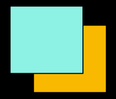
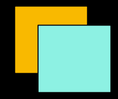
TestBench Guide

10/12/2021



NTHU Logic Design Laboratory (Fall. 2021)



By Prof. Chun-Yi Lee

Agenda

TestBench Walk Through

TestBench Walk Through (1/9)

- First of all, please specify the timescale

  of the simulation

- The first number specify time_unit

- The second one specify time_precision

TestBench Walk Through (2/9)

```
// `timescale time_unit / time_precision
`timescale 100ns / 10ns

module tb;

reg [4:0] set;
parameter d1 = 20,
          d2 = 1.5,
          d3 = 1.54,
          d4 = 1.55;

initial begin
    #d1   set = 0;      // real_delay = round(20*100)   = 2000ns
    #d2   set = 1;      // real_delay = round(1.5*100)  = 150ns
    #d3   set = 2;      // real_delay = round(1.54*100) = 150ns
    #d4   set = 3;      // real_delay = round(1.55*100) = 160ns
end

endmodule
```

TestBench Walk Through (3/9)

- Then declare the I/O for the module you want to test.
- inputs : **reg/wire**
- Outputs : **wire**

```
module RippleCarryAdder_t;
// global clock signal: change input at engedge clk
reg CLK;

// inputs to module
reg [8-1:0] A, B;
reg Cin;

// outputs from module
wire [8-1:0] Sum;
wire Cout;
```

TestBench Walk Through (4/9)

- Instantiate the module you want to test.
- Connection by name is recommanded.

```
// instantiate testing module
RippleCarryAdder testing_instance (
  .A (A),
  .B (B),
  .Cin (Cin),
  .Sum (Sum),
  .Cout (Cout)
);
```

TestBench Walk Through (5/9)

- Simulate clk signal

  (useful for timing management)

- Gives input at one clk edge, test the

  output at the other.

- Use **repeat**.

- Do remember to call **finish**.

```
// simulated clk period = 5 * 2 = 10 ns
always #5 CLK = ~CLK;

// main testing
initial begin
  {A, B, Cin} = 9'd0;

  repeat (2 ** 9) begin
    @ (posedge CLK)
        Test;
    @ (negedge CLK)
        {A, B, Cin} = {A, B, Cin} + 1'b1;
  end

  $finish;
end
```

TestBench Walk Through (6/9)

- Debugging by looking at waveforms can be hard .....
- Design tasks to check the correctness for you!
- Also, use display to print out debugging log for you.

```
// utility task for testing
task Test;
begin
    if({Cout, Sum} !== (A + B + Cin))begin
        $display("[ERROR]");
        $write("A:%d",A);
        $write("B:%d",B);
        $write("Cin:%d",Cin);
        $write("Cout:%d",Cout);
        $write("Sum:%d",Sum);
        $display;
    end
end
endtask
```

TestBench Walk Through (7/9)

- Use ‘===’ or ‘!==’ to compare the output of the module to the answer.
- Errors such as x and z can only be detected this way

==	z	x	1	0
z	x	x	x	x
x	x	x	x	x
1	x	x	1	0
0	x	x	0	1

!=	z	x	1	0
z	x	x	x	x
x	x	x	x	x
1	x	x	0	1
0	x	x	1	0

===	z	x	1	0
z	1	0	0	0
x	0	1	0	0
1	0	0	1	0
0	0	0	0	1

!==	z	x	1	0
z	0	1	1	1
x	1	0	1	1
1	1	1	0	1
0	1	1	1	0

TestBench Walk Through (8/9)

- Test every input combination, if possible.
- Use tasks make code cleaner and easier to maintain
- Use display tasks to help you.

```
// utility task for testing
task Test;
begin
    if({Cout, Sum} !== (A + B + Cin))begin
        $display("[ERROR]");
        $write("A:%d",A);
        $write("B:%d",B);
        $write("Cin:%d",Cin);
        $write("Cout:%d",Cout);
        $write("Sum:%d",Sum);
        $display;
    end
end
endtask
```

TestBench Walk Through (9/9)

```
for (i = 0; i < 16; i = i + 1) begin
    in1 = i;
    in2 = i;
    in3 = i;
    in4 = i;
    #1;
end
```

```
module Lab1_TeamX_Crossbar_4x4_4bit_t;

reg [3:0] in1 = 4'b0001;
reg [3:0] in2 = 4'b0010;
reg [3:0] in3 = 4'b0011;
reg [3:0] in4 = 4'b0100;
reg [4:0] control = 5'b00000;
wire [3:0] out1, out2, out3, out4;

Crossbar_4x4_4bit CSBR4x4(in1, in2, in3, in4, out1, out2, out3, out4, control);

always#(2) control = control + 1;


endmodule
```