NMK20603

Computer Architecture

MY1

Describing Combinational Logic (cont.)

MY1

Codes so far:

✔ mux21_1b, mux21_1b_tb

✔ mux41_1b, mux41_1b_tb

MY1

Homework?

⇒ mux41_2b

⇒ mux41_2b_tb

MY1

'Hidden' work?

⇒ mux21_2b

⇒ mux21_2b_tb

MY1

```
// notice size mismatch?
assign tA = ~iS & iA;
```

MY1

- Extend / replicate

⇒ concatenation!

{ iS, iS }

@ {2{iS}}

mux41_2b_tb

- checked waveform?
- truth table rows?

MY1

Self-checking Testbench

MY1

...

- Stimuli generator ✔
- Response monitor ?

MY1

Verilog System Task

MY1

- $write / $display (like printf)
- $time (current simulation time)
- $stop / $finish (end simulation)

$write / $display

❄ %b (byte), %h (hex)

❄ %d (decimal)

❄ %g (general format fp)

⇒ numeric prefix (display size limit)

$stop

⇒ ends simulation only

⇒ use in gui

MY1

$finish

⇒ exit modelsim

⇒ use in scripts

```
module systask1_tb ();
initial begin
  $display("-- Stopping");
  $stop;
  $display("-- Finishing");
  $finish;
end
endmodule
```

```
module systask2_tb ();
integer keep,loop;
reg[9:0] temp;
initial begin
  keep=$time;
  #10 $display("-- SimTime:[%%d:%d][%%g:%g]",keep,keep);
  #10 $display("-- SimTime:[%%5d:%5d] => numeric prefix sets spacing",$time);
  #10 $display("-- SimTime:[%%05d:%05d] => zero prefix for 0-padding",$time);
  #10 $display("-- SimTime:[%%5g:%5g] => same for g",$time);
  #10 $display("-- SimTime:[%%05g:%05g]",$time);
  temp=$time;
  #10 $display("-- SimTime:[%%8b:%8b] =>  b is always 0-padded",temp);
  #10 $display("-- SimTime:[%%08b:%08b]",temp);
  for (loop=0;loop<3;loop=loop+1) begin
    #10 $display("-- SimTime:[%05g]",$time);
  end
  $stop;
end
endmodule
```

MY1

```
module systask3_tb ();
initial begin
  $write("-- 1st Line... ");
  $write("still here!\n");
  $display("-- Next Line");
  $display("-- Next Line");
  $stop;
end
endmodule
```

MY1

Literal bit definition

MY1

`1` vs `1'b1`

Truth Table

Output @ Array

MY1

A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

// W1 homework

MY1

```
// this is in _tb
reg[0:7] cY;
// in initial block
cY = { 1'b1, 1'b1, 1'b0, 1'b0,
  1'b0, 1'b0, 1'b1, 1'b0 };
```

MY1

How to verify?

MY1

```
// this is in _tb
integer ecnt; // error count
// in for loop
if (mY!==cY[loop]) begin
  ecnt = ecnt + 1;
end
```

MY1

if ecnt is 0

when loop ends

⇒ verified!

Let's do some 'programming'

MY1

Describe:

Response 'Monitor'

MY1

```
module logic_3i1o_tb ();
reg dA,dB,dC;
wire mY;
reg[0:7] cY;
integer loop, ecnt;
initial begin
  cY = {1'b1,1'b1,1'b0,1'b0,1'b0,1'b0,1'b1,1'b0};
  ecnt = 0;
  for (loop=0;loop<8;loop=loop+1) begin
    {dS,dA,dB} = loop;
    $write("## | %2b | %2b | %2b |",dA,dB,dC);
    #5; // we check at half cycle
    $write(" %2b | %2b |",mY,cY);
    if (mY!==cY) begin
      $write("** error **");
      ecnt = ecnt + 1;
    end
    $write("\n");
    #5; // complete the 10 TU cycle
  end
  if (ecnt==0) begin
    $display("-- Logic 3In1Out Verified!");
  end
  $stop; // or $finish , if you use scripts
end
logic_3i1o dut (dS,dA,dB,mY);
endmodule
```

Note:

⇒ check at half-cycle (#5)

⇒ setup & hold time!

Back to mux!

⇒ mux21 n-bits?

Keyword:

- parameter
- defparam

Let's rewrite

mux21_1b

Start a new ModelSim project!

MY1

Describe:

dmux21

MY1

```
module dmux21 (iS,iA,iB,oY);
parameter BITS = 4;
input iS;
input[BITS-1:0] iA,iB;
output[BITS-1:0] oY;
wire[BITS-1:0] oY, tA, tB;
assign tA = {BITS{~iS}}&iA;
assign tB = {BITS{iS}}&iB;
assign oY = tA | tB;
endmodule
```

MY1

Describe:

dmux21_tb

MY1

```
module dmux21_tb ();
parameter BITS = 4;
reg dS;
reg[BITS-1:0] dA,dB;
wire[BITS-1:0] mY;
integer loop,ecnt;
initial begin
  ecnt = 0;
  for (loop=0;loop<2**((BITS*2)+1);loop=loop+1) begin
    {dS,dA,dB} = loop;
    #5;
    if (dS===1'b1) begin
      if (mY!==dB) begin
        ecnt = ecnt + 1;
      end
    end
    else if (dS==1'b0) begin
      if (mY!==dA) begin
        ecnt = ecnt + 1;
      end
    end
    #5;
  end
  if (ecnt==0) begin
    $display("-- Module dmux21 Verified!");
  end
  else begin
    $display("** Module dmux21 with errors! (%d)",ecnt);
  end
  $stop;
end
defparam dut.BITS = BITS;
dmux21 dut (iS,iA,iB,oY);
// or, dmux21 #(BITS) dut (iS,iA,iB,oY); // verilog2k1
endmodule
```

All future tb MUST BE sctb! (optional in assessment)

MY1

Practice: (optional)

⇒ sctb: mux21_2b_tb

⇒ sctb: mux41_2b_tb

MY1

Homework:

⇒ circ: dmux41

⇒ sctb: dmux41_tb

MY1