

AS GATE COUNTS INCREASE AT AN ENORMOUS RATE, THE SCAN-DESIGN METHODOLOGY IS BECOMING NECESSARY TO PRODUCE HIGH-QUALITY CHIPS.

10 tips for successful scan design: part one

ALTHOUGH SCAN-DESIGN METHODOLOGIES have existed for several years, many companies are just starting to explore the use of scan, particularly as these companies create more complex, system-on-chip (SOC) designs. With gate counts increasing at an enormous rate, producing high-fault-coverage production tests without using scan techniques becomes increasingly difficult. Scan is becoming a necessary design methodology to produce high-quality chips.

If you're just starting out in scan design, this two-part series provides useful design tips to ensure successful adoption of scan-design methodologies within your company or design group. Part one reviews the scan methodology and techniques. Part two presents the tips. By adhering to these tips, you can produce chips that current automatic-test-pattern-generation (ATPG) tools can process to generate scan-based test vectors providing high fault coverage.

Before starting your project, research scan-design methodologies and read as much as you can. In addition to this series, **references 1 and 2** are great introductions to scan. (You can also download "Simple case study" from EDN's Web site, www.ednmag.com/ednmag/reg/2000/02172000/04ms604.htm.) Don't expect just the "scan expert" of the project to learn scan techniques. Get everyone involved. The more the designers know, the easier it is for them to produce scan-friendly designs. (See **sidebar** "Glossary" for a list of definitions and acronyms.)

Also, don't underestimate the amount of time it takes to produce a scan design. If your company is just starting out, scan design will require a large amount of time. You may encounter many pitfalls and unexpected problems. Your managers may want to send the chip to the fab without allowing you much time to simulate the test patterns produced by the ATPG tools. But if you don't run back-annotated simulations of the test patterns, then don't expect

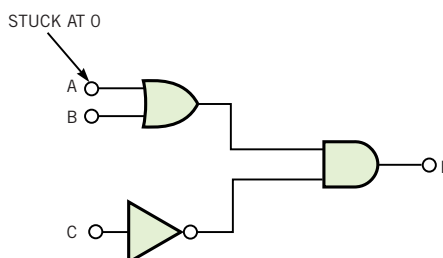
the patterns to work. These simulations will alert you to timing problems, tool problems, and functional problems.

WHY IS SCAN IMPORTANT?

Designers create functional simulations to verify the proper operation of their designs. For example, a designer of a memory controller creates simulations to verify that the design operates correctly within the system. This approach is fine for the virtual world, in which the chip is only bits and pieces of HDL coding. However, you ultimately want to manufacture a real chip and verify that it works properly. Production tests can verify that the manufacturer correctly implemented the design and that the design is free from flaws, such as power and ground shorts, open interconnections due to dust particles, and others. In short, production testing ensures that the customer receives high-quality parts with low failure rates.

In the past, you could use functional simulations to generate test vectors, which you could then use to verify newly manufactured chips on a tester. But because of the high gate counts and extreme com-

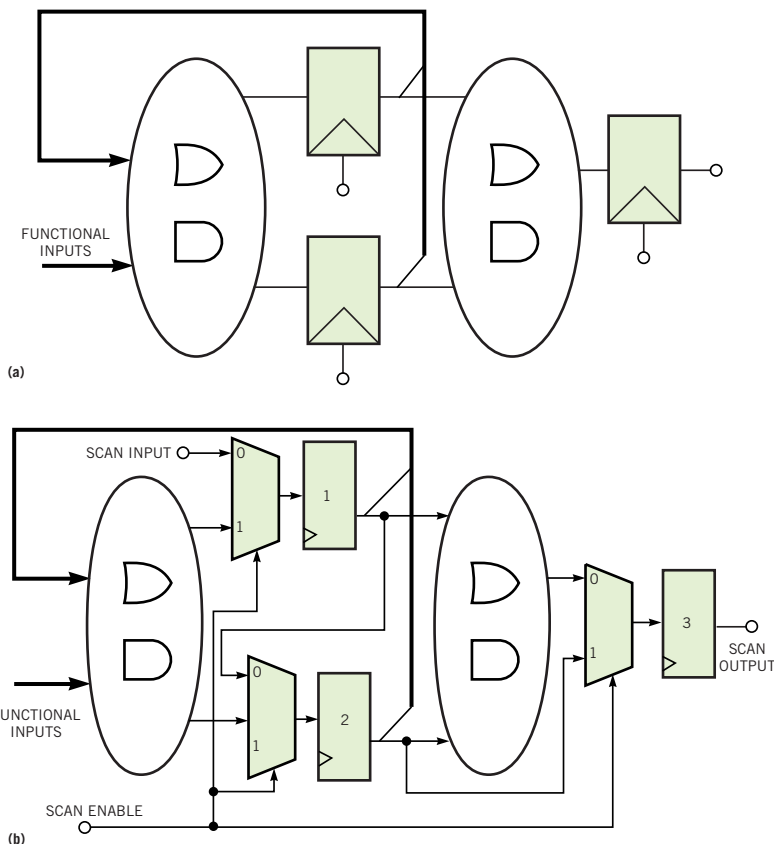
Figure 1



For a simple circuit, you can manually create test vectors to verify that each node is not stuck high or low. However, this approach isn't feasible for more complicated designs.

plexity of today's SOC designs, these production-test techniques are quickly running out of steam. You can

Figure 2



For an example design that contains three flip-flops and some combinational logic, which could represent a simple state machine with a registered output (a), adding scan testing requires just three additional pins (b).

WHAT IS SCAN?

The goal of any production-testing scheme is to verify that the chip is manufactured correctly. Consider the simple circuit in **Figure 1**. If you want to verify that node A is not stuck at 0, due to a manufacturing flaw that shorts this node to ground, you can create the vector (A=1, B=0, C=0). Setting B and C low allows modifications of A to directly control the output at D. If A is stuck at 0, then D will be 0, no matter what value drives A. Similarly, you can create other vectors to verify that each node is not stuck high or low. You can manually create these simple test vectors with little effort. However, if the design is complicated and contains thousands of flip-flops and hundreds of thousands of combinational logic gates, manually creating these test vectors is laborious.

The use of scan-design techniques allows you to use all of the flip-flops in a design as a big shift register during scan testing. Thus, you can shift patterns into the chip, for example to drive the inputs of **Figure 1**'s circuit with A=1, B=0, and C=0. The flip-flops capture the functional data that results from the test pattern, for example to capture the value at D, which is 1, and you can shift out the results. This internal scan increases the controllability and observability of internal nodes by connecting storage cells into a long shift register, or scan chain, and by enhancing the logic of these cells to support a scan-shift mode that allows for serial loading and unloading of scan chain's contents.

In normal operational mode, the scan chain does not affect system operation. When you select scan mode, however, the

outputs of each storage cell in the scan design become primary inputs to the combinational logic, which increases controllability. The inputs of each scan storage cell allow registering of the outputs of the combinational logic, which increases observability. ATPG tools are proficient at generating test patterns to provide high fault coverage for combinational logic. Scan allows the tools to have easy access to all the combinational logic in the design.

Figure 2a shows a simple circuit without any scan circuitry. The circuit contains three flip-flops and some combinational logic. This circuit could represent a simple state machine with a registered output. You add scan to this design to create the design in **Figure 2b**. You need to add only three additional pins, or pads, for scan testing: a scan-input pin for se-

serial data input, a scan-output pin for serial data output, and a scan-enable pin for scan-mode control. This design shares the scan clock with the system clock. You can often use multiplexers to combine these pads with system-operation pads, which reduces the I/O overhead. This example forms a serial scan chain from the scan-input pad, connecting each flip-flop into a scan register that is 3 bits long. The output of the final flip-flop connects to the scan-output pad. Each of the flip-flops in **Figure 2b** is now a flip-flop with a multiplexed input. The scan-enable signal selects between the normal functional-data input, which comes from the combinational-logic clouds, and the scan data, which comes from the scan input or the previous flip-flop. The scan-chain ordering is from flip-flop 1 to flip-flop 3.

Figure 3 illustrates the timing for scan testing of **Figure 2b**'s circuit. The timing sequence consists of three stages: first scan mode, then normal system mode, then scan mode again. Scan enable=1 selects the scan mode, during which data serially loads into the scan chain from the scan-input signal. When the scan chain is fully loaded, which requires one scan clock for each storage cell in the scan chain, scan enable=0 selects the normal system mode. In this mode, the system applies one system clock, applies data at the primary inputs of the chip, and observes data at the primary outputs of the chip. This procedure captures data from the combinational-logic elements of the design into the scan-storage cells. The system asserts and deasserts the scan-enable signal on the falling edge of the clock, which helps ease timing, especially the hold-time constraints. In the last and third part of the sequence, the system again selects scan mode with scan enable=1 and uses the scan clock to unload the scan chain through the scan output. The tester then checks this output data against expected values. While captured data shifts out of the scan chain, the system can load input data from the next scan-test pattern into the scan chain.

Figure 4 shows a circuit with two clock domains and with the scan circuit in place. The upper portion of the figure shows clock domain 1 using clk1 and consists of two flip-flops and some com-

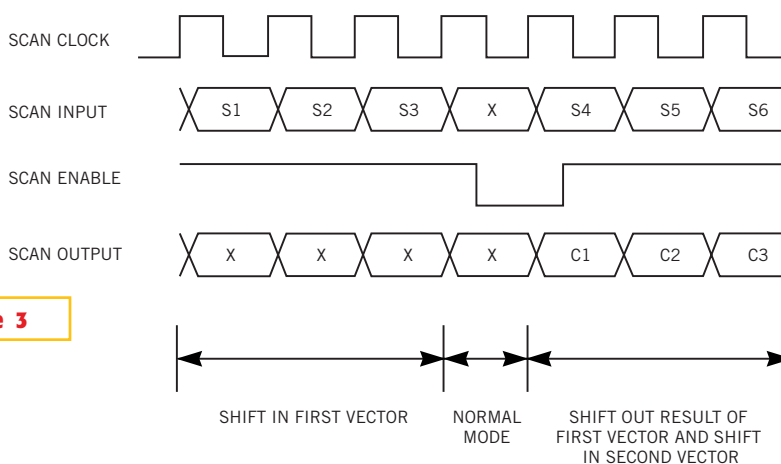


Figure 3

NOTES:

S1, S2, S3=SCAN DATA FOR THE FIRST TEST VECTOR.

S4, S5, S6=SCAN DATA FOR THE SECOND TEST VECTOR.

C1, C2, C3=CAPTURE DATA FROM THE FIRST TEST VECTOR.

A scan sequence involves three stages, which the scan-enable input controls.

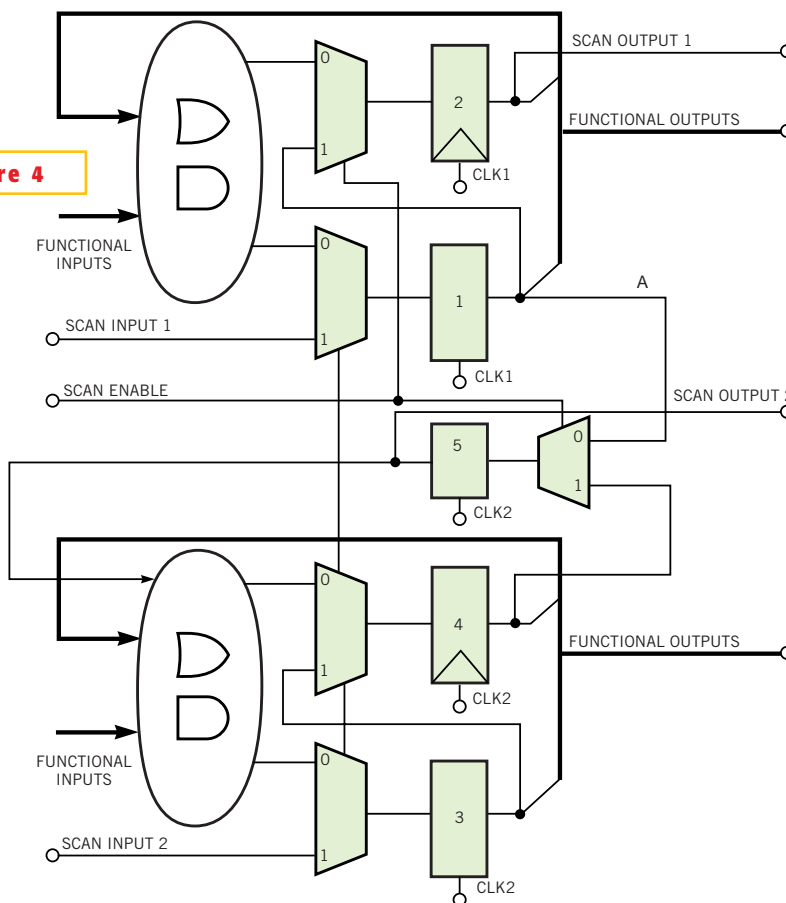
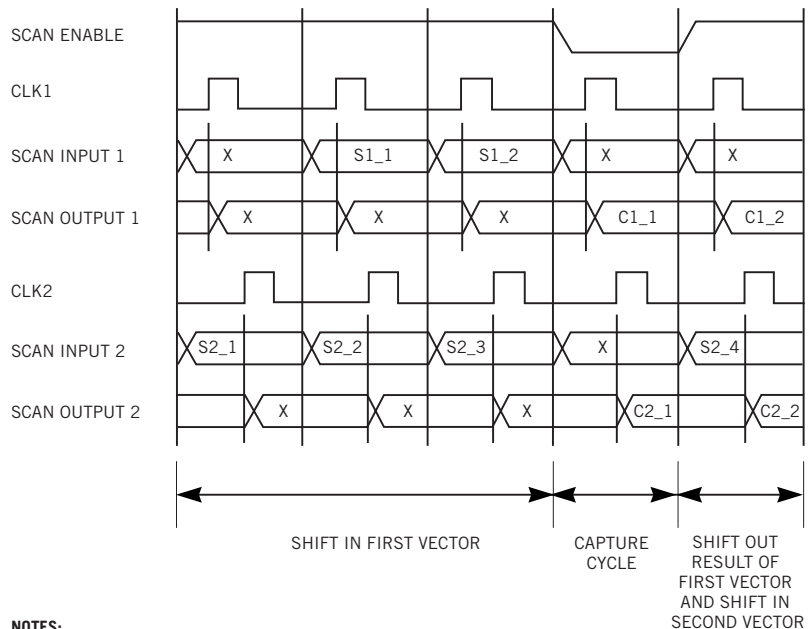


Figure 4

When a circuit uses two clock domains, the scan design involves two scan chains, one for each clock domain.



NOTES:

S2_1, S2_2, S2_3=SCAN CHAIN 2 DATA FOR THE FIRST TEST VECTOR.
 S1_1, S1_2=SCAN CHAIN 1 DATA FOR THE FIRST TEST VECTOR.
 C2_1, C2_2, C2_3=SCAN CHAIN 2 CAPTURE DATA FOR THE FIRST TEST VECTOR.
 C1_1, C1_2=SCAN CHAIN 1 CAPTURE DATA FOR THE FIRST TEST VECTOR.
 S2_4, S2_5, S2_6=SCAN CHAIN 2 DATA FOR THE SECOND TEST VECTOR.
 S1_3, S1_4=SCAN CHAIN 1 DATA FOR THE SECOND TEST VECTOR.

Figure 5

As in Figure 3, the scan operation for multiple scan chains involves three stages. In this case, however, the capture cycle uses staggered clocks for the two clock domains.

binational logic. The bottom portion of the figure shows clock domain 2 using clk2; it consists of three flip-flops and some combinational logic. You can think of the circuit as two state machines with a shared communication signal, A, that flip-flop 5 synchronizes to clk2. This example includes two scan chains, one for each clock domain. The first scan chain starts with scan input 1 going through flip-flops 1 and 2 to scan output 1. The second scan chain starts with scan input 2 going through flip-flops 3, 4, and 5 to scan output 2. The circuit has only one scan-enable signal, even though there are two scan chains. Because the two scan chains interact—via the functional path from flip-flop 1 to flip-flop 5—you have to be careful when you assert the clocks during the capture cycle, or you could end up with a hold-time violation at flip-flop 5.

Figure 5 illustrates the timing for scan testing of this circuit. Similar to the timing diagram in Figure 3, this sequence has three stages. During scan mode,

when scan enable=1, data loads serially into the scan chains from scan input 1 and scan input 2. Each of the two scan chains loads in parallel. The length of the longest scan chain determines the length of the scan-shift operation. Scan chain 2 is three flip-flops long. Therefore, the scan-shift operation takes three clocks. Scan chain 1 is only two flip-flops long. However, you can still shift three data values into this chain as long as the first value is a “don’t care.”

As in Figure 3, when the scan chains are loaded, scan enable=0 selects normal mode: The system applies one clock, applies data at the primary inputs of the chip, and observes the data at the primary outputs of the chip. As before, this action captures data from the combination-logic elements of the design into the scan-storage cells. However, the capture cycle in Figure 5 differs from the cycle in Figure 3, which has only one clock domain. Because two clock domains exist and interact, the capture cycle must stagger the assertion of the clocks. Stagger-

ing the clocks prevents any timing problems with the data crossing the clock domains. The capture cycle first clocks data into scan chain 1, at the first clock after scan enable goes low, to capture data into the clk1-based flops. The capture cycle then clocks data into scan chain 2, at the second clock after scan enable goes low, to capture data into the clk2 based flops. After capture data latches into the flip-flops on scan chain 1, the functional input to flip-flop 5 of the second scan chain will change. Only sequential-based ATPG tools can handle this situation. Purely combinational-based ATPG tools cannot. If you're using a combinational ATPG tool, you should tell the tool to assert only one of the clocks during the capture cycle. The tool then asserts clk1 or clk2 only during the capture cycle, which results in a higher number of patterns for the same level of fault coverage.

Finally, the cycle again selects scan mode, and the system uses the scan clocks to unload the scan chains through the scan-output pins. While capture data shifts out of the scan chains, input data from the next scan-test pattern can begin loading. Note that **Figure 5** shows the first cycle of the shift.

SCAN TECHNIQUES AND ELEMENTS

All of the examples so far use flip-flops with multiplexed inputs for the scan-storage elements. These multiplexed flip-flops are only one type of scan-storage element. The other types of scan elements are clocked scan elements and level-sensitive-scan-design (LSSD) elements. Each type of scan element provides its own benefits. Multiplexed flip-flop and clocked-scan techniques are better suited for designs that contain edge-triggered flip-flops. LSSD techniques are better suited for latch-based designs. The type of scan element you decide to use depends on your design and your ASIC vendor. The examples in this article use the multiplexed flip-flop technique.

A multiplexed flip-flop scan element contains a single D-type flip-flop with multiplexed inputs that allows selection of either normal functional data or scan-input data. **Figure 6a** shows a multiplexed flip-flop scan element. In normal mode (scan enable=0), the system data, or functional input, goes through to the

flip-flop, which registers the data. In scan mode (scan enable=1), scan data goes through to the flip-flop so that the flip-flop registers the scan data.

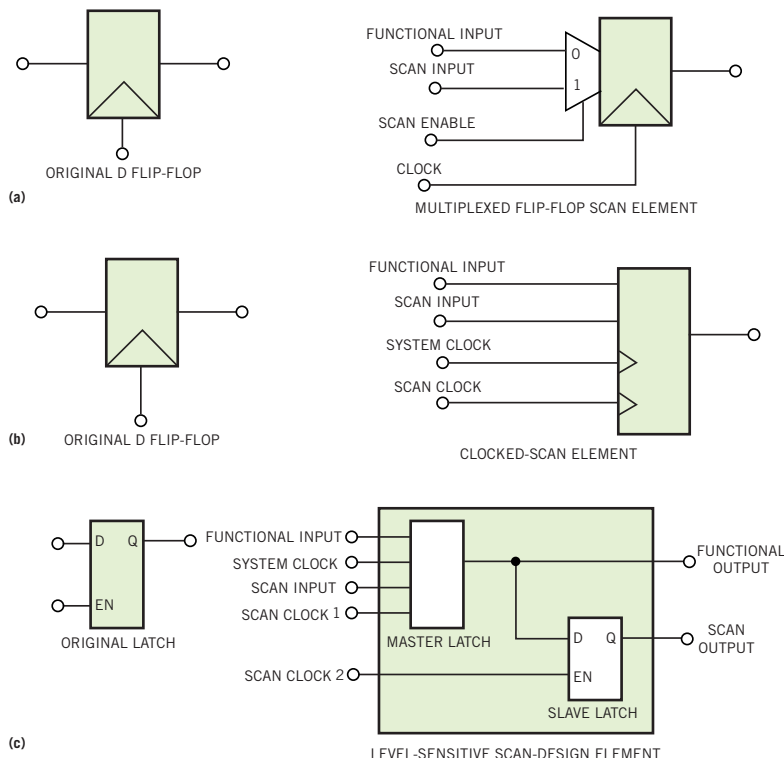
Figure 6

Clocked-scan elements are similar to multiplexed flip-flop elements but use a dedicated test clock to register scan data into the flip-flop (**Figure 6b**). During normal operation, the system clock registers the system data at the functional input into the flip-flop. During scan mode, the scan clock registers the scan data into the flip-flop.

LSSD uses three independent clocks to capture data into the two latches within the scan cell (**Figure 6c**). During normal mode, the master latch uses the system clock to latch system data at the functional input and to output this data to the normal functional-data output path. During scan mode, the two scan clocks control the latching of data through the master and slave latches to generate the data at the scan output.

LOCKUP LATCHES

Scan chains are vulnerable to clock-skew problems for two main reasons. The first reason has to do with layout and propagation delay. The same clock may drive hundreds or thousands of scan-storage cells with no circuitry between them. Logically adjacent storage cells in the scan chain may be physically separated in the layout. Clock skew between



Three types of scan-storage elements exist: multiplexed flip-flop elements (a), clocked-scan elements (b), and level-sensitive-scan-design (LSSD) elements (c).

GLOSSARY

Automatic test-pattern

generation. (ATPG)—the process of generating scan-based production-test patterns automatically via a CAD tool.

Capture cycle—the clock cycle during scan mode in which the system switches the scan flip-flop multiplexed inputs to select the normal functional inputs rather than the scan inputs. At this point, the scan flip-flops “capture” functional data after a scan pattern has shifted into the device under test.

Combinational versus sequential ATPG—applies to how the ATPG tool handles scan data during capture cycles. This concept is important if you have multiple

clock domains and the logic between domains interacts during capture cycles. When these conditions exist, you normally stagger your clocks for pattern generation. For example, if you have two clock domains, clk1 and clk2, you assert clk1 at time 200 and deassert it at time 250 while asserting clk2 at time 300 and deasserting it at time 350. This staggering of clocks helps to avoid any hold-time violations between clock domains.

Combinational scan tools cannot handle this situation. They assume that capture data from one clock domain does not affect the capture data of another chain. If your ATPG tool does not support

sequential scan and your design has interaction between the clock domains, then you must tell the tool to generate only one scan clock at a time during capture cycles. This constraint makes the tool’s job more difficult because it has to generate more patterns to get the same level of fault coverage as if the tool had been able to assert multiple clocks during capture cycles.

Sequential scan tools can handle this situation. The tool knows that data from clk1 changes during the capture cycle and that, in turn, changes the capture data of some flip-flops in the clk2 scan chain.

Fault grading—the process of de-

termining the percentage of manufacturing faults a set of test patterns can detect within a chip. Production testing—the process of verifying the correct manufacture of a chip. To accomplish this testing, you usually create a set of test vectors to run on a tester that tests packaged parts.

Sequential ATPG—See combinational versus sequential ATPG.

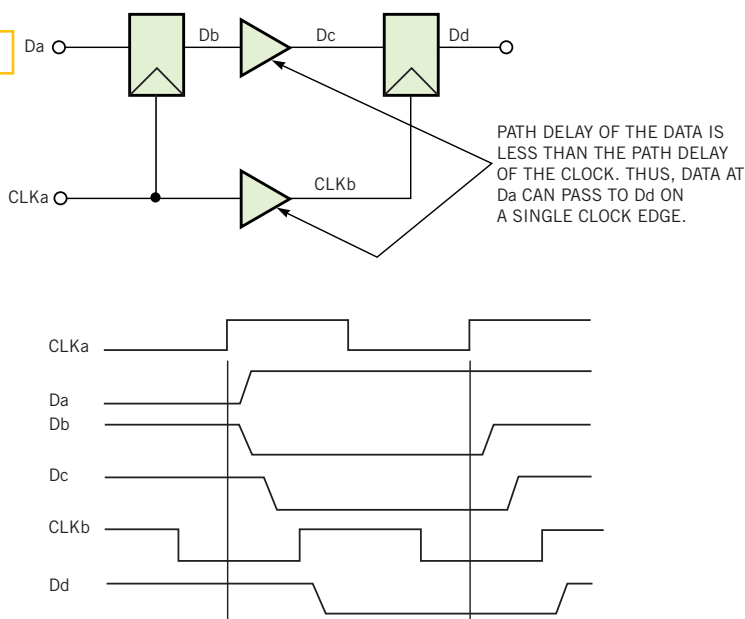
Shift mode—the clock cycles during scan mode in which the scan flip-flop’s multiplexed inputs switch to select the scan inputs rather than the normal functional inputs. This mode allows you to shift in the current test pattern prior to performing a capture cycle.

successive scan-storage cells must be less than the propagation delay between the scan output of the first storage cell and the scan input of the next storage cell. Otherwise, data slippage may occur. Thus, data that latches into the first scan cell also latches into the second scan cell. This situation results in an error because the second scan cell should latch the first scan cell's "old" data rather than its "new" data. **Figure 7** demonstrates that the path delay for the data is less than that of the clock. Thus, the "new" data at Da passes all the way through to Dd in one clock period. The second flip-flop should have latched the "old" value at Dc (a logic high) rather than the "new" value.

The second reason for clock skew is that clock domains separate the scan chains. For example, all the flip-flops from the clk1 clock domain in **Figure 4** are linked in the same scan chain. Likewise all the flip-flops from the clk2 clock domain form a second scan chain. If you want to link these two scan chains to form a single scan chain, timing problems could result. Two clock trees generate the two clocks, which introduces some amount of skew between the two clocks. You cannot link the two scan chains unless you handle this clock-skew problem. The timing for this scenario would be very much the same as that in **Figure 7** except there would be two separate clocks rather than a single clock.

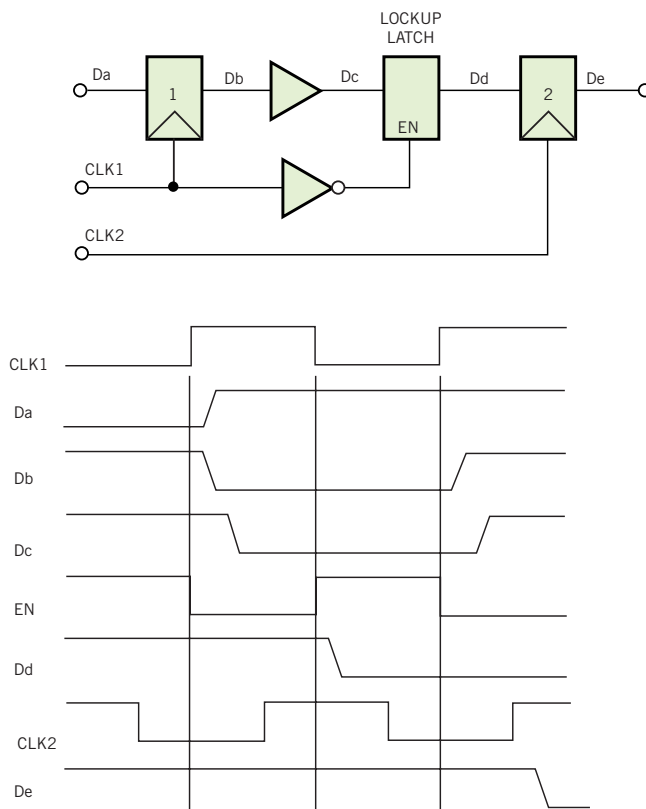
Lockup latches are nothing more than transparent latches. You use them to connect two scan-storage elements in a scan chain in which excessive clock skew exists. **Figure 8** illustrates the use of lockup latches. The circuit contains two flip-flops. Flip-flop 1 represents the end of the scan chain that contains only elements that are in the clk1 clock domain. Flip-flop 2 represents the beginning of the scan chain that contains only elements that are in the clk2 clock domain. Although the figure doesn't show it, these flip-flops have multiplexed inputs. The inputs of these flip-flops represent the scan inputs of the multiplexers. The latch has an active-high enable, which becomes transparent only when clk1 goes low and effectively adds a half clock of hold time to the output of flip-flop 1. In this figure, you assume that the system synchronously asserts clk1 and clk2,

Figure 7



Clock skew can cause data slippage. In this case, because the "new" data at Da passes all the way through to Dd in one clock period, the second flip-flop incorrectly latches this "new" value instead of the correct "old" value at Dc.

Figure 8



To reduce clock-skew problems, you can use lockup latches to connect two scan-storage elements in a scan chain. In this case, the latch effectively adds a half-clock of hold time to the output of flip-flop 1.

which would be the normal case during scan-mode operation. Although the clocks synchronously assert, some amount of clock skew between them still exists because they come from different clock trees.

Figure 8 shows the use of lockup latches to connect scan chains from different clock domains. However, you can just as easily use these latches to connect scan chains from various blocks within a chip that, although on the same scan chain, are physically remote from each other on the die. You want to make the latch transparent during the inactive part of the clock. For example, both flip-flops in **Figure 8** trigger on the rising edge of the clock. Therefore, you want to make the lockup latch transparent during the low period of the clock. If the flip-flops trigger on the falling edge of the clock, you want the latch to be transparent when the clock is high. □

REFERENCES

1. Scan Synthesis User Guide, Synopsys, www.synopsys.com.
2. ASIC/IC Design-for-Test Process Guide, Mentor Graphics, www.mentorgraphics.com.

AUTHORS' BIOGRAPHIES

Ken Jaramillo is a principal engineer at Philips Semiconductors, where he has worked for four years. He has worked as designer and architect of ASICs, FPGAs, and boards for products including avionics, high-speed serial buses, high-performance gaming platforms, PCI-bus-related products, high-performance PC audio, and high-speed networking products. He has a BSEE from the University of Missouri (Kansas City) and a BSCE from the University of Missouri (Columbia).

Subbu Meiyappan is a senior design engineer at VLSI Technology, a subsidiary of Philips Semiconductors. He has worked for the company for three years, designing, developing, synthesizing, simulating, and validating high-performance intellectual-property blocks for PCI, ARM-ASB-based devices, and high-performance ASICs. He has a BE from Annamalai University (Annamalai Nagar, India) and an MS from Tennessee Technological University (Cookeville, TN).