

NMK20603

# Computer Architecture

# Describing Combinational Logic (cont.)

Codes so far:

✓ mux21\_1b / mux21\_1b\_tb

✓ mux41\_1b / mux41\_1b\_tb

# Homework?

⇒ mux41\_2b

⇒ mux41\_2b\_tb

'Hidden' work?

⇒ mux21\_2b

⇒ mux21\_2b\_tb

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

- Extend / replicate  
⇒ concatenation!

{ iS, iS }

@ { 2 { iS } }



mux41\_2b\_tb

- ▶ checked waveform?
- ▶ truth table rows?

# Self-checking Testbench

...

- ▶ Stimuli generator ✓
- ▶ Response monitor ?

# Verilog

# System Task

- ▶ `$write` / `$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)

`$stop`

⇒ ends simulation only

⇒ use in gui

`$finish`

⇒ `exit modelsim`

⇒ `use in scripts`



# Literal bit definition

1 vs 1'b1

# Truth Table

# Output @ Array

	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

```
// 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 };
```

How to  
verify?

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

if ecnt is 0

when loop ends

⇒ verified!



Let's do some  
'programming'

Describe:

Response 'Monitor'

```

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!



Describe:

dmux21

```
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
```

Describe:

dmux21\_tb

```

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)

Practice: (optional)

⇒ sctb: mux21\_2b\_tb

⇒ sctb: mux41\_2b\_tb

# Homework:

⇒ circ: dmux41

⇒ sctb: dmux41\_tb