LV600 User's Manual

Authors:
Alfred Norman Gifford IV
Daniel Khoury

Project Advisor:
Ken Stevens

University of Utah

April 30, 2016

Contents

1 Introduction and Motivation

At the University of Utah, students enrolled in Digital VLSI Design (CS/ECE 5710/6710) work in teams to design an integrated circuit (IC) for their final project. Optionally, students may send their chip designs to MOSIS, an IC foundry, for fabrication. As per MOSIS's rules, students who fabricate their chip designs are required to test their received chips and report their results back to MOSIS.

At least one student from any team that chooses to fabricate their chip must enroll in CS/ECE 6712, which teaches students how to test their chips using a device called an application-specific integrated circuit (ASIC) tester. Presently, the University of Utah has two ASIC testers: a Verigy 93000 and a Tektronix LV500. As of this writing, both machines are in troubled states. The Verigy 93000 is no longer operational due to a hard drive failure. The LV500 has many failed sectors, so the system as a whole may prove unreliable when testing a chip.

Simply purchasing a new ASIC tester is not feasible. ASIC testers are extremely expensive machines (millions to billions of dollars) – partly because of a niche market, but mainly because they are incredibly difficult systems to engineer, construct, and maintain.

Our project, which we term the LV600, is an ASIC tester based largely upon the LV500. The project's intention is to provide an ASIC tester that is good enough for most student-designed VLSI chips. This document serves as both the final report for our project and the user's manual for the system that we hope to ultimately provide to the University.

In this document, we also include ideas for how to build upon our project, and we strongly encourage students to consider building an ASIC tester for their senior project if they have any interest in such machines. The project provides so many learning experiences – it's conceptually simple, but the devil lies in the details.

2 Overview of ASIC Testers

All ASIC testers build upon the same simple interface:

1. The user specifies what outputs to expect from their chip in response to what inputs.

2. The system applies the user-specified inputs to the device under test (DUT) and later samples the outputs of the DUT.

3. The machine reports how the sampled outputs compare to the outputs expected by the user.

Below, we dissect and analyze these three steps both abstractly and via the LV500's implementation. Throughout this section, we define some terminology borrowed from the LV500.

2.1 Specifying Input Vectors and Output Vectors

Terminology:

- **Input vector**: An ordered list of 0s and 1s that correspond to some ordering of a subset of the DUT's input pins.

- **Output vector**: An ordered list of *expected* 0s and 1s that correspond to some ordering of a subset of the DUT's output pins.

As ASIC testers are generally not designed with any specific DUT in mind, much more information accompanying input and output vectors must be provided.

Pin Locations

Any ASIC tester will define "pin locations" that are associated with some kind of socket used to connect a DUT to the tester. Obviously, mapping pins on a DUT to pin locations on an ASIC tester varies per tester. The key idea to keep in mind is that the user does not have to inform the tester of every input and output signal on the DUT. Only the pins on the DUT that the user wishes to test must be mapped to the tester's pin locations. Depending on the tester, any unspecified DUT pins may be left floating or pulled high/low with a resistor.

Inputs and Outputs

Terminology:

- **Pin direction**: A designator that specifies whether a DUT pin is an input or output.
- **Template**: A bit vector that maps DUT pins to pin directions.

In addition to providing the locations of the DUT pins to test, the user must specify the direction of these types of pins using templates. Every input and output vector is associated with some template. For bidirectional pins, different templates can be applied at different phases throughout a test. One template may designate a signal X as an input while a different template may designate that same signal X as an output.

Thus far, we have described (abstractly) how to tell an ASIC tester about the pins on a DUT that we wish to test, which of those pins are inputs/outputs, and the signals that we expect to see in response to the signals that we wish to apply to the input pins. We must answer two other questions for the ASIC tester:

1. When should input signals be applied to the DUT's input pins, and when should signals from the DUT's output pins be sampled?
2. After we apply the input signals to the DUT, what should we do with the input signals before and after the output pins are sampled?

2.2 Test Cycles

One thing we must specify is when input signals to the DUT should be applied and when output signals of the DUT should be sampled.

Terminology:

- **Test cycle**: A window of time wherein an input vector is forced upon the DUT and the output signals of the DUT are (later) sampled.
- **Test cycle length**: The length (in time) of a test cycle.
- **Test cycle delay**: The amount of time between the beginning of the test cycle and when an input vector should be forced upon the DUT.

- **Test cycle width**: The amount of time between when an input vector should be forced upon the DUT and when the output signals of the DUT should be sampled.
- **Test**: The execution of all input vectors specified by the user.

Figure 1 below illustrates how a test cycle can be thought of as just a configurable pulse (though test cycles don't necessarily have to be implemented as an actual pulse). It is important to note that the test cycle itself is NOT sent to the DUT. Test cycles use two edges (a rising edge and falling edge) to signify when to force inputs and sample outputs, respectively.

Terminology:

- **Leading edge**: A rising edge that corresponds to a test cycle's delay.
- **Trailing edge**: A falling edge that corresponds to a test cycle's width.

To get a sense of how test cycles are used, consider Figure 2 below. Assume the following setup as the context of this figure:

- Our DUT contains three pins mapping to a two-input AND gate that we wish to test.
- We have informed the ASIC tester of where these three pins are located, which two of the three pins are inputs, and which of the three pins is the output. (We have also named the input pins A and B and the output pin Y for convenience.)
- We have specified two input vectors: A = 1, B = 1 and A = 0, B = 0, along with two corresponding output vectors: Y = 1 and Y = 0.
- We specified a test cycle length of 100 ns, a test cycle delay of 20 ns, and a test cycle width of 50 ns.

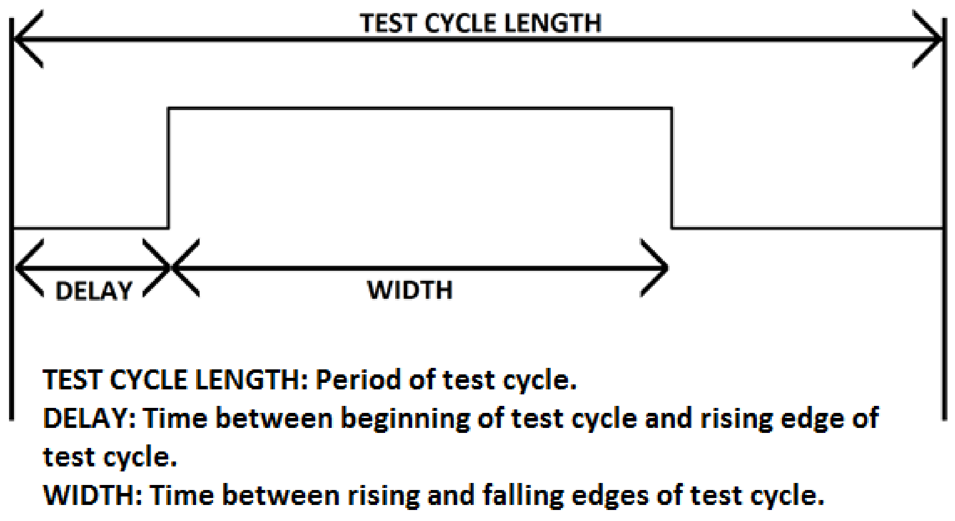
TEST CYCLE LENGTH
DELAY
WIDTH
TEST CYCLE LENGTH: Period of test cycle.
DELAY: Time between beginning of test cycle and rising edge of test cycle.
WIDTH: Time between rising and falling edges of test cycle.

Figure 1: A single test cycle.

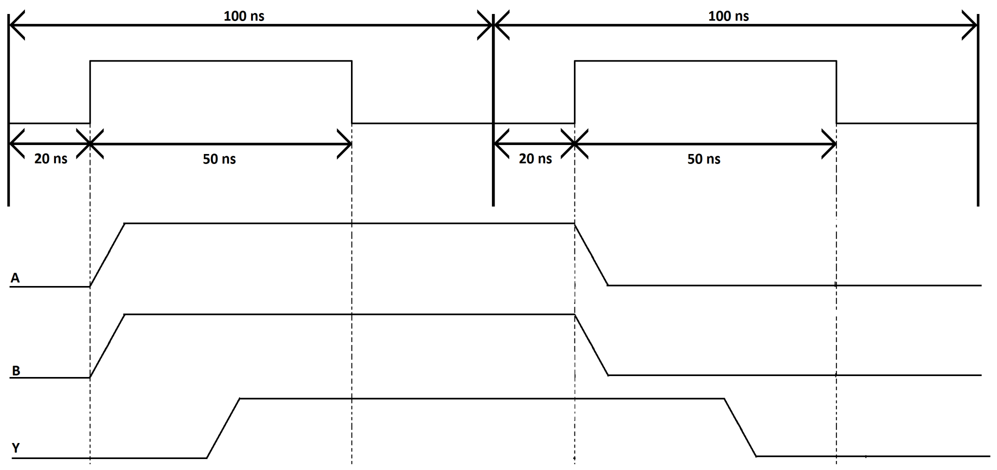

Figure 2: Testing an AND gate with a test cycle configured to have a length of 100 ns, a delay of 20 ns, and a width of 50 ns. Two input vectors (and thus two cycles) are applied to the DUT.

2.3 Force Formats

Note that in Figure 2, the inputs A and B latch onto whatever value they are assigned at every leading edge. It may be desirable to have A and B behave differently, however. For instance, we may want A and B to latch onto whatever value they are assigned at every leading edge, but return to 0 at every trailing edge. For versatile testing, an ASIC tester must also provide (what the LV500 calls) force formats to be associated with every input pin.

Terminology:

- **Force format**: A specification for how an input signal should behave during a test cycle.

An aside: The LV500 uses the term ”force” to refer to applying inputs to DUT pins, hence the phrase ”force format”.

The LV500 defines five natural force-formats:

- **R0**: Inputs latch onto their assigned values on leading edges, but return to 0 on trailing edges. (Return to zero.)
- **R1**: Inputs latch onto their assigned values on leading edges, but return to 1 on trailing edges. (Return to one.)
- **DNRZ L**: Inputs latch onto their assigned values on leading edges. (Delayed non-return-to-zero, leading edge.)
- **DNRZ T**: Inputs latch onto their assigned values on trailing edges. (Delayed non-return-to-zero, trailing edge.)
- **R INH**: Inputs latch onto their assigned values on leading edges, but return to high-Z on trailing edges. (Return to inhibited.)

Figure 3 below is the same setup as Figure 2, but with the R0 force-format used for A and B (as opposed to the DNRZ L force-format). Note that A and B do not have to be assigned the same force-format! Any ASIC tester should allow every individual pin to be associated with its own force-format. In this case, there is nothing invalid about input A being associated force-format R0 and input B being associated force-format R1, for instance.

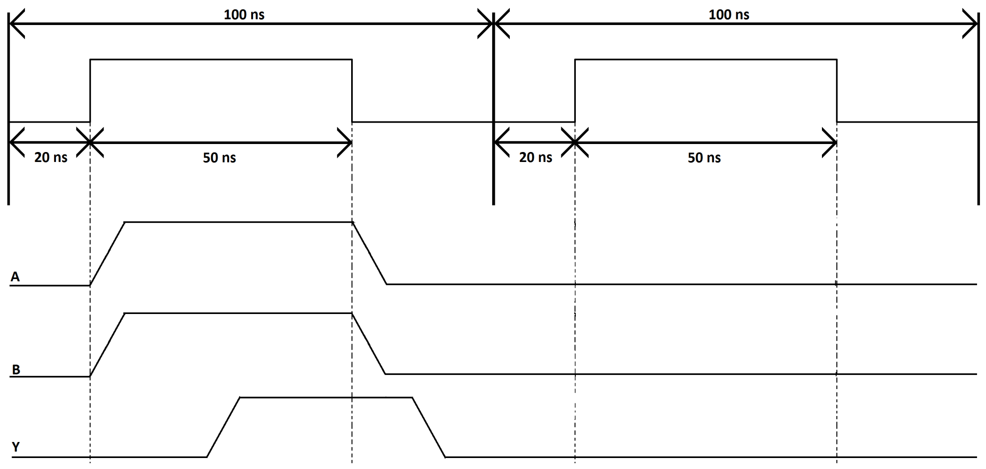

Figure 3: The same configuration as Figure 2 but with a different force format for the inputs.

2.4 Combinational Circuits vs. Sequential Circuits

As shown with Figures 2 and 3, testing a combinational circuit is fairly straightforward. On the other hand, testing a sequential circuit requires more functionality from an ASIC tester. We certainly don't want to transition the clock signal at the same time as the inputs – otherwise, we may violate the setup times of registers within the DUT. A solution to this problem is to allow every pin to be associated with a test cycle. Consider Figure 4 below, which demonstrates testing a D Flip-Flop (DFF). We associate the D input to the DFF with the uppermost test cycle and the CLK input to the DFF with the lowermost test cycle. As seen in the figure, we allow D 20 ns to settle on the input to the DFF before we transition the clock. Note that the output of the DFF, Q, needs to be associated with one of the two test cycles in the figure. Whichever test cycle Q is associated with, we sample Q on the trailing edge of the cycle.

2.5 Electrical and Dynamic Configurations

A DUT may require a variety of different VDDs – 5V, 3.3V, 1.8V, etc., so an ASIC tester needs to provide different VDDs for different chips. More interesting, however, is how a DUT may respond to unideal high and low

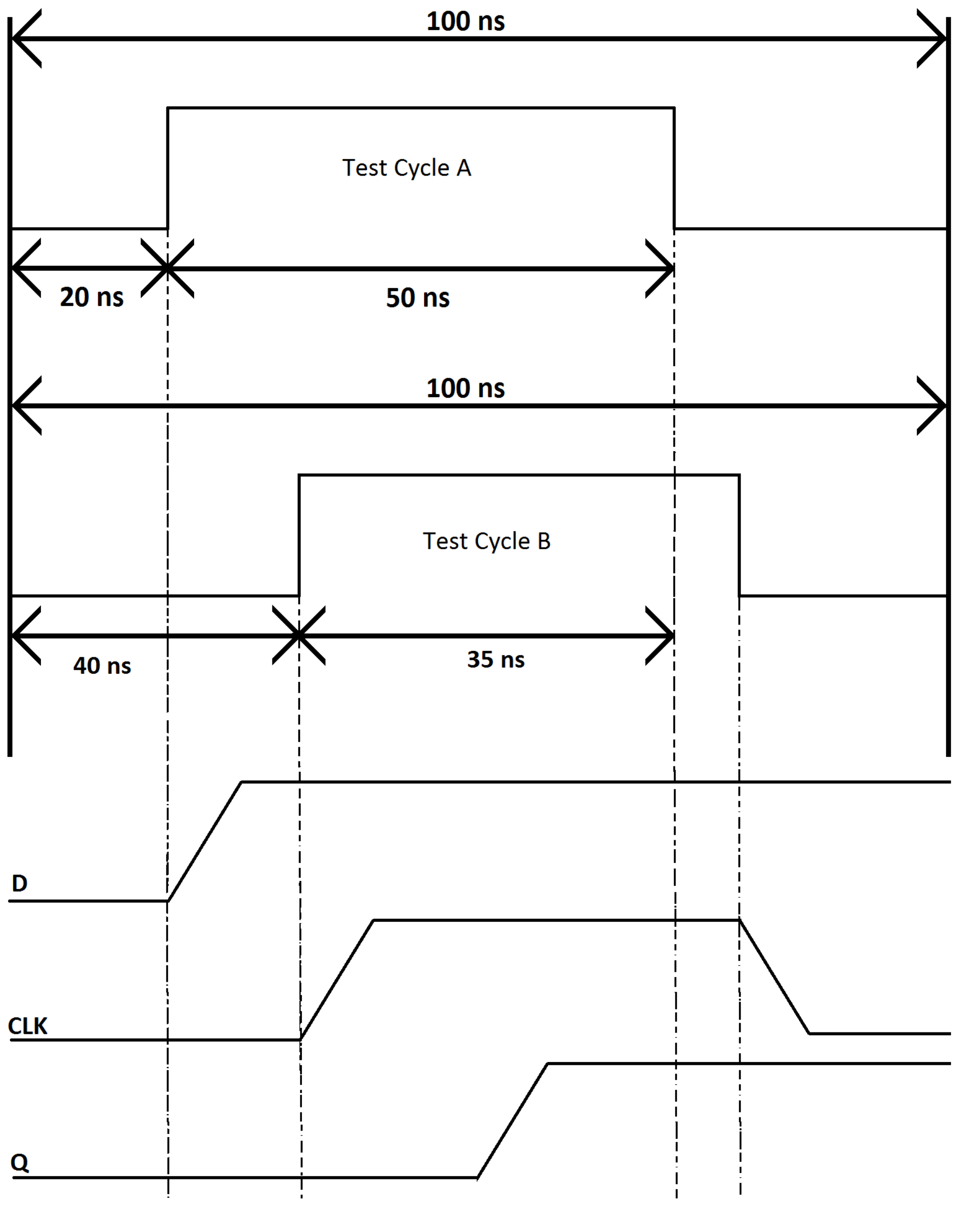

Figure 4: Testing a D Flip-Flop.

voltage values. For instance, if we know that logic low values may range between 0V and 0.5V and that logic high values may range between 4.2V and 5V, it'd be interesting to know how a DUT will respond to these kinds of voltage values.

Additionally, it may be worth knowing how fast our DUT is. Consider Figure 4, where we test a DFF. One way to estimate the setup time of the DFF would be to gradually bring the two leading edges closer to each other until the Q output fails to meet our expectations.

Terminology:

- **Schmooing**: Repeating a test some number of times, with just one electrical or timing characteristic differing between tests.

Consider Figure 3, where we perform a simple test of an AND gate using two input vectors. An example of schmooing would be repeating the entire test, but with the VDD for the input signals starting at 5V for the first test and dropping by 0.2V on each subsequent test until VDD reaches 4.2V. (So, a total of five tests, where each test consists of two test cycles.) Another example, with Figure 4, would be to increase the delay of the uppermost test cycle by 1 ns on each subsequent test up to 40 ns. (So, a total of 20 tests.)

Even better, the LV500 allows users to generate plots of the percentage of output vectors that pass in comparison to the one or two electrical/timing characteristics schmooed.

2.6 Further Reading

Refer to these slides for more information on how the LV500 works.

3 Senior Project: Overview

TODO: Norm, Daniel

3.1 Project Architecture

3.2 What Worked

3.3 Areas of Improvement

4 Master's Project: Design Decisions

4.1 BeagleBone Black Web Server

TODO: Norm

4.2 Numato Saturn FPGA Development Board

TODO: Daniel

4.3 Real-Time Testing

TODO: Norm, Daniel

4.4 Timing Characteristics

TODO: Norm, Daniel

5 Master's Project: Implementation

5.1 BeagleBone Black Web Server and C Code

TODO: Norm (web server code), Daniel (C code)

5.2 Verilog for Numato Saturn FPGA Development Board

TODO: Daniel

5.3 PCB Boards

TODO: Norm, Daniel

6 Master's Project: Overview

TODO: Norm, Daniel

6.1 What Worked

6.2 Difficulties Encountered

6.3 Shortcomings and Areas of Improvement

7 Using Our System

TODO: Norm, Daniel

8 Advice for Future Work

TODO: Norm, Daniel

9 Special Thanks

We are sincerely grateful to the following people:

- Ken Stevens, our Master's project advisor, for providing us with proper guidance throughout this project.
- Erik Brunvand, our senior project advisor, for providing us with FPGA boards for prototyping and the ZIF socket we used for our DUT board.
- Michelle Gifford, Norm's wife, for helping to create the housing for our project.
- Advanced Circuits, our go-to PCB manufacturer, for offering student prices that cannot be found anywhere else.
- Our families, who put up with our absences and will need to get to know us again.

10 Contact Information

Feel free to email us any questions/comments regarding our project, ASIC testers in general, or your own ASIC tester project.

Alfred ”Norm” Gifford: agiffiv@gmail.com
Daniel Khoury: d.khoury@outlook.com

11 References