Verification engineers can aid test

Paul Taubman - July 01, 2002

To test complex devices, test engineers must rely on the vector sets generated by verification engineers. Unfortunately, verification engineers—who work in a software simulation environment—often have little understanding of the characterization and production test functions that test engineers perform on real silicon. Cooperation between test and verification engineers is critical to the debug and production release of the physical product. Fostering communication with verification engineers can enhance both the quality and flow of the vector generation process.

The term "verification" is used at several steps within the design flow (**Table 1**). In this article, I will focus on functional verification, which is the process of simulating the design to ensure that the logic adheres to the specification. It is the functional verification simulation that is converted to tester-specific vectors.

ATE program development relies upon cycle-based testing. Verification engineers may not be familiar with the term "cycle-based test," but they often employ a similar concept when simulating and verifying the functionality of a design. Knowing this, you can explain cycle-based testing in terms and methods that the verification engineer uses each day.

Verification engineers generally use waveforms during design simulation to debug an ASIC's functionality. As a convenience, they add the vertical grid lines to the waveform to make it easier to determine when a logic transition has occurred. In most cases, an engineer will set the grid lines based on the clock being used in the system. Such grid lines most likely will be set on each edge of the clock cycle or upon the active edge of each clock period. In **Figure 1**, the grid lines were set on each edge of the clock.

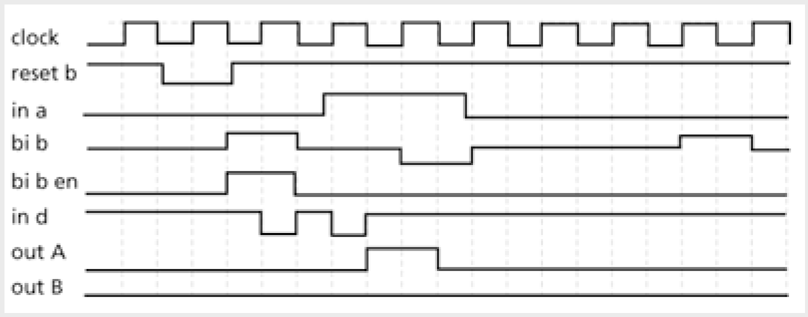
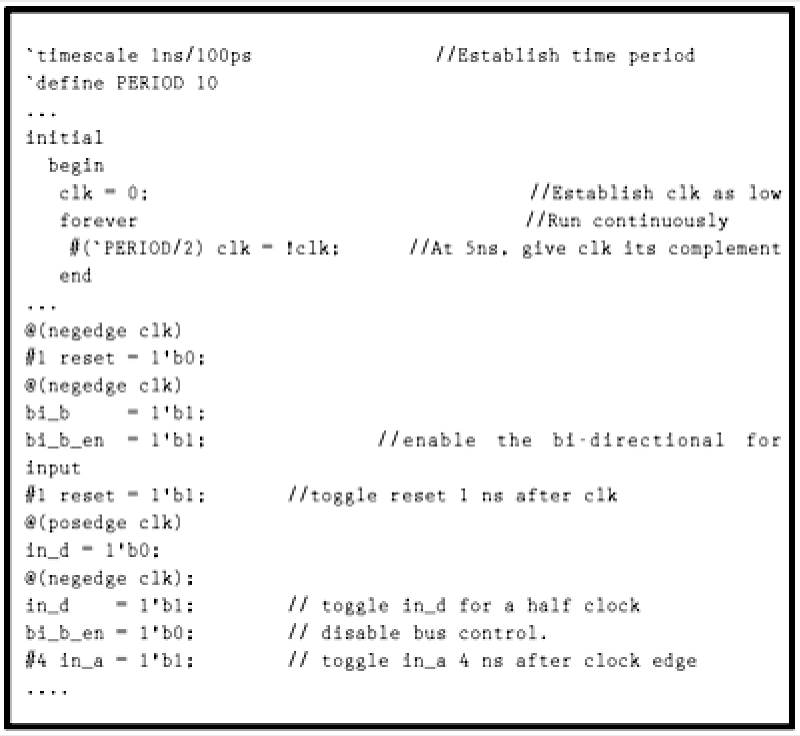


Figure 1. Verification engineers make extensive use of waveforms and employ vertical grid lines to help pinpoint where transitions occur.

The method that verification engineers use to create the grid in their waveforms can be the basis for their understanding of cycle-based timing but with a simple exception. The grid is no longer a viewing aide, but rather the boundaries of the cycle.

With this foundation, the general concepts of cycle-based timing fall into place. The size of the time cycle is usually based upon the system clock of the device under test (DUT). Cycle size is tailored to accommodate the amount of activity on the individual pins. Optimally, the cycle size is selected in such a way that a non-clock input pin on a DUT does not switch more than once during any given cycle. The waveforms of Figure 1 would require a cycle of one-half clock period, because the signal in_d has two transitions in a single clock period. All cycles must be of equal length. Each cycle translates into a single vector: a collection of signals and their values at a specific point in time.

For cycle-based timing to function properly, the test and verification engineers may need to establish rules regarding when a signal transition may occur. During software simulation, a verification engineer can easily toggle a signal at any time without regard to such rules (**Figure 2**), and the test engineer may need to explain that the pin electronics on ATE can impose electronic and physical limitations on signal transitions.



```
`timescale 1ns/100ps               //Establish time period
`define PERIOD 10
...
initial
  begin
   clk = 0:                                //Establish clk as low
   forever                                 //Run continuously
    #(`PERIOD/2) clk = !clk:      //At 5ns, give clk its complement
   end
...
@(negedge clk)
#1 reset = 1'b0:
@(negedge clk)
bi_b     = 1'b1:
bi_b_en  = 1'b1:               //enable  the  bi-directional  for
input
#1 reset = 1'b1:       //toggle reset 1 ns after clk
@(posedge clk)
in_d = 1'b0:
@(negedge clk):
in_d    = 1'b1:        // toggle in_d for a half clock
bi_b_en = 1'b0:        // disable bus control.
#4 in_a = 1'b1:        // toggle in_a 4 ns after clock edge
....
```

Figure 2. Sample Verilog code may represent the input signals shown in Figure 1. Verification engineers are not limited or required by their simulation software to stay within any timing bounds. The simulator will give them what they want when they want it. They can add delay or toggle signals at any point in the simulation.

The analogy of a machine gun works well to explain ATE limitations. While one bullet is being fired, another bullet is being loaded. The gun will jam or misfire if the period of time between loading and firing is inconsistent. The pin electronics of an ATE system work in a similar fashion. While a vector is being fired into the DUT, the pin electronics are already preparing for the next vector. If the appropriate amount of the time is not given, the pin electronics may not give the DUT the proper signal.

The pin electronics for a single input signal need to know at what point during a cycle they should toggle the pin, if it needs to be toggled. If the vector set calls for the same pin on the DUT to be toggled at different points in different cycles, more timing resources will need to be allocated to the input pin, usually at the cost of using the pin electronics of an unused pin. Though this may not sound drastic to someone not in contact with ATE, it is a critical point that the verification engineer must realize. Furthermore, two ATE systems with the same model number can be configured differently. What may have worked on the ATE used to test a prototype device may not be available on the ATE used at the manufacturing plant.

Waveform examples

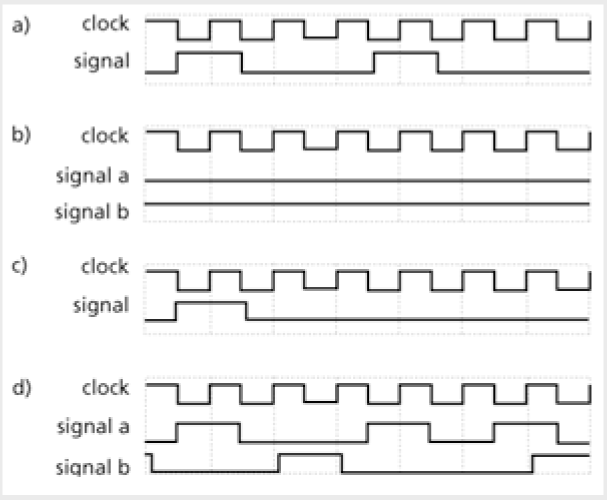


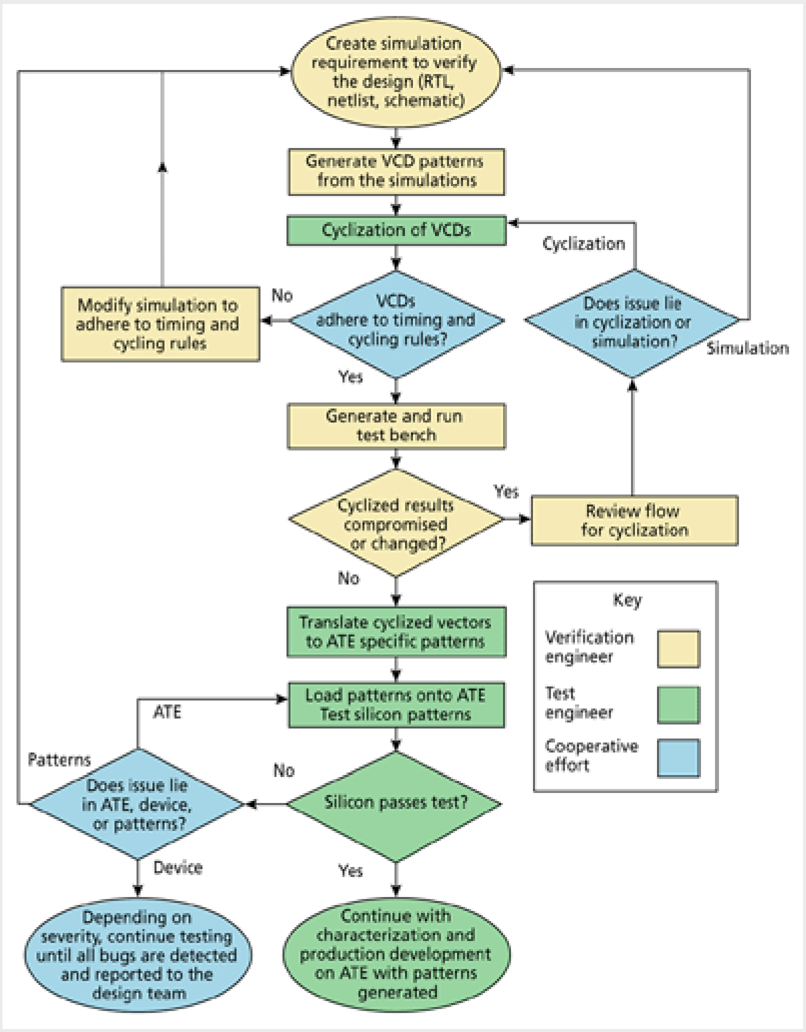
Figure 3. Sample waveforms demonstrate the required ATE timing resources. a) and c) require two timing resources, whereas b) and d) don't introduce timing issues.

Examples clarify a number of questions regarding the time in a cycle at which the signal is allowed to switch. For the examples in **Figure 3**, assume the cycle will be based on the positive rising edge of the clock, which also happens to be the active edge. In Figure 3a, the signal would require two timing resources. The first two transition points occur on the falling edge of the clock. The third and fourth transition points occur 1 ns after the falling edge. In Figure 3b, signals a and b are not toggled in the vector pattern. It is acceptable for a set of vectors to include signals that do not toggle.

In Figure 3c, the signal rises on the falling edge of the clock and falls 1 ns after the falling edge. Though there are only two transitions, two timing resources would be needed to handle the rise and the fall of the signal. In Figure 3d, there are no timing issues. The point at which signal a and b switch within their respective cycles is consistent. Remember, one timing resource has been allocated to each pin. Signals a and b have mutually exclusive timing resources. If the signal must be toggled, it should toggle at the same relative point in time within any given cycles—for instance, signal a will always switch on the falling edge of the clock if it needs to be switched, and signal b will switch 1 ns after the positive rising edge of the clock if it needs to be switched.

From VCD to cyclized test patterns

One of the issues that strains the test-verification relationship is the time gap that exists between the completion of functional verification and the arrival of silicon. This time gap is normally measured in months. To ensure translatable vectors, test needs to be involved with verification during the generation of VCDs (vector change dumps). Test can provide immediate feedback and can correct any discrepancies that may prevent the cyclization of a VCD pattern. If left until silicon arrives, this transition exercise becomes extremely difficult (**Figure 4**). The VCD is an ASCII file that contains header information, defines variables, and lists the value changes for all variables. You will use the VCD when creating test patterns, so a high-quality VCD will go a long way toward ensuring the quality of your tests.



Create simulation requirement to verify the design (RTL, netlist, schematic)
Generate VCD patterns from the simulations
Cyclization of VCDs
Cyclization
Modify simulation to adhere to timing and cycling rules
No
VCDs adhere to timing and cycling rules?
Does issue lie in cyclization or simulation?
Simulation
Yes
Generate and run test bench
Cyclized results compromised or changed?
Yes
Review flow for cyclization
No
Translate cyclized vectors to ATE specific patterns
Key
Verification engineer
Test engineer
Cooperative effort
ATE
Load patterns onto ATE Test silicon patterns
Patterns
Does issue lie in ATE, device, or patterns?
No
Silicon passes test?
Device
Yes
Depending on severity, continue testing until all bugs are detected and reported to the design team
Continue with characterization and production development on ATE with patterns generated

Figure 4. The test engineer and verification engineer each have dedicated areas of responsibility in the design flow. Cooperation on key tasks can ease the jobs of both.

Once you receive the VCDs, you need to translate them into loadable test patterns. The methods used by engineering firms to create and translate VCDs differ. Some companies use proprietary tools for the task, while others use commercial tools. In addition, the model and make of the ATE will play a role.

Cyclization is the first process during which you must examine timing issues of the VCD to determine whether a signal may have multiple timing assignments and to establish cycle boundaries for the ATE. If the VCD contains incorrect timing information, you need to work with the verification engineer to correct them. Most commercial tools allow the test engineer to correct timing discrepancies, but avoid doing this. If you enter changes into the cyclization tool manually, your changes might be lost for subsequent cyclization runs.

If timing problems exist, you need to provide feedback to the verification engineer. Identify the signals that are violating the rules, the VCDs where the problems are located, and the point in the simulation where the discrepancy occurs. This information will expedite the corrections.

Cyclization yields two end products. One is an ASCII formatted file, like the standard WGL, that can be translated into ATE-specific binary patterns. The other is a test bench in an RTL-type language like Verilog or VHDL. These files contain a number of wave sets that the verification engineer can view to ensure that the cyclization of the original VCD to a WGL format has not lost, changed, or added information.

Though cyclization may flush timing issues out of the VCDs, it can also introduce other errors: It can alter the states of bidirectional buffers; it can affect the window strobes the ATE uses to check the output results of the device; and it can result in missing or delayed waveforms. Because it is a waste of time to debug vectors on the ATE, the verification engineer should review the vectors. If the test bench has problems, the corresponding WGL will have problems that will become apparent during test efforts.

Once the load board is in place and the vectors have been loaded, you can begin the debug and setup efforts. At this stage, many verification engineers may believe their purpose has been fulfilled, so they move on to other projects. Yet, the verification engineer must still play a critical role in evaluating the failure data and deciding whether to proceed with characterization or re-spin.

Other points to communicate

There are several steps you can take to ensure that you receive useful VCDs:

- Be part of the design team. Your involvement during the early stages of the project can help prevent some fundamental problems. Because many members of the team will likely deal only with software, you may need to explain the limitations of the test hardware. For example, you may need to explain that your company's 200-pin tester can't handle the planned 400-pin device.
  Another reason test engineers should get involved at the beginning of the project is to prevent duplication of efforts. If multiple verification engineers set off to simulate the device using different timing delays in the cycle simulation, a number of problems may arise. These include the inability to combine the vector sets from multiple verification engineers, the unnecessary use of ATE resources, and exceeding the tester resources available on the ATE.
- Aim to create a single test program comprising the VCD patterns. Loading vector patterns into ATE memory can take several minutes. It is not economical for a test engineer or a production facility to load and test each test pattern separately. The VCD supplied by the verification engineer should represent every I/O pin on the DUT, regardless of usage.
- To help smooth the cyclization process, encourage verification engineers to avoid the use of metacharacters (!@#$%^&*()_+|~-=\`, etc.) in naming device pins—what may work in the creation of Verilog, C/C++, and VHDL may not work in the tools used to cyclize or translate vectors.
- Encourage verification engineers to develop many smaller VCD patterns rather than one large pattern. Debugging large VCD files during cyclization is extremely difficult. If the verification engineer is writing custom code to simulate the device, there is a fair chance he or she may accidentally add extra delay to a signal that will propagate into the VCD file.
  Short tests with specific purposes will help during device debug on the ATE. It is easier for a verification engineer to interpret a specific functional fail than to receive an approximation of the vector fail in a large all-purpose vector set. Neither test nor verification engineers want to dig out a problem that occurred on vector 634,678 in a functional pattern.
  Another advantage to smaller test pattern sets is that you can create a Pareto chart of the failures to determine yield; you also can use this Pareto chart to determine which vector sets are capturing failures. By eliminating vectors that do not capture failures, you reduce the overall test time.
  Test economics tell a convincing story about the need for short tests. Most production facilities charge about $350/hr for the use of their testers, so each second costs $0.097. For a volume of 1 million chips, each second of test time per chip costs $97,000.
- Build universal vectors. It's important that the verification engineers create universal VCDs if the ATE used for debug and characterization will differ from the ATE used for production. The translator used to convert the VCDs to test vectors may impose its own restrictions. Many companies still use custom translators that may not have been updated to the capabilities of the

newer test systems. Software flow may play a part, especially if a double translation is required, such as VCD to WGL to a specific make and model of an ATE system.

Adhering to the restrictions imposed by your hardware and software leads to quality vectors. Even if your ATE system can handle any waveform thrown at it, you should not get sloppy when generating vectors.

If your project team follows the preceding recommendations, you may be able to take advantage of various compression methods to quickly load vectors into memory, and you'll have the flexibility to manipulate the patterns for characterization. For example, it is easier to expand and compress an existing test pattern set than to create individual VCDs for different speeds.

The verification engineer may worry about the limitations he or she faces—such as the number of licenses available for the simulation—but is probably not aware of the limitations you face: for example, that only one test engineer can use an ATE system at a given time. These facts need to be communicated. Test time needs to be spent debugging the device, not dealing with vector issues. The verification engineer should know the tester schedule for the device to ensure that any vector changes are delivered with ample time for you to translate them and verify that the vectors are good for loading onto the test system.

There are hundreds of reasons why a device might fail on the ATE. Working together to understand the problem accelerates the debug effort, and it becomes a team effort. If the test facility is on site, invite the verification engineer to the test floor to see the ATE in action. The verification engineer should understand the fundamental test capability of the ATE systems. During characterization or debug, the verification engineer should be able to explain what is expected of the device. The verification engineer can also use the simulations and test bench to create simple tests that may aid in the debug or characterization process. Taking advantage of the verification engineer's skills starts with education about the test process.

Table 1. The roles of verification

Functional verification	Test of the logic and functionality against the specification. Usually performed under ideal clock conditions. Requires a test bench and vectors.
Timing verification	Performed during and after the place-and-route process to ensure the placement has not created excessive delay or race conditions. This is a net-t--net evaluation based upon desired clock speed.
Functional verification,	Same concept as functional verification with the delays of the place-and-route back annotated added to the netlist.
Formal verification	The comparison between the original schematic or RTL description against an equivalent behavioral model or processed result of the schematic or RTL such as a netlist. Does not require a test bench or vectors.
Physical verification	Verification pertaining to the layout, placement, and routing of the device. This includes, but is not limited to, LVS (layout versus schematic,) DRC (design rule checking), and ERC (electrical rule checking).