Verilog Test Benches

ECE 2372 | Modern Digital System Design | Texas Tech University

Lecture Overview

- The Anatomy of a Test Bench
- Verifying Your Test Cases
- Tracking Your Test Results
- Verilog Test Benching at Scale
- Importing Test Cases

The Unit Under Test

During this lecture, we will be testing a simple, 2-bit adder with the filename **adder.v**, and the following implementation:

$$\{\boldsymbol{C}, \boldsymbol{S_1}, \boldsymbol{S_0}\} = \{\boldsymbol{A_1}, \boldsymbol{A_0}\} + \{\boldsymbol{B_1}, \boldsymbol{B_0}\}$$

The Anatomy of a Test Bench

Verilog Test Benches | Modern Digital System Design

The Anatomy of a Test Bench

All test benches must have the following elements:

1. Include the unit under test.
2. Set the time unit and time scale.
3. Test bench module:
    1. Input register instantiation
    2. Output wire instantiation
    3. Unit Under Test instantiation
    4. Test execution

The Anatomy of a Test Bench

```
 9  // INCLUDE THE UNIT UNDER TEST
10  `include "adder.v"
```

The Anatomy of a Test Bench

```
12  // SET THE TIME UNIT / TIME SCALE
13  `timescale 1ps/1ps
```

The Anatomy of a Test Bench

```
15  // YOUR TESTBENCH MODULE
16  module adder_test;
17      // TEST CODE GOES HERE
18  endmodule
```

The Anatomy of a Test Bench

```
18    // INPUT REGISTERS
19    reg [1:0] A, B;
```

The Anatomy of a Test Bench

```
21    // OUTPUT WIRES
22    wire [1:0] S;
23    wire C;
```

The Anatomy of a Test Bench

```
25   // UNIT UNDER TEST
26   adder UUT(A, B, S, C);
```

The Anatomy of a Test Bench

```
28  // TEST EXECUTION
29  initial begin
30      // ...
31  end
```

The Anatomy of a Test Bench | **Execution**

```
31  // TEST CASE 0
32  A = 2'b00; B = 2'b00; #10;
33  $display("%b + %b = %b%b", A, B, C, S);
```

```
91  // TEST CASE 15
92  A = 2'b11; B = 2'b11; #10;
93  $display("%b + %b = %b%b", A, B, C, S);
```

```
00 + 00 = 000
00 + 01 = 001
00 + 10 = 010
00 + 11 = 011
01 + 00 = 001
01 + 01 = 010
01 + 10 = 011
01 + 11 = 100
10 + 00 = 010
10 + 01 = 011
10 + 10 = 100
10 + 11 = 101
11 + 00 = 011
11 + 01 = 100
11 + 10 = 101
11 + 11 = 110
```

What questions do you have?

Practice Problem

What Verilog datatype is used to specify an input to a test bench?

A. integer
B. reg
C. wire
D. parameter

Practice Problem

What Verilog datatype is used to specify an input to a test bench?

~~A. integer~~

B. reg

~~C. wire~~

~~D. parameter~~

Verifying Your Test Cases

Verilog Test Benches | Modern Digital System Design

Verifying Your Test Cases

- Manually reading the output of each test case is time-consuming.
- Let’s write a testbench that can check the logic for us.

If/Else Syntax in Verilog

Verilog support an if/else syntax like most languages.

```
42  if (CONDITION)
43  begin
44      // EXECUTE IF TRUE
45  end
46  else
47  begin
48      // EXECUTE IF FALSE
49  end
```

Verifying Your Test Cases

```
31  // TEST CASE 0
32  A = 2'b00; B = 2'b00; #10;
33  if({C,S} == 3'b000)
34  begin
35      $display("PASSED");
36  end
37  else
38  begin
39      $display("FAILED");
40  end
```

PASSED
PASSED
PASSED
PASSED
PASSED
PASSED
PASSED
PASSED
PASSED
PASSED
PASSED
PASSED
PASSED
PASSED
PASSED
PASSED
PASSED
PASSED
PASSED

Verifying Your Test Cases

- Our testbench file is now **209 lines long**!
- We can reduce the size since we only really care about failing test cases.

Verifying Your Test Cases

```
31  // TEST CASE 0
32  A = 2'b00; B = 2'b00; #10;
33  if({C,S} != 3'b000)
34  begin
35      $display("TEST FAILED FOR %b + %b", A, B);
36  end
```

This change reduced our testbench to 149 lines from 209 lines.

```
TEST FAILED FOR 01 + 10
TEST FAILED FOR 10 + 11
TEST FAILED FOR 11 + 10
```

What questions do you have?

Practice Problem

What will be the test case output for the following module and testbench?

```
assign F = A | ~B & ~C;
```

A.) PASSED

B.) FAILED

```
{A, B, C} = 3'b010; #10;
if (F != 0)
begin
    $display("PASSED");
end
else
begin
    $display("FAILED");
end
```

Practice Problem

What will be the test case output for the following module and testbench?

```
assign F = A | ~B & ~C;
```

~~A.) PASSED~~

B.) FAILED

```
{A, B, C} = 3'b010; #10;
if (F != 0)
begin
    $display("PASSED");
end
else
begin
    $display("FAILED");
end
```

Tracking Your Test Results

Verilog Test Benches | Modern Digital System Design

Tracking Your Test Results

- If all our tests pass, then we get no output at all.
- It would be nice to see some report at the end.
- We will need to keep track of which tests pass/fail.

Tracking Your Test Results

```
28  // TEST PARAMETERS
29  integer TOTAL_TESTS = 15;
30  integer TOTAL_PASSED = 15;
```

This should be outside your test execution block.

We will decrement the TOTAL_PASSED value every time a test fails.

Tracking Your Test Results

```
35  // TEST CASE 0
36  A = 2'b00; B = 2'b00; #10;
37  if({C,S} != 3'b000)
38  begin
39      $display("TEST FAILED FOR %b + %b", A, B);
40      TOTAL_PASSED = TOTAL_PASSED - 1;
41  end
```

Tracking Your Test Results

```
163  // END OF TEST REPORT
164  $display("%d of %d TEST CASES PASSED.", TOTAL_PASSED, TOTAL_TESTS);
```

```
          15 of          15 TEST CASES PASSED.


TEST FAILED FOR 11 + 11
          14 of          15 TEST CASES PASSED.
```

The spacing of the numbers is strange. This is because the spacing is set for a 32-bit integer (10 decimal digits).

Tracking Your Test Results

```
28  // TEST PARAMETERS
29  reg [3:0] TOTAL_TESTS = 4'd15;
30  reg [3:0] TOTAL_PASSED = 4'd15;
```

```
TEST FAILED FOR 11 + 11
14 of 15 TEST CASES PASSED.
```

What questions do you have?

Practice Problem

If the logic for a Verilog module should be `assign F = A | ~B & ~C` but instead, is typed as `assign F = A | B & ~C`, how many tests will pass?

A. 7 of 7 tests pass
B. 6 of 7 tests pass
C. 5 of 7 tests pass
D. 4 of 7 tests pass

Practice Problem

If the logic for a Verilog module should be assign F = A | ~B & ~C but instead, is typed as assign F = A | B & ~C, how many tests will pass?

~~A. 7 of 7 tests pass~~

~~B. 6 of 7 tests pass~~

C. 5 of 7 tests pass

~~D. 4 of 7 tests pass~~

Practice Problem

A	B	C	A+B'C'	A+BC'
0	**0**	**0**	1	0
0	**0**	**1**	0	0
0	**1**	**0**	0	1
0	**1**	**1**	0	0
1	**0**	**0**	1	1
1	**0**	**1**	1	1
1	**1**	**0**	1	1
1	**1**	**1**	1	1

Verilog Test Benching at Scale

Verilog Test Benches | Modern Digital System Design

Verilog Test Benching at scale

- Even with only 4-bits of input, our test bench is 168 lines long.
- The length of the testbench increases exponentially with input
- Big-O Notation: $\boldsymbol{O[n^2]}$
- We can bring our testbenches down to $\boldsymbol{O[1]}$.

For Loop Syntax in Verilog

- Verilog also supports For Loop syntax like other languages

```
for (START; END_CONDITION; CHANGE)
begin
    // EXECUTE
end
```

Verilog Test Benching at Scale

```
35  for (integer TEST = 0; TEST < 16; TEST = TEST + 1)
36  begin
37      A = TEST[3:2];
38      B = TEST[1:0];
39      #10;
40      if ({C, S} != A + B)
41      begin
42          $display("TEST FAILED FOR %b + %b", A, B);
43          TOTAL_PASSED = TOTAL_PASSED - 1;
44      end
45  end
```

Verilog Test Benching at Scale

```
35  for (integer TEST = 0; TEST < 16; TEST = TEST + 1)
36  begin
37      A = TEST[3:2];
38      B = TEST[1:0];
39      #10;
40      if ({C, S} != A + B)
41      begin
42          $display("TEST FAILED FOR %b + %b", A, B);
43          TOTAL_PASSED = TOTAL_PASSED - 1;
44      end
45  end
```

Even though TEST is a 32-bit integer, we can tap into the last 4-bits to get out 0 – 15 test cases.

15 of 15 TEST CASES PASSED.

Verilog Test Benching at Scale

```
35  for (integer TEST = 0; TEST < 16; TEST = TEST + 1)
36  begin
37      A = TEST[3:2];
38      B = TEST[1:0];
39      #10;
40      if ({C, S} != A + B - 1)
41      begin
42          $display("TEST FAILED FOR %b + %b", A, B);
43          TOTAL_PASSED = TOTAL_PASSED - 1;
44      end
45  end
```

Let's intentionally mess up our test case.

```
TEST FAILED FOR 00 + 00
TEST FAILED FOR 00 + 01
TEST FAILED FOR 00 + 10
TEST FAILED FOR 00 + 11
TEST FAILED FOR 01 + 00
TEST FAILED FOR 01 + 01
TEST FAILED FOR 01 + 10
TEST FAILED FOR 01 + 11
TEST FAILED FOR 10 + 00
TEST FAILED FOR 10 + 01
TEST FAILED FOR 10 + 10
TEST FAILED FOR 10 + 11
TEST FAILED FOR 11 + 00
TEST FAILED FOR 11 + 01
TEST FAILED FOR 11 + 10
TEST FAILED FOR 11 + 11
15 of 15 TEST CASES PASSED.
```

```
TEST FAILED FOR 00 + 00
TEST FAILED FOR 00 + 01
TEST FAILED FOR 00 + 10
TEST FAILED FOR 00 + 11
TEST FAILED FOR 01 + 00
TEST FAILED FOR 01 + 01
TEST FAILED FOR 01 + 10
TEST FAILED FOR 01 + 11
TEST FAILED FOR 10 + 00
TEST FAILED FOR 10 + 01
TEST FAILED FOR 10 + 10
TEST FAILED FOR 10 + 11
TEST FAILED FOR 11 + 00
TEST FAILED FOR 11 + 01
TEST FAILED FOR 11 + 10
TEST FAILED FOR 11 + 11
15 of 15 TEST CASES PASSED.
```

???

Blocking vs Nonblocking Assignment

Verilog supports two separate types of assignment.

- **Blocking –** Will hold all execution until complete (synchronous)
- **Nonblocking –** Will not hold up execution (asynchronous)

```
54  // Nonblocking assignment
55  TOTAL_PASSED = TOTAL_PASSED - 1;
56
57  // Blocking assignment
58  TOTAL_PASSED <= TOTAL_PASSED - 1;
```

Blocking vs Nonblocking Assignment

```
35  for (integer TEST = 0; TEST < 16; TEST = TEST + 1)
36  begin
37      A = TEST[3:2];
38      B = TEST[1:0];
39      #10;
40      if ({C, S} != A + B - 1)
41      begin
42          $display("TEST FAILED FOR %b + %b", A, B);
43          TOTAL_PASSED <= TOTAL_PASSED - 1;
44      end
45  end
```

```
TEST FAILED FOR 00 + 00
TEST FAILED FOR 00 + 01
TEST FAILED FOR 00 + 10
TEST FAILED FOR 00 + 11
TEST FAILED FOR 01 + 00
TEST FAILED FOR 01 + 01
TEST FAILED FOR 01 + 10
TEST FAILED FOR 01 + 11
TEST FAILED FOR 10 + 00
TEST FAILED FOR 10 + 01
TEST FAILED FOR 10 + 10
TEST FAILED FOR 10 + 11
TEST FAILED FOR 11 + 00
TEST FAILED FOR 11 + 01
TEST FAILED FOR 11 + 10
TEST FAILED FOR 11 + 11
 0 of 15 TEST CASES PASSED.
```

What questions do you have?

Practice Problem

In the following For Loop structure, what will the values of A and B be on the 21st iteration?

```
63  for (integer N = 0; N < 32; N = N + 1)
64  begin
65      A = N[5:3];
66      B = N[2:0];
67  end
```

A. A = 5, B = 4

B. A = 5, B = 0

C. A = 2, B = 5

D. A = 2, B = 4

Practice Problem

In the following For Loop structure, what will the values of A and B be on the 21st iteration?

```
63  for (integer N = 0; N < 32; N = N + 1)
64  begin
65      A = N[5:3];
66      B = N[2:0];
67  end
```

~~A. A = 5, B = 4~~

~~B. A = 5, B = 0~~

~~C. A = 2, B = 5~~

D. A = 2, B = 4

Practice Problem

In the following For Loop structure, what will the values of A and B be on the 21st iteration?

```
63  for (integer N = 0; N < 32; N = N + 1)
64  begin
65      A = N[5:3];
66      B = N[2:0];
67  end
```

~~A. A = 5, B = 4~~

~~B. A = 5, B = 0~~

~~C. A = 2, B = 5~~

D. A = 2, B = 4

On the 21st iteration, N = 20 = 010100.
A = N[5:3] = 010 = 2
B = N[2:0] = 100 = 4

Importing Test Cases

Verilog Test Benches | Modern Digital System Design

Importing Test Cases

```
35  for (integer TEST = 0; TEST < 16; TEST = TEST + 1)
36  begin
37      A = TEST[3:2];
38      B = TEST[1:0];
39      #10;
40      if ({C, S} != A + B)
41      begin
42          $display("TEST FAILED FOR %b + %b", A, B);
43          TOTAL_PASSED = TOTAL_PASSED - 1;
44      end
45  end
```

We're inclined to mess up our logic in the module too.

Importing Test Cases

- If you mess up the logic in your module, you'll probably mess it up in your testbench too.
- We need a way to get our test cases from the outside world.

Importing Test Cases

adder_data.txt

1	000	9	010
2	001	10	011
3	010	11	100
4	011	12	101
5	001	13	011
6	010	14	100
7	011	15	101
8	100	16	110

Importing Test Cases

```
32  // TEST DATA
33  // [2:0] <- 3-bit data
34  // [0:15] <- 16 rows in the data
35  reg [2:0] DATA [0:15];
36
37  // TEST EXECUTION
38  initial begin
39  // Read in the test data.
40  // READ MEMory Binary
41  $readmemb("C:\\PATH_TO_FILE\\adder_data.txt", DATA);
```

```
41  // Run the test.
42  for (integer TEST = 0; TEST < 16; TEST = TEST + 1)
43  begin
44      A = TEST[3:2];
45      B = TEST[1:0];
46      #10;
47      if ({C, S} != DATA[TEST])
48      begin
49          $display("TEST FAILED FOR %b + %b", A, B);
50          TOTAL_PASSED <= TOTAL_PASSED - 1;
51      end
52  end
```

What questions do you have?

Practice Problem

Suppose that a Verilog module is written as assign F = A + B'. If the text of the *data.txt* file is given below, how many test cases will pass?

A. 4 of 4 Tests Pass
B. 3 of 4 Tests Pass
C. 2 of 4 Tests Pass
D. 1 of 4 Tests Pass

data.txt:

1
1
1
0

Practice Problem

Suppose that a Verilog module is written as assign F = A + B'. If the text of the *data.txt* file is given below, how many test cases will pass?

~~A. 4 of 4 Tests Pass~~

~~B. 3 of 4 Tests Pass~~

C. 2 of 4 Tests Pass

~~D. 1 of 4 Tests Pass~~

data.txt:

1

1

1

0

Practice Problem

Suppose that a Verilog module is written as assign F = A + B'. If the text of the *data.txt* file is given below, how many test cases will pass?

~~A. 4 of 4 Tests Pass~~

~~B. 3 of 4 Tests Pass~~

C. 2 of 4 Tests Pass

~~D. 1 of 4 Tests Pass~~

data.txt:	**A + B'**
1	1
1	0
1	1
0	1

Lecture Recap

- The Anatomy of a Test Bench
- Verifying Your Test Cases
- Tracking Your Test Results
- Verilog Test Benching at Scale
- Importing Test Cases

Verilog Test Benches

ECE 2372 | Modern Digital System Design | Texas Tech University