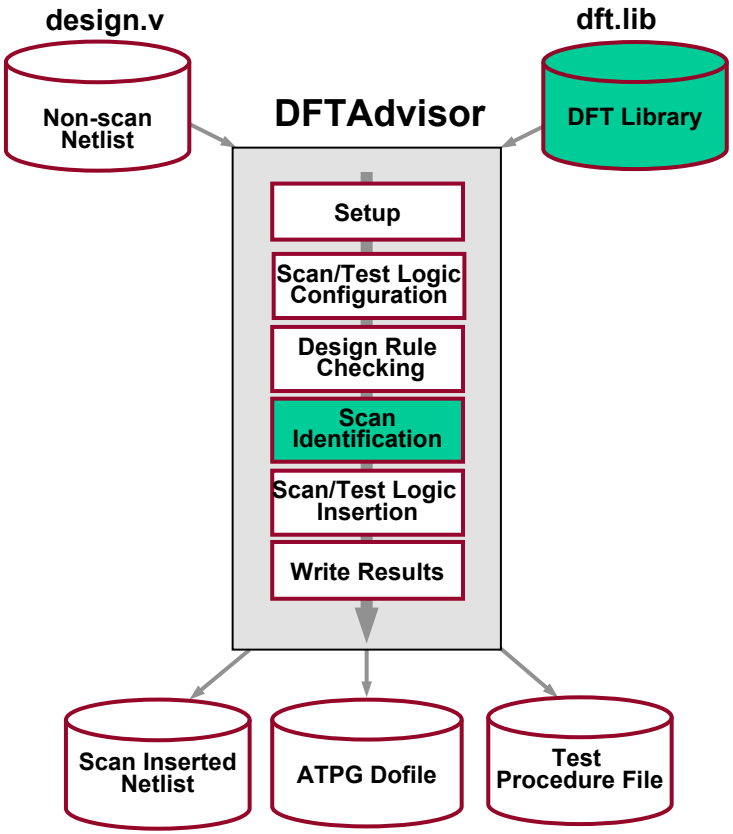


- **Specify `Full scan` for scan identification.**
  - `SETup SCan Identification Full scan` //default

Scan Methodology: Full Scan versus Partial Scan

- **Full scan (default)**
  - Most commonly used
  - Converts all memory elements to scan
    - Sometimes a few non-scan restrictions
  - Provides high test coverage/high quality
  - Uses a combinational ATPG tool
    - Requires minimal test generation effort
- **Partial scan**
  - Not commonly used
  - Converts a portion of memory elements to scan
  - Requires less silicon area
    - Increases CPU time to obtain a certain test coverage

Scan Methodology: DFT Library and Scan Identification



- **The DFT library:**
  - **A model description defines a single cell in the technology library.**
  - **A cell description (model or macro) describes a component in a specified design.**
  - **A library is simply a set of models.**

Non scan cell model

```
// ================================
// Model: DFF
// ================================
```

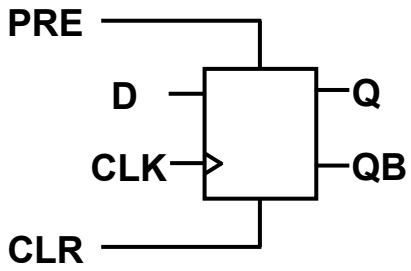
```
model DFF (PRE, CLR,  CLK,  D,  Q, QB)  (
input (PRE,  CLR,  D) ( )
input (CLK)  (clock = rise_edge;)
output(Q,  QB)  (primitive = _dff (PRE, CLR, CLK, D

Q, QB) ;

)
```

Scan Methodology: DFT library and Scan Identification (Cont.)

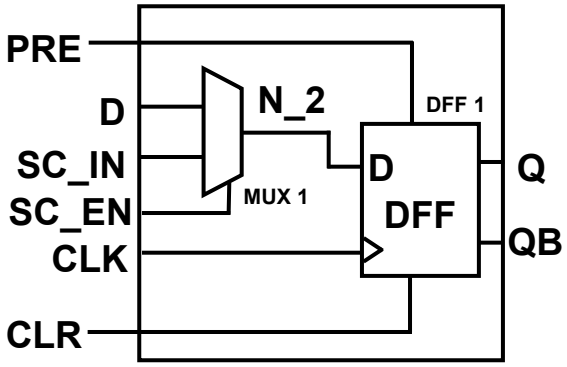
- **If a library model is a scan cell, the model description contains a scan definition attribute:**
  - **Provides information for mapping non-scan sequential models (dffs and latches) to their associated scan cell models.**

```
//=========================================
                Model: DFF
//=========================================

     model DFF ( PRE,CLR,CLK,D, Q, QB) (
           input ( PRE, CLR, D) ()
           input(CLK) (clock = rise_edge;)
           output(Q QB) (primitive = _dff(PRE,CLR,CLK,D,QB);
                        )
//=============================================
            Model: MUX_SCAN _CELL
//=============================================
     model MUX_SCAN_CELL (PRE, CLR, SC_IN, SC_EN, CLK, D, Q, QB) (
                scan_definition (
                type = mux_scan
                scan_in = SC_IN;
                scan_enable = SC_EN;
                scan_out = Q, QB;
                non_scan_model = DFF ( PRE, CLR, CLK, D, Q, QB);
                         )
     input (PRE, CLR, SC_IN, SC_EN, CLK ()
     intern(N_2) (primitive = _mux mux1 (D, SC_IN, SC_EN,N_2);)
     output(Q, QB) (primitive = _dff dff1(PRE, CLR, CLK, N_2, Q, QB);)
                         )
//=========================================
```

Original Flip-Flop

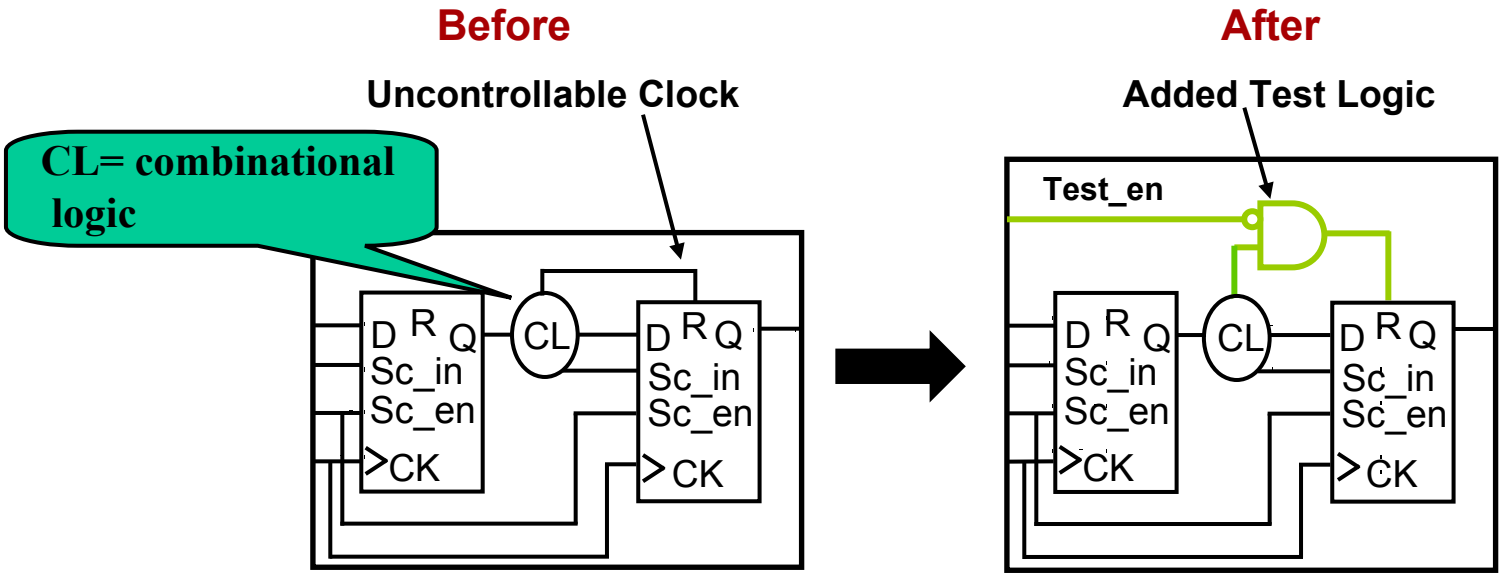


Replaced with: Mux_Scan_Cell



Test Logic

- **Why do we add test logic?**
  - **Some designs contain uncontrollable clock circuitry.**
  - **Sequential devices must be controllable to be converted to scan.**
  - **RAM and three-state logic must be controllable to be testable.**

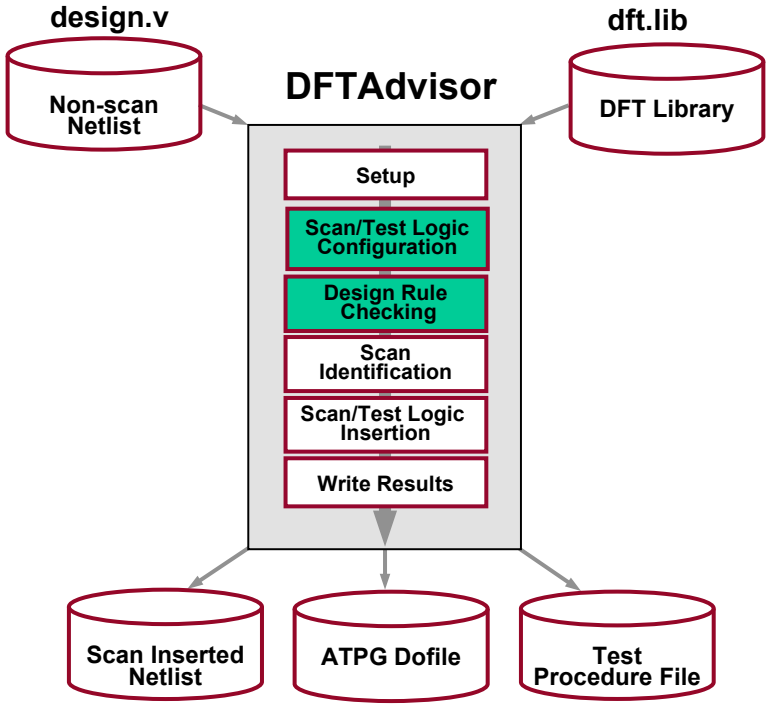


Test Logic (Cont.)

- To add test logic circuitry, DFTAdvisor uses a number of combinational gates from the ATPG library. For example:
  - AND MUX Latch
  - OR INV
- Can also use the `ADD Cell Models` command.
- A *cell_type* attribute defines valid test logic.

```
// ============================
// Model: AN2
// ============================

model AN2   (A, B, Z) (
  cell_type = AND;
  input (A,  B) ( )

  output  (Z)  (function = A * B;)

)
```

Test Logic (Cont.)



- **Define library models used for test logic:**

SETUP> **ADD CEll Models dftlib_model**

  - **Or automatically defined by DFT library if the model has a *cell_type* attribute.**

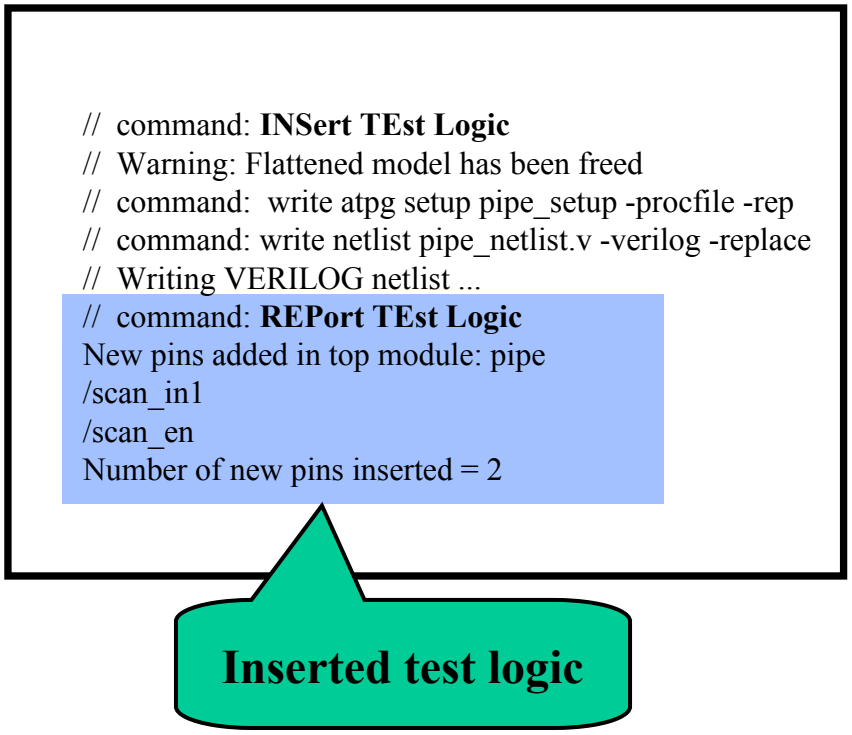
- **Generate scannability check results for non-scan instances:**

DFT> **REPort DFT Check**

Helpful Tip:
Lists identified pins that require test logic.

```
SETUP> ANAlyze COntrol Signals
SETUP> ADD Clocks Clk 0
SETUP> SET TEst Logic -set on
SETUP> SET SYstem Mode dft
DFT> REPort DFT Check
```

Test Logic: Defining Library Models



- Define library models when inserting test logic for the following situations:
  - Set/reset clock access.
  - Lockup latch between clock domains.
  - RAM control.
  - Three-state bus control.
  - Control points.
  - Observe points.

Typically added in 2nd pass

- Display added test logic during scan insertion:

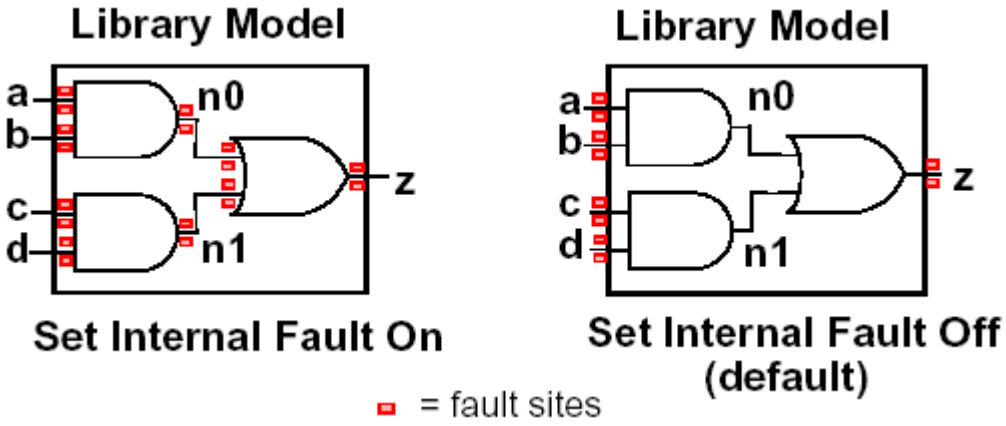
```
DFT> REPort TEst Logic
```

Pins

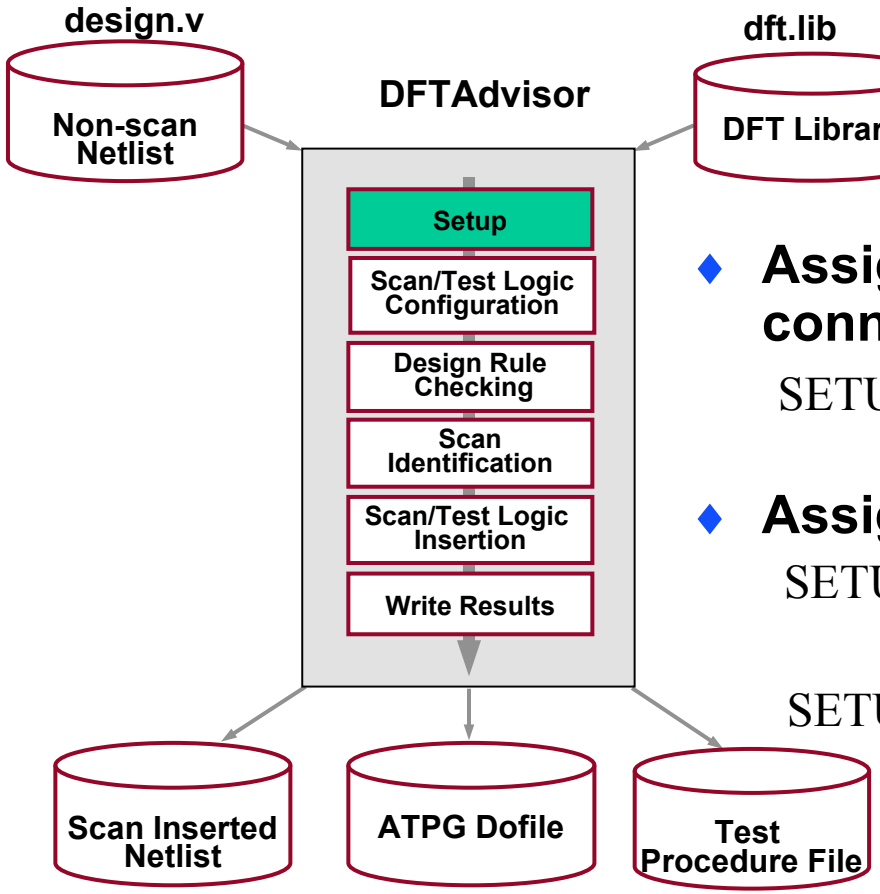
- **Fault sites include only:**
  - **PI, PO.**
  - **Library model pins.**
  - **DFT primitive pins (internal).**
  - **Pins are identified by unique pin names:**
    - **/I116/q**
- **Three types of pins:**
  - **Inputs (top-level, primary input).**
  - **Output (top-level, primary output).**
  - **Bidi.**

Fault Locations

- **By default, faults reside at the inputs and outputs of library models.**
- **Faults can reside at the inputs and outputs of gates within library models instead if you turn internal faulting on.**



Defining Pins



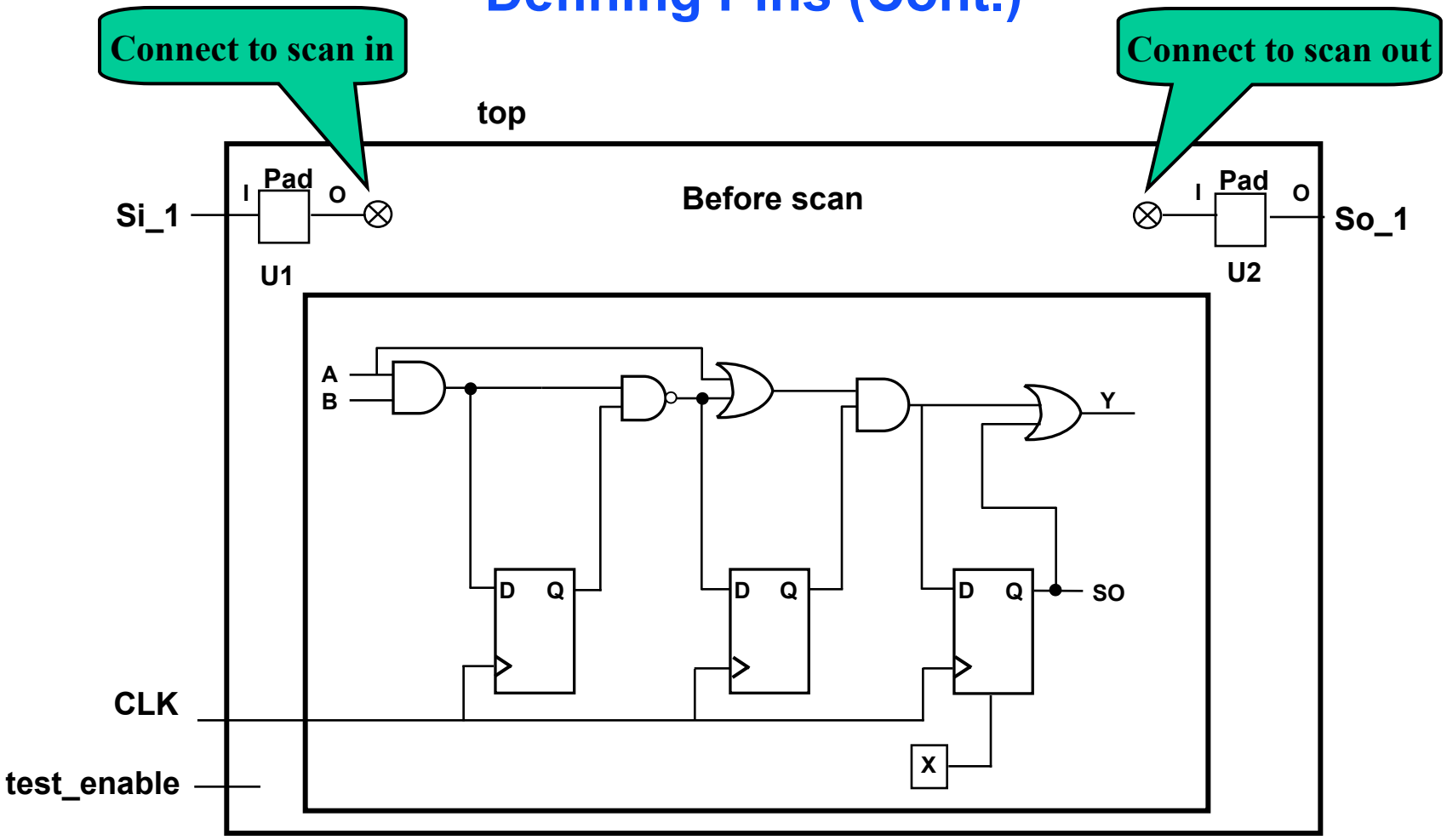
- **Assign scan input and scan output pin connections to existing logic:**

```
SETUP> ADD SCan Pins Chain1 /U1/O \
         /U2/I
```

- **Assign scan control pin connections:**

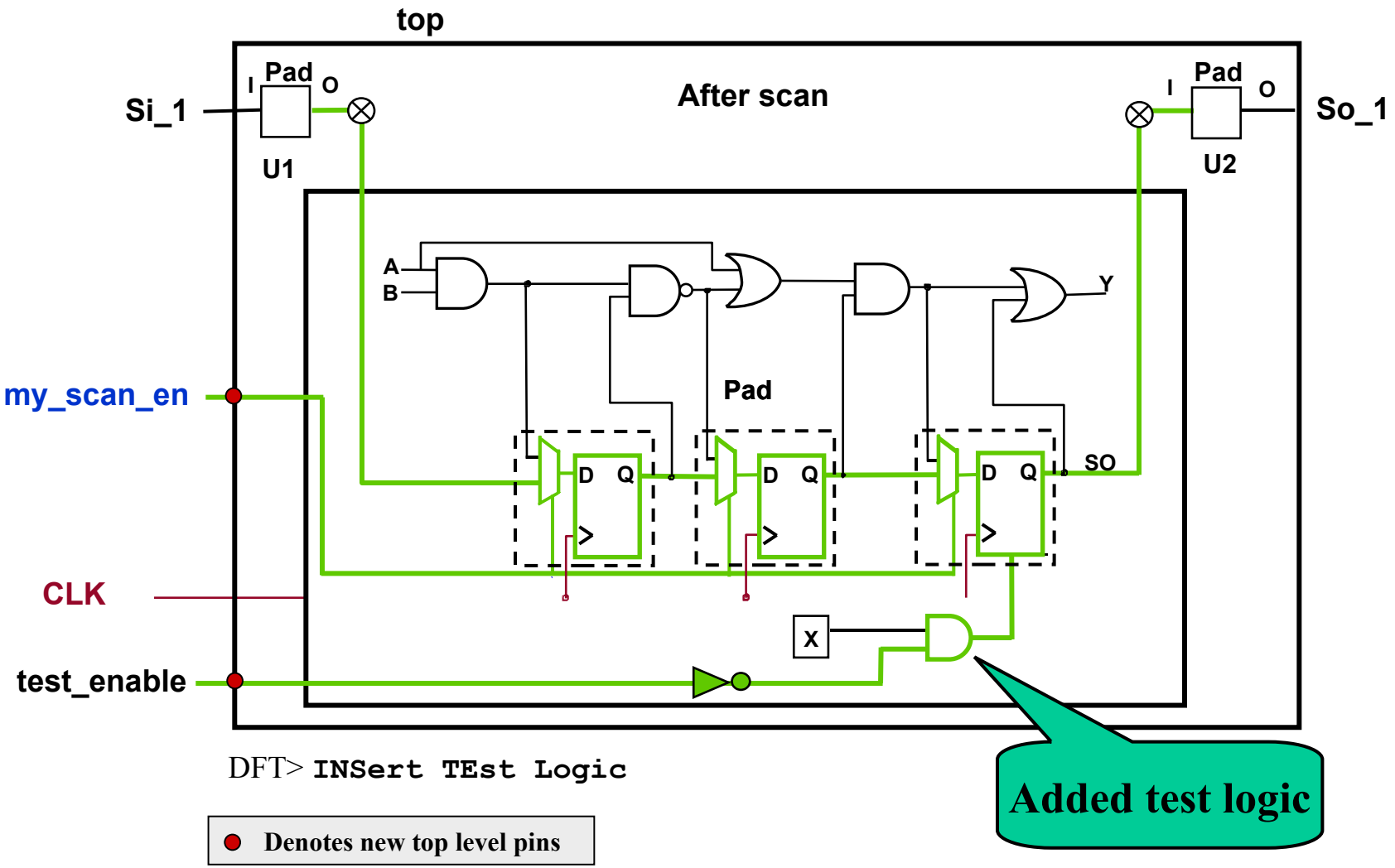
```
SETUP> SETup SCan Insertion \
         -TEN test_en
SETUP> SET Scan Enable my_scan_en
```

Defining Pins (Cont.)



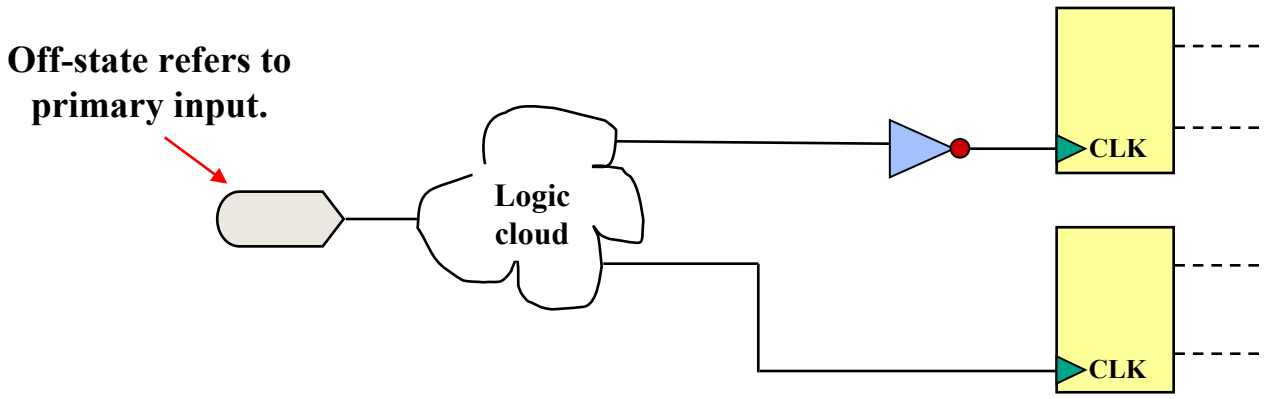

```
SETUP> ADD SCan Pins /U1/O /U2/I
SETUP> SETup SCan Insertion -TEN test_enable
SETUP> SET Scan Enable my_scan_enable
```

Defining Pins (Cont.)



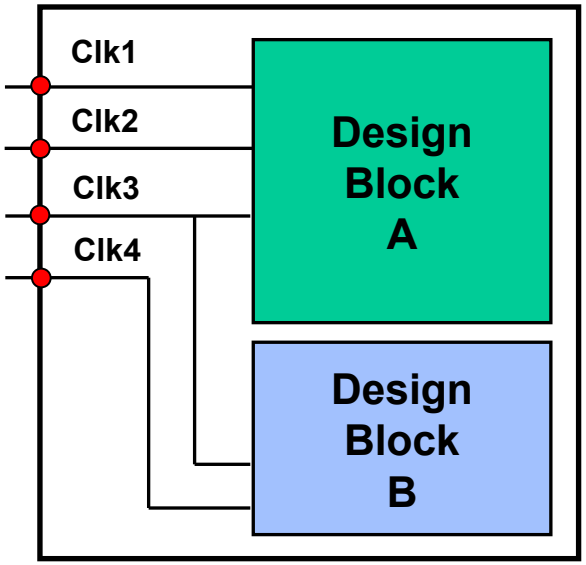
Clocks

- **The following applies to clocks:**
  - **Most designs have multiple clocks.**
  - **Clocks have a defined "off-state"**
  - **Clocks have two types of behavior:**
    - **Shift clocks shift data through the scan chain.**
    - **Capture clocks capture data into scan cells.**



Multiple Clock Issues

- Designs with multiple clock domains can produce clock skew during test.
- Different clock inputs and clock edges can cause the following skew problems:
  - Shifting data through scan chains.
  - Capturing data into scan chains.

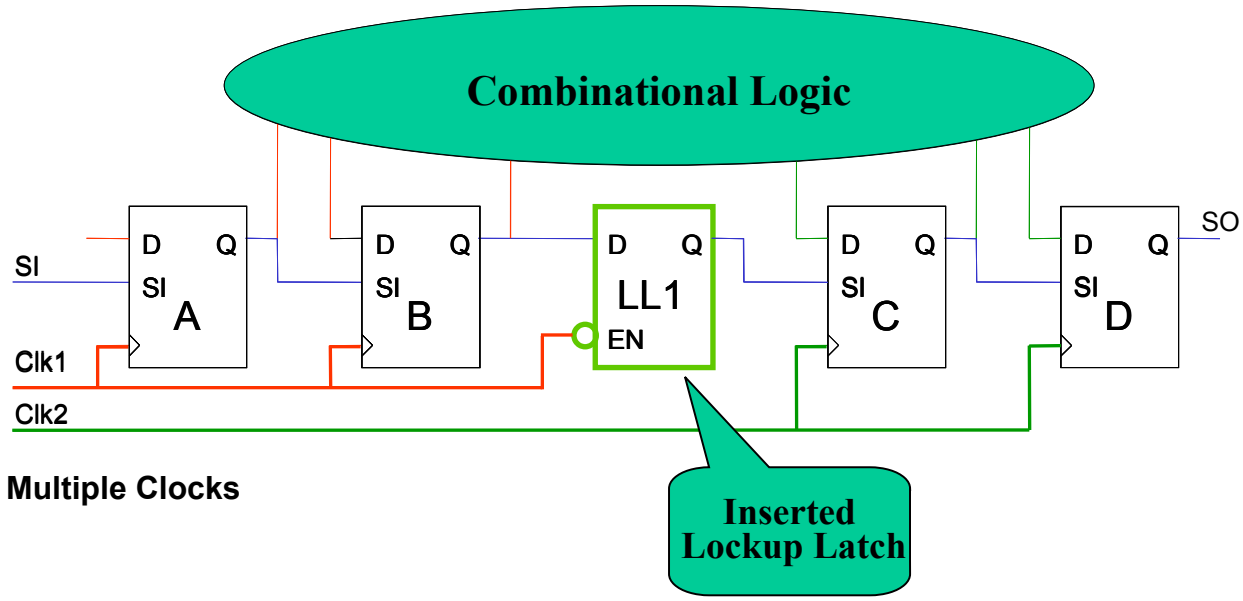


Multiple Clock Issues (Cont.)

- **During shift, all shift clocks are pulsed at the same frequency and time.**
  - **Clock skew results because of different clock domains.**
  - **Clock skew results because of an unbalanced clock tree.**

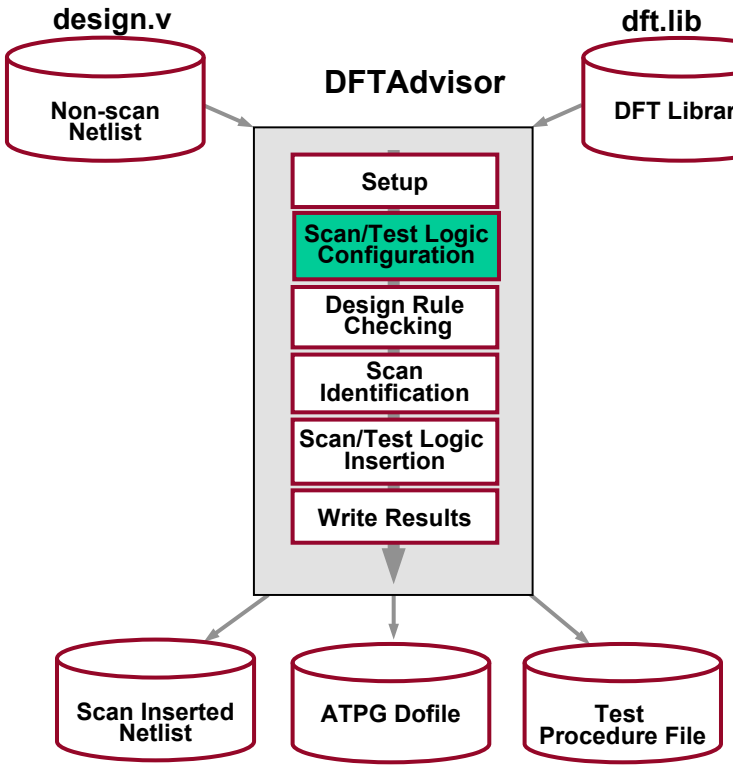
Multiple Clocks: Minimizing Clock Skew

- **Minimize clock skew during shift.**
  - **Order scan chains:**
    - **Group flip-flops together into one clock domain.**
    - **Insert lockup latches where domains cross.**



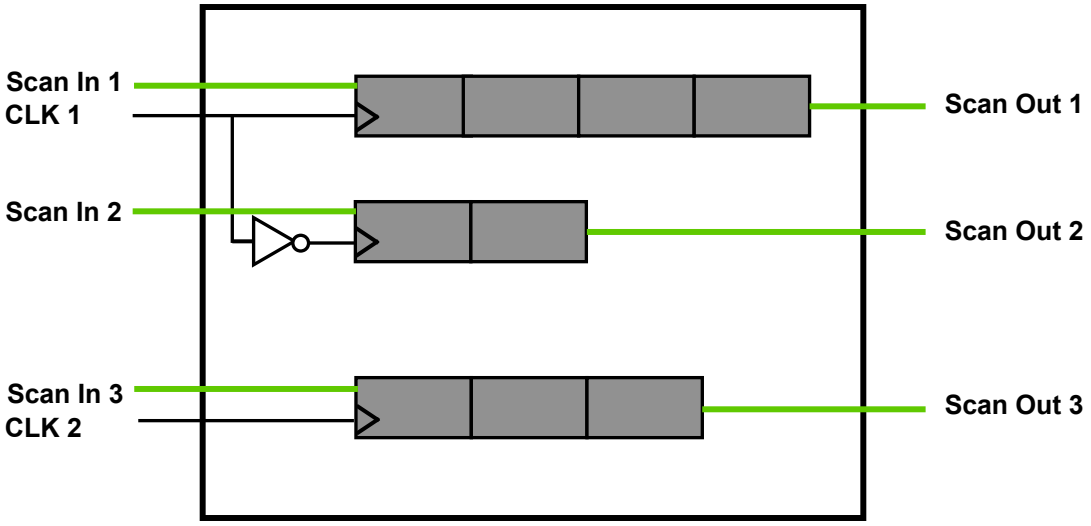
Multiple Clocks

Multiple Clocks



- **By default all clock primary inputs and edges are placed in separate scan chains.**

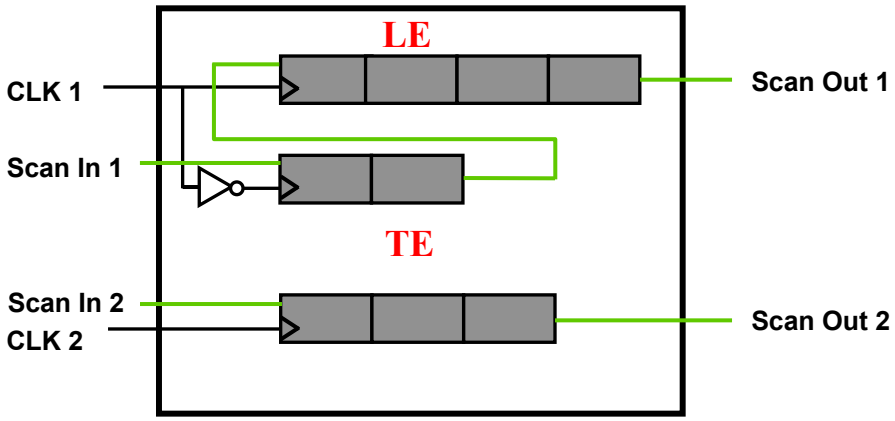
DFT> **INSert TEst Logic**



Multiple Clocks: Merging Clock Edges

- **Leading and trailing edge clocks can be combined into the same scan chain.**
- **DFTAdvisor groups all trailing edge clock scan cells first.**

```
DFT> INSert TEst Logic -Edge Merge
```



Multiple Clocks: Merging Different Clocks

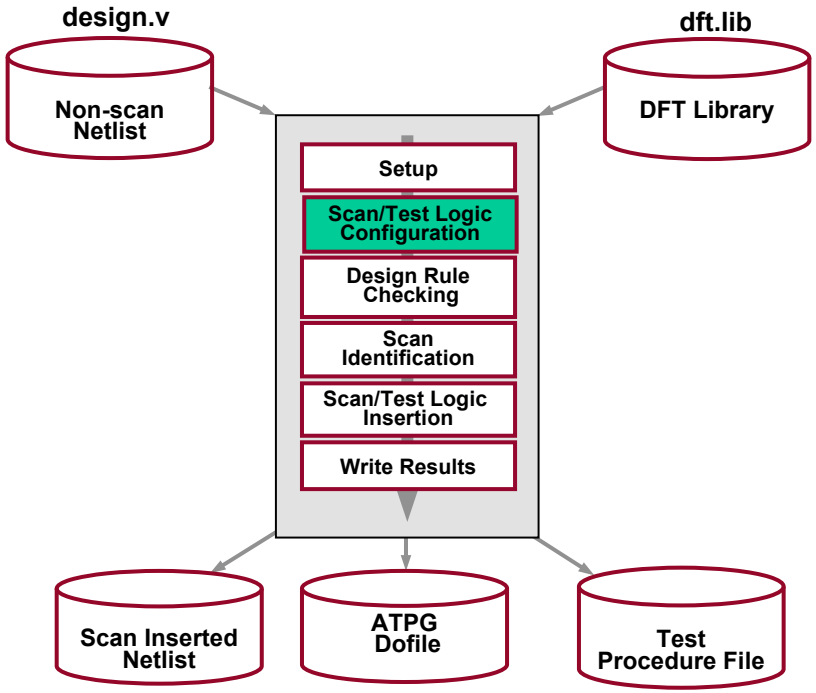
- **Different clocks can be merged into the same chain.**
  - **DFTAdvisor selects scan cells to be merged.**
  - **Tessent DFTAdvisor places lockup latches between each clock domain.**

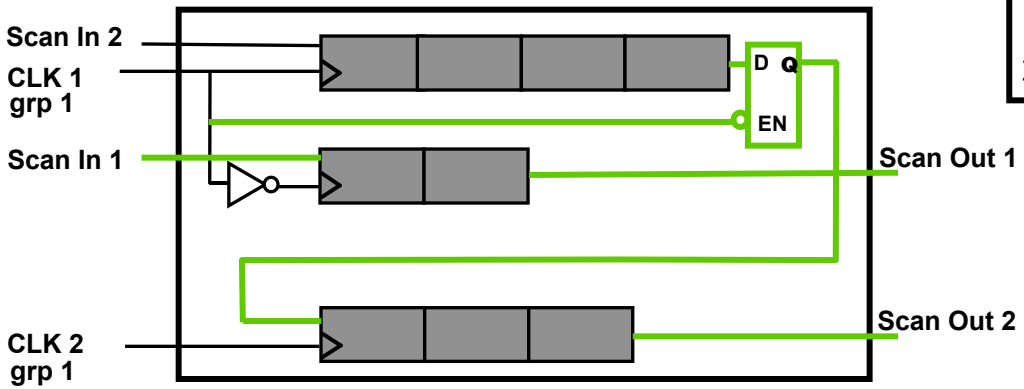
```
DFT> SET LOckup Cell on
DFT> INSert TEst Logic -clock merge
```

- **Multiple clocks can be combined into 'clock groups'.**
  - **Explicitly defines which clocks can be placed into the same scan chain.**

```
DFT> ADD CLocks 0 clk1 clk2 clk3
DFT> ADD CLock Groups group1 clk1 clk2 clk3
DFT> SET LOckup Cell on
DFT> INSert TEst Logic -clock merge
```

Multiple Clocks: Using Lockup Latches





DFT> **INSert TEst Logic -Clock Merge**

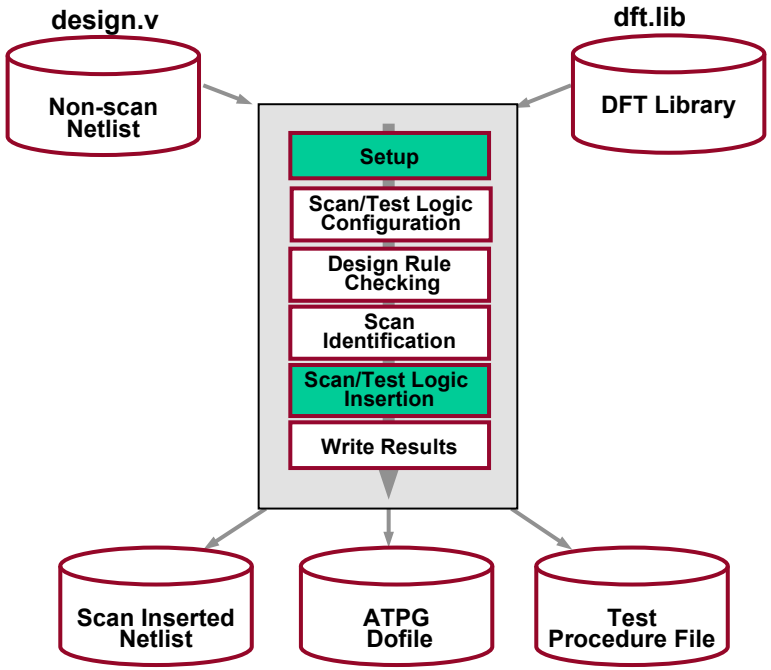
- **Insert lockup latch.**

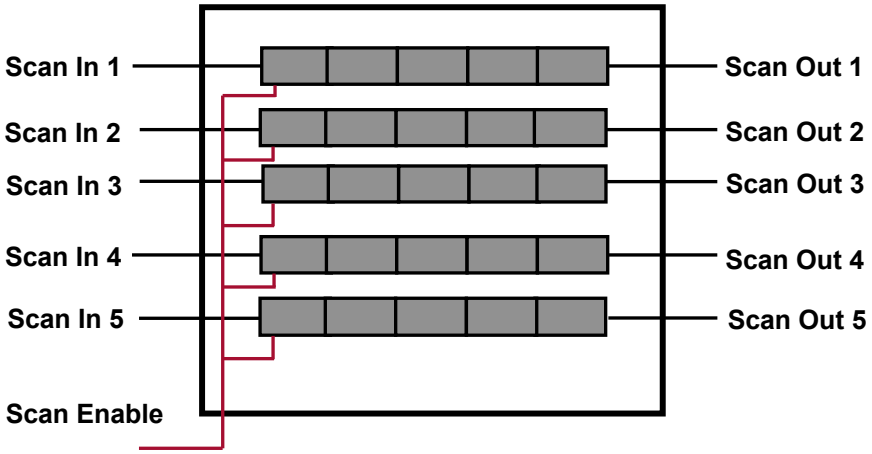
```
SETUP> ADD CLocks 0 clk1  clk2

SETUP> SET System Mode dft

DFT> ADD CLock Groups grp1
       clk1 clk2
DFT> ADD CEll Model DLat1
       -Type DLat enable data
       -Active Low
DFT> SET LOckup Cell ON -Type DLat
       .
       .
       .
DFT> INSert TEst Logic -Clock
Merge
DFT> REPort TEst Logic
```

Balancing Scan Chains

- Testers need deep serial memory for every scan input and output pin.
- Functional pins can be shared as scan pins in test mode.
- Test time and cost is reduced with more and shorter scan chains.
- The number of scan chains is dependent upon tester capabilities.

Balancing Scan Chains (Cont.)





- **Balance scan chains by defining their maximum length or by setting their total number.**

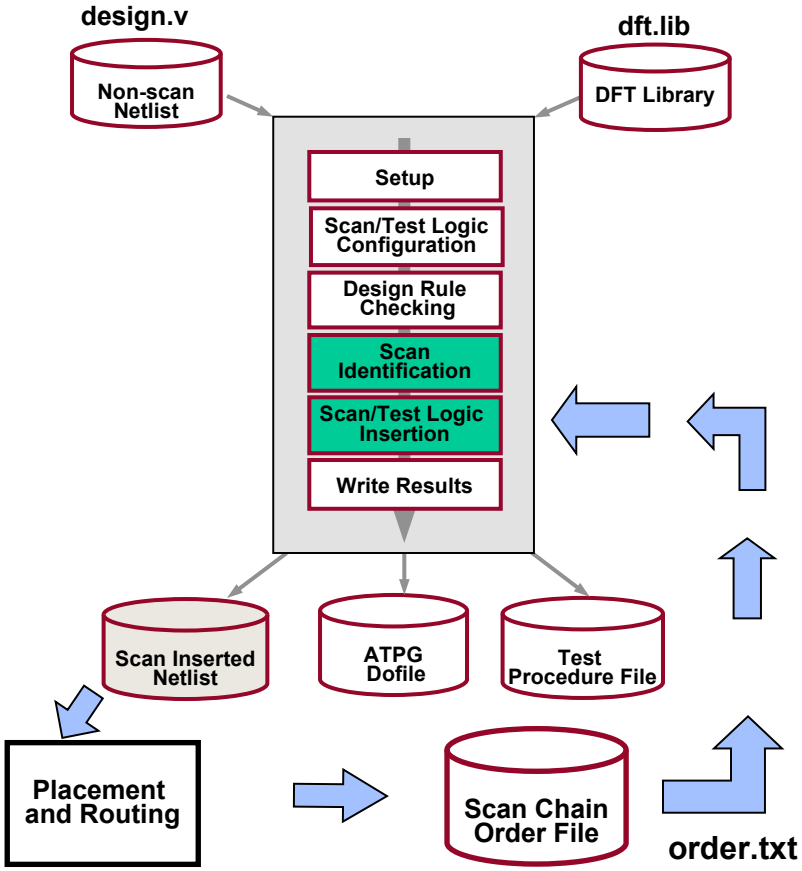
```
SETUP> ADD CEll Models DLat1 -Type \
       DLat enable data -Active Low
SETUP> SET LOckup Cell ON -Type DLat
DFT> INSert TEst Logic -Clock Merge \
     -Edge Merge -NUmber 5
```

5 scan chains will be balanced automatically.

Scan Chain Ordering and Stitching

- **Mux-scan designs:**
  - **Scan cells must be correctly ordered to prevent skew during shift.**
  - **Better placement and routing of scan cells results in better stitching.**
- **To optimize a scan design layout:**
  - **Remove all previous scan chains from the design.**
  - **Reorder the scan cells and write a scan cell order file.**
  - **Stitch scan cells into scan chains using the scan cell order file.**

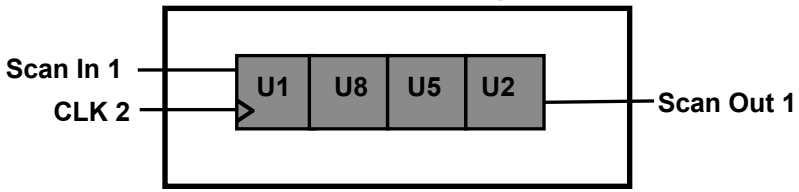
Scan Chain Ordering and Stitching Flow



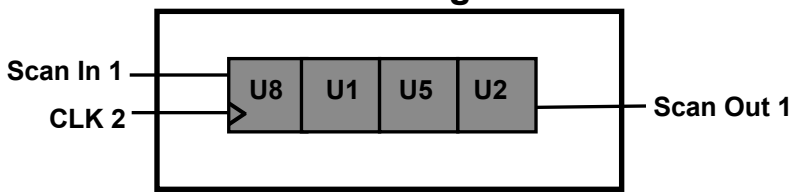
Dofile

```
.
.
.
SETUP> SET SYstem Mode DFT
DFT> RIPup SCan Chains -All
DFT> RUN
DFT> INSert TEst Logic \
                order.txt -
fixed
```

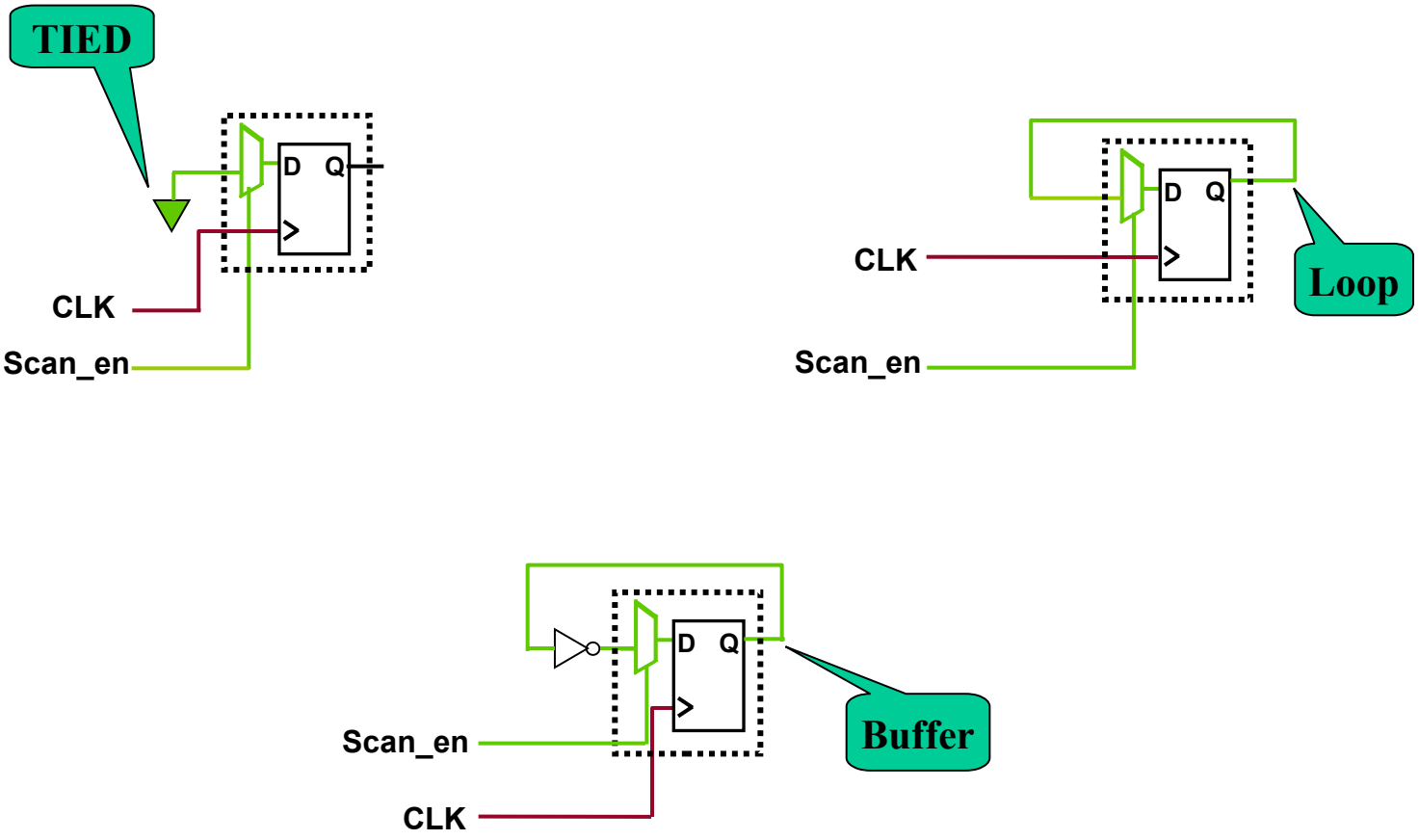
Before Reordering



After Reordering



Scan Cell Configuration After Ripup

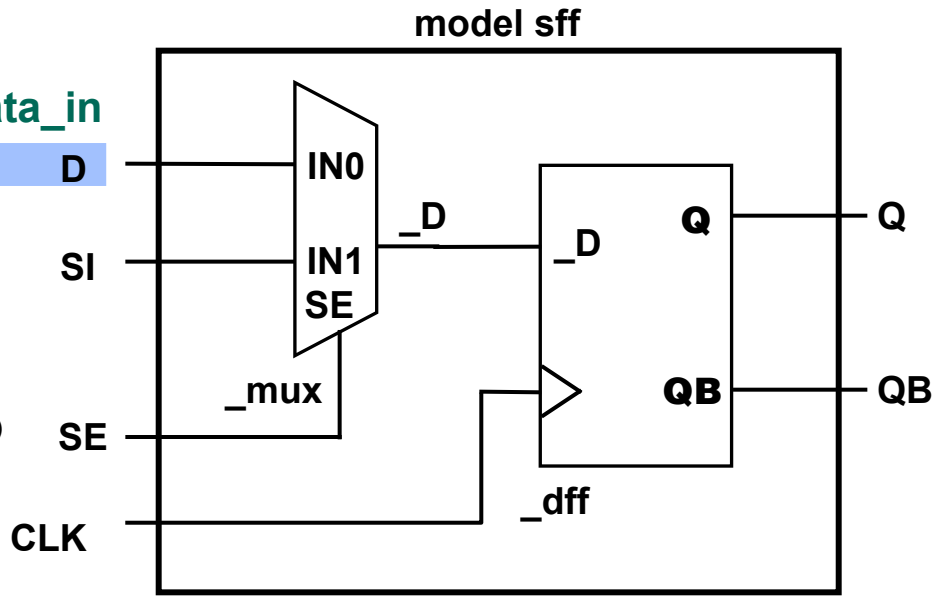


Scan Chain Stitching: Stitching Existing Scan Cells

- **When stitching existing scan, define the following:**
  - **Scan enable (If, previously connected).**

    SETUP> **`SET SCan Enable`**
  - **The "data_in" field of the DFT library model.**

```
model sff (D, SI, SE, CLK, Q, QB) (
 scan_definition (
  type = mux_scan;
     data_in = D;
  scan_in = SI;
  scan_enable = SE;
  scan_out = Q, QB;
  non_scan_model = dff (D, CLK, Q, QB);
)
 input (D, SI, SE, CLK) ( )
 intern(_D) (primitive = _mux (D, SI, SE, _D);)
 output(Q, QB) (primitive = _dff(, , CLK, _D, Q, QB);)
```



Lab 3: Configuring Scan Chains/Test Logic

During this lab, you will

- **Configure scan chains and insert test logic in a full scan flow.**
- **Set up scan pins**
- **Balance scan chains with multiple domains**
- **Stitch scan chains**