Digital System Design and Lab: HW4

Lo Chun, Chou
R13922136

May 18, 2025

1

(1)

Present State			Next State		
C	B	A	C^+	B^+	A^+
0	0	1	0	1	1
0	1	1	0	1	0
0	1	0	1	1	0
1	1	0	1	1	1
1	1	1	1	0	1
1	0	1	1	0	0
1	0	0	0	0	1

BA \ C	0	1
00	X	0
01	0	1
11	0	1
10	1	1

(a) C^+

BA \ C	0	1
00	X	0
01	1	0
11	1	0
10	1	1

(b) B^+

BA \ C	0	1
00	X	1
01	1	0
11	0	1
10	0	1

(c) A^+

Figure 1: K-maps

(2)

The K-maps for D flip-flop are the same as the K-maps for C^+, B^+, and A^+ in (1), since we assign the values of D_C, D_B, and D_A to be C^+, B^+, and A^+ respectively.

Thus, the following D_C is the same as C^+:

		C	
		0	1
BA	00	X	0
	01	0	1
	11	0	1
	10	1	1

Figure 2: D flip-flop (D_C)

The minimum SOP expression for D_C is:

$$D_C = AC + A'B$$

(3)

Since C is toggled for $CBA = \{010, 100\}$, we can derive the following K-map for T_C:

BA \ C	0	1
00	X	1
01	0	0
11	0	0
10	1	0

Figure 3: T flip-flop (T_C)

The minimum SOP expression for T_C is:

$$T_C = A'B' + A'C'$$

(4)

From the truth table is subproblem (1), we have:

$$\{B, B^+\} = \begin{cases} \{0,0\} & \text{for } CBA = \{101, 100\} \rightarrow \{S, R\} = \{0, X\} \\ \{0,1\} & \text{for } CBA = \{001\} \rightarrow \{S, R\} = \{1, 0\} \\ \{1,0\} & \text{for } CBA = \{111\} \rightarrow \{S, R\} = \{0, 1\} \\ \{1,1\} & \text{for } CBA = \{011, 010, 110\} \rightarrow \{S, R\} = \{X, 0\} \end{cases}$$

BA \ C	0	1
00	X	0
01	1	0
11	X	0
10	X	X

(a) S_B

BA \ C	0	1
00	X	X
01	0	X
11	0	1
10	0	0

(b) R_B

The minimum SOP expression for S_B, R_B are:

$$S_B = C'$$
$$R_B = AC$$

(5)

From the truth table is subproblem (1), we have:

$$\{A, A^+\} = \begin{cases} \{0,0\} & \text{for } CBA = \{010\} \rightarrow \{J,K\} = \{0, X\} \\ \{0,1\} & \text{for } CBA = \{110, 100\} \rightarrow \{J,K\} = \{1, X\} \\ \{1,0\} & \text{for } CBA = \{011, 101\} \rightarrow \{J,K\} = \{X, 1\} \\ \{1,1\} & \text{for } CBA = \{001, 111\} \rightarrow \{J,K\} = \{X, 0\} \end{cases}$$

BA \ C	0	1
00	X	1
01	X	X
11	X	X
10	0	1

(a) J_A

BA \ C	0	1
00	X	X
01	0	1
11	1	0
10	X	X

(b) K_A

The minimum SOP expression for J_A, K_A are:

$$J_A = C$$
$$K_A = BC' + B'C$$

2

(1)

Following the steps to construct the state table at lecture slides 13, p. 15, we first determine the flip-flop and output equations:

$$
\begin{aligned}
&J_1 = X,\\
&K_1 = X \barwedge Q_2' = (XQ_2')' = X' + Q_2,\\
&J_2 = X,\\
&K_2 = Q_1 \barwedge X = (Q_1X)' = Q_1' + X',\\
&Z = Q_2' \oplus X = Q_2'X' + Q_2X
\end{aligned}
$$

Then, using the next state equation for JK flip-flop, which is:

$$Q^+ = JQ' + K'Q$$

we have:

$$
\begin{aligned}
&Q_1^+ = J_1Q_1' + K_1'Q_1 = XQ_1' + (X' + Q_2)'Q_1 = \underline{XQ_1' + XQ_1Q_2'}\\
&Q_2^+ = J_2Q_2' + K_2'Q_2 = XQ_2' + (Q_1' + X')'Q_2 = \underline{XQ_2' + XQ_1Q_2}\\
&Z = \underline{Q_2'X' + Q_2X}
\end{aligned}
$$

We then plot the next state map for each flip-flop:

Q_1Q_2 \ X	0	1
00	0	1
01	0	1
11	0	0
10	0	1

(a) Q_1^+

Q_1Q_2 \ X	0	1
00	0	1
01	0	0
11	0	1
10	0	1

(b) Q_2^+

Q_1Q_2 \ X	0	1
00	1	0
01	0	1
11	0	1
10	1	0

(c) Z

We can then use the K-maps to form the state table:

Q_1Q_2	$Q_1^+Q_2^+$		Z	
	$X=0$	$X=1$	$X=0$	$X=1$
00	00	11	1	0
01	00	10	0	1
11	00	01	0	1
10	00	11	1	0

(2)

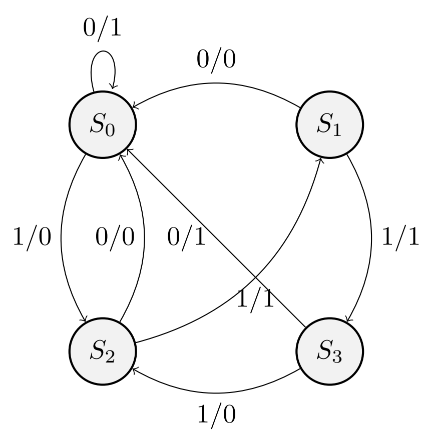
We first replace part of the resulting state table in subproblem (1) with the following symbols:

$$S_0 = 00, \quad S_1 = 01, \quad S_2 = 11, \quad S_3 = 10$$

Then we'll have:

Q_1Q_2	$Q_1^+Q_2^+$		Z	
	$X=0$	$X=1$	$X=0$	$X=1$
S_0	S_0	S_2	1	0
S_1	S_0	S_3	0	1
S_2	S_0	S_1	0	1
S_3	S_0	S_2	1	0

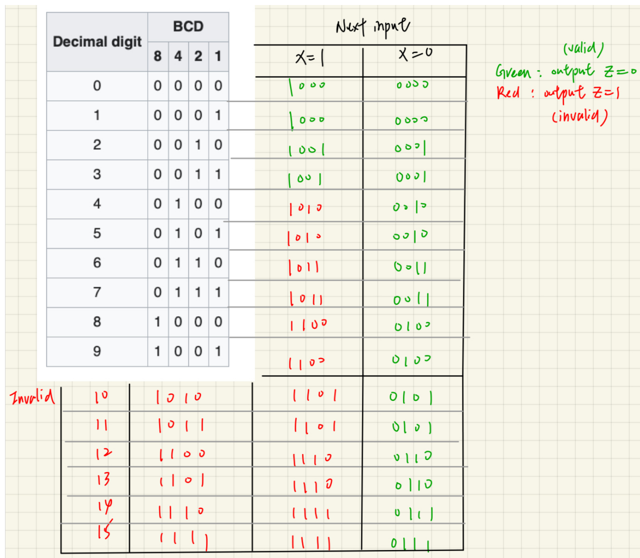
Using this table, we can construct the state diagram:



3

(1)

From the following image, we can check that for all possible situations 0000 to 1111, what the next state should be after the input $X = \{0, 1\}$ is applied.


Next input

(valid)
Green : output Z=0
Red : output Z=1
(invalid)

Decimal digit	BCD 8 4 2 1	X=1	X=0
0	0 0 0 0	1000	0000
1	0 0 0 1	1000	0000
2	0 0 1 0	1001	0001
3	0 0 1 1	1001	0001
4	0 1 0 0	1010	0010
5	0 1 0 1	1010	0010
6	0 1 1 0	1011	0011
7	0 1 1 1	1011	0011
8	1 0 0 0	1100	0100
9	1 0 0 1	1100	0100
Invalid 10	1010	1101	0101
11	1011	1101	0101
12	1100	1110	0110
13	1101	1110	0110
14	1110	1111	0111
15	1111	1111	0111

Let S_0 be the initial state, which assumes all previous inputs were 0 as given in the problem. We then add two states S_1 and S_2 to represent the cases when the current input along with the previous 3 inputs is valid and invalid respectively.

This also means that the green states in the above image would go to S_1 and the red states would go to S_2.

Thus, we can form the state table as follows:

Current State	Input		Z	
	$X = 0$	$X = 1$	$X = 0$	$X = 1$
S_0	S_0	S_1	0	0
S_1	S_1	S_1, S_2	0	0
S_2	S_1, S_2	S_1, S_2	1	1

(2)

We shall use the following four states, with each of them containing the previous 4 inputs as follows:

$$S_0 = \{0000, 0001, 0010, 0011\}$$
$$S_1 = \{0100, 0101, 0110, 0111\}$$
$$S_2 = \{1000, 1001\}$$
$$S_3 = \{1010, 1011, 1100, 1101, 1110, 1111\}$$

For S_0, this state contains all the cases when no matter we receive any input ($X = 0$ or 1), it would still be valid.

For S_1, this state contains the cases when it receives 0 as input, it would go to S_0, and would be invalid (go to S_3) if it receives 1.

For S_2, this state contains the cases when it receives 0 as input, it would go to S_1, and would be invalid (go to S_3) if it receives 1. (The distinction between S_1, S_2 is that it is possible for S_1 to enter the next state S_0, which would be valid no matter what next input is.)

For S_3, this state contains all the invalid cases.

We can then form the state table as follows:

Current State	Input		Z	
	$X = 0$	$X = 1$	$X = 0$	$X = 1$
S_0	S_0	S_0	0	0
S_1	S_0	S_3	0	1
S_2	S_1	S_3	0	1
S_3	S_1	S_3	0	1

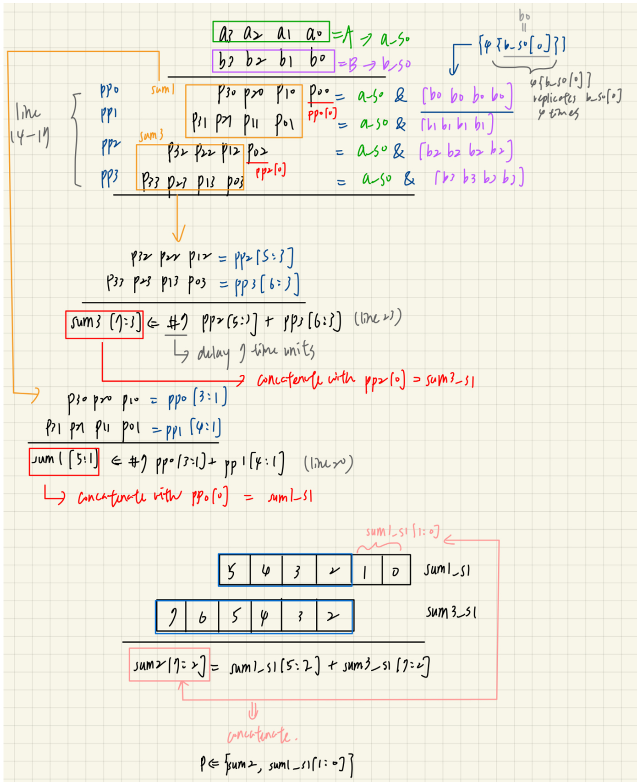
4

(1)

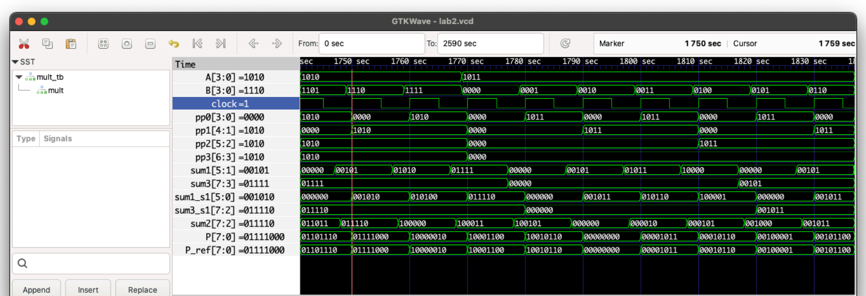
The following screenshot is the module mult_fast

```verilog
// pipelined fast multiplier
module mult_fast(
    output reg[7:0] P,  // product
    input[3:0] A, B,    // multiplicand and multiplier
    input clk           // clock (posedge)
    );
    // stage 0 (input)
    reg[3:0] a_s0, b_s0;
    always @(posedge clk) begin
        a_s0 <= A;
        b_s0 <= B;
    end
    // stage 1: sum each two rows of partial products
    wire[3:0] pp0 = a_s0 & {4{b_s0[0]}}; // ignore the delays of AND gates
    wire[4:1] pp1 = a_s0 & {4{b_s0[1]}}; // ignore the delays of AND gates
    wire[5:2] pp2 = a_s0 & {4{b_s0[2]}}; // ignore the delays of AND gates
    wire[6:3] pp3 = a_s0 & {4{b_s0[3]}}; // ignore the delays of AND gates
    reg[5:1] sum1;
    always @(pp0, pp1)
        sum1[5:1] <= #7 pp0[3:1] + pp1[4:1]; // delay of the 4-bit adder
    reg[7:3] sum3;
    always @(pp2, pp3)
        sum3[7:3] <= #7 pp2[5:3] + pp3[6:3]; // delay of the 4-bit adder
    reg[5:0] sum1_s1;
    reg[7:2] sum3_s1;
    always @(posedge clk) begin
        sum1_s1 <= {sum1, pp0[0]};
        sum3_s1 <= {sum3, pp2[2]};
    end
    // stage 2 (outout)
    reg[7:2] sum2;
    always @(sum1_s1, sum3_s1)
        sum2[7:2] <= #8 sum1_s1[5:2] + sum3_s1[7:2]; // delay of the 6-bit adder
    always @(posedge clk) begin
        P <= {sum2, sum1_s1[1:0]};
    end
endmodule
```

And also not required, the following figure shows how the indices are decided:



(2)



(3)

We can find the latency by the above waveform, for example, if we look at the input at the input at 1750, we have inputs $A = 1010$ and $B = 1110$, which corresponds to 10 and 14 respectively.

Thus, the output P at 1750 is $10 \times 14 = 140$, which can be represented as 10001100 in binary, and this correct product appears at 1770.

Therfore, the latency is $1770 - 1750 = 20$ ticks.

5

(1)

The minimum clock cycle is 8 ticks

(2)

The waveform is as follows: