NMK20603

Computer Architecture

MY1

Describing Combinational Logic (cont.)

Codes so far:

✔ dmux21, dmux41

✔ add_1b, add_4b

✔ sub_1b, sub_4b

✔ and_4b, or_4b

✔ alu_4b

⇒ all with sctb!

- We have

4-bit ALU!

(4 functions)

MY1

Wrapping Up Combinational Logic

MY1

1-bit Comparator

MY1

iA	iB	LT	GT	EQ
0	0	0	0	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	1

Get SOP equation!

Describe:

cmp_1b

MY1

```
module cmp_1b (iA,iB,oLT,oGT,oEQ);
input iA,iB;
output oLT,oGT,oEQ;
wire oLT,oGT,oEQ;
wire tL,tG; // output cannot be RHS
assign tL = ~iA & iB;
assign tG = iA & ~iB;
assign oLT = tL;
assign oGT = tG;
assign oEQ = ~(tL|tG);
endmodule
```

Practice:

⇒ cmp_1b_tb

Tri-State Logic!

MY1

Special bit values:

⇒ high-impedance ('z')

⇒ unknown ('x')

Syntax

- boolean op

⇒ === vs ==

Tri-state

Buffers

⇒ switch!

Syntax

- if-else op

⇒ ? :

Only for

z-buff

(and tb)

MY1

Describe:

zbuff

```
module zbuff (iE,iA,oY);
input iE,iA;
output oY;
wire oY;
assign oY = iE ? iA : 1'bz;
endmodule
```

Practice:

zbuff_tb

2-4 Decoder

MY1

A1	A0	Y0	Y1	Y2	Y3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

⇒ Active-HI!

MY1

Get SOP equation!

Describe:

decode24

```
module decode24 (iA,oY);
input[1:0] iA;
output[3:0] oY;
wire[3:0] oY;
assign oY[0] = ~iA[1] & ~iA[0];
assign oY[1] = ~iA[1] & iA[0];
assign oY[2] = iA[1] & ~iA[0];
assign oY[3] = iA[1] & iA[0];
endmodule
```

Practice:

decode24_tb

Special: (optional)

⇒ mux41 using decode24

Homework: (optional)

⇒ encode24

⇒ 7-segment decoder

MY1

Homework:

⇒ zbuff (>1b)

⇒ decode38

⇒ cmp_2b

✔ LA2 candidate questions