Virtual Test: Simulating ATE Vectors in a SystemVerilog Testbench for Faster Time to Market

John Mackintosh, Matthew Borto, Cecil Stone, David Brownell, and Courtney Fricano
Analog Devices, Inc.
3 Technology Way
Norwood, MA 02062

***Abstract*- Testcase bring-up on the ATE is often in the critical path to product sampling and release. Virtual test can be used to fully validate scan and functional patterns before the arrival of silicon to reduce Time-To-Market. This paper describes a SystemVerilog virtual testbench framework that is low cost, easy to re-use, has a low learning curve, and fast vector processing. A product development case study is included to illustrate the benefits of using this approach.**

I. INTRODUCTION

Reducing Time-To-Market (TTM) is a critical goal of most companies in the semiconductor industry. Lower TTM results in faster time to revenue, higher lifetime product revenue and lower product development costs. Silicon bring-up on the ATE is often in the critical path to market. Specifically, the level of effort required to bring-up scan and functional testcases plays a large role in defining TTM for semiconductor products.

To reduce the ATE silicon bring-up effort, it is desired to identify and correct as many testcase deficiencies as possible before the arrival of silicon. Verification practices such as testbench force commands and memory preload can render a testcase unusable in silicon. This often goes undetected until patterns are run across Process, Voltage, and Temperature (PVT) in silicon. A methodology is required that generates and validates ATE testcases across PVT before getting on the tester with silicon. This will prevent the consumption of valuable ATE time for debugging non-silicon related issues and speed the overall silicon bring-up task.

The technique commonly used to accomplish this task is virtual test. Commercial virtual test tools are available for purchase. However, it has been the experience of the authors that these commercially available tools do not satisfy all of our requirements:

- Low cost
- Very small learning curve
- Easy re-use from project to project
- Fast vector processing throughput

The Analog Devices, Inc. (ADI) design verification and product test teams co-developed a Virtual Test Framework in order to reduce the amount of time between first silicon and customer samples with these key requirements in mind. This reusable framework enables the simulation of ATE patterns in a SystemVerilog testbench to identify any test-related functional or timing issues before first silicon arrives.

This paper will describe the implementation of the tool and results achieved with the tool on recent development projects. A simple Device Under Test (DUT) example will be used to demonstrate how the tool works.

II. IMPLEMENTATION

The Virtual Test framework has two main components. A diagram of the Virtual Test architecture is shown in Figure 1 below.

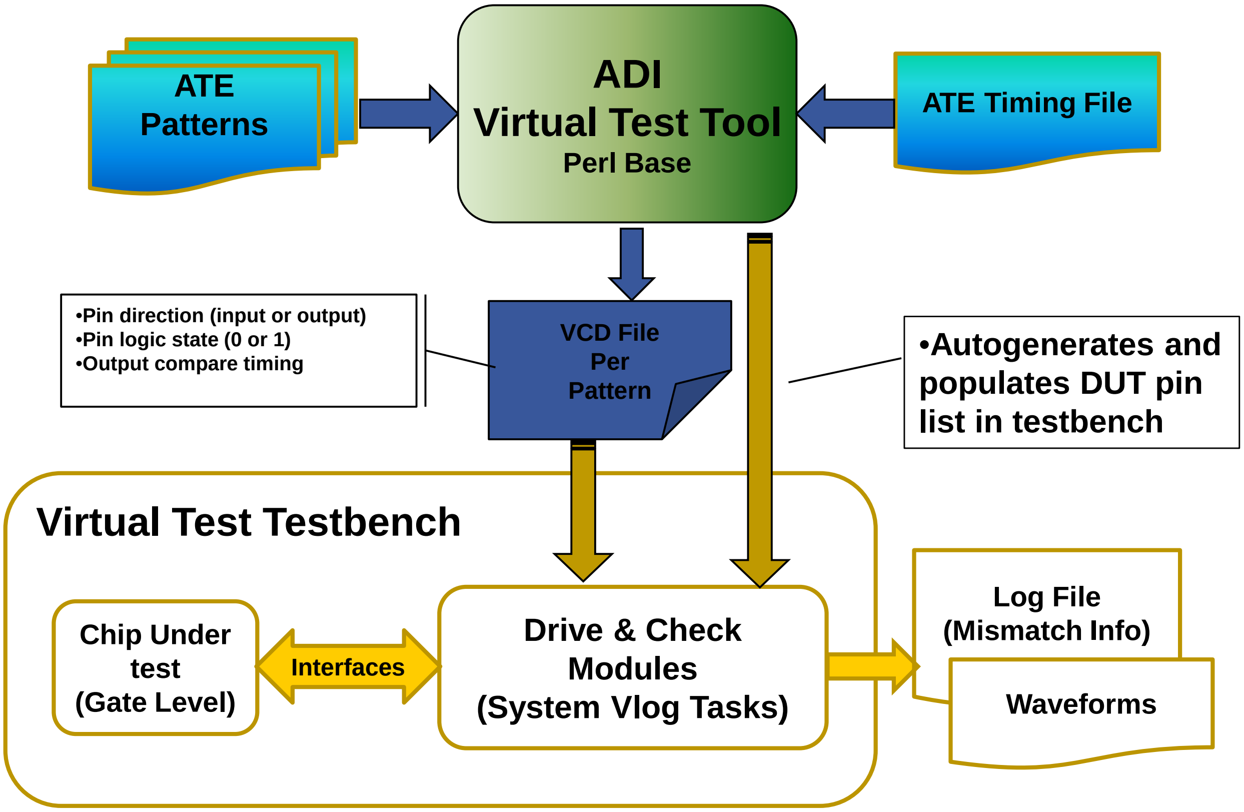


Figure 1. Virtual Test Framework

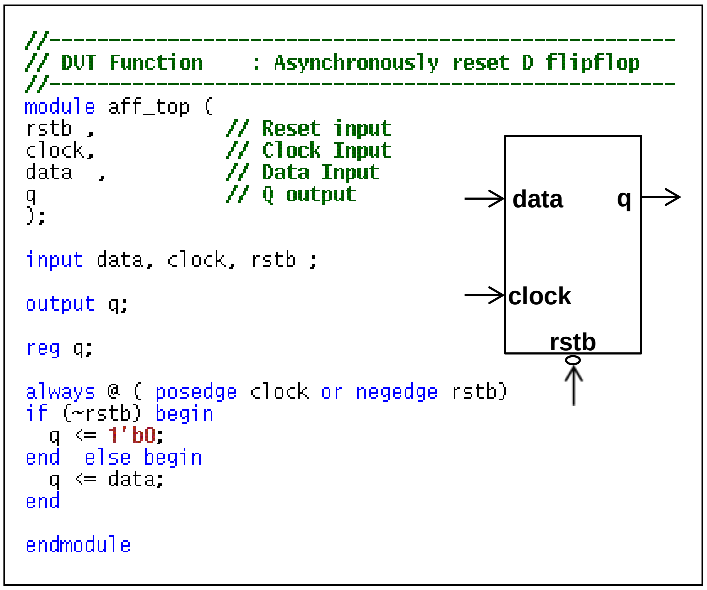
A. VIRTUAL TEST TOOL

The first component of the Virtual Test Framework is the Virtual Test Tool. The Virtual Test Tool is a PERL script used to generate a VCD file from the ATE pattern and timing files. Generation of the ATE patterns and timing file can be accomplished with commercially available or in-house tools and is beyond the scope of this paper. The Virtual Test Tool was developed inside ADI and required approximately 1 person-month of development effort.

The Virtual Test Tool generates one VCD file per ATE pattern. The VCD file is formatted per the IEEE-1364 VCD standard [1] and contains 3 variables per DUT pin:

1. Pin State. Naming convention: <pin_name>
   Value may be 1,0,or x. Matches ATE pattern data.
   Time stamp of value change matches ATE timing file input edge timing.
2. Pin Direction. Naming convention: oeb_<pin_name>
   1 when signal is output. 0 when signal is input or tristate.
3. Pin Output Strobe Timing. Naming convention: stb_<pin_name>
   Pulse placed where tester strobe exists (per ATE timing file) for output compare.
   Negative going edge of pulse indicates edge strobe position.

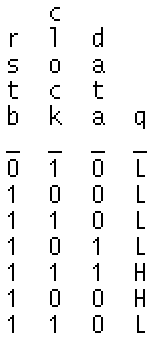
An example DUT is shown in Figure 2. It is a simple asynchronously reset D-type flip-flop with 3 inputs (rstb, clock, and data) and 1 output (q). An ATE pattern for this DUT is shown in Figure 3. The Virtual Test Tool will process this ATE test pattern and generate a VCD file representation of the test pattern which includes signals for pin state, pin direction, and pin output strobe timing information. See Figure 4 for the waveform representation of the output VCD file. The file retains ATE timing for input drive edges, I/O switching, and output strobe placement. The full VCD file generated for the example test pattern is included as Appendix A.



```
//-----------------------------------------------------
// DUT Function    : Asynchronously reset D flipflop
//-----------------------------------------------------
module aff_top (
rstb ,              // Reset input
clock,              // Clock Input
data  ,             // Data Input
q                   // Q output
);

input data, clock, rstb ;

output q;

reg q;

always @ ( posedge clock or negedge rstb)
if (~rstb) begin
  q <= 1'b0;
end  else begin
  q <= data;
end

endmodule
```

data
q
clock
rstb

Figure 2. Example Device Under Test



rstb	clock	data	q
0	1	0	L
1	0	0	L
1	1	0	L
1	0	1	L
1	1	1	H
1	0	0	H
1	1	0	L

Figure 3. Example Test Pattern (Tabular Representation)

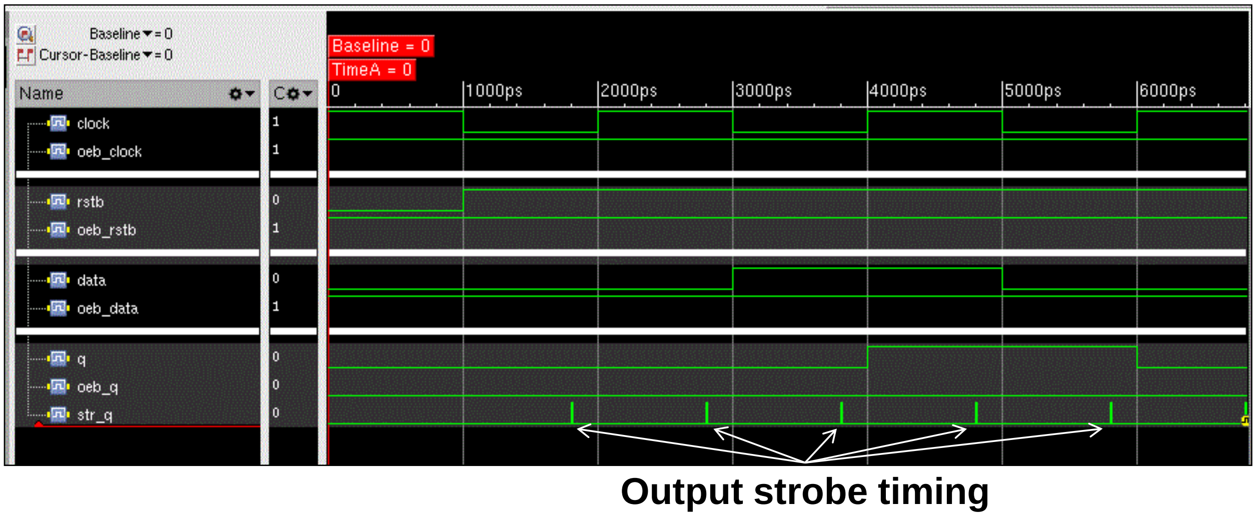


Figure 4. Waveform Representation of Output VCD File

B. Virtual Test Testbench

The second component of the Virtual Test Framework is the Virtual Test Testbench. The Virtual Test Testbench is a SystemVerilog testbench with tasks that translate the previously generated test pattern VCD file to drive the input pins of the design under test with accurate ATE timing. The testbench also compares the output pins to expected values at the strobe timing specified in the VCD file and reports a pass/fail result. Waveforms can optionally be generated to support Virtual Test debug.

SystemVerilog was chosen for implementation to take advantage of data structures such as associative arrays and SystemVerilog's object oriented programming structure. The Virtual Test Testbench is based on a generic, re-usable template that is populated with DUT specific information. The DUT specific information is auto-generated once at the start of a project by invoking the Virtual Test Tool Perl script (described in section II-A) with a special auto-generate switch. This facilitates fast bring-up and easy re-use from one project to the next.

Figure 5 shows the Virtual Test Testbench block diagram. A pin interface is defined and instantiated for each DUT pin. The pin interface contains low level force and release functions used to control each DUT pin. The DUT is connected to the testbench using these pin interfaces. SystemVerilog tasks manage pin drive and check functionality in the testbench.

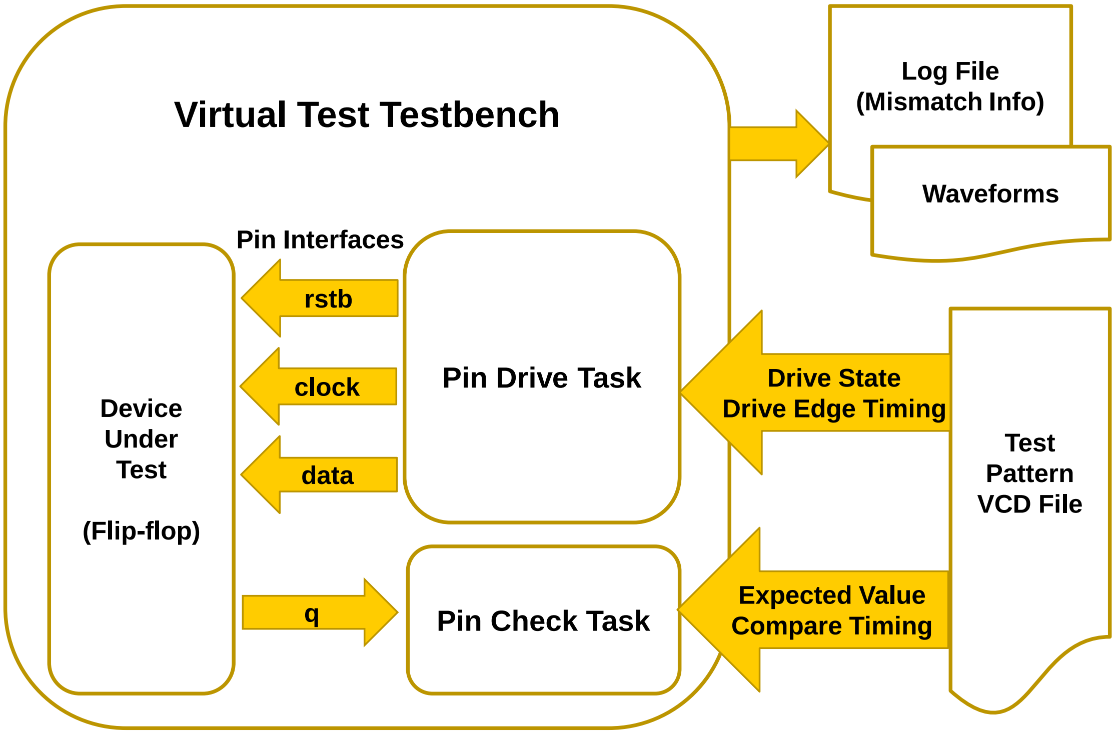


Figure 5. Virtual Test Testbench Block Diagram

The virtual test flow diagram is shown the Figure 6. The test pattern VCD file is parsed to collect pin state, direction, and output strobe timing information at a given timestamp. Once all data is collected for the timestamp, all pin states are updated by the testbench. Inputs are driven and outputs are strobed. Then the tool proceeds to the next timestamp in the VCD to collect and process the next set of pin data. All output strobe mismatches are logged and strobe positions are marked in the optional waveform file for debug and analysis. For the simple flip-flog DUT, an output delay fault was introduced that results in an output strobe failure on pin "q" at timestamp 4,810ns. The resulting virtual test log file and waveforms are shown in Figures 7 and 8.

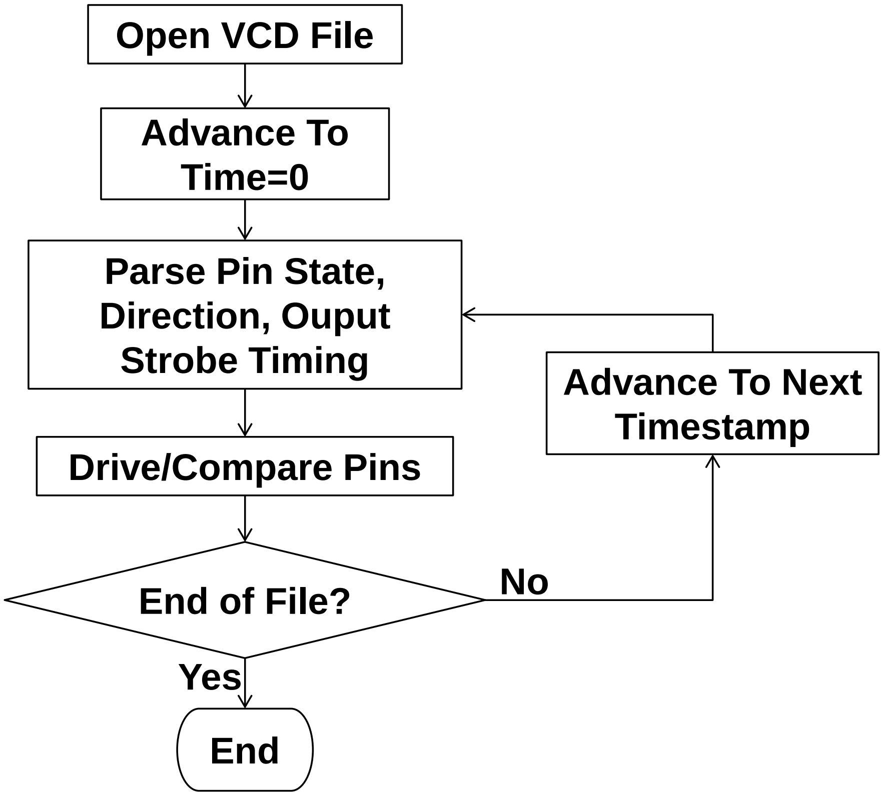
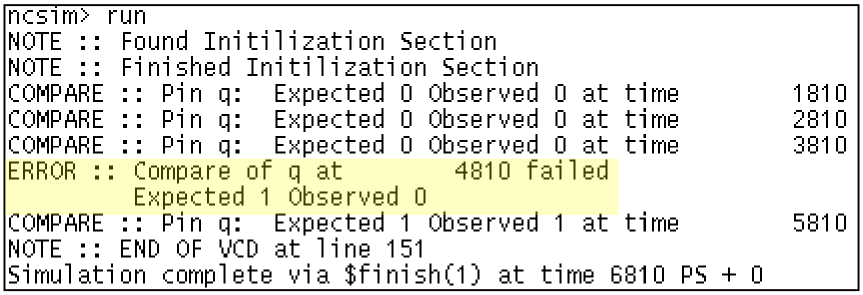
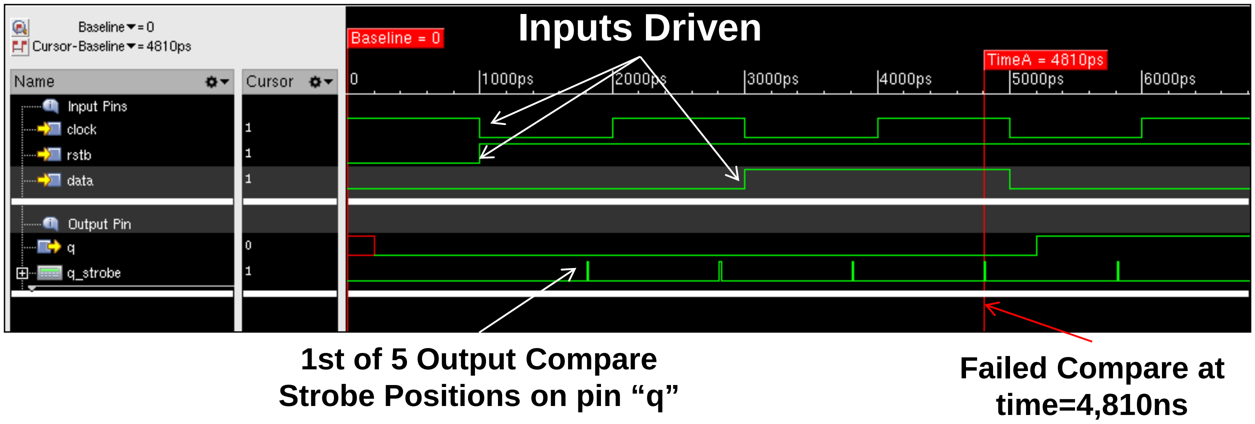

Open VCD File
Advance To Time=0
Parse Pin State, Direction, Ouput Strobe Timing
Drive/Compare Pins
End of File?
No
Advance To Next Timestamp
Yes
End

Figure 6. Virtual Test Flow Diagram



```
ncsim> run
NOTE :: Found Initilization Section
NOTE :: Finished Initilization Section
COMPARE :: Pin q:  Expected 0 Observed 0 at time        1810
COMPARE :: Pin q:  Expected 0 Observed 0 at time        2810
COMPARE :: Pin q:  Expected 0 Observed 0 at time        3810
ERROR :: Compare of q at        4810 failed
         Expected 1 Observed 0
COMPARE :: Pin q:  Expected 1 Observed 1 at time        5810
NOTE :: END OF VCD at line 151
Simulation complete via $finish(1) at time 6810 PS + 0
```

Figure 7. Virtual Test Simulation Log File For Flip Flop DUT With Delay Fault Inserted



Inputs Driven

1st of 5 Output Compare Strobe Positions on pin "q"

Failed Compare at time=4,810ns

Figure 8. Virtual Test Simulation Waveform For Flip Flop DUT With Delay Fault Inserted

III. RESULTS

The virtual test methodology has been used to fully validate scan and functional patterns before the arrival of silicon at ADI with some very compelling results. Performance benchmarks for two different products are shown in Table 1. Benchmarks were measured for a single test pattern on a standard 64-bit workstation. Simulations are done using gate level netlist with back annotated timing. An in-house simulation wrapper script is typically used for parallel batch processing to achieve very good throughput when processing the chip's entire testcase suite.

TABLE 1
BENCHMARKS: PROCESSING A SINGLE TEST PATTERN

Project	**Transistor Count**	**Pin Count**	**VCD Generation (vectors/sec)**	**Testbench Sim Time (vectors/sec)**
1 - AMS	650K	32	1.2k	2.1k
2 – SOC	100M	184	373[1]	40[1]

[1] Stuck-at-fault scan test pattern of 28k vector length. This results in a vcd generation time of 40s and testbench simulation time of 714s.

Virtual test simulations discovered multiple issues prior to tapeout that saved many weeks of debug time. Table 2 lists 3 categories of issues detected during one of the projects.

TABLE 2
CASE STUDY: VIRTUAL TEST DISCOVERIES

I.	**Silicon Bugs: prevented mask set revision**
	Clock mux bug caused glitching, resulted in non-deterministic behavior for all functional tests
	Race condition in scan pattern at cold, FF silicon, high voltage
II.	**Major Pattern Issues: required regeneration of ALL patterns**
	Scan patterns fail because a bond option was included in ATPG constraints
	Start up sequence wrong
	Test clock period not integer multiple of tester period
	Inaccurate modeling of oscillator startup delay
III.	**Minor Test Issues: saved debug time**
	Inputs floated in port testmode
	Longer than expected pad delays identified – adjusted tester output delay timing specs
	Communication port input timing needed adjustment

Because one of the issues was a serious silicon bug that would have prevented ATE determinism on all functional test patterns, Virtual Test prevented a serious project delay and the added expense of a metal mask silicon revision. Virtual Test also identified test issues like incorrect startup sequences, inaccurate oscillator modeling, and scan patterns that were not valid due to a bond option that was not included in ATPG constraints. These issues would have required complete pattern regenerations, using up valuable post-silicon schedule time. Virtual Test also helped saved time by adjusting ATE specs before first silicon to account for very long pad delays and to improve ATE determinism.

Virtual Test can also help determine the root cause of issues found during silicon evaluation by providing a simulation environment with full waveform dumping that replicates the ATE test. For example, one new product had unexpected scan pattern yield loss at cold temperature, high voltage. Virtual test was used to reproduce the failure and expose a race condition that escaped detection in the design flow.

It should be noted that this testbench implementation assumes a digital interface between the ATE and the DUT which can be modeled in a VCD file. Embedded analog DUT functionality can be simulated in the environment with a co-simulation approach. However, there may be limitations when verifying DUTs with analog pins.

IV. CONCLUSIONS

In general, virtual test is an important component of any modern product development flow. There are many benefits of virtual test:

- Achieve faster TTM
- Prevent wasting valuable ATE system time
- Verify that testbench "shortcuts" are not masking testability issues
- Vector validation is accomplished prior to silicon while the design and verification resources are still allocated to the project.

The approach described in this paper is a low cost, short learning curve solution that has demonstrated the capability to improve product time to market. It has been successfully used on three ADI projects to date and will be used for all ATE-tested projects going forward. The framework is easily re-usable across projects.

Because virtual test is implemented in a familiar SystemVerilog environment, simulated with standard verilog simulators, and analyzed with common waveform viewers, all members of the design and product test teams can collaborate to debug and solve problems. Unlike commercially available virtual test solutions, there is no large learning curve required to work with the Virtual Test framework. All members of the team can be up and running quickly.

As all new product development teams know, there is very high visibility when first silicon arrives. Upper management is often literally looking over the shoulder of the bring-up team. They want real quick results, not testcase conversion issues that delay silicon evaluation and sampling to customers. Validating product patterns ahead of silicon with this Virtual Test Framework will greatly increase the likelihood of having real silicon results to report soon after silicon arrives.

ACKNOWLEDGMENT

The authors wish to acknowledge the following Analog Devices employees for their thoughtful comments and contributions during the writing of this paper: Todd Honan, Steven Pesut, and Roberta Myers

REFERENCES

[1] *IEEE Standard Verilog Hardware Description Language*, IEEE Standard 1364, 2001.

Appendix A – VCD File Generated by Virtual Test Tool for Example Test Pattern

```
$date
   Dec 17, 2012 12:16:53
$end

$version
   Signalscan 6.8s6
$end

$timescale
   1ps
$end


$scope module aff_top $end

$scope module PE_module0 $end
$var wire 1 R rstb  $end
$var wire 1 r oeb_rstb  $end
$var wire 1 1 stb_rstb  $end
$var wire 1 C clock  $end
$var wire 1 c oeb_clock  $end
$var wire 1 2 stb_clock  $end
$var wire 1 D data  $end
$var wire 1 d oeb_data  $end
$var wire 1 3 str_data  $end
$var wire 1 Q q  $end
$var wire 1 q oeb_q  $end
$var wire 1 4 str_q  $end

$upscope $end

$upscope $end

$enddefinitions $end
#0
$dumpvars
0R
1r
01
1C
1c
02
0D
1d
03
0q
0Q
04
$end
#1000
1R
1r
01
0C
1c
02
0D
1d
03
0q
0Q
```

```
04
#1800
14
#1810
04
#2000
1R
1r
01
1C
1c
02
0D
1d
03
0q
0Q
04
#2800
14
#2810
04
#3000
1R
1r
01
0C
1c
02
1D
1d
03
0q
0Q
04
#3800
14
#3810
04
#4000
1R
1r
01
1C
1c
02
1D
1d
03
0q
1Q
04
#4800
14
#4810
04
#5000
1R
1r
01
0C
1c
```

```
02
0D
1d
03
0q
1Q
04
#5800
14
#5810
04
#6000
1R
1r
01
1C
1c
02
0D
1d
03
0q
0Q
04
#6800
14
#6810
04
```