Technical Publication

Designs with Multiple Clock Domains: Avoiding Clock Skew and Reducing Pattern Count Using DFTAdvisor*tm* and FastScan*tm*

Mentor Graphics Corporation
March 2001

www.mentor.com/dft

Designs with Multiple Clock Domains: Avoiding Clock Skew and Reducing Pattern Count Using DFTAdvisor^{tm} and FastScan^{tm}

Abstract

This whitepaper discusses DFT and ATPG issues that commonly occur for designs with multiple clock domains. Multiple vs. single clock in test mode and different scan cell structures will be discussed with respect to the different modes of operation in a scan based test. Different pattern generation methods will be discussed with respect to safe operation, pattern count, and test generation runtime.

Table of contents:

Introduction

Nothing accelerates a DFT engineer's receding hairline as the addition of further clock domains to the latest SoC or IC design. Avoiding clock skew during test is becoming one of the biggest DFT challenges for designs with multiple clock domains. Such designs might have several clock domains, usually internally generated. In test mode, these might be combined to one, or they may all be brought out to primary input pins in test mode. The problem that designers worry about is unpredictable clock skew between these clock domains. This can cause difficulties shifting data through the scan chains as well as unreliable ATPG patterns. This issue needs to be taken into consideration during the design stage, since the clock access will affect several aspects of ATPG.

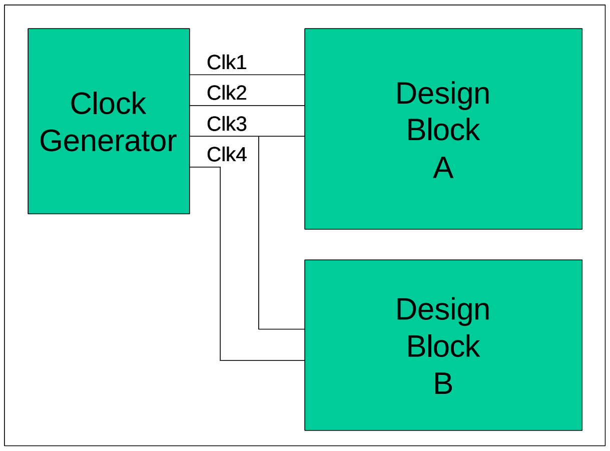

Figure 1 - Design with multiple internally generated clocks

Problem Definition

Usually, clock trees are synthesized with functional operation in mind. For instance, clocks might be pulsed in a specific sequence, and certain clocks might not pulse at the same time. During functional operation, internal clock generators may skew manage the internal clocks going to internal domains. However, the internal domain issues will still exist during test since the tester will bypass the skew managed clock generator. During a scan based test, the clocks might be operated in a different way:

- For ATPG, clocks need to be accessible from primary input pins. For a design with multiple internally generated clocks, this means that each clock must be accessible from a dedicated input pin, or one test clock is used to clock all domains during test. These two alternatives will be discussed later.
- When data is shifted through the scan chains, all clocks are pulsed at the same frequency and at the same time. This means that if multiple clocks are used within the same scan chain there is a danger of clock skew between the clock domains during shift.

- During capture, one or more clocks might be pulsed at the same time. The same sequence that occurs in functional mode might not be feasible for ATPG.

These issues need to be taken into consideration during the design process, scan chain stitching, and ATPG.

Multiple clocks: Considerations during shift:

When a design has multiple clocks in test mode (as illustrated in Figure 4), clock skew can occur between the different domains. We can separate the problem into two issues. Clock skew can occur during shift and during capture. To minimize skew during shift, all scan chains should be ordered such that all flops clocked by one clock domain are grouped together. This minimizes the locations where clock skew can occur. Then, to avoid skew completely where the domains cross, a lockup latch can be inserted. This is illustrated in Figure 2.

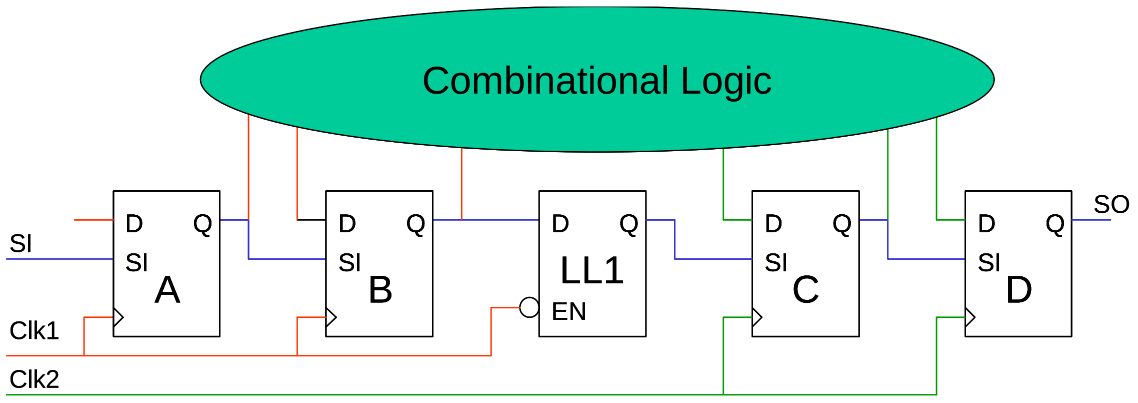

Figure 2 – Multiple clocks

Multiple clocks: Considerations during capture:

The above methodology will only solve the skew problem during shift. During capture, there can be multiple paths between the clock domains. Since there can be a path from both domain 1 to domain 2 and vice versa, simply skewing the clock inputs might not help. The most conservative approach, which is the approach traditionally chosen by ATPG tools, is to only pulse one clock per pattern during capture.

Handling of multiple clocks during capture can be controlled by the ATPG tool. This is discussed in the ATPG section of this paper.

Test Solutions

There are two sides to this problem, hardware design and handling during ATPG. One side will affect the other, and it is important to determine the ATPG strategy before choosing the hardware solution.

Hardware

For ATPG, clocks need to be accessible from primary input pins. For a design with multiple internally generated clocks, this can be done in two ways. One can use only one clock in test mode, or route all the internal clocks to primary input pins during test. Different scan methodologies will also have different requirements.

Mux-DFF: Using one clock in test mode

The first alternative is to use one clock in test mode, as shown in Figure 3. In functional mode, multiple clocks are generated internally. In test mode, the internal clock signals are bypassed with one external clock signal. From the ATPG tool's point of view, the design has one clock.

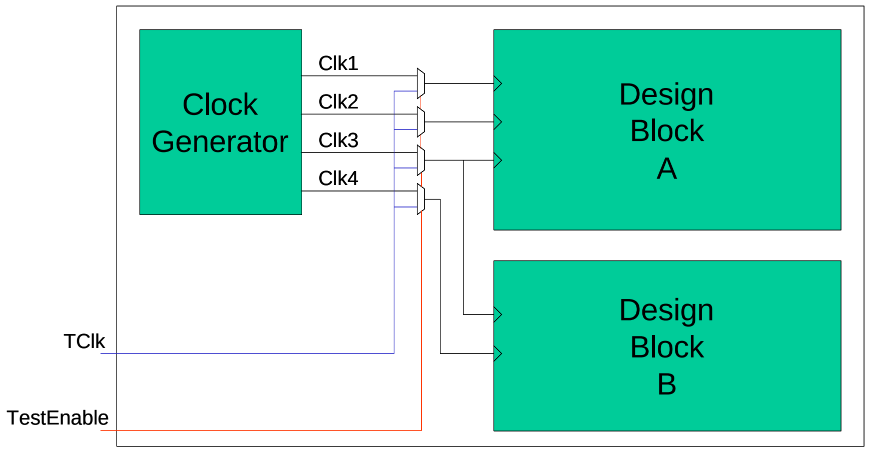

Figure 3 – One clock in test mode

Advantages

✓ Only one pin needs to be used as a dedicated clock pin in test mode.

✓ Compact test pattern set. As will be discussed in a later section, using multiple clocks can increase the test pattern set.

✓ Short ATPG runtime. Since the design has only one clock in test mode, the tool does not have to activate algorithms that help handle multiple clock domains.

Disadvantages

- ✗ This methodology requires very careful clock analysis. For Mux-DFF designs, the clock tree needs to be synthesized separately for test mode. Even though Clk1 and Clk2 are correctly skewed in functional mode, they will now be clocked at the same time. This is required to avoid clock skew during both shift and capture.
- ✗ Complicates scan chain reordering (discussed later).
- ✗ No flexibility. If clock skew occurs, it is very difficult to resolve the problem without modifying the design or loosing test coverage. The ATPG tool's only option is to ignore observing data from scan cells with skew problems, which will reduce test coverage.

Mux-DFF: Using multiple clocks in test mode

A different approach is to use one dedicated pin per internally generated clock, as in Figure 4. In functional mode, multiple clocks are generated internally. In test mode, each internal clock has a different clock pin. Now, also from the ATPG tool's point of view, the design has multiple clocks. All clocks are still clocked at the same time during shift, but the ATPG tool is now free to handle the clock domains in different ways during the capture cycle.

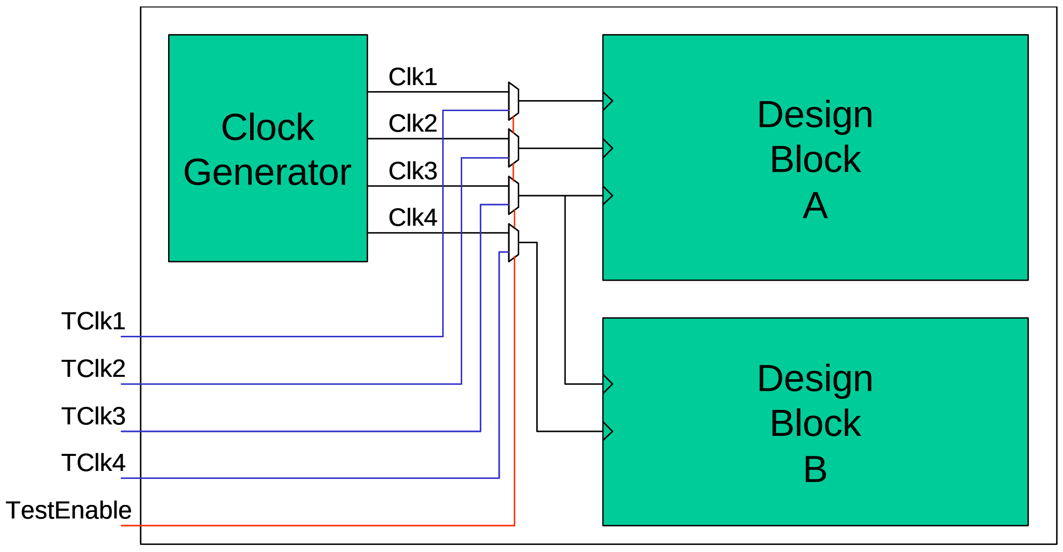

Figure 4 – Multiple clocks in test mode

Advantages

- ✓ Safest approach. Not necessary to do a complete clock tree synthesis for test mode.
- ✓ More flexibility for the ATPG tool. Different methods can be used by the ATPG tool. Each clock can be pulsed separately or at the same time.

Disadvantages

- ✖ Some additional routing and more pins dedicated as clock pins in test mode. It is possible to share test clock pins with functional pins. Such sharing can reduce test coverage (usually insignificantly), because these pins need to be operated as clocks and not regular input pins during test.
- ✖ Increased pattern count. Since each pattern probably will not use all the clocks, the pattern count will increased. Advanced techniques exist to limit the increase in patterns. See the ATPG section of this paper.
- ✖ Solutions that minimize the increase in pattern count will increase the ATPG tool runtime.
- ✖ May not be practical if too many internal clock domains exist.

Mux-DFF: Scan chain ordering

For Mux-DFF based scan designs, the scan chains must be correctly ordered to prevent skew during shift. This is necessary independent of how many clocks are used in test mode.

Correct scan chain ordering typically includes using data from a placement tool to optimize ordering of scan cells. Several placement tools have the ability to perform layout based scan chain stitching, or to recommend a scan chain ordering (and then have a separate tool utilize this ordering information). Such reordering typically helps reduce clock skew during shift.

If one scan chain contains flops clocked by different clocks, or that trigger on different clock edges, most scan insertion tool will group the flops triggered by each domain together, to minimize the risk of clock skew. Some placement tools capable of scan chain reordering do not automatically take his grouping into consideration. Therefore, even though layout based scan chain reordering is recommended, it can introduce the same problem it is supposed to solve if the tool is not set up correctly.

One possible flow is to use DFTAdvisor for scan chain insertion, and have the tool insert lockup latches between each group of scan cells (as illustrated in Figure 2). In the placement tool where scan chains can be reordered, it might be possible to define the lockup latch as an "endpoint". Then, one can first reorder the cells between scan input and the lockup latch (which is domain 1), and then the cells between the lockup latch and scan chain output (domain 2). That way, the clock grouping (and correct location of the lockup latch) is preserved.

DFTAdvisor usage

DFTAdvisor can automatically insert lockup latches between clock domains when two or more clocks are used within one scan chain. The following shows the commands needed:

```
SETUP> add clocks 0 clk1 clk2 clk3
SETUP> add clock groups group1 clk1 clk2 clk3
SETUP> add cell model dlat1 -type dlatn enable data –active low
SETUP> add cell model inv1 –type inv
SETUP> set lockup latch on
….
DFT> insert test logic -clock merge
```

The flow above requires using multiple clocks in test mode. For additional information, see the *set lockup latch on* command in the DFTAdvisor Reference Manual. If one clock is used in test mode, or if lockup latches are needed within one clock domain, one of the following methods can be used:

The tool can be “tricked” into believing that internal nets are primary inputs without physically adding these as input pins to the design. That way, they can be added as clocks and the same method as above can be used:

```
SETUP> add primary input –cut clkbuf1/out  // Net is intclk1
SETUP> add primary input –cut clkbuf2/out  // Net is intclk2
SETUP  add clocks 0 intclk1 intclk2
SETUP> add clock groups group1 intclk1 intclk2
SETUP> add cell model dlat1 -type dlatn enable data –active low
SETUP> add cell model inv1 –type inv
SETUP> set lockup latch on
….
DFT> insert test logic -clock merge
```

The second method to insert lockup latches within one clock domain or when one clock is used in test mode, is to use a scan chain order file. Such a file is also required when inserting scan chains in a specific order based on information from a placement tool.

To generate a scan chain order file from DFTAdvisor, use the following commands:

```
DFT> insert test logic
DFT> report scan cells –file <filename>
```

To insert scan cells in a specific order, use the following command:

```
DFT> insert test logic <filename> -fixed
```

<filename> refers to a scan chain order file. Before this file is reused for the *insert test logic* command, the order can be rearranged and lockup cells can be specified manually, for instance within one clock domain. For details on this file format, see the *insert test logic* command in the DFTAdvisor Reference Manual.

LSSD

Using an LSSD based scan approach rather than Mux-DFF solves many issues related to scan chain shifting. An LSSD scan cell requires three clocks, a system clock, and two non-overlapping clocks that are used during shift, as shown in Figure 5.

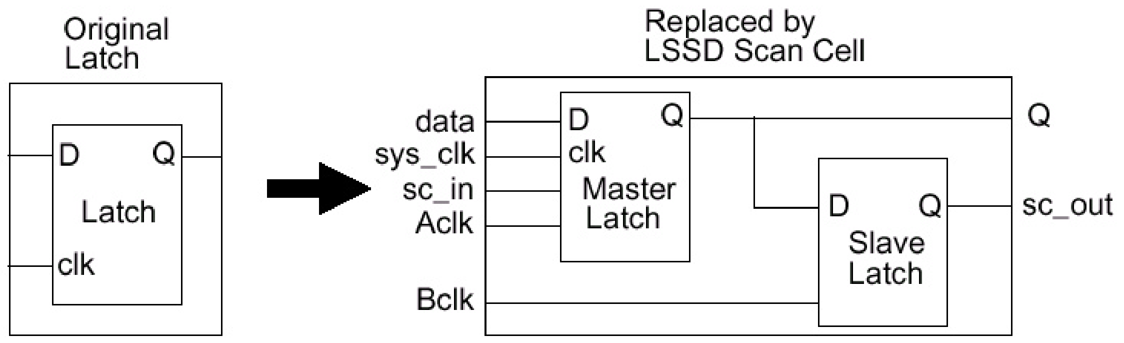

Figure 5 -- LSSD

LSSD cell illustrated in Figure 5 is used to replace a nonscan latch, and is suitable for latch based designs. Similar cells exist for DFF based designs. Such LSSD cells have an edge triggered system clock, but use the same non-overlapping clocks during shift as regular LSSD. Therefore, it is not necessary to have a latch based design prior to scan to use an LSSD approach.

Advantages

- ✓ Clock skew during shift is eliminated since two non-overlapping clocks are used.
- ✓ Approach can work for both latch and DFF based designs.
- ✓ Only two additional primary inputs are needed (shift clocks)

Disadvantages

- ✗ Still skew problems similar to Mux-DFF during capture.
- ✗ Additional area and routing overhead compared to Mux-DFF.

D-mimic

The main advantage of LSSD is to eliminate clock skew during shift. The d-mimic model takes this idea one step further. This is a model that also eliminates clock skew problems during capture by using non-overlapping clocks also during capture. This way, one clock can safely be used in test mode. During scan insertion, the D-mimic scan cell replaces nonscan DFFs. There are different variations of D-mimic cells. Figure 6 shows a simple implementation.

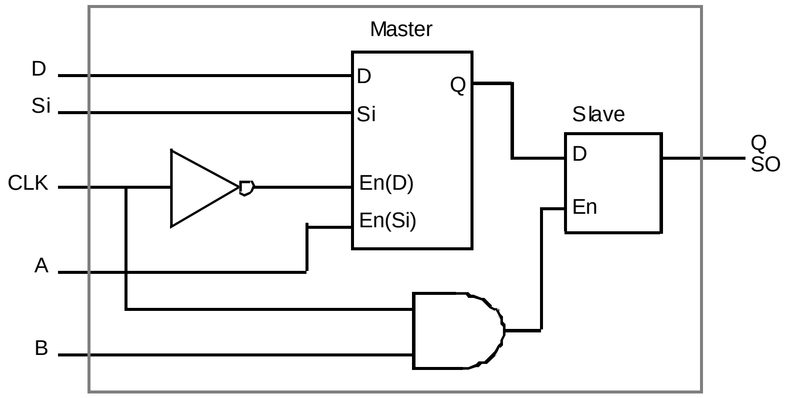

Figure 6 – Simple D-mimic scan cell (Type 1)

For the D-mimic cell in Figure 6, CLK is the functional clock input. The shift clocks A and B do not have to be routed as clocks. Figure 7 illustrates how these clocks operate in functional mode, as well as shift and capture during a scan test. After capture, a *master_observe* procedure is used to copy the contents of the master cell into the slave.

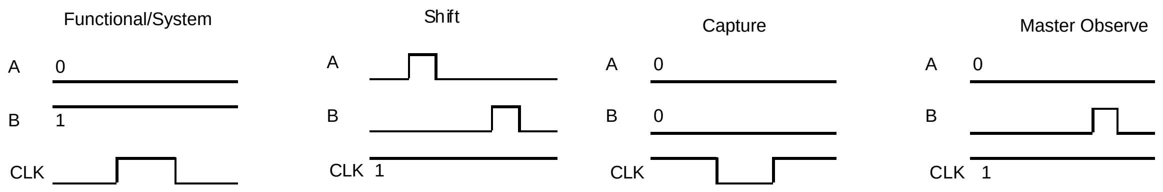

Figure 7 – Operation of Type 1 D-mimic cell

One major disadvantage of the D-mimic cell described in Figures 6 and 7 is that this cell does not support a mixture of rising edge-triggered and falling-edge triggered cells in the same design, even if these cells are located in different scan chains. This is because CLK needs to be "off" during shift, and the off-state will differ for rising edge and falling edge triggered devices. A more complex D-mimic cell that supports a mixture of rising edge and falling edge triggered devices is shown in Figure 8.

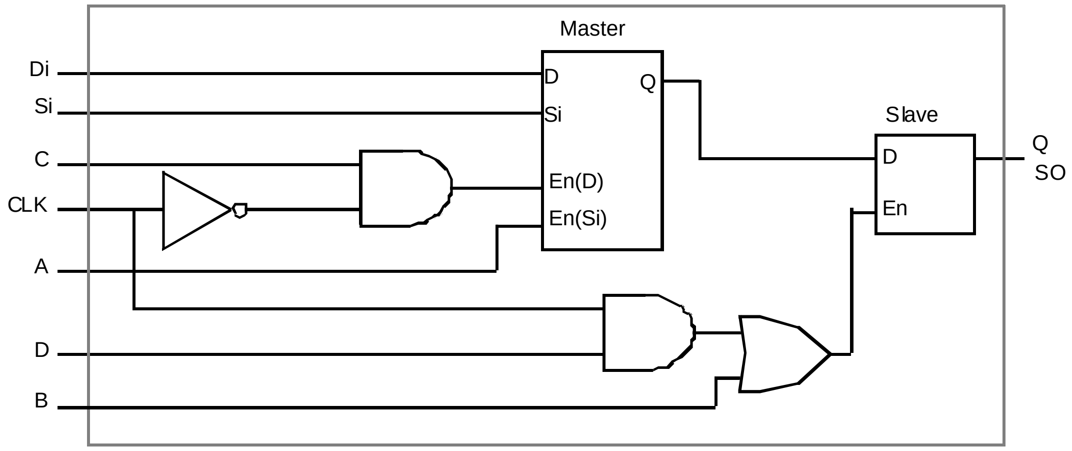

Figure 8 – Complex D-mimic cell (Type 2)

For the cell in Figure 8, additional signals are used to disable the functional clock CLK. Notice that even though two additional nets (C and D) exist, only CLK needs to be routed as a clock. The operation of the Type 2 D-mimic cell is illustrated in Figure 9.

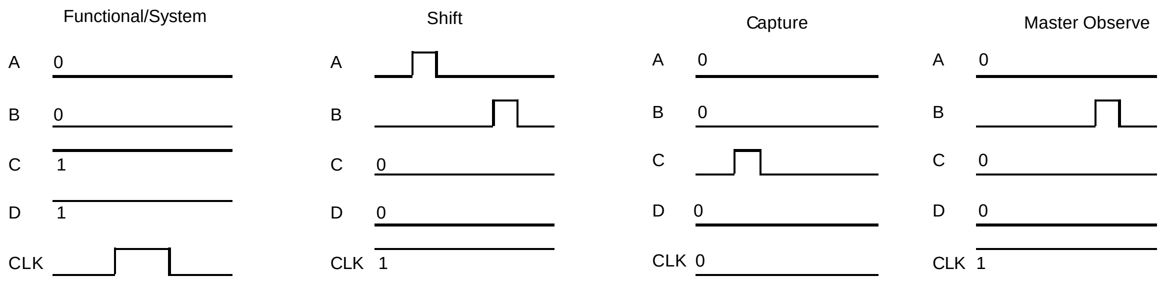

Figure 9 – Operation of Type 2 D-mimic cell

See Appendix A for details on D-mimic support in DFTAdvisor and FastScan.

Advantages
- ✓ Eliminates clock skew for all test modes (capture and shift)
- ✓ No need for layout based scan chain reordering
- ✓ Most compact pattern set

Disadvantages
- × Higher area and routing overhead than both Mux-DFF and LSSD.
- × Most models do not support at-speed capture using functional (system) clock.
- × Not supported by all tools and ASIC vendors

ATPG

In the previous section, we discussed hardware approaches that enabled the use of either one or multiple clocks in test mode. This section will discuss different ways an ATPG tool can handle multiple clocks in test mode. For all of these methods, it is assumed that clock skew during shift is taken care of by correct ordering of the scan chains and usage of lockup latches.

Pulsing one clock per pattern

The traditional way to avoid clock skew is to pulse only one clock per pattern. During shift, all clocks are pulsed at the same time, but only one clock is selected per pattern, as illustrated in Figure 10. In this example, each scan chain has 2 scan cells, and there is a total of four scan clocks.

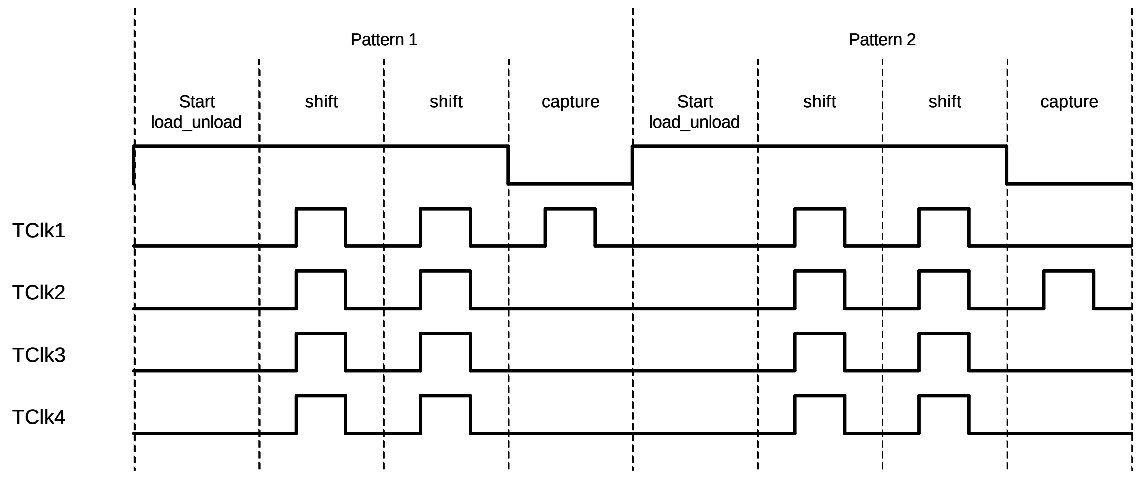

Figure 10 – Pulsing one clock per pattern

Advantages

- ✓ The easiest way to avoid clock skew during capture.
- ✓ Little ATPG effort (short ATPG runtime).

Disadvantages

- × High pattern count. For a design with ten clocks, one can experience up to ten times more patterns than if the design had one clock.

FastScan usage

In FastScan, this is the default behavior when combinational patterns are generated, and when dynamic compression is either on or off. This behavior can be forced to occur with dynamic compression for both combinational and sequential patterns using:

```
ATPG> SET ATpg Compression on –NOMulti_clock_capture
```

Utilizing clock independence

One flow is to first route all clocks to separate inputs in test mode, and then analyze which clock domains are independent (i.e. have no functional paths between them). This can be between certain domains, or between all domains. After such analysis has been done, the user can tell the ATPG tool to treat multiple clocks as equivalent clocks. That will have the same effect as using one pin for these clock pins. In the example shown in Figure 11, analysis shows that there are no functional paths between clock domains 1 and 3. Therefore, for Pattern 1 clocks TClk1 and TClk3 can be pulsed at the same time. Since there is interaction between other domains, clock TClk2 is pulsed alone for Pattern 2.

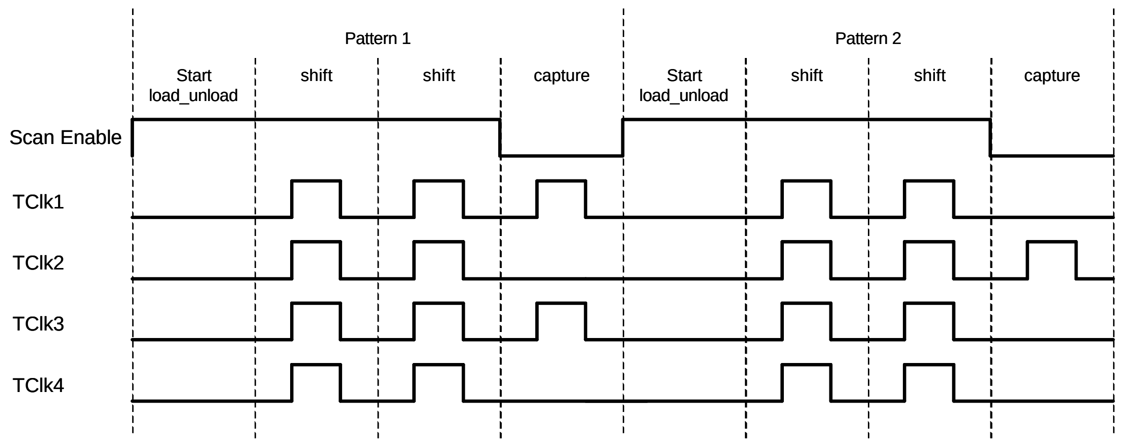

Figure 11 – Clock domain analysis allows Tclk1 and Tclk3 to be pulsed simultaneously

Advantages

- ✓ Low pattern count (the same as when using one clock in test mode).
- ✓ Flexibility, if there is a problem between certain domains, one can use a different approach without changing the hardware.

Disadvantages

- ✗ Analysis required. One has to find out which clocks can be pulsed at the same time without causing problems.

FastScan usage

In FastScan, this can be done using the *add pin equivalence* command. In the following example, TClk1 and TClk3 will be pulsed at the same time, while TClk2 and Tclk4 will be pulsed separately as shown in Figure 11:

```
SETUP> ADD CLocks 0 TClk1 TClk2 TClk3 TClk4
SETUP> ADD PIn Equivalence TClk1 TClk3
```

Pulsing clocks sequentially

This is an approach that results in a compact pattern set without risking clock skew. The capture takes place over multiple cycles. Only one clock is pulsed per cycle. Since the same timeplate is used for all cycles, no additional timeplates or other settings are required. This method is illustrated in Figure 12 and is utilized by FastScan's multi clock compression. Figure 13 shows another example with values for a design with two clocks.

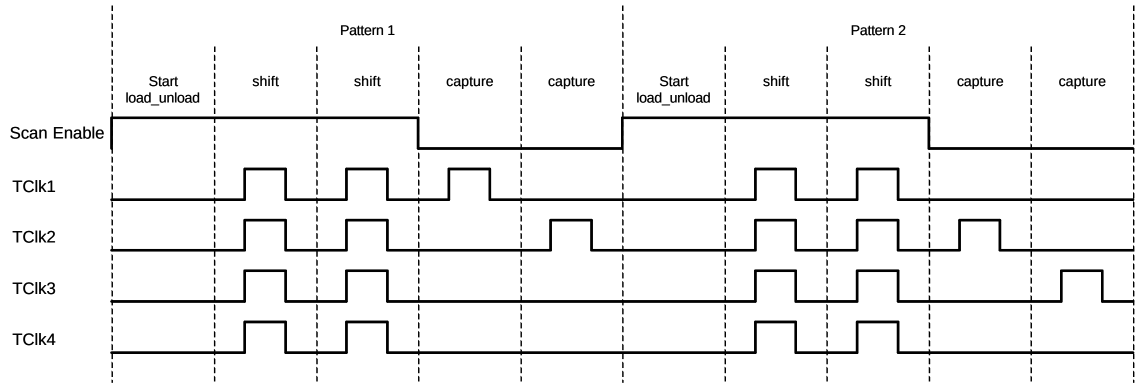

Figure 12 – Two capture clocks pulsed sequentially in a dual-cycle capture.

Advantages:

- ✓ Reduced pattern count (compared to pulsing one clock per pattern)
- ✓ Very good compression results with only a sequential depth of 2 or 3 even for designs with many clocks.
- ✓ No risk for clock skew during capture

Disadvantages

- ✗ Increased ATPG runtime

FastScan usage

In FastScan, this is the default when dynamic compression is enabled and the sequential depth is set to 2 or greater. Increasing the depth further will allow even more than 2 clocks to be pulsed sequentially per pattern.

```
ATPG> SET ATpg Compression on –Multi_clock_capture // Default
ATPG> SET Simulation Mode comb –depth 2
ATPG> create patterns –compace           // Generate compressed pats.
```

The *–multi_clock_capture* option is the default when atpg compression is set on. However, to achieve multi clock compression, the sequential depth must always be set to 2 or higher using the *set simulation mode* command (the default depth is 0). The lowest pattern count will be achieved when the depth is equivalent to the number of clocks in the design. Since increasing the depth also means increasing the runtime, the recommended depth is 2 – 3.

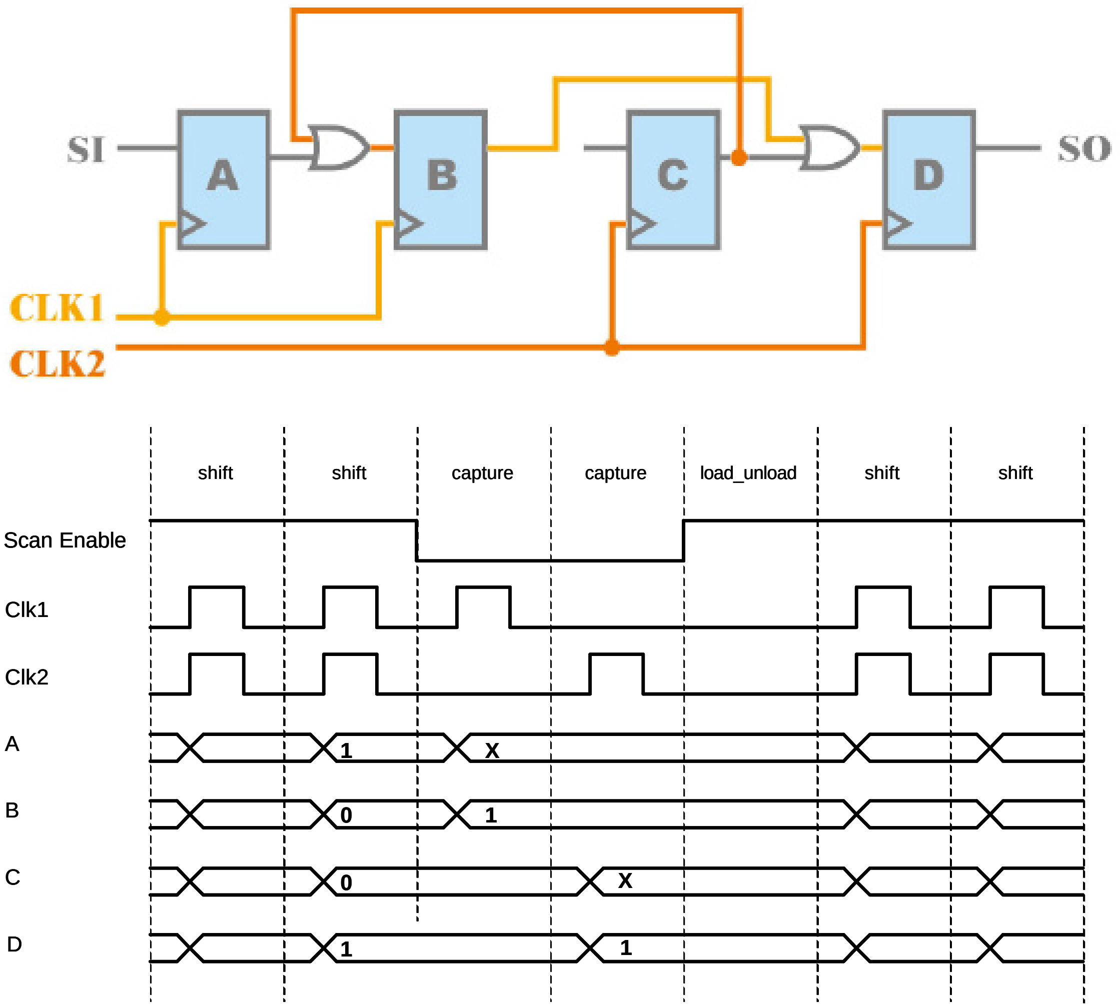

Figure 13 – Example with values

Recommendations and Conclusions

For Mux-DFF based designs, the best solution for designs with many clock domains is to let each internal clock domain have a clock pin in test mode. This will leave the most options to the ATPG tool. To achieve the most compact pattern set, one should first analyze which domains do not interact with each other, and allow the ATPG tool to pulse these clocks simultaneously. Then, the remaining clocks should be pulsed sequentially using a multi-clock compression technique. These two methods combined will result in the safest approach to a minimized pattern set.

Using an LSSD approach will reduce the need for scan chain reordering and clock skew during shift. However, to reduce pattern count and eliminate clock skew during capture, the same approach as is described above for Mux-DFF based designs should be taken for system clocks for designs using LSSD cells with edge triggered system clocks.

Using a D-mimic approach will reduce the need for both scan chain reordering, make the ATPG process simpler, eliminate the need for multiple clock pins in test mode, and produce the lowest possible pattern count. However, the footprint is increased, and such cells are not available from all ASIC vendors. Also, system clock at-speed capture using transition or path delay fault models is not possible using D-mimc.

Table 1 shows normalized runtimes and pattern sizes for different methodologies for a small design with 38,000 gates, 2,120 scan cells and four clock domains. It is found that clocks 3 and 4 do not have interaction between them. The runs using pin equivalence (clock independence) do therefore effectively have 3 clocks, clk1, clk2, and clk3/clk4 combined. All runs are using dynamic compression, and result in 99.6% test coverage.

Table 1

Method	Runtime	Pattern size	Comment
One clock in test mode	0.39	0.40	Dangerous!!! Clock skew will occur!
Four clocks in test mode	1.00	1.00	Safe & simple
Four clocks in test mode and multi clock compression (2 clocks)	1.39	0.58	Sequential depth = 2 for multi clock compression
Four clocks, multi-clock compression (3 clocks)	2.73	0.46	Sequential depth = 3 for multi clock compression
Pin equivalence (clk3=clk4)	0.83	0.80	Clock domain analysis required. Found clk3 and clk4 independent.
Pin equivalence and multi-clock compression (2)	2.36	0.54	Sequential depth = 2, clk3=clk4.
D-mimic, one clock in test mode	0.39	0.40	Safe for both shift and capture

Appendix A –D-mimic cells in DFTAdvisor and FastScan

Two D-mimic cells were presented in this Whitepaper. What's defined as a Type 1 cell is described in Figures 6 and 7 and they Type 2 cell is described in figures 8 and 9. To utilize these cells in DFTAdvisor and FastScan, ATPG library models have to be defined and the waveforms have to be correctly implemented in test procedures and commands.

Current versions of DFTAdvisor and FastScan (v8.9_1.10) allow usage of D-mimic scan cells. Some manual modifications are necessary to the enhanced procedure file and dofile produced by DFTAdvisor.

Type 1 D-mimic (Figure 6)

This D-mimic type does not allow a mixture of rising-edge and falling-edge triggered devices in the same design when these devices are using the same system clock.

ATPG library definition for DFTAdvisor and FastScan:

```
model dmimic_type1 (Q, QBAR, SI, D, CLK, A, B) (
     scan_definition (
       type = lssd;
       scan_in = SI;
       data_in = D;
       scan_master_clock=A;
       scan_slave_clock=B;
       scan_out=Q,QBAR;
       non_scan_model=dff(Q,QBAR,CLK,D);)
     input(SI,D,CLK,A,B) ()
     intern (CLK_INV)   ( primitive = _inv ( CLK, CLK_INV ) ; )
     intern (SL_EN)     ( primitive = _and ( CLK, B, SL_EN ) ; )
     intern (Q_MASTER)  ( primitive = _dlat ( , , CLK_INV, D, A, SI,
                                    Q_MASTER, ) ; )
     output (Q, QBAR)   ( primitive = _dlat ( , , SL_EN, Q_MASTER, Q,
                                    QBAR ) ; )
)
```

DFTAdvisor dofile:

```
analyze control signals  -auto_fix
// or manually use "add clocks"
set scan type lssd
setup scan insertion  -smclk A -ssclk B
set system mode dft
setup scan identification full_scan
run
insert test logic  -number 2
write netlist test_scan.v -verilog -replace
write atpg setup test_scan -replace -procfile
```

Dofile for FastScan:

```
//
//  Generated by DFTAdvisor at Wed Mar  7 10:55:17 2001
//  Manual changes marked with "MODIFY"
//
add scan groups grp1 test_scan_modify.testproc    // MODIFY
add scan chains chain1 grp1 scan_in1 scan_out1
add scan chains chain2 grp1 scan_in2 scan_out2
add clocks 1 CLOCK                                // MODIFY
add clocks 0 B
add clocks 0 A
```

Enhanced procedure file for FastScan:

```
//
// Generated by DFTAdvisor at Wed Mar  7 10:55:17 2001
// Manual changes marked with "MODIFY"
//
set time scale 1.000000 ns ;
 timeplate gen_tp1 =
    force_pi 0 ;
    measure_po 10 ;
    pulse A 20 10;
    pulse B 40 10;
    pulse CLOCK 30 20;    // MODIFY
    period 60 ;
 end;

 procedure shift =
    scan_group grp1 ;
    timeplate gen_tp1 ;
    cycle =
        force_sci ;
        measure_sco ;
        pulse A ;
        pulse B ;
    end;
 end;

 procedure load_unload =
    scan_group grp1 ;
    timeplate gen_tp1 ;
    cycle =
        force A 0 ;
        force B 0 ;
        force CLOCK 1 ;         // MODIFY
    end ;
    apply shift 3;
 end;

 procedure master_observe =
    scan_group grp1 ;
    timeplate gen_tp1 ;
    cycle =
        force A 0 ;
        force B 0 ;
        force CLOCK 1 ;         // MODIFY
        pulse B ;
    end;
 end;
```

Type 2 D-mimic (Figure 8)

This D-mimic type does allow a mixture of rising-edge and falling-edge triggered devices in the same design when these devices are using the same system clock. Notice that some undetected faults will occur on the additional control lines C and D. Also notice that CLK is not defined as a clock in DFTAdvisor or FastScan. A *skew_load* procedure can be added to catch additional faults on this model.

ATPG library definitions for DFTAdvisor and FastScan:

```
// Rising edge triggered
model dmimic_type2 (Q, QBAR, SI, DI, CLK, A, B, C, D) (
     scan_definition (
       type = lssd;
       scan_in = SI;
       data_in = DI;
       scan_master_clock=A;
       scan_slave_clock=B;
       test_clock=C;
       test_enable=D;
       scan_out=Q,QBAR;
       non_scan_model=dff(Q,QBAR,CLK,DI);)
     input(SI,DI,CLK,A,B,C,D) ()
     intern (CLK_INV)   ( primitive = _inv ( CLK, CLK_INV ) ; )
     intern (D_EN)      ( primitive = _and ( C, CLK_INV, D_EN ) ; )
     intern (SL_EN1)    ( primitive = _and ( CLK, D, SL_EN1 ) ; )
     intern (SL_EN2)    ( primitive = _or  ( SL_EN1, B, SL_EN2 ) ; )
     intern (Q_MASTER)  ( primitive = _dlat ( , , D_EN, DI, A, SI,
                             Q_MASTER, ) ; )
     output (Q, QBAR)   ( primitive = _dlat ( , , SL_EN2, Q_MASTER, Q,
                             QBAR ) ; )
)

// Falling edge triggered
model dmimic_type2i (Q, QBAR, SI, DI, CLK, A, B, C, D) (
     scan_definition (
       type = lssd;
       scan_in = SI;
       data_in = DI;
       scan_master_clock=A;
       scan_slave_clock=B;
       test_clock=C;
```

```
        test_enable=D;
        scan_out=Q,QBAR;
        non_scan_model=dffi(Q,QBAR,CLK,DI);)
      input(SI,DI,CLK,A,B,C,D) ()
      intern (CLK_INV)   ( primitive = _inv ( CLK, CLK_INV ) ; )
      intern (D_EN)      ( primitive = _and ( C, CLK, D_EN ) ; )
      intern (SL_EN1)    ( primitive = _and ( CLK_INV, D, SL_EN1 ) ; )
      intern (SL_EN2)    ( primitive = _or  ( SL_EN1, B, SL_EN2 ) ; )
      intern (Q_MASTER)  ( primitive = _dlat ( , , D_EN, DI, A, SI,
                               Q_MASTER, ) ; )
      output (Q, QBAR)   ( primitive = _dlat ( , , SL_EN2, Q_MASTER, Q,
                               QBAR ) ; )
)
```

DFTAdvisor dofile:

```
analyze control signals  -auto_fix
// or manually use "add clocks"
set scan type lssd
setup scan insertion -tclk C -smclk A -ssclk B -ten D
set system mode dft
setup scan identification full_scan
run
insert test logic  -number 2
report test logic
write netlist test_scan.v -verilog -replace
write atpg setup test_scan -replace -procfile
```

Dofile for FastScan:

```
//
//  Generated by DFTAdvisor at Thu Mar  8 08:58:34 2001
//  Items changed marked with "MODIFY"
//

add scan groups grp1 test_scan_modify.testproc // MODIFY
add scan chains chain1 grp1 scan_in1 scan_out1
add scan chains chain2 grp1 scan_in2 scan_out2
add clocks 0 C
// add clocks 0 CLOCK       // MODIFY (commented out)
add clocks 0 B
add clocks 0 A
// add pin constraints D C1 // MODIFY (commented out)
```

Enhanced procedure file for FastScan:

```
//
// Generated by DFTAdvisor at Thu Mar  8 09:07:07 2001
// Items changed marked with "MODIFY"
//

set time scale 1.000000 ns ;
 timeplate gen_tp1 =
    force_pi 0 ;
```

```
     measure_po 10 ;
    pulse A 20 10;
    pulse B 40 10;          // MODIFY
    pulse C 30 20;          // MODIFY
    //pulse CLOCK 20 10;    // MODIFY (commented out)
    period 60 ;             // MODIFY
 end;

// procedure test_setup =
//    timeplate gen_tp1 ;
//    cycle =
//        force D 0 ;        // MODIFY
//    end;
// end;

 procedure shift =
    scan_group grp1 ;
    timeplate gen_tp1 ;
    cycle =
        force_sci ;
        measure_sco ;
        pulse A ;
        pulse B ;
    end;
 end;

 procedure load_unload =
    scan_group grp1 ;
    timeplate gen_tp1 ;
    cycle =
        force A 0 ;
        force B 0 ;
        force C 0 ;
        force D 0 ;          // MODIFY
        force CLOCK 1 ;      // MODIFY
    end ;
    apply shift 3;
 end;

 procedure master_observe =
    scan_group grp1 ;
    timeplate gen_tp1 ;
    cycle =
        force A 0 ;
        force B 0 ;
        force C 0 ;
        force D 0 ;          // MODIFY
        force CLOCK 1 ;      // MODIFY
        pulse B ;
    end;
 end;
```

For more information, call us or visit: www.mentor.com/dft