CS150 Spring 2004
Problem Set #3 Solutions

1) a)

I_3	I_2	I_1	O_1	O_0
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	0	1
1	0	0	1	1
1	0	1	0	1
1	1	0	1	0
1	1	1	0	1

c)

O_1

I_1 \ I_3I_2	00	01	11	10
0	0	1	1	1
1	0	0	0	0

$$O_1 = I_2\bar{I}_1 + I_3\bar{I}_1 = (I_2 + I_3)\bar{I}_1$$

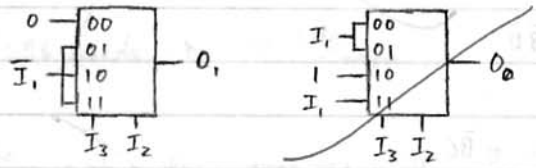
O_0

I_1 \ I_3I_2	00	01	11	10
0	0	0	0	1
1	1	1	1	1

$$O_0 = (I_3 + I_1)(\bar{I}_2 + I_1) = I_1 + I_3\bar{I}_2$$

b) Attached

d)

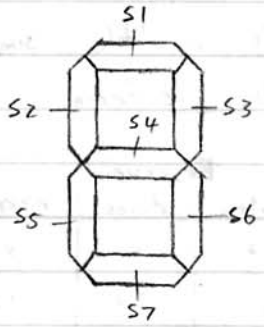


e) The ROM would have to be 8 words by 2 bits/word because there are 8 different input combinations and the output is 2 bits.

f) The answer from (c) is simpler than the answer from (d). The answer from (c) uses a total of four 2-input gates. The muxes in (d) use a total of eight 2-input gates and two 4-input gates.

2) a)

	A	B	C	D	S1	S2	S3	S4	S5	S6	S7
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	1	0
2	0	0	1	0	0	0	1	0	0	1	0
3	0	0	1	1	1	0	1	0	0	1	0
4	0	1	0	0	1	1	1	0	0	1	0
5	0	1	0	1	1	1	1	0	1	1	0
6	0	1	1	0	1	1	1	1	1	1	0
7	0	1	1	1	1	1	1	1	1	1	1
8	1	0	0	0	1	1	1	1	0	1	1
9	1	0	0	1	1	1	1	0	0	1	1
A	1	0	1	0	1	0	1	0	0	1	1
B	1	0	1	1	0	0	1	0	0	1	1
C	1	1	0	0	0	0	0	0	0	1	1
D	1	1	0	1	0	0	0	0	0	0	1
E	1	1	1	0	0	0	0	1	0	0	1
F	1	1	1	1	0	0	0	1	0	1	1



b) Attached

c)

S1

CD \ AB	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	1	1	0	0
10	0	1	0	1

S2

CD \ AB	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	0
10	0	1	0	0

S3

CD \ AB	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	1	1	0	1
10	1	1	0	1

S4

CD \ AB	00	01	11	10
00	0	0	0	1
01	0	0	0	0
11	0	1	*	0
10	0	1	1	0

S5

CD \ AB	00	01	11	10
00	0	0	0	0
01	0	1	0	0
11	0	1	0	0
10	0	1	0	0

S6

CD \ AB	00	01	11	10
00	0	1	1	1
01	1	1	0	1
11	1	1	1	1
10	1	1	0	1

S7

CD \ AB	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	0	1	1	1
10	0	0	1	1

$S1 = \bar{A}B + A\bar{B}\bar{C} + A\bar{B}\bar{D} + \bar{A}CD$

$S2 = \bar{A}B + A\bar{B}\bar{C}$ $\quad S3 = \bar{A}C + \bar{A}B + A\bar{B}$

$S4 = BC + A\bar{B}\bar{C}\bar{D}$ $\quad S5 = \bar{A}BD + \bar{A}BC$

$S6 = \bar{A}D + \bar{A}C + CD + \bar{A}B + A\bar{B} + B\bar{C}\bar{D}$

$S7 = A + BCD$ (16 different terms)

d)

$S1 = \bar{A}B + \bar{A}CD + A\bar{B}\bar{C} + A\bar{B}\bar{D}$

$S2 = \bar{A}B + A\bar{B}\bar{C}$

$S3 = A\bar{B} + \bar{A}B + \bar{B}C$

$S4 = \bar{A}BC + BC + A\bar{B}\bar{C}\bar{D}$

$S5 = \bar{A}BC + \bar{A}BD$

$S6 = A\bar{B} + BCD + \bar{A}B + \bar{B}C + \bar{A}C\bar{D} + \bar{A}D$

$S7 = A\bar{B} + BCD + A$ (14 different terms)

e) (d) is a little simpler when using the number of different terms as the criteria.

3) a)

Reservoir quarters	Reservoir dimes	Reservoir nickels	Output quarters	Output dimes	Output nickels
4+	x	x	4	0	0
3	2+	1+	3	2	1
3	x	0	no change		
3	1	3+	3	1	3
	0	5+		0	5
3-	0	0	no change		
2	5+	x	2	5	0
2	4	2+	2	4	2
	3	4+		3	4
	2	6+		2	6
	1	8+		1	8
2	0	10+	2	0	10
1	x	0	no change		
1	7+	1+	1	7	1
	6	3+		6	3
	5	5+		5	5
	4	7+		4	7
	3	9+		3	9
	2	11+		2	11
	1	13+		1	13
1	0	15+	1	0	15

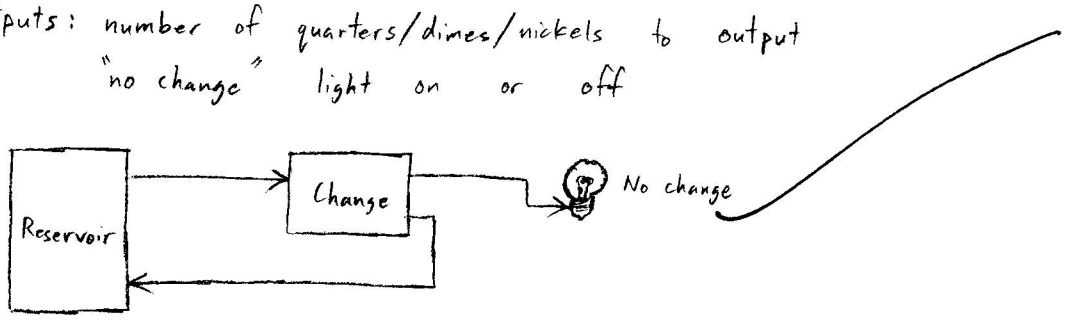
(continued)

0	10+	x	0	10	0
	9	2+		9	2
	8	4+		8	4
	7	6+		7	6
	6	8+		6	8
	5	10+		5	10
	4	12+		4	12
	3	14+		3	14
	2	16		2	16
	1-	4		no change	

27 unique ways

b) Inputs: number of quarters/dimes/nickels in reservoir
bit to indicate that a user has inserted a dollar bill

Outputs: number of quarters/dimes/nickels to output
"no change" light on or off



c) 4096 words by 12 bits/word

The range for... quarters is 0 to 4 (3 bits). (Above 5 quarters is just like having 4)
... dimes is 0 to 10 (4 bits).
... nickels is 0 to 16 (5 bits).

So the input signal and resulting output are 12 bits wide. The number of words = $2^{\#\text{ of input bits}}$.

d) Attached

```
1   // Problem 1, part b
2
3   module lowBit (I, O);
4     input [3:1] I;
5     output [1:0] O;
6     reg [1:0] O;
7
8     assign O[1] = (I[3] | I[2]) & (~I[1]);
9     assign O[0] = I[1] | (I[3] & ~I[2]);
10  endmodule
11
12
13
14  // Problem 2, part b
15
16  module toMartian (in, led);
17    input [3:0] in;
18    output [7:1] led;
19    reg [7:1] led;
20
21    assign led[7] = in[3] | (in[2] & in[1] & in[0]);
22    assign led[6] = (~in[3] & in[2]) | (in[3] & ~in[2]) | (in[2] & ~in[1] & ~in[0])
23                  | (~in[3] & in[0]) | (~in[3] & in[1]) | (in[1] & in[0]);
24    assign led[5] = (~in[3] & in[2] & in[0]) | (~in[3] & in[2] & in[1]);
25    assign led[4] = (in[2] & in[1]) | (in[3] & ~in[2] & ~in[1] & ~in[0]);
26    assign led[3] = (~in[3] & in[2]) | (in[3] & ~in[2] & in[1]);
27    assign led[2] = (~in[3] & in[2]) | (in[3] & ~in[2] & ~in[1]);
28    assign led[1] = (~in[3] & in[2]) | (in[3] & ~in[2] & ~in[1]) | (in[3] & ~in[2] & ~in
    [0])
29                  | (~in[3] & in[1] & in[0]);
30  endmodule
31
32
33
34  // Problem 3, part d
35
36  module changeMe (inq, ind, inn, outq, outd, outn, nc, go);
37    input [2:0] inq;
38    input [3:0] ind;
39    input [4:0] inn;
40    input go;
41    output [2:0] out1;
42    output [3:0] outd;
43    output [4:0] outn;
44    output nc;
45    reg [2:0] outq, q;
46    reg [3:0] outd, d;
47    reg [4:0] outn, n;
48    reg nc;
49
50    integer i, j, k;
51
52    always @ (inq or ind or inn) begin
53      begin : find_change
54        for (i = inq; i >= 0; i = i - 1) begin
55          for (j = ind; j >= 0; j = j - 1) begin
56            for (k = inn; k >= 0; k = k - 1) begin
57              if ((inq*25) + (ind*10) + (inn*5) == 100) begin
58                nc <= 1'b0;
59                q <= i; d <= j; n <= k;
60                disable find_change;
```

```
61              end
62            end
63          end
64        end
65      end
66    end
67
68    always @ (posedge go) begin
69      outq <= q;
70      outd <= d;
71      outn <= n;
72    end
73  endmodule
```