CPE 322
Digital Hardware Design Fundamentals

Electrical and Computer Engineering

Finite State Machine Representation &
Verilog

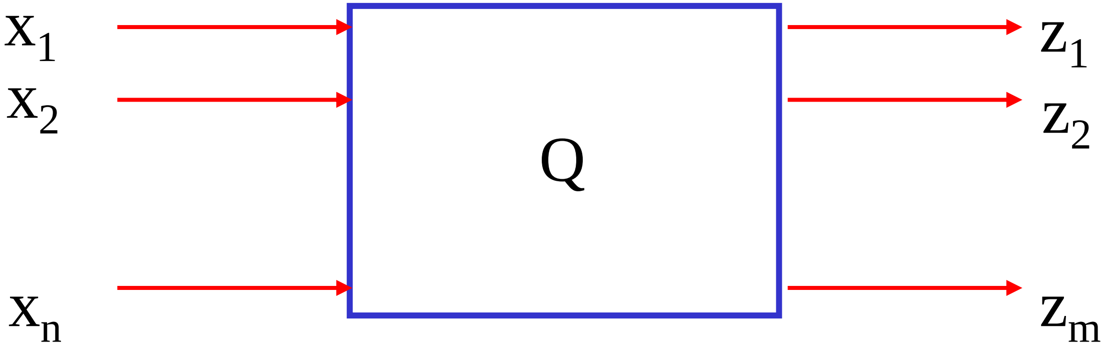
Sequential Networks

- Have memory (state)
  - Present state depends not only on the current input, but also on all previous inputs (history)
  - Future state depends on the current input and state

$$X = x_1\ x_2 \ldots\ x_n$$

$$Q = Q_1\ Q_2 \ldots\ Q_k$$

$$Z = z_1\ z_2 \ldots\ z_m$$



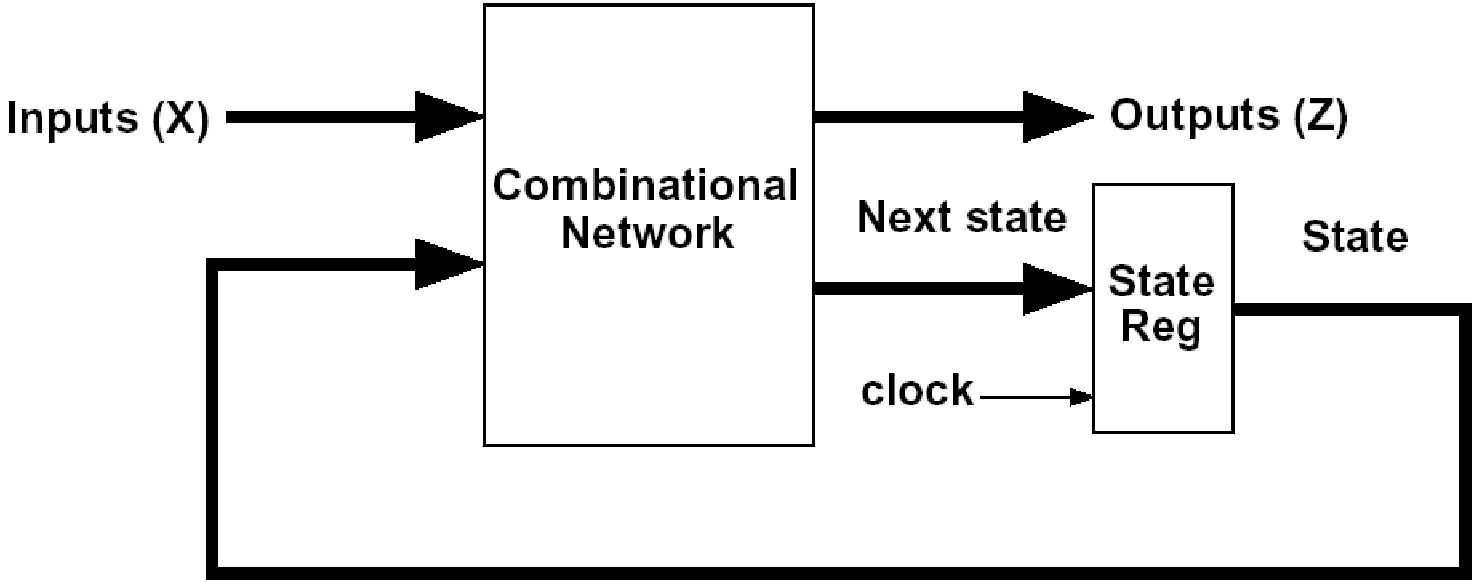
$$Z(t) = F(X(t), Q(t))$$

$$Q(t^+) = G(X(t), Q(t))$$

Flip-flops are commonly used as storage devices: D-FF, JK-FF, T-FF

Mealy Sequential Networks

General model of Mealy Sequential Network



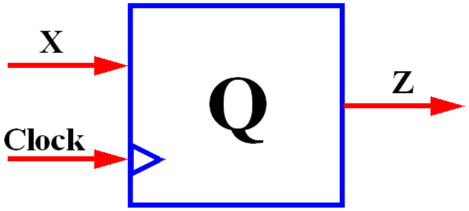
$$Q(t^{+}) = G(X(t), Q(t)) \qquad Z(t) = F(X(t), Q(t))$$

(1) X inputs are changed to a new value

(2) After a delay, the Z outputs and next state appear at the output of CM (input of State Register)

(3) The next state is clocked into the state register and the state changes

An Example: 8421 BCD to Excess3 BCD Code Converter

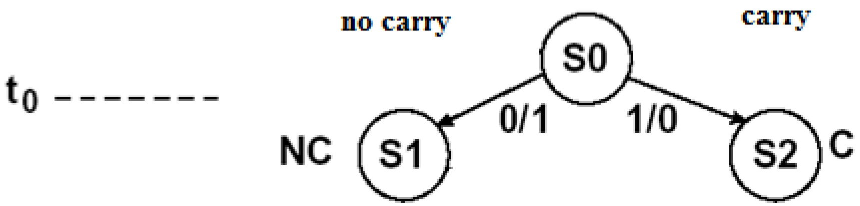


X (inputs)				Z (outputs)			
t3	t2	t1	t0	t3	t2	t1	t0
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

State Graph
BCD to Exess3 Code Converter

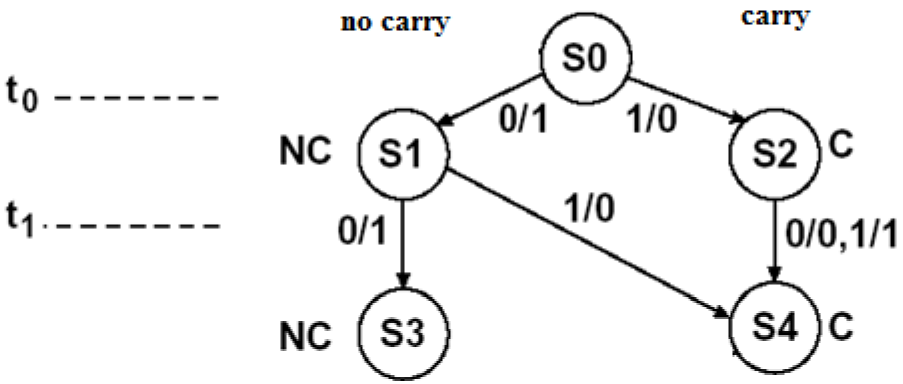
State Graph
BCD to Exess3 Code Converter

X/Z



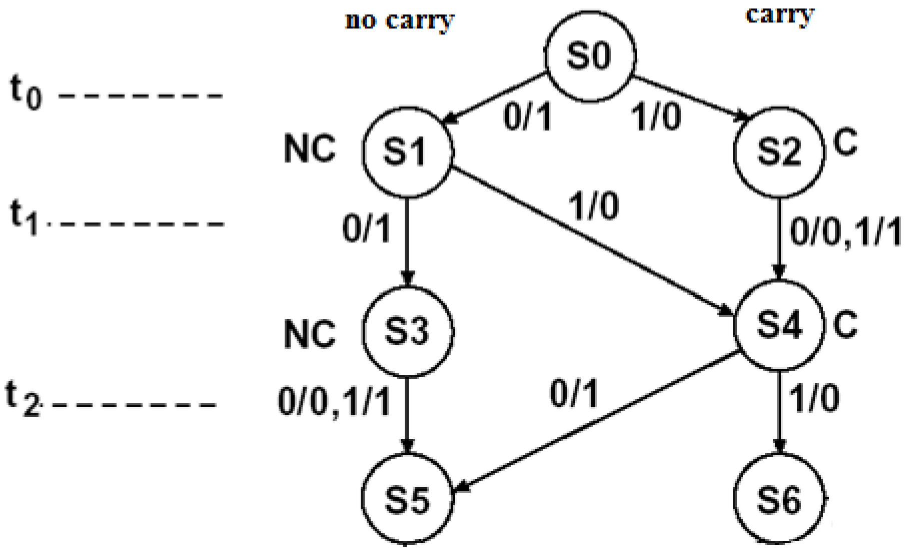
State Graph
BCD to Exess3 Code Converter

X/Z



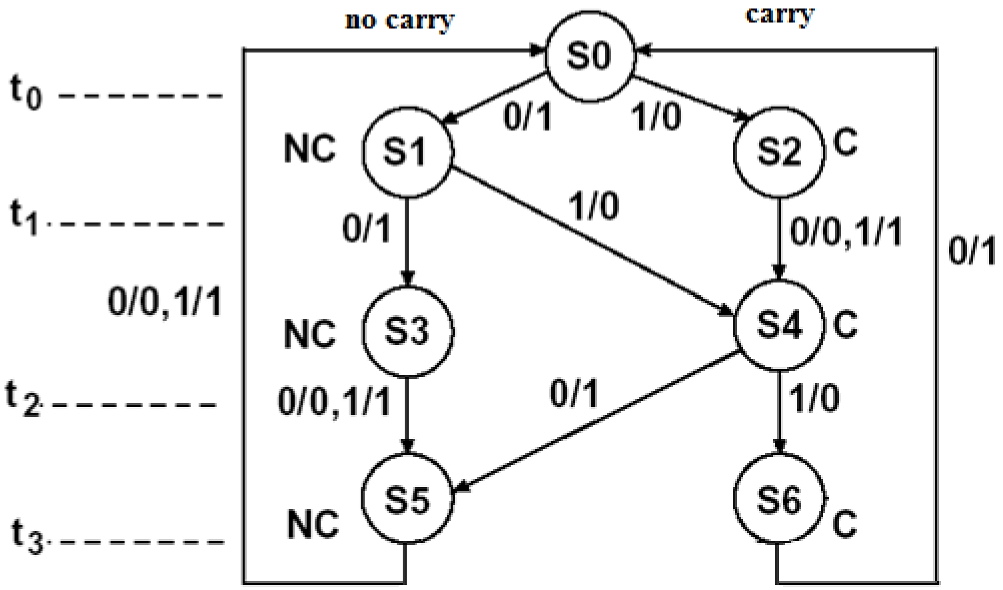
State Graph
BCD to Exess3 Code Converter

X/Z

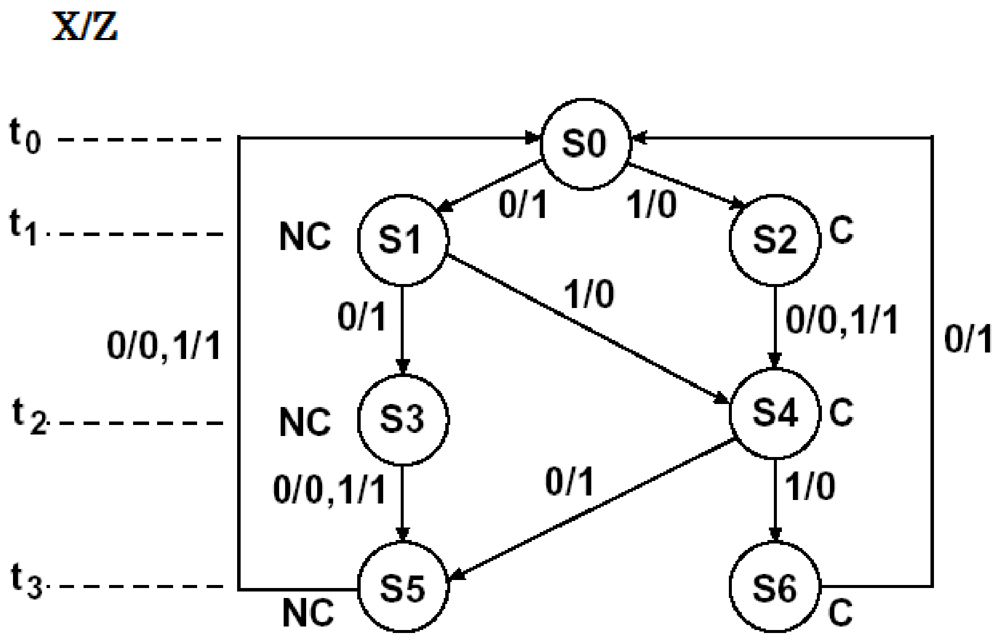


State Graph
BCD to Exess3 Code Converter

X/Z



State Graph and Table for BCD to Exess3 Code Converter

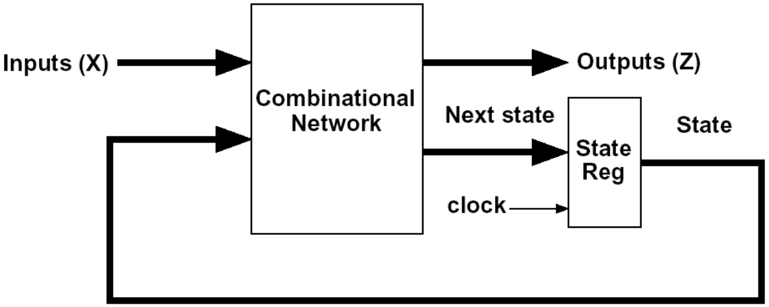


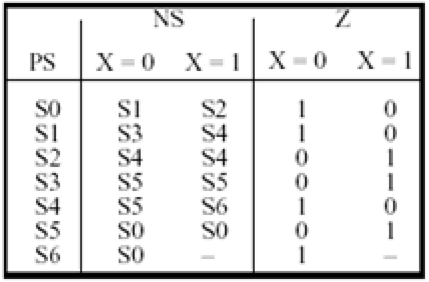
	NS		Z	
PS	X = 0	X = 1	X = 0	X = 1
S0	S1	S2	1	0
S1	S3	S4	1	0
S2	S4	S4	0	1
S3	S5	S5	0	1
S4	S5	S6	1	0
S5	S0	S0	0	1
S6	S0	–	1	–

Behavioral Verilog HDL Model

8421 BCD to Excess3 Code Converter (Mealy FSM)

```
// bcd to execess 3 converter
// Mealy Implementation
// Behavioral Model
module bcd_ex3_mealy(input X,CLK,output reg Z);
      reg [2:0]  State=0,Nextstate=0;

   // Combinational Network
   always @ (State,X)
      case (State)
         0 : if (!X) begin Nextstate=1;Z=1; end
             else begin Nextstate=2; Z=0; end
         1 : if (!X) begin Nextstate=3; Z=1; end
             else begin Nextstate=4; Z=0; end
         2 : if (!X) begin Nextstate=4; Z=0; end
             else begin Nextstate=4; Z=1; end
         3 : if (!X) begin Nextstate=5; Z=0; end
             else begin Nextstate=5; Z=1; end
         4 : if (!X) begin Nextstate=5; Z=1; end
             else begin Nextstate=6; Z=0; end
         5 : if (!X) begin Nextstate=0; Z=0; end
             else begin Nextstate=0; Z=1; end
         6 : if (!X) begin Nextstate=0; Z=1; end
             else begin Nextstate=0; Z=1'b?; end
        default : begin Nextstate=0; Z=0;end
      endcase

   // State Reg Portion of Design
   always @ (posedge CLK)
      State=Nextstate;

endmodule
```





	NS		Z	
PS	X = 0	X = 1	X = 0	X = 1
S0	S1	S2	1	0
S1	S3	S4	1	0
S2	S4	S4	0	1
S3	S5	S5	0	1
S4	S5	S6	1	0
S5	S0	S0	0	1
S6	S0	–	1	–

Two always sections:

- the first represents the combinational network;
- the second represents the state register

State Assignment Rules

I. States which have the same next state (NS) for a given input should be given adjacent assignments (look at the columns of the state table).

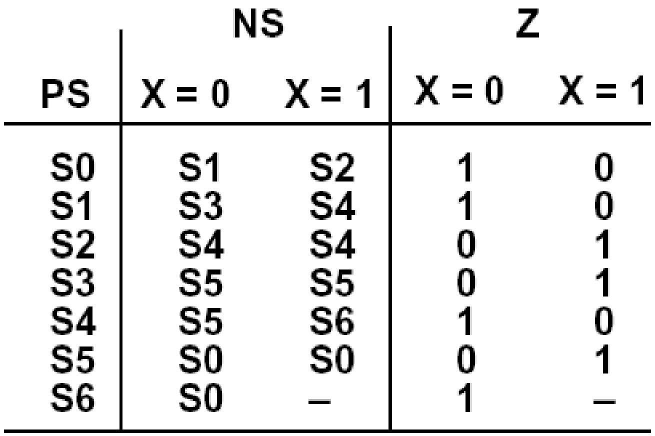
II. States which are the next states of the same state should be given adjacent assignments (look at the rows).

III. States which have the same output for a given input should be given adjacent assignments.

I. (1,2) (3,4) (5,6) (in the X=1 column, S_1 and S_2 both have NS S_4;
in the X=0 column, S_3 & S_4 have NS S_5, and S_5 & S_6 have NS S_0)

II. (1,2) (3,4) (5,6) (S_1 & S_2 are NS of S_0; S_3 & S_4 are NS of S_1;
and S_5 & S_6 are NS of S_4)

III. (0,1,4,6) (2,3,5)



	NS		Z	
PS	X = 0	X = 1	X = 0	X = 1
S0	S1	S2	1	0
S1	S3	S4	1	0
S2	S4	S4	0	1
S3	S5	S5	0	1
S4	S5	S6	1	0
S5	S0	S0	0	1
S6	S0	–	1	–

State Assignment Rules

I. States which have the same next state (NS) for a given input should be given adjacent assignments (look at the columns of the state table).

II. States which are the next states of the same state should be given adjacent assignments (look at the rows).

III. States which have the same output for a given input should be given adjacent assignments.

I. (1,2) (3,4) (5,6) (in the X=1 column, S_1 and S_2 both have NS S_4;
in the X=0 column, S_3 & S_4 have NS S_5, and S_5 & S_6 have NS S_0)

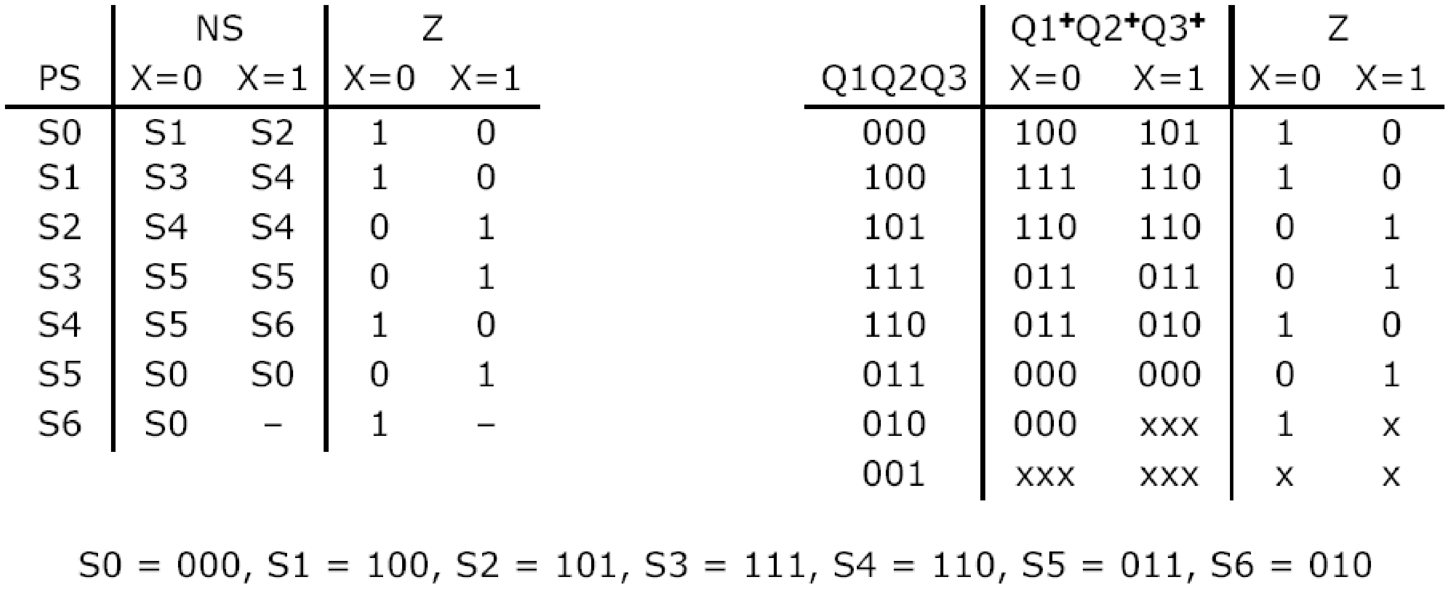
II. (1,2) (3,4) (5,6) (S_1 & S_2 are NS of S_0; S_3 & S_4 are NS of S_1;
and S_5 & S_6 are NS of S_4)

III. (0,1,4,6) (2,3,5)

	NS		Z	
PS	X = 0	X = 1	X = 0	X = 1
S0	S1	S2	1	0
S1	S3	S4	1	0
S2	S4	S4	0	1
S3	S5	S5	0	1
S4	S5	S6	1	0
S5	S0	S0	0	1
S6	S0	–	1	–

Q2 Q3 \ Q1	0	1
00	S0	S1
01		S2
11	S5	S3
10	S6	S4

Transition Table

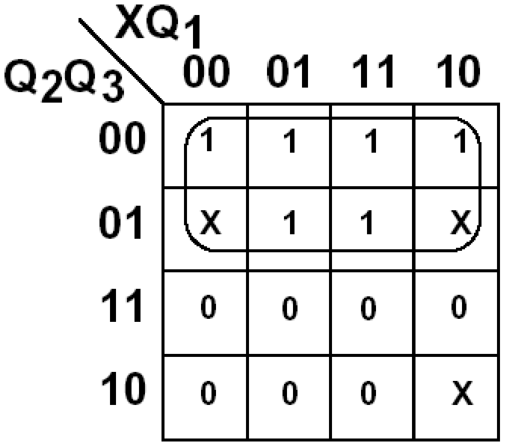


	NS		Z	
PS	X=0	X=1	X=0	X=1
S0	S1	S2	1	0
S1	S3	S4	1	0
S2	S4	S4	0	1
S3	S5	S5	0	1
S4	S5	S6	1	0
S5	S0	S0	0	1
S6	S0	–	1	–

	$Q1^+Q2^+Q3^+$		Z	
Q1Q2Q3	X=0	X=1	X=0	X=1
000	100	101	1	0
100	111	110	1	0
101	110	110	0	1
111	011	011	0	1
110	011	010	1	0
011	000	000	0	1
010	000	xxx	1	x
001	xxx	xxx	x	x

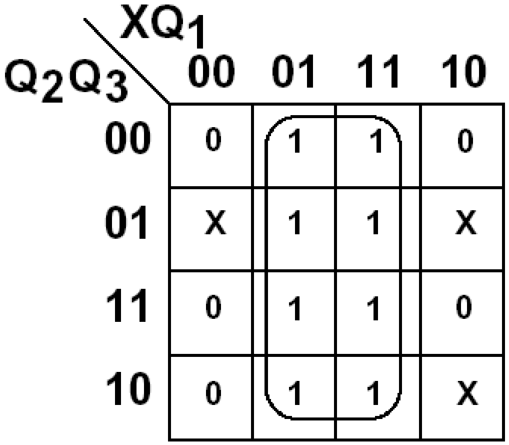
S0 = 000, S1 = 100, S2 = 101, S3 = 111, S4 = 110, S5 = 011, S6 = 010

K-maps



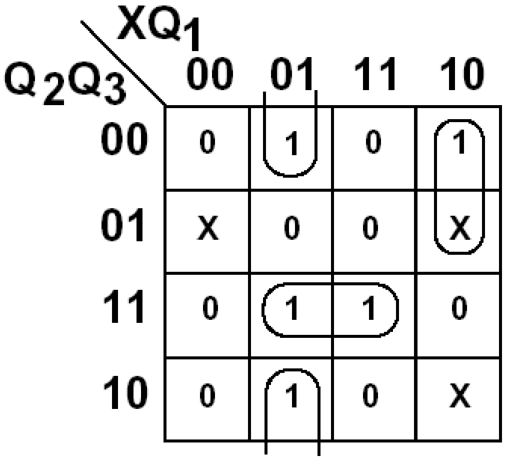
Q_2Q_3 \ XQ_1	00	01	11	10
00	1	1	1	1
01	X	1	1	X
11	0	0	0	0
10	0	0	0	X

$D_1 = Q_1^+ = Q_2'$



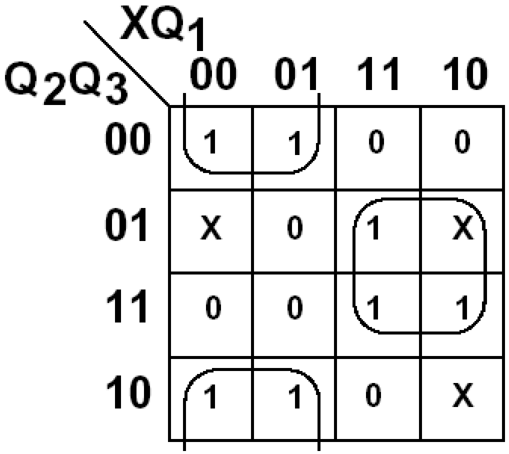
Q_2Q_3 \ XQ_1	00	01	11	10
00	0	1	1	0
01	X	1	1	X
11	0	1	1	0
10	0	1	1	X

$D_2 = Q_2^+ = Q_1$



Q_2Q_3 \ XQ_1	00	01	11	10
00	0	1	0	1
01	X	0	0	X
11	0	1	1	0
10	0	1	0	X

$D_3 = Q_3^+ = Q_1Q_2Q_3 + X'Q_1Q_3' + XQ_1'Q_2'$



Q_2Q_3 \ XQ_1	00	01	11	10
00	1	1	0	0
01	X	0	1	X
11	0	0	1	1
10	1	1	0	X

$Z = X'Q_3' + XQ_3$

Data Flow Representation Verilog HDL

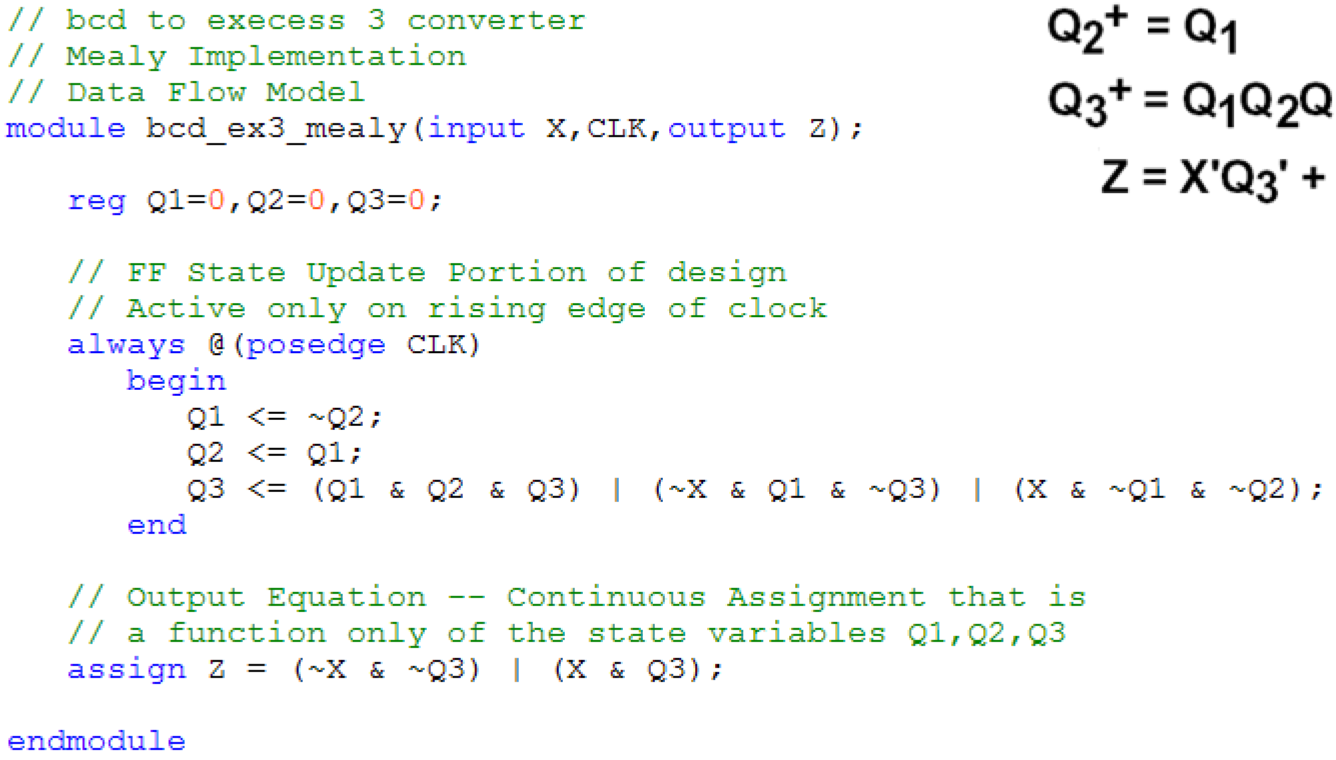
8421 BCD to Excess3 Code Converter (Mealy FSM)

$$Q_1^+ = Q_2'$$

$$Q_2^+ = Q_1$$

$$Q_3^+ = Q_1Q_2Q_3 + X'Q_1Q_3' + XQ_1'Q_2'$$

$$Z = X'Q_3' + XQ_3$$



```
// bcd to execess 3 converter
// Mealy Implementation
// Data Flow Model
module bcd_ex3_mealy(input X,CLK,output Z);

   reg Q1=0,Q2=0,Q3=0;

   // FF State Update Portion of design
   // Active only on rising edge of clock
   always @(posedge CLK)
      begin
         Q1 <= ~Q2;
         Q2 <= Q1;
         Q3 <= (Q1 & Q2 & Q3) | (~X & Q1 & ~Q3) | (X & ~Q1 & ~Q2);
      end

   // Output Equation -- Continuous Assignment that is
   // a function only of the state variables Q1,Q2,Q3
   assign Z = (~X & ~Q3) | (X & Q3);

endmodule
```

Schematic Realization

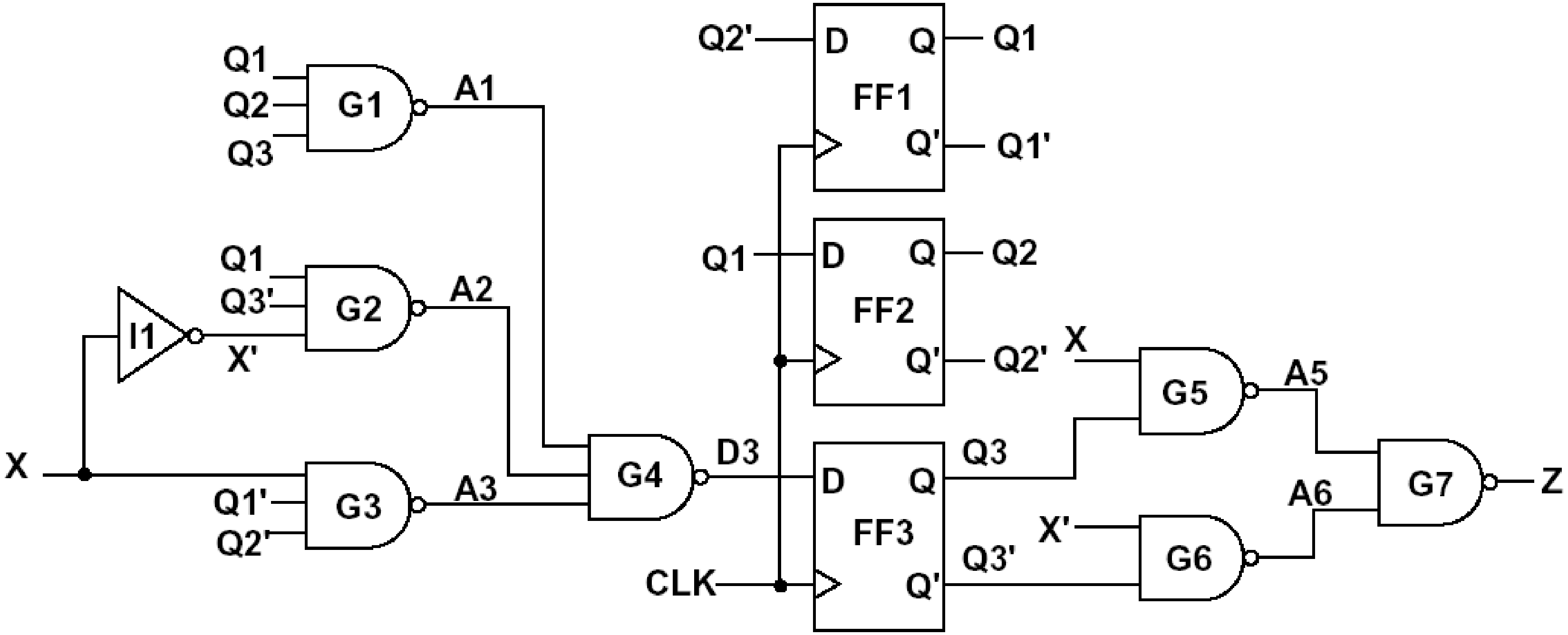
8421 BCD to Excess3 Code Converter (Mealy FSM)

$Q_1^+ = Q_2'$

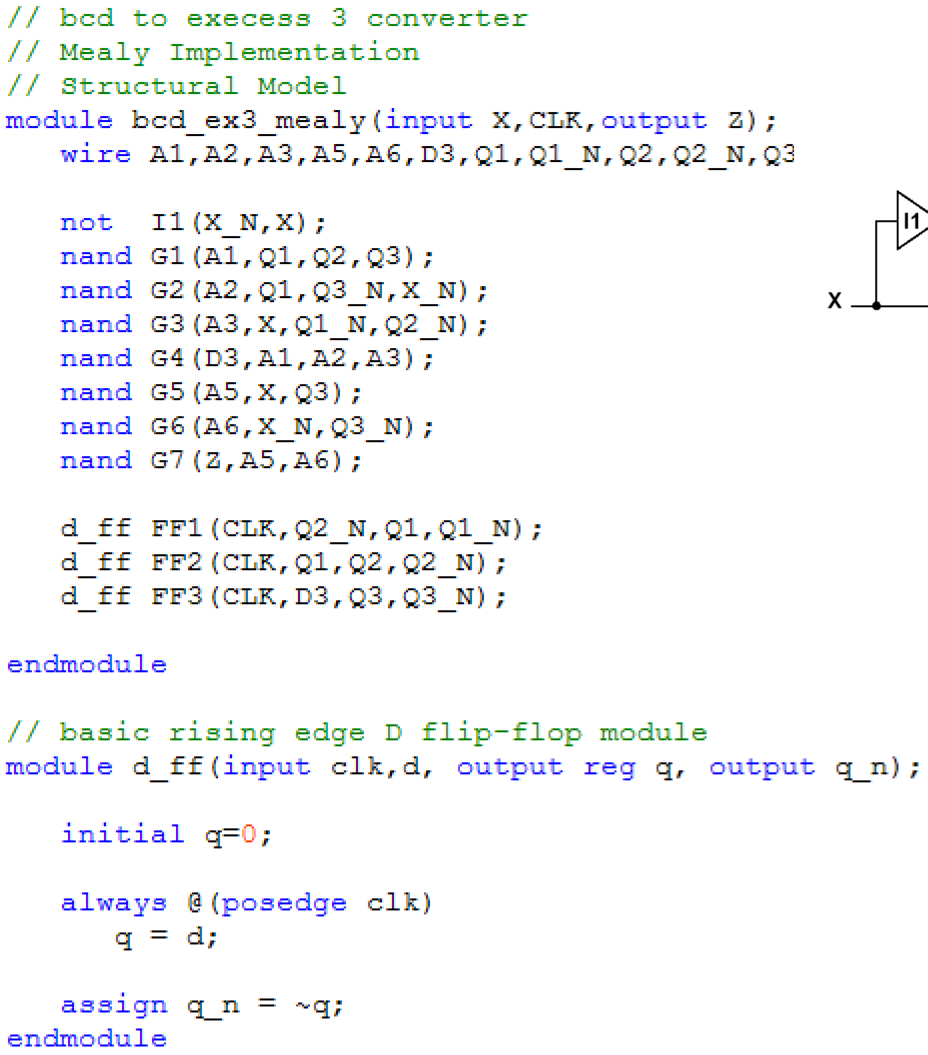
$Q_2^+ = Q_1$

$Q_3^+ = Q_1Q_2Q_3 + X'Q_1Q_3' + XQ_1'Q_2'$

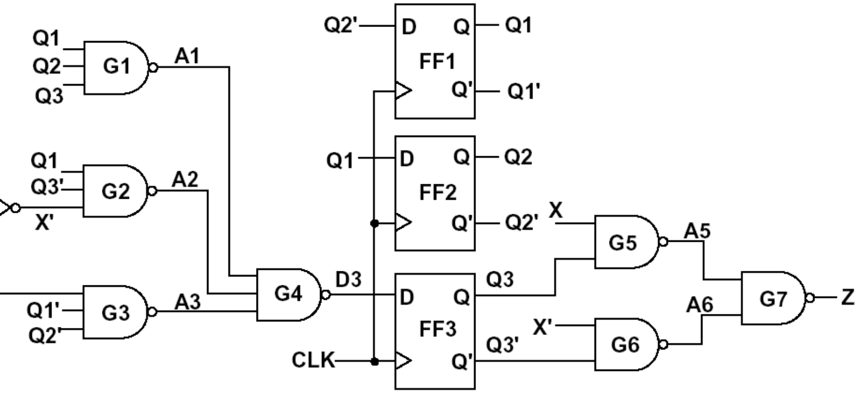
$Z = X'Q_3' + XQ_3$



Structural Representation Verilog HDL
8421 BCD to Excess3 Code Converter (Mealy FSM)



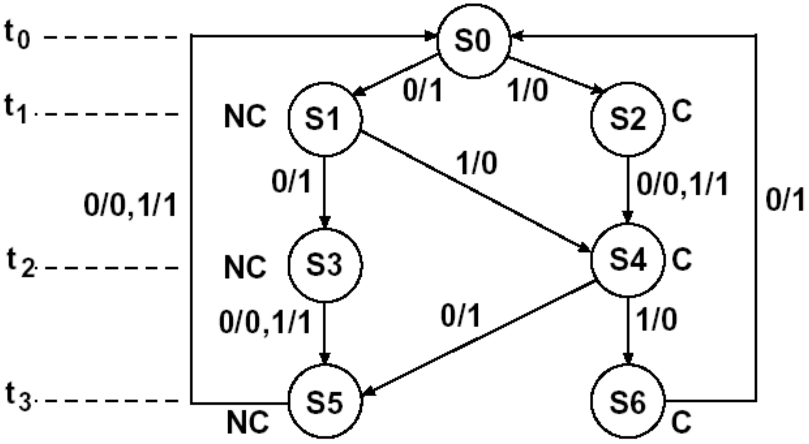
```
// bcd to execess 3 converter
// Mealy Implementation
// Structural Model
module bcd_ex3_mealy(input X,CLK,output Z);
   wire A1,A2,A3,A5,A6,D3,Q1,Q1_N,Q2,Q2_N,Q3

   not  I1(X_N,X);
   nand G1(A1,Q1,Q2,Q3);
   nand G2(A2,Q1,Q3_N,X_N);
   nand G3(A3,X,Q1_N,Q2_N);
   nand G4(D3,A1,A2,A3);
   nand G5(A5,X,Q3);
   nand G6(A6,X_N,Q3_N);
   nand G7(Z,A5,A6);

   d_ff FF1(CLK,Q2_N,Q1,Q1_N);
   d_ff FF2(CLK,Q1,Q2,Q2_N);
   d_ff FF3(CLK,D3,Q3,Q3_N);

endmodule

// basic rising edge D flip-flop module
module d_ff(input clk,d, output reg q, output q_n);

   initial q=0;

   always @(posedge clk)
      q = d;

   assign q_n = ~q;
endmodule
```

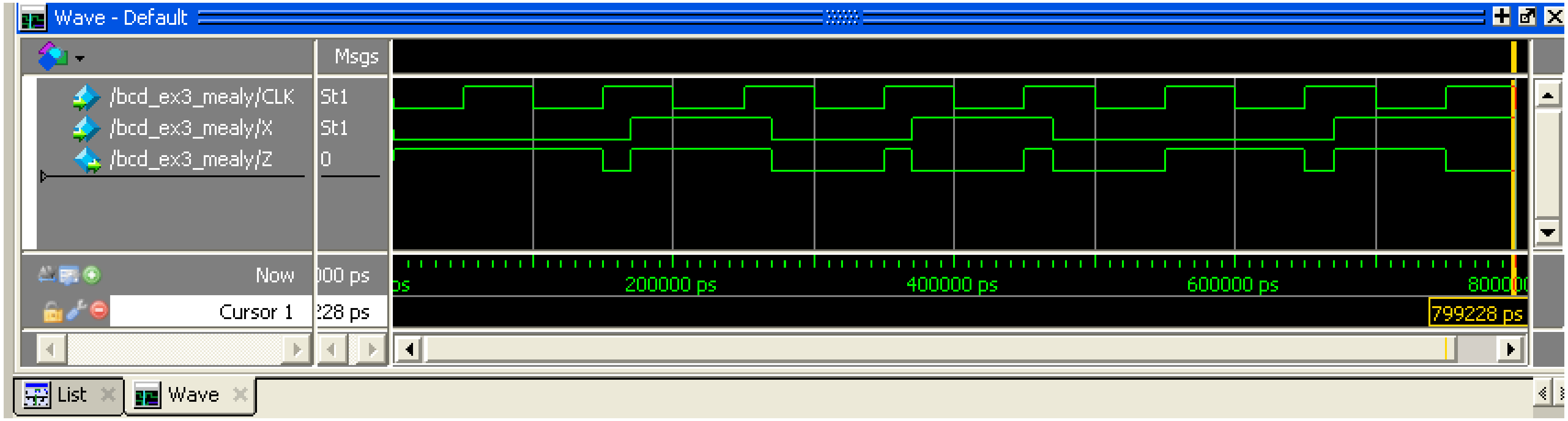


Sequential Network Timing

- Code converter
  - X = 0010_1001 => Z = 1110_0011

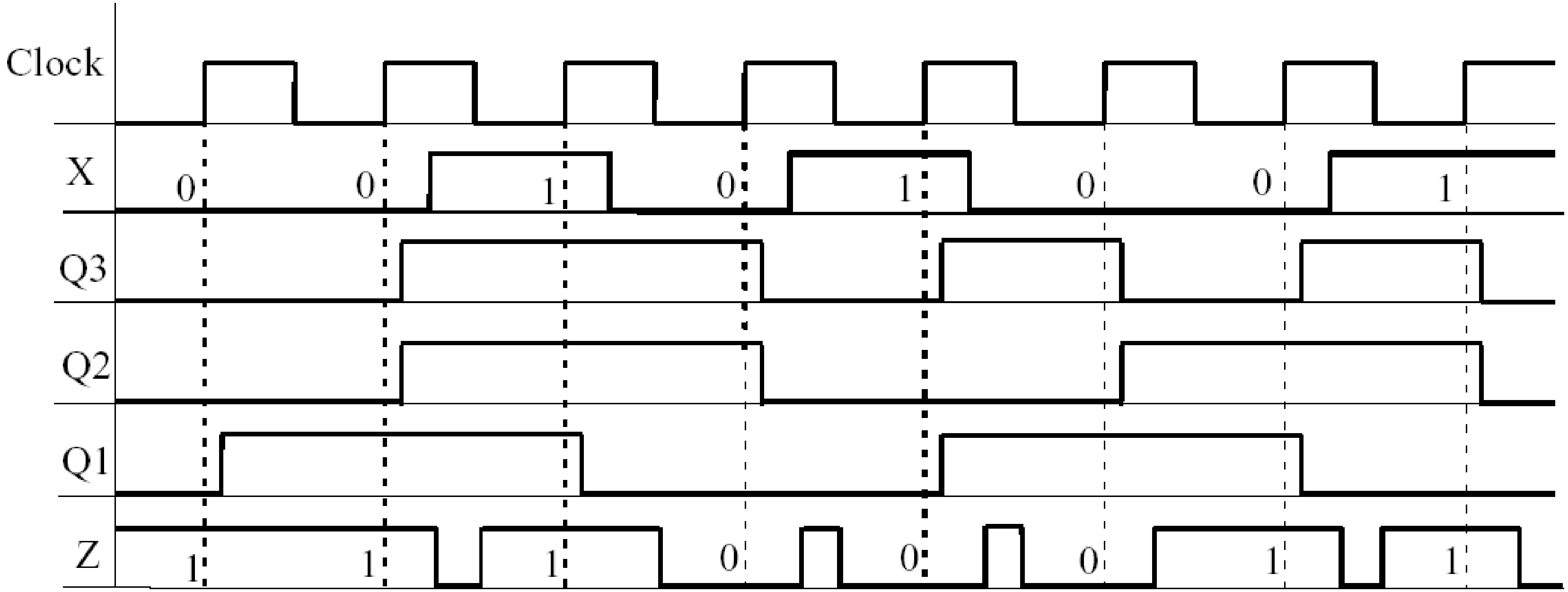
Changes in X are not synchronized with active clock edge => glitches (false output), e.g. at tb



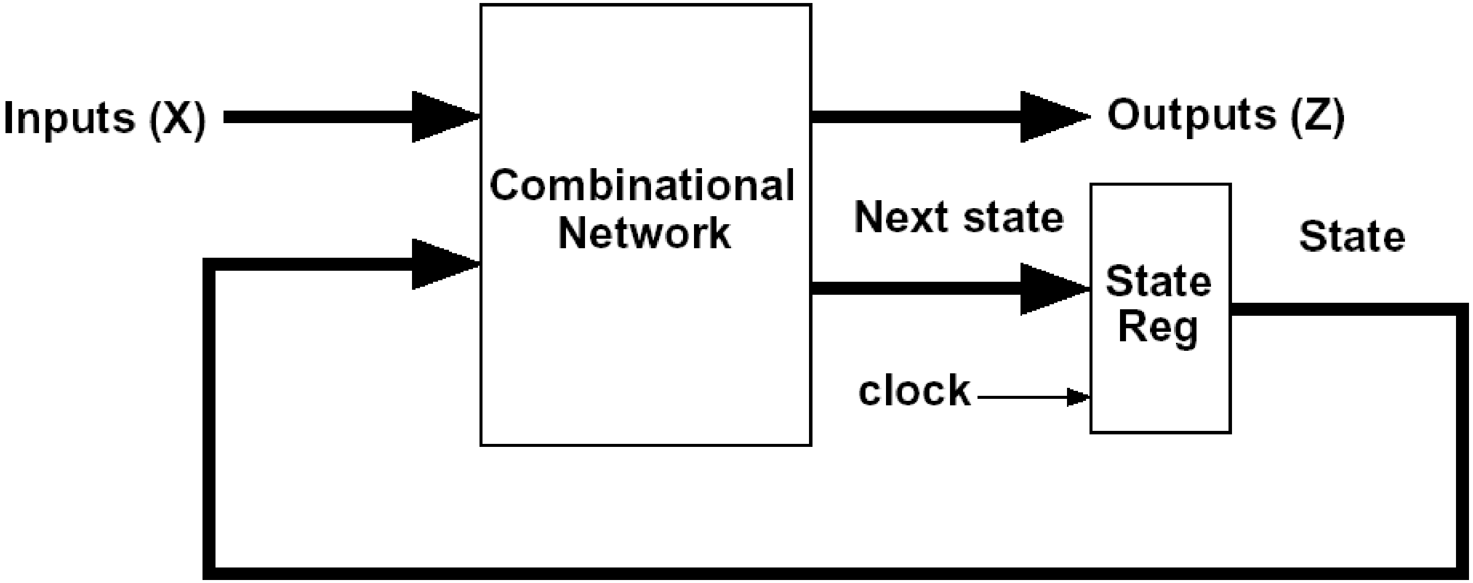


Sequential Network Timing (cont’d)

Timing diagram assuming a propagation delay of 10 ns for each flip-flop and gate (State has been replaced with the state of three flip-flops)



Review: Mealy Sequential Networks



$$Q(t^{+}) = G(X(t), Q(t)) \qquad Z(t) = F(X(t), Q(t))$$

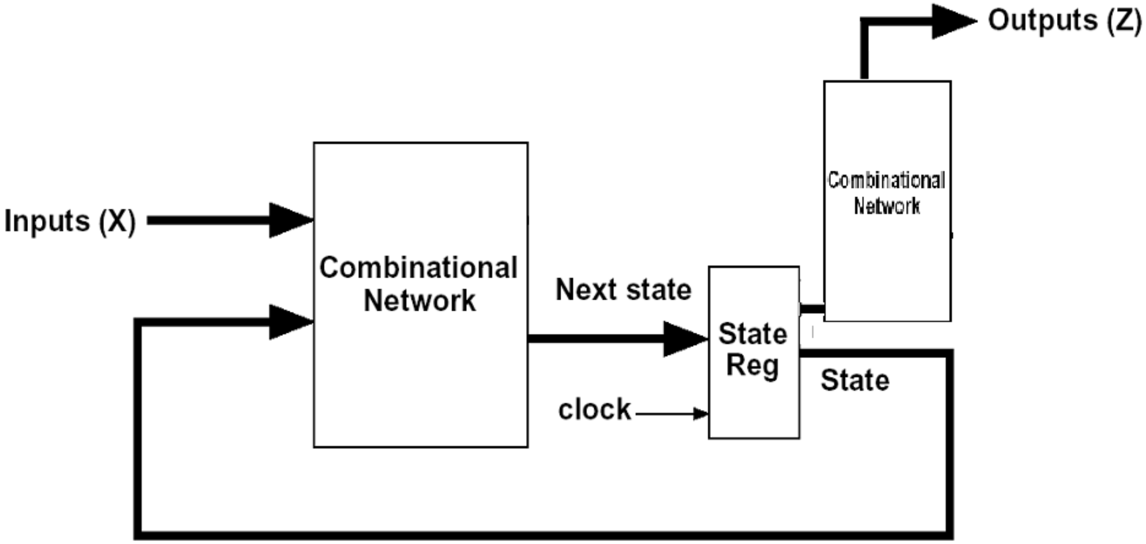
(1) X inputs are changed to a new value

(2) After a delay, the Z outputs and next state appear at the output of CN (input of state register)

(3) The next state is clocked into the state register and the state changes

Moore Sequential Networks

General model of Moore Sequential Network



(1) X inputs are changed to a new value
(2) The next state is clocked into the state registers
(3) The outputs appear which are a function of the current state only (not the inputs)

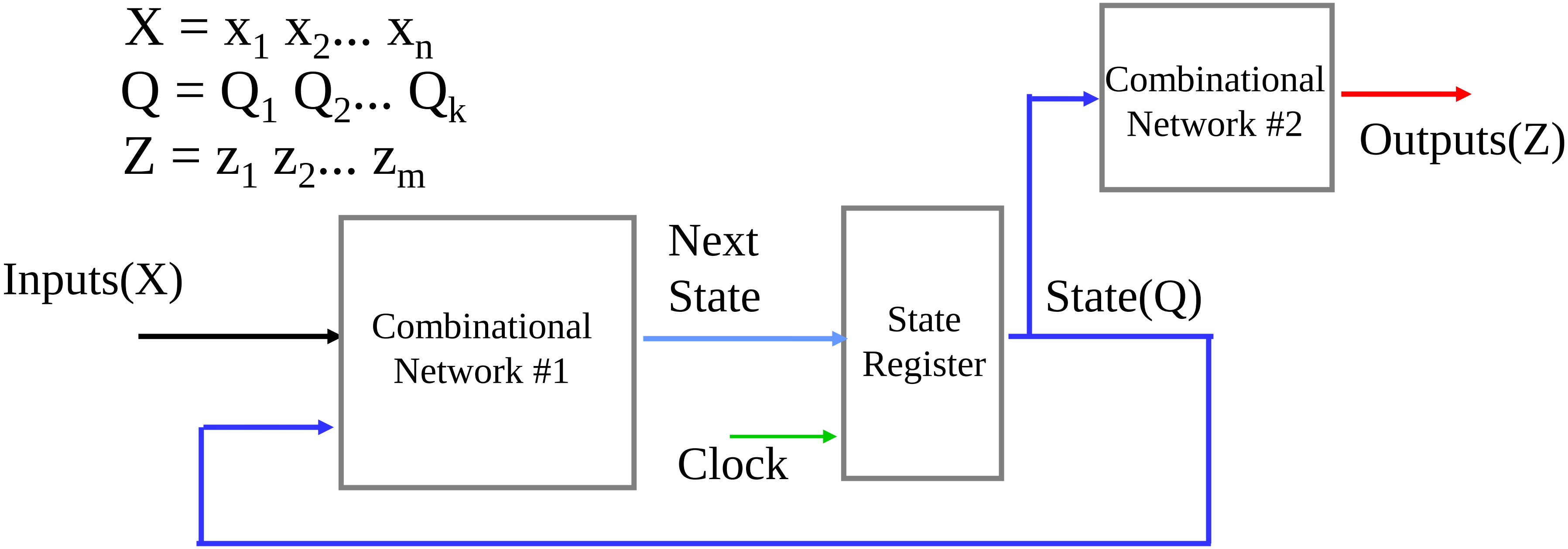
General Model of Moore Sequential Machine

Outputs depend only on present state!

$X = x_1 \, x_2 \ldots \, x_n$

$Q = Q_1 \, Q_2 \ldots \, Q_k$

$Z = z_1 \, z_2 \ldots \, z_m$



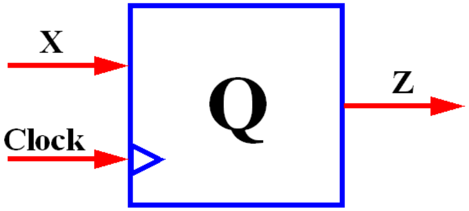
$$Q(t^+) = G(X(t), Q(t)) \qquad Z(t) = F(Q(t))$$

(1) X inputs are changed to a new value

(2) After a delay, the output of CN appears on the inputs of the state register

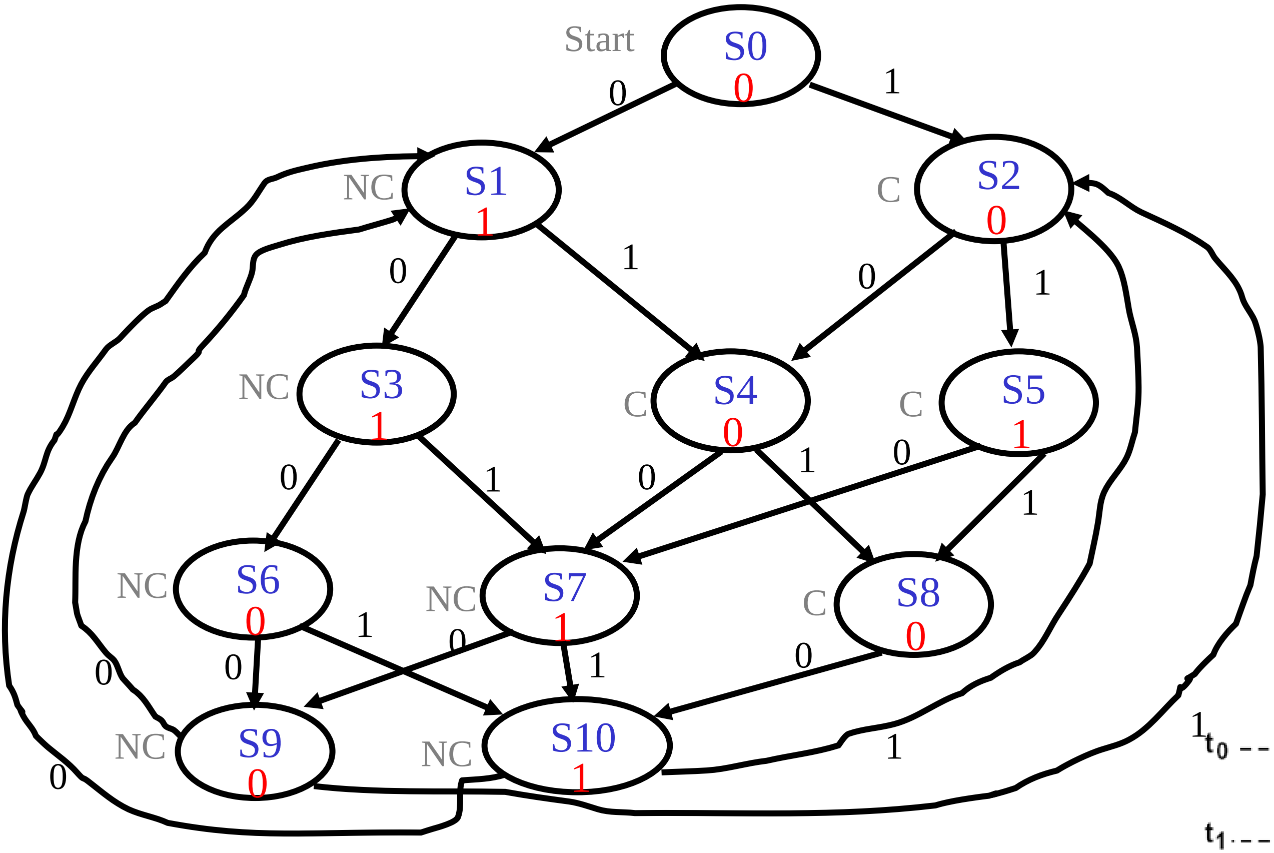
(3) The next state is clocked into the state register resulting in a possible change in state, and possible new Z outputs appear after a CN delay

An Example: 8421 BCD to Excess3 Code Converter

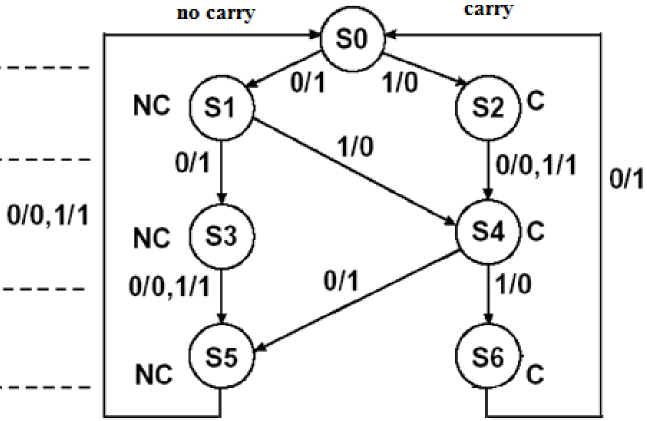


X (inputs)				Z (outputs)			
t3	t2	t1	t0	t3	t2	t1	t0
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

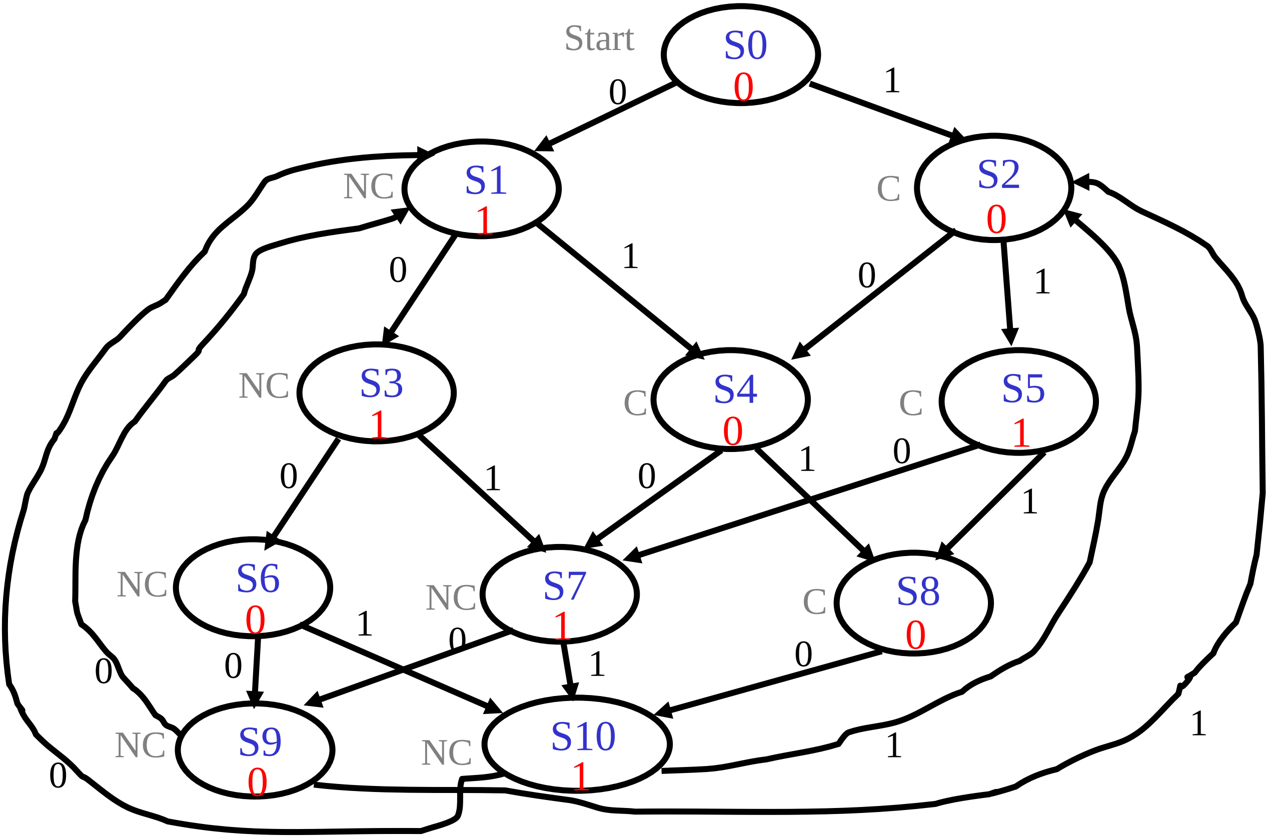
Moore Machine



Note: state S0 could be eliminated (S0 == S9), if S9 was start state!



Moore Machine



PS	NS		Z
	X=0	X=1	
S0	S1	S2	0
S1	S3	S4	1
S2	S4	S5	0
S3	S6	S7	1
S4	S7	S8	0
S5	S7	S8	1
S6	S9	S10	0
S7	S9	S10	1
S8	S10	-	0
S9	S1	S2	0
S10	S1	S2	1

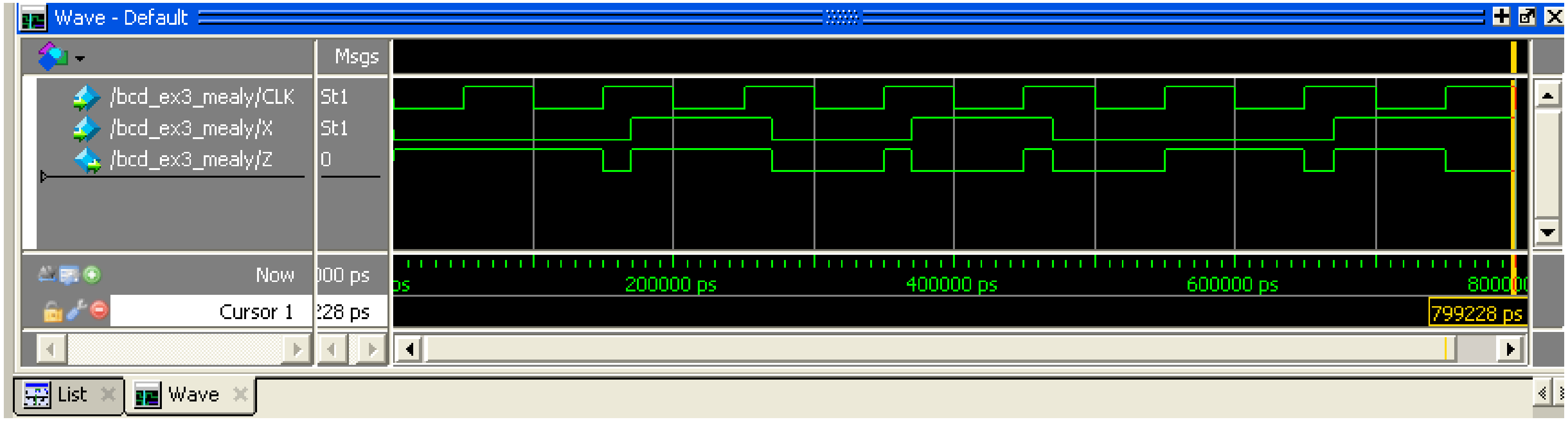
Note: state S0 could be eliminated (S0 == S9), if S9 was start state!

FSM Timings

Differences between Mealy and Moore

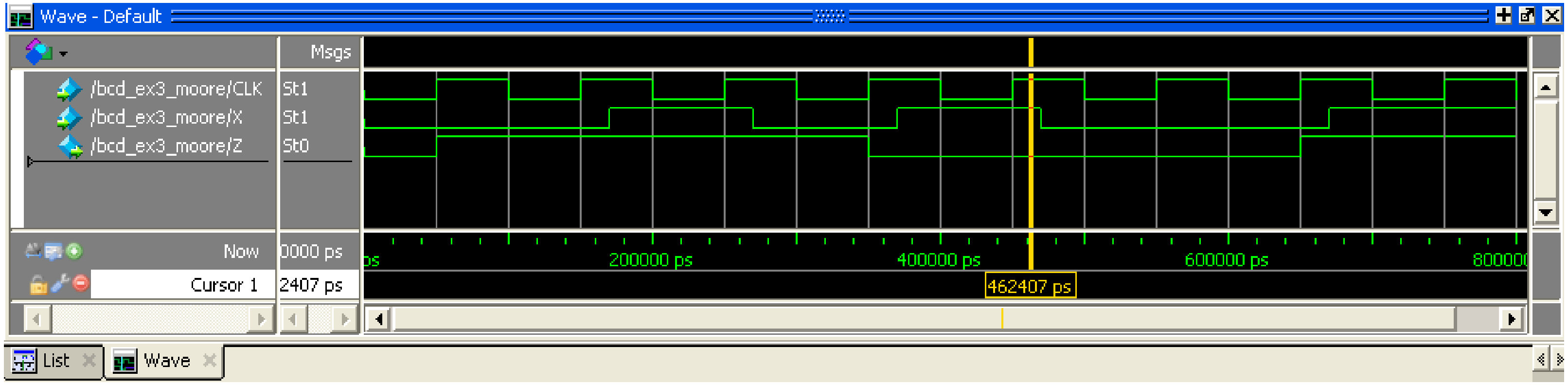
- X = 0010_1001 => Z = 1110_0011

Mealy



/bcd_ex3_mealy/CLK St1
/bcd_ex3_mealy/X St1
/bcd_ex3_mealy/Z 0
Now 00 ps
Cursor 1 28 ps
799228 ps

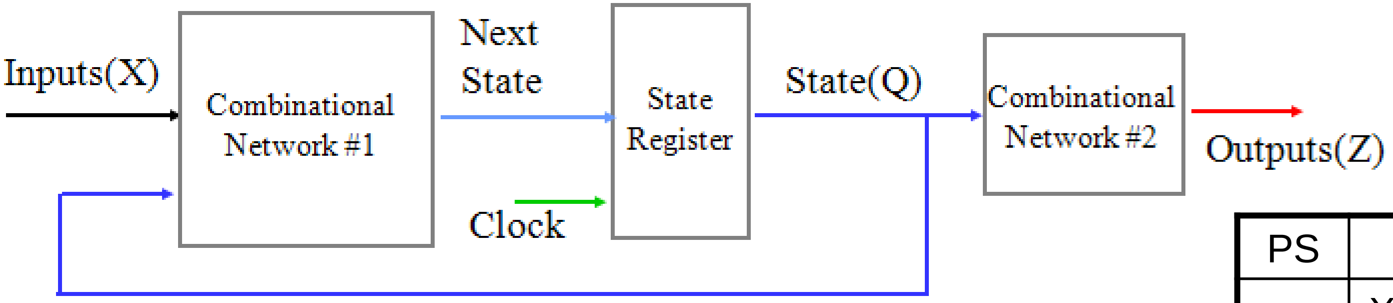
Moore

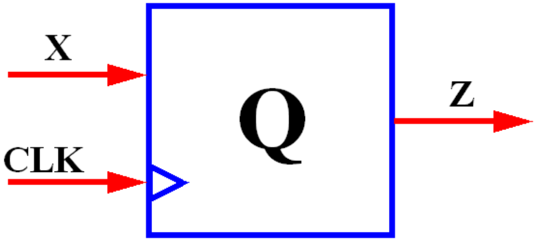


/bcd_ex3_moore/CLK St1
/bcd_ex3_moore/X St1
/bcd_ex3_moore/Z St0
Now 0000 ps
Cursor 1 2407 ps
462407 ps

Behavioral Verilog HDL Model

8421 BCD to Excess3 Code Converter (Moore FSM)





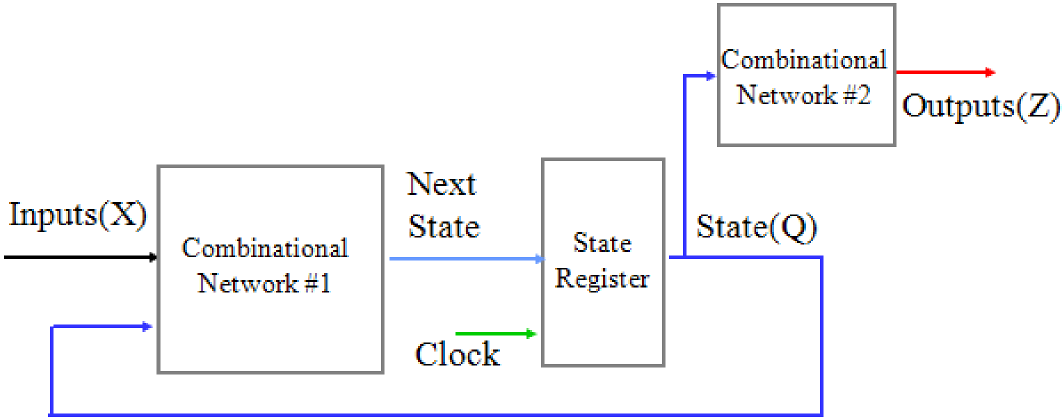
PS	NS		Z
	X=0	X=1	
S0	S1	S2	0
S1	S3	S4	1
S2	S4	S5	0
S3	S6	S7	1
S4	S7	S8	0
S5	S7	S8	1
S6	S9	S10	0
S7	S9	S10	1
S8	S10	-	0
S9	S1	S2	0
S10	S1	S2	1

```
// bcd to execess 3 converter
// Moore Implementation
// Behavioral Model
module bcd_ex3_moore(input X,CLK,output reg Z);

   reg [3:0]  State=0,Nextstate=0;
    •
    •
    •
endmodule
```

Behavioral Verilog HDL Model

8421 BCD to Excess3 Code Converter (Moore FSM)

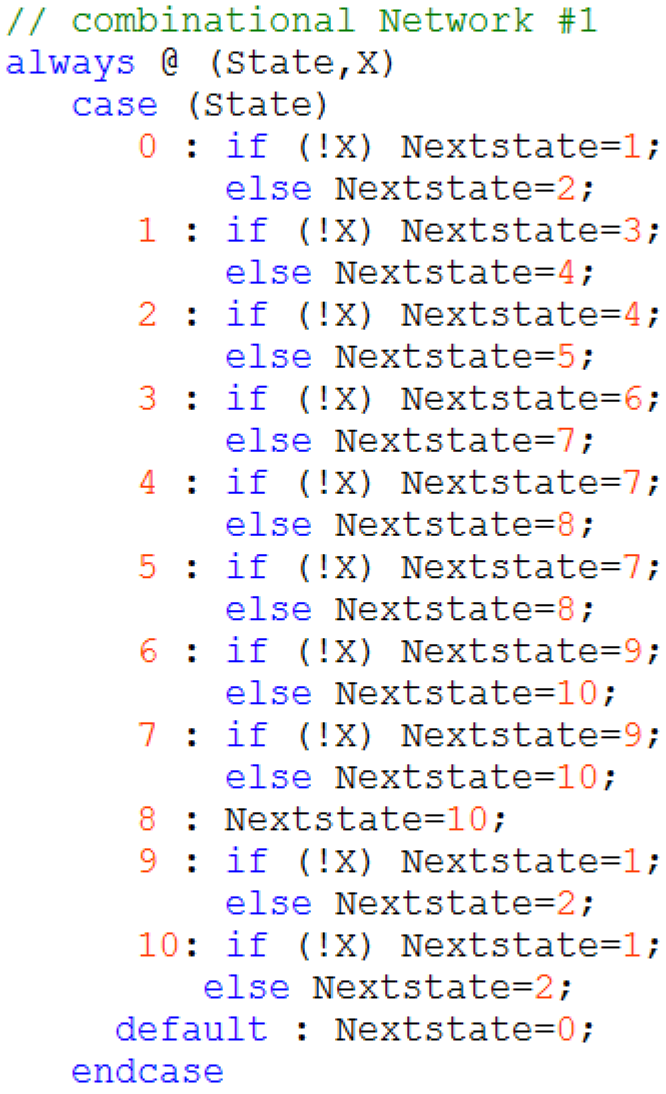
PS	NS		Z
	X=0	X=1	
S0	S1	S2	0
S1	S3	S4	1
S2	S4	S5	0
S3	S6	S7	1
S4	S7	S8	0
S5	S7	S8	1
S6	S9	S10	0
S7	S9	S10	1
S8	S10	-	0
S9	S1	S2	0
S10	S1	S2	1



Behavioral Verilog HDL Model

8421 BCD to Excess3 Code Converter (Moore FSM)

PS	NS		Z
	X=0	X=1	
S0	S1	S2	0
S1	S3	S4	1
S2	S4	S5	0
S3	S6	S7	1
S4	S7	S8	0
S5	S7	S8	1
S6	S9	S10	0
S7	S9	S10	1
S8	S10	-	0
S9	S1	S2	0
S10	S1	S2	1

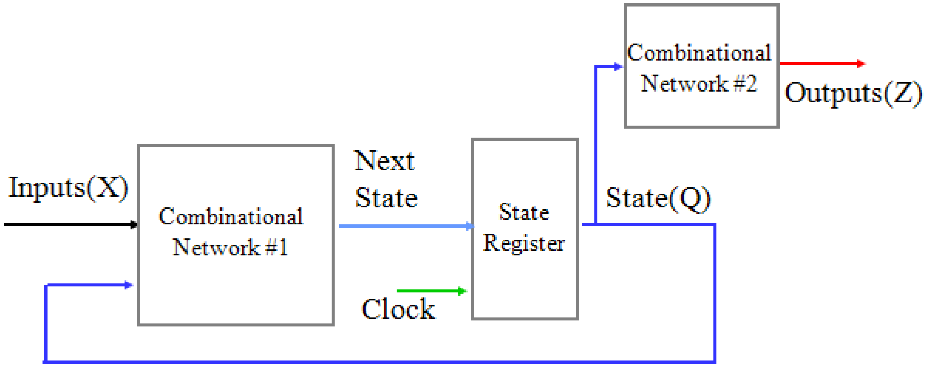


```
// combinational Network #1
always @ (State,X)
   case (State)
      0 : if (!X) Nextstate=1;
          else Nextstate=2;
      1 : if (!X) Nextstate=3;
          else Nextstate=4;
      2 : if (!X) Nextstate=4;
          else Nextstate=5;
      3 : if (!X) Nextstate=6;
          else Nextstate=7;
      4 : if (!X) Nextstate=7;
          else Nextstate=8;
      5 : if (!X) Nextstate=7;
          else Nextstate=8;
      6 : if (!X) Nextstate=9;
          else Nextstate=10;
      7 : if (!X) Nextstate=9;
          else Nextstate=10;
      8 : Nextstate=10;
      9 : if (!X) Nextstate=1;
          else Nextstate=2;
      10: if (!X) Nextstate=1;
         else Nextstate=2;
     default : Nextstate=0;
   endcase
```

Behavioral Verilog HDL Model

8421 BCD to Excess3 Code Converter (Moore FSM)

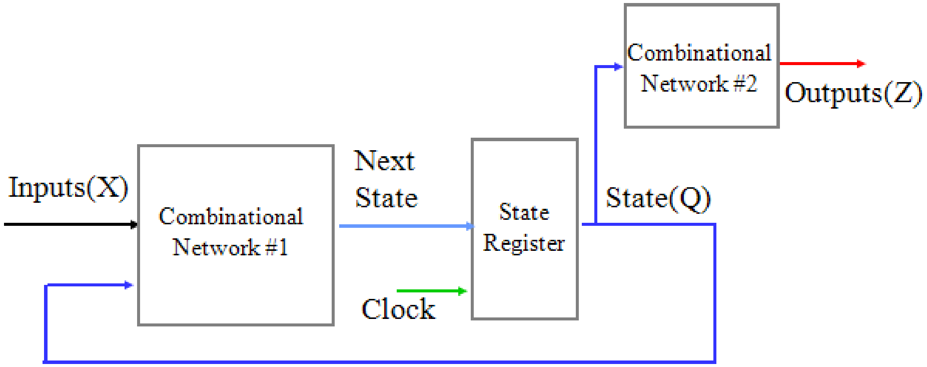
PS	NS		Z
	X=0	X=1	
S0	S1	S2	0
S1	S3	S4	1
S2	S4	S5	0
S3	S6	S7	1
S4	S7	S8	0
S5	S7	S8	1
S6	S9	S10	0
S7	S9	S10	1
S8	S10	-	0
S9	S1	S2	0
S10	S1	S2	1



```
// combinational Network #2
always @ (State)
   case (State)
       0 : Z=0;
       1 : Z=1;
       2 : Z=0;
       3 : Z=1;
       4 : Z=0;
       5 : Z=1;
       6 : Z=0;
       7 : Z=1;
       8 : Z=0;
       9 : Z=0;
       10: Z=1;
     default : Z=0;
   endcase
```

Behavioral Verilog HDL Model
8421 BCD to Excess3 Code Converter (Moore FSM)

PS	NS		Z
	X=0	X=1	
S0	S1	S2	0
S1	S3	S4	1
S2	S4	S5	0
S3	S6	S7	1
S4	S7	S8	0
S5	S7	S8	1
S6	S9	S10	0
S7	S9	S10	1
S8	S10	-	0
S9	S1	S2	0
S10	S1	S2	1



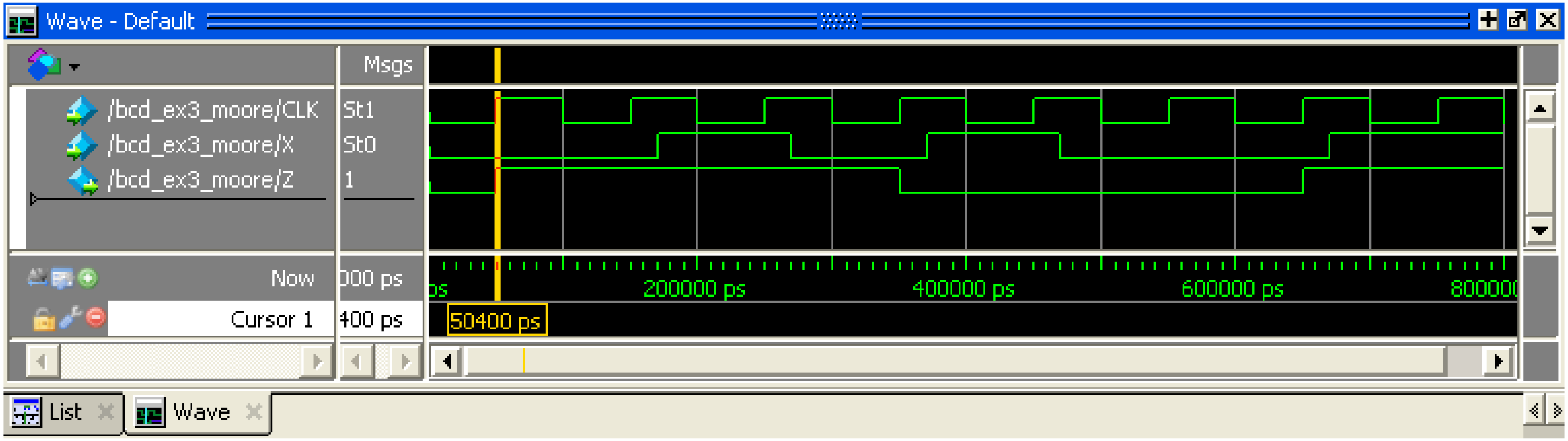
```
// State Reg Portion of Design
always @ (posedge CLK)
   State=Nextstate;
```

ModelSim™ Waveform

Behavioral 8421 BCD to Excess3 Code Converter (Moore FSM)

- add wave CLK X Z
- force CLK 0 0 ns, 1 50 ns -r 100 ns
- force X 0 0 ns, 1 170 ns, 0 270 ns, 1 370 ns, 0 470 ns, 1 670 ns
- run 800 ns

- X = 0010_1001 => Z = 1110_0011



/bcd_ex3_moore/CLK St1
/bcd_ex3_moore/X St0
/bcd_ex3_moore/Z 1

Now 000 ps

Cursor 1 400 ps 50400 ps

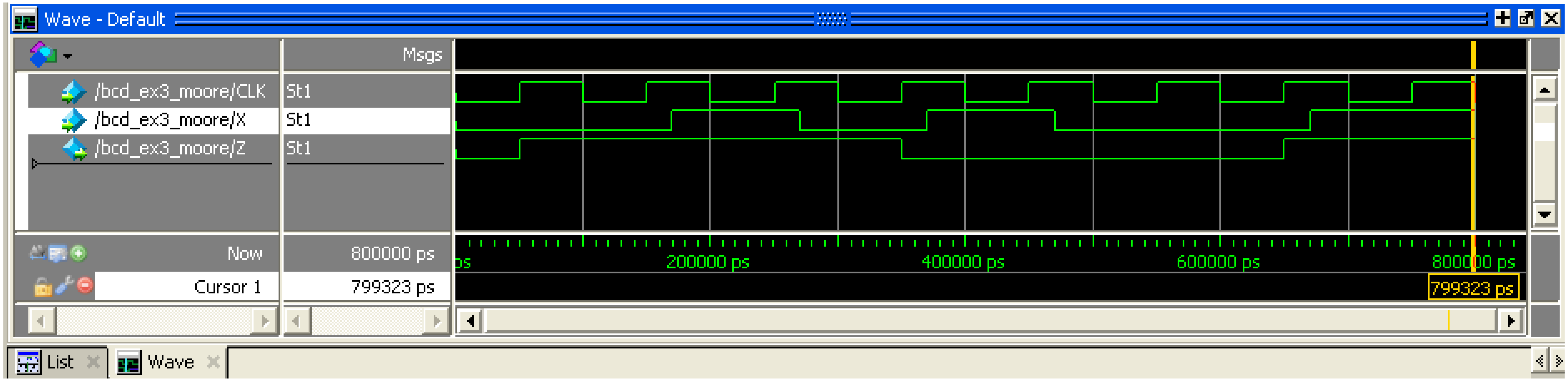
One-hot State Encoding

```
// bcd to execess 3 converter
// Moore Implementation
// Data Flow Model -- one hot state assignment
module bcd_ex3_moore(input X,CLK,output Z);
   // flip flop outputs
   reg Q0=1,Q1=0,Q2=0,Q3=0,Q4=0,Q5=0,Q6=0,Q7=0,Q8=0,Q9=0,Q10=0;

   // FF State Update Portion of design
   // Active only on rising edge of clock
   always @(posedge CLK)
      begin
         Q0 <= 0;
         Q1 <= (Q0 & ~X) | (Q9 & ~X) | (Q10 & ~X);
         Q2 <= (Q0 & X) | (Q9 & X) | (Q10 & X);
         Q3 <= Q1 & ~X;
         Q4 <= (Q1 & X) | (Q2 & ~X);
         Q5 <= Q2 & X;
         Q6 <= Q3 & ~X;
         Q7 <= (Q3 & X) | (Q4 & ~X) | (Q5 & ~X);
         Q8 <= (Q4 & X) | (Q5 & X);
         Q9 <= (Q6 & ~X) | (Q7 & ~X);
         Q10 <= (Q6 & X) | (Q7 & X) | (Q8 & ~X);
      end

   // Output Equation -- Continuous Assignment that is
   // a function only of the state variables QA,QB,QC,QD
   assign Z = Q1 | Q3 | Q5 | Q7 | Q10;

endmodule
```

PS	NS		Z
	X=0	X=1	
S0	S1	S2	0
S1	S3	S4	1
S2	S4	S5	0
S3	S6	S7	1
S4	S7	S8	0
S5	S7	S8	1
S6	S9	S10	0
S7	S9	S10	1
S8	S10	-	0
S9	S1	S2	0
S10	S1	S2	1

- X = 0010_1001 => Z = 1110_0011

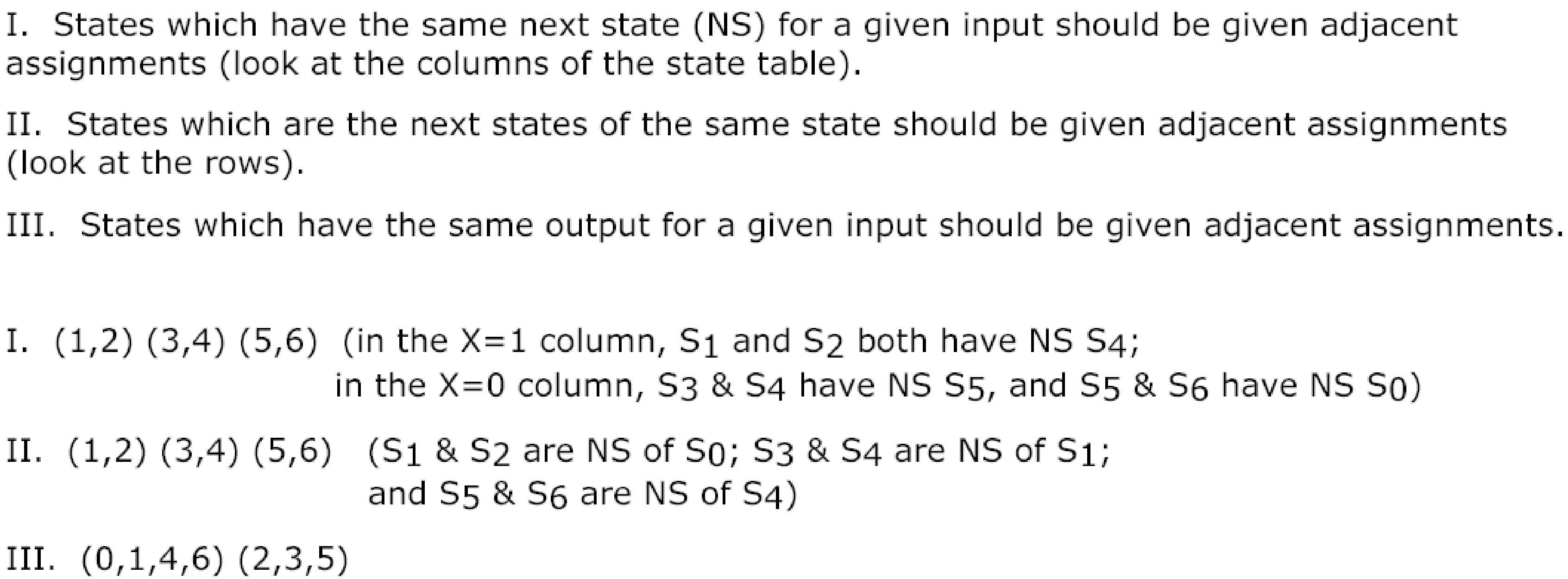
- add wave CLK X Z
- force CLK 0 0 ns, 1 50 ns -r 100 ns
- force X 0 0 ns, 1 170 ns, 0 270 ns, 1 370 ns, 0 470 ns, 1 670 ns
- run 800 ns



Wave - Default

	Msgs
/bcd_ex3_moore/CLK	St1
/bcd_ex3_moore/X	St1
/bcd_ex3_moore/Z	St1
Now	800000 ps
Cursor 1	799323 ps

State Assignment Rules



I. States which have the same next state (NS) for a given input should be given adjacent assignments (look at the columns of the state table).

II. States which are the next states of the same state should be given adjacent assignments (look at the rows).

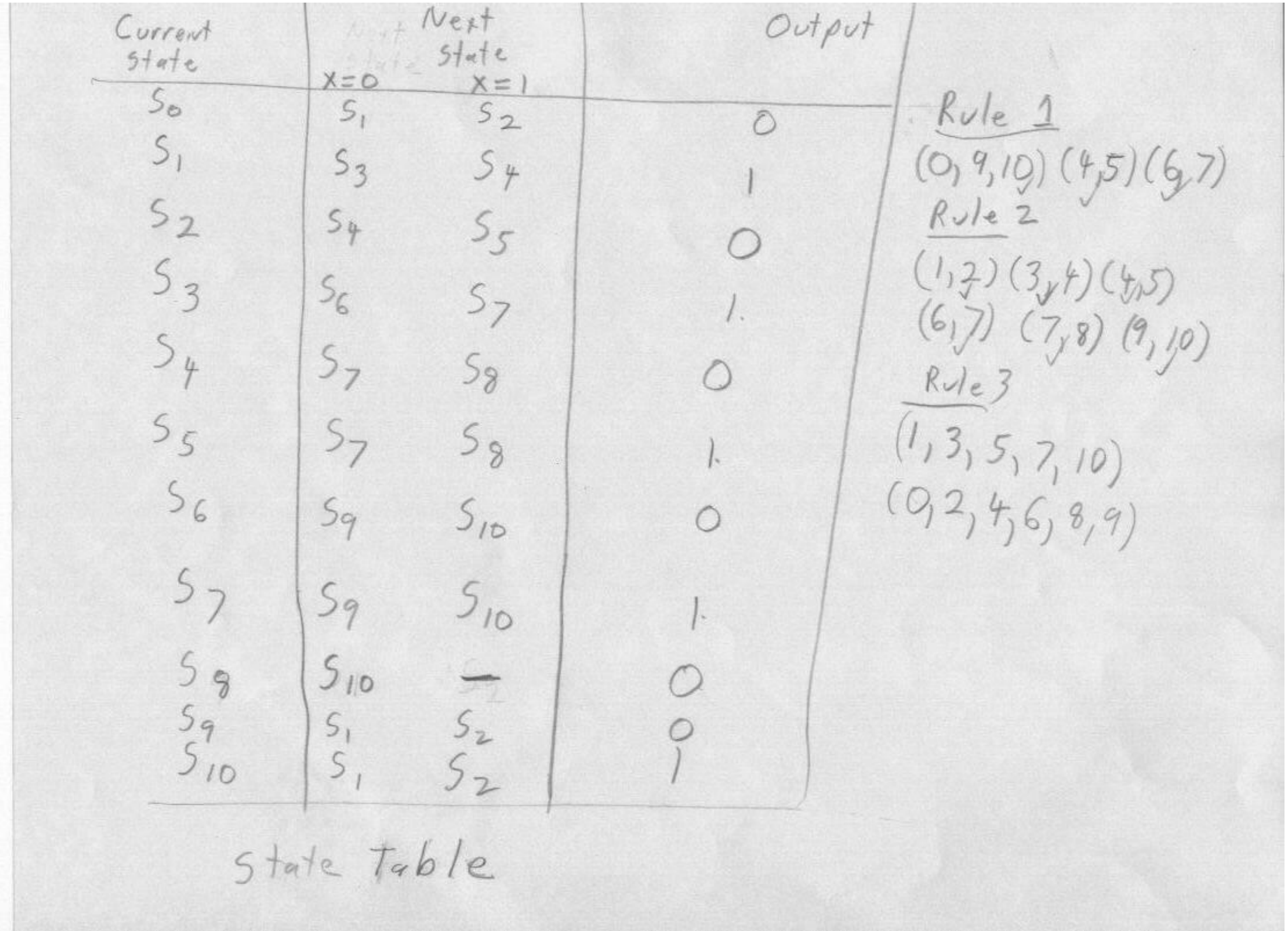
III. States which have the same output for a given input should be given adjacent assignments.

I. (1,2) (3,4) (5,6) (in the X=1 column, S_1 and S_2 both have NS S_4;
in the X=0 column, S_3 & S_4 have NS S_5, and S_5 & S_6 have NS S_0)

II. (1,2) (3,4) (5,6) (S_1 & S_2 are NS of S_0; S_3 & S_4 are NS of S_1;
and S_5 & S_6 are NS of S_4)

III. (0,1,4,6) (2,3,5)

Moore State Table



Current state	Next State x=0	Next State x=1	Output
S_0	S_1	S_2	0
S_1	S_3	S_4	1
S_2	S_4	S_5	0
S_3	S_6	S_7	1.
S_4	S_7	S_8	0
S_5	S_7	S_8	1.
S_6	S_9	S_{10}	0
S_7	S_9	S_{10}	1.
S_8	S_{10}	—	0
S_9	S_1	S_2	0
S_{10}	S_1	S_2	1

State Table

Rule 1

(0, 9, 10) (4, 5) (6, 7)

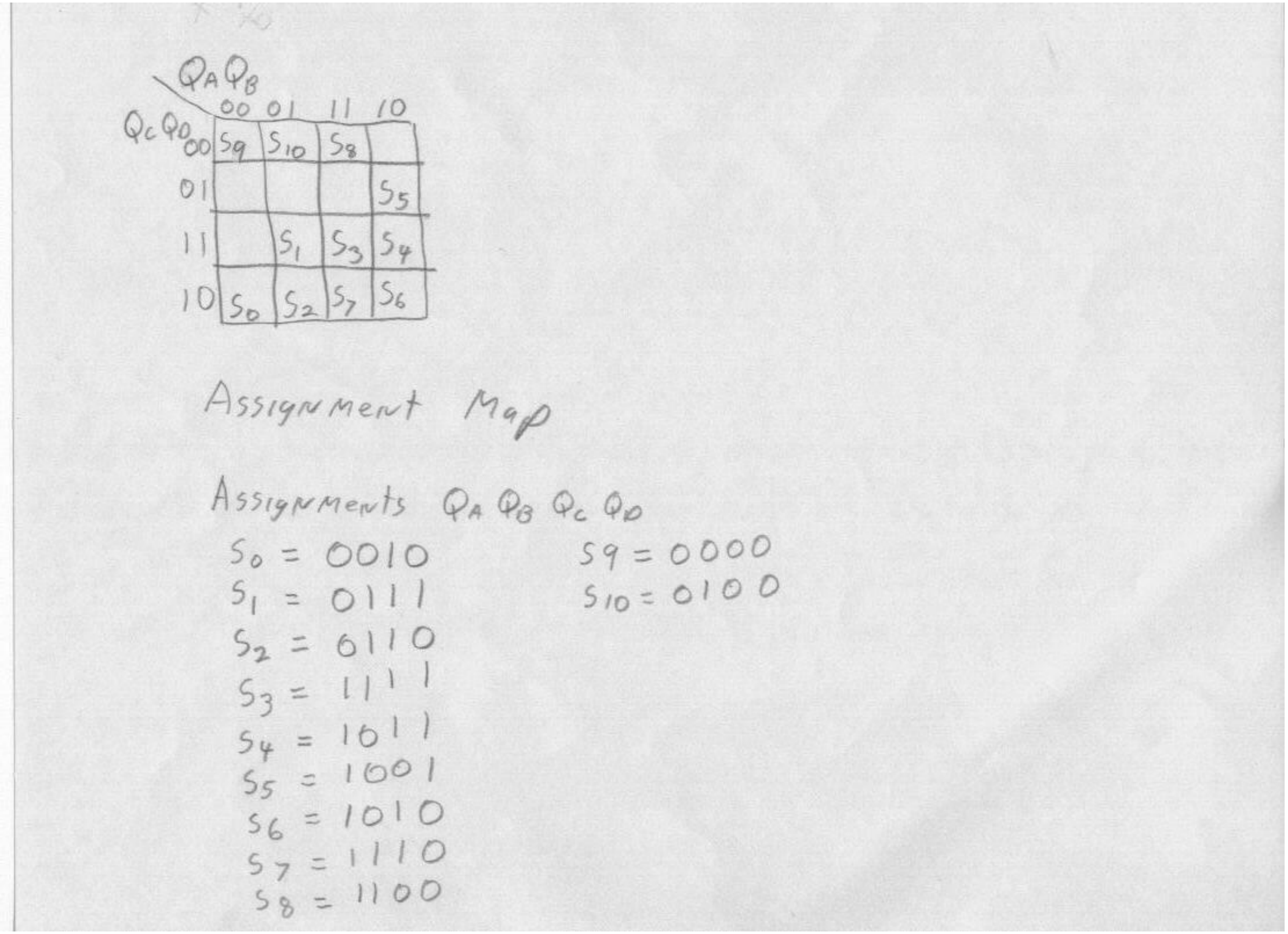
Rule 2

(1, 2) (3, 4) (4, 5)
(6, 7) (7, 8) (9, 10)

Rule 3

(1, 3, 5, 7, 10)
(0, 2, 4, 6, 8, 9)

State Assignments



Q_CQ_D \ Q_AQ_B	00	01	11	10
00	S_9	S_{10}	S_8	
01				S_5
11		S_1	S_3	S_4
10	S_0	S_2	S_7	S_6

Assignment Map

Assignments Q_A Q_B Q_C Q_D

S_0 = 0010

S_1 = 0111

S_2 = 0110

S_3 = 1111

S_4 = 1011

S_5 = 1001

S_6 = 1010

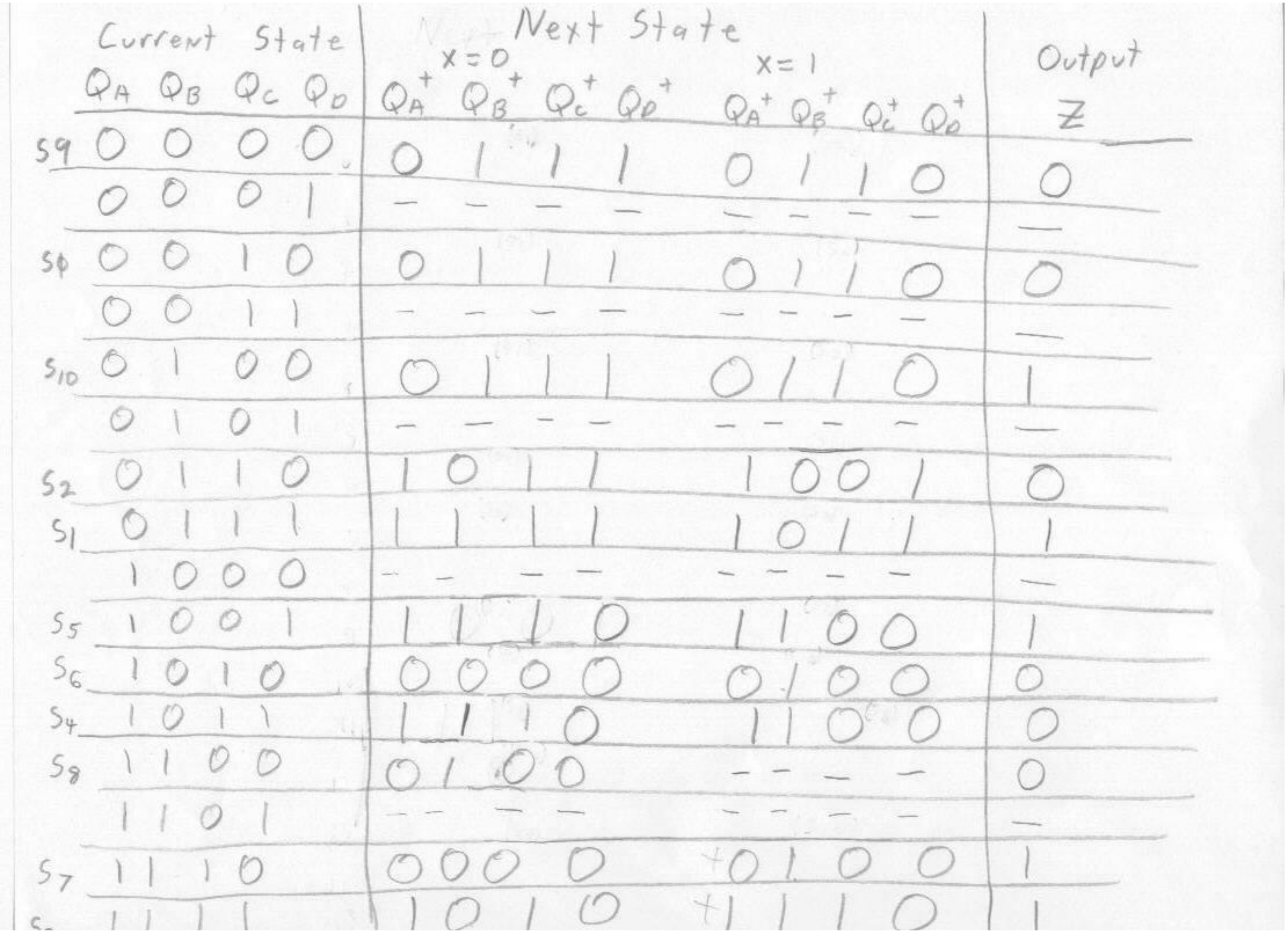
S_7 = 1110

S_8 = 1100

S_9 = 0000

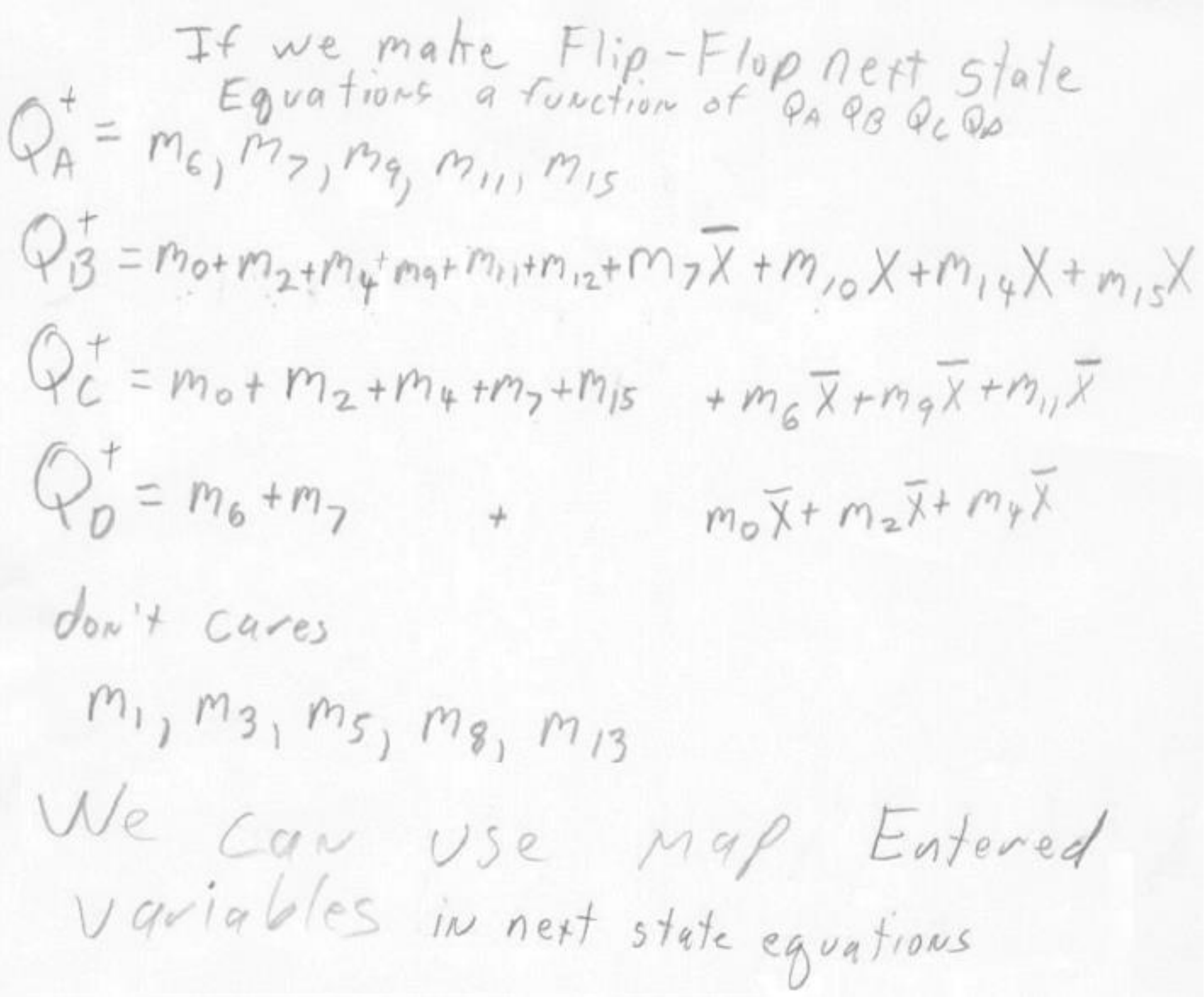
S_{10} = 0100

Transition Diagram



	Current State				Next State								Output
					x=0				x=1				
	Q_A	Q_B	Q_C	Q_D	Q_A^+	Q_B^+	Q_C^+	Q_D^+	Q_A^+	Q_B^+	Q_C^+	Q_D^+	Z
S_9	0	0	0	0	0	1	1	1	0	1	1	0	0
	0	0	0	1	–	–	–	–	–	–	–	–	–
S_ϕ	0	0	1	0	0	1	1	1	0	1	1	0	0
	0	0	1	1	–	–	–	–	–	–	–	–	–
S_{10}	0	1	0	0	0	1	1	1	0	1	1	0	1
	0	1	0	1	–	–	–	–	–	–	–	–	–
S_2	0	1	1	0	1	0	1	1	1	0	0	1	0
S_1	0	1	1	1	1	1	1	1	1	0	1	1	1
	1	0	0	0	–	–	–	–	–	–	–	–	–
S_5	1	0	0	1	1	1	1	0	1	1	0	0	1
S_6	1	0	1	0	0	0	0	0	0	1	0	0	0
S_4	1	0	1	1	1	1	1	0	1	1	0	0	0
S_8	1	1	0	0	0	1	0	0	–	–	–	–	0
	1	1	0	1	–	–	–	–	–	–	–	–	–
S_7	1	1	1	0	0	0	0	0	+0	1	0	0	1
S	1	1	1	1	1	0	1	0	+1	1	1	0	1

Resulting Boolean Equations



If we make Flip-Flop next state Equations a function of $Q_A\ Q_B\ Q_C\ Q_D$

$Q_A^+ = m_6, m_7, m_9, m_{11}, m_{15}$

$Q_B^+ = m_0 + m_2 + m_4 + m_9 + m_{11} + m_{12} + m_7\overline{X} + m_{10}X + m_{14}X + m_{15}X$

$Q_C^+ = m_0 + m_2 + m_4 + m_7 + m_{15} \quad + m_6\overline{X} + m_9\overline{X} + m_{11}\overline{X}$

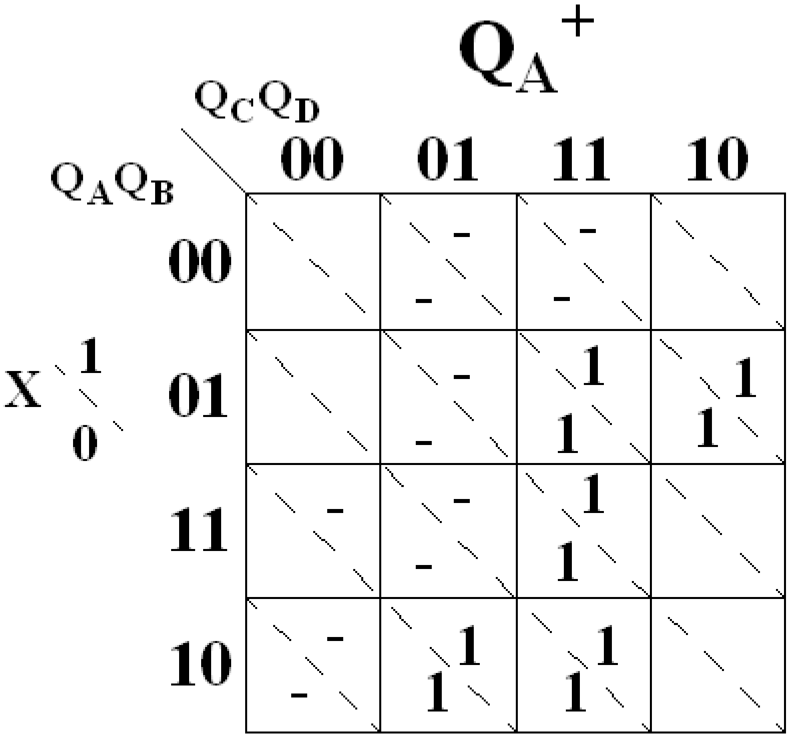
$Q_D^+ = m_6 + m_7 \quad + \quad m_0\overline{X} + m_2\overline{X} + m_4\overline{X}$

don't cares

$m_1, m_3, m_5, m_8, m_{13}$

We can use map Entered variables in next state equations

Flip-flop Q_A



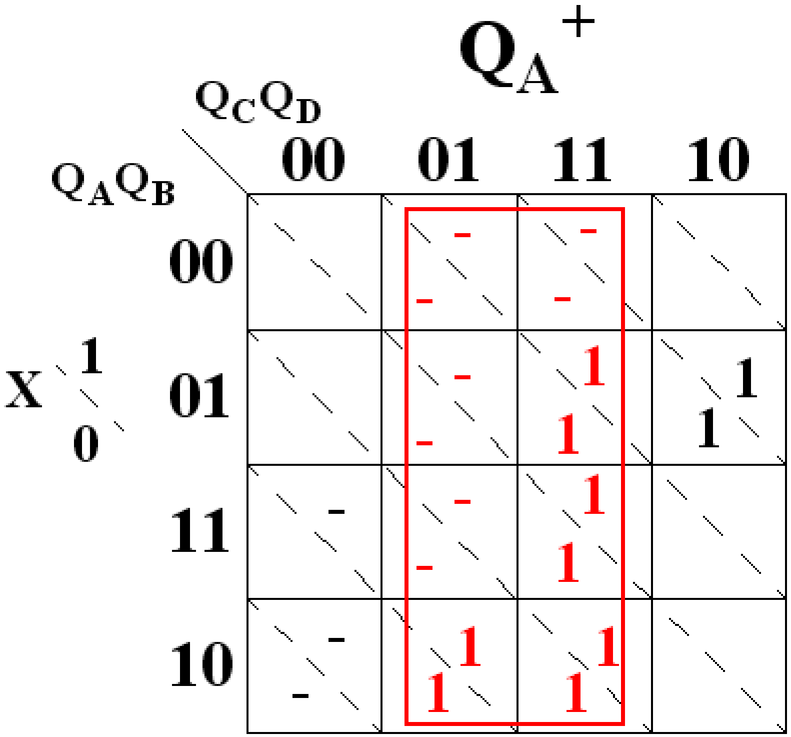
Q_A^+

X: 1 / 0

Q_AQ_B \ Q_CQ_D	00	01	11	10
00		- -	- -	
01		- -	1 1	1 1
11	-	- -	1 1	
10	- -	1 1	1 1	

$Q_A^+ =$

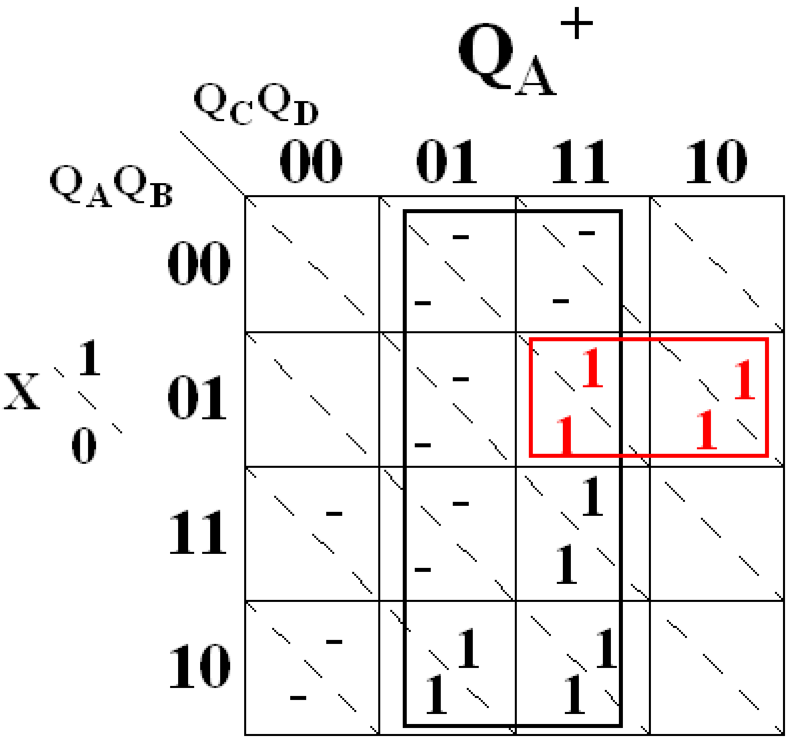
Flip-flop Q_A



Q_AQ_B \ Q_CQ_D	00	01	11	10
00		- -	- -	
01		- -	1 1	1 1
11	-	- -	1 1	
10	- -	1 1	1 1	

$$Q_A^+ = Q_D$$

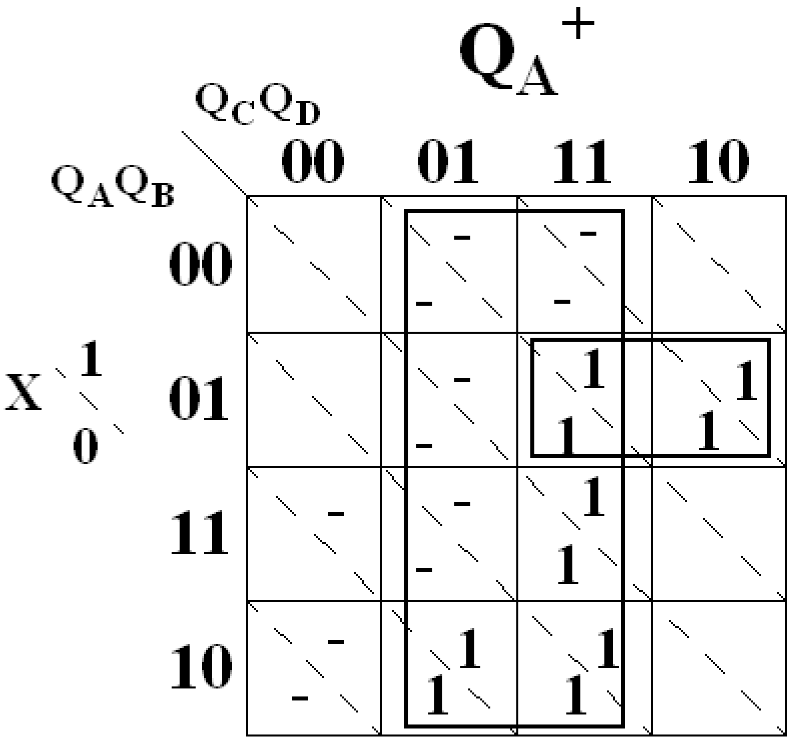
Flip-flop Q_A



Q_AQ_B \ Q_CQ_D	00	01	11	10
00		- -	- -	
01		- -	1 1	1 1
11	-	- -	1 1	
10	- -	1 1	1 1	

$$Q_A^+ = Q_D + Q_A'Q_BQ_C$$

Flip-flop Q_A



Q_AQ_B \ Q_CQ_D	00	01	11	10
00		- -	- -	
01		- -	1 1	1 1
11	-	- -	1 1	
10	- -	1 1	1 1	

$$Q_A^+ = Q_D + Q_A' Q_B Q_C$$

Flip-flop Q_B

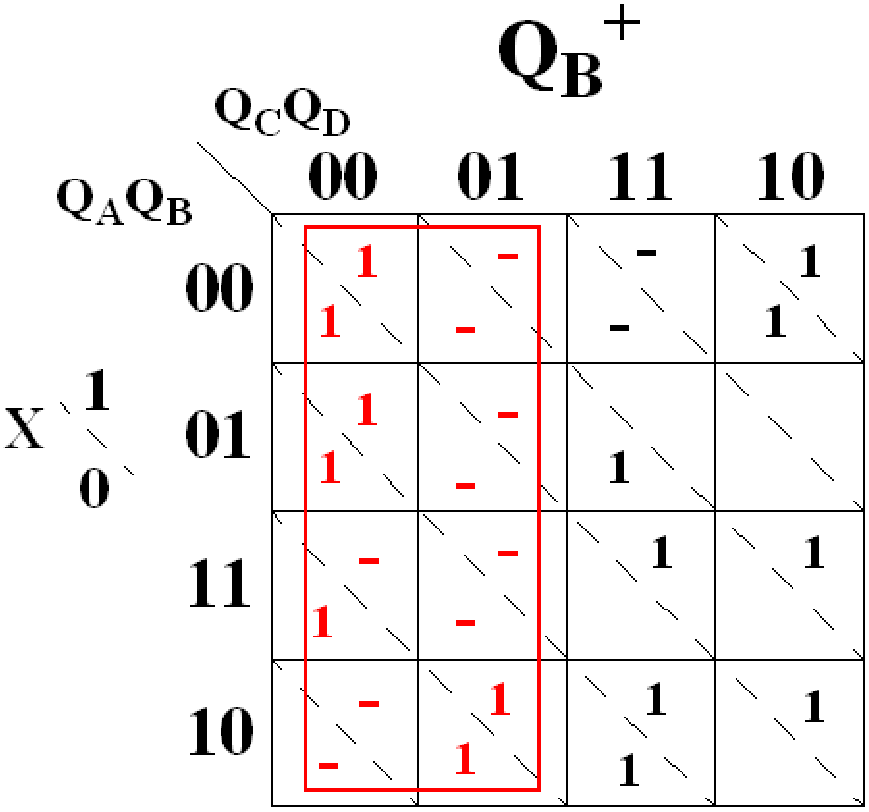
Q_B^{+}

X: 1 \ 0

Q_AQ_B \ Q_CQ_D	00	01	11	10
00	1 1	- -	- -	1 1
01	1 1	- -	 1	
11	- 1	- -	1 	1
10	- -	1 1	1 1	1

$Q_B^{+} =$

Flip-flop Q_B



Q_B^+

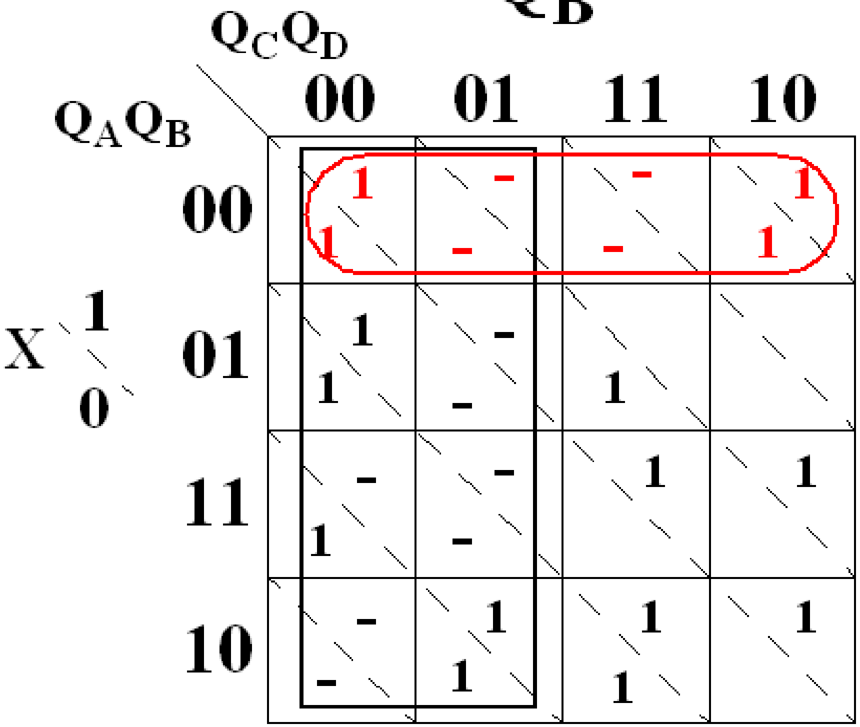
Q_AQ_B \ Q_CQ_D	00	01	11	10
00	1 1	- -	- -	1 1
01	1 1	- -	 1	
11	- 1	- -	1 	1
10	- -	1 1	1 1	1

X: 1 / 0

$Q_B^+ = Q_C'$

Flip-flop Q_B

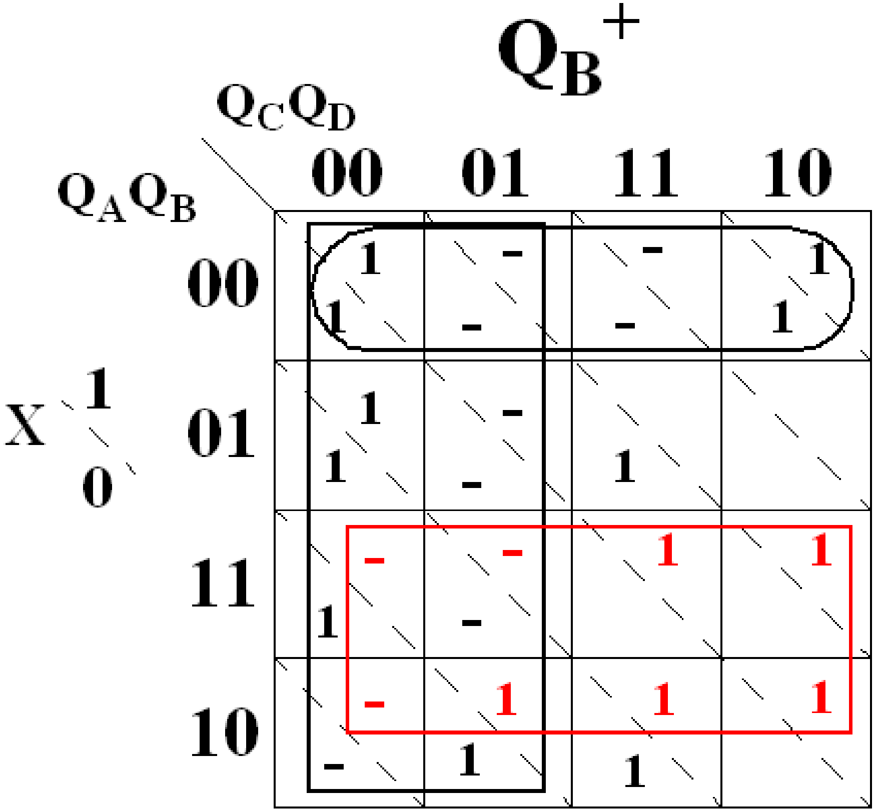
Q_B^+



Q_AQ_B \ Q_CQ_D	00	01	11	10
00	1 1	- -	- -	1 1
01	1 1	- -	 1	
11	- 1	- -	1 	1
10	- -	1 1	1 1	1

$$Q_B^+ = Q_C' + Q_A'Q_B'$$

Flip-flop Q_B



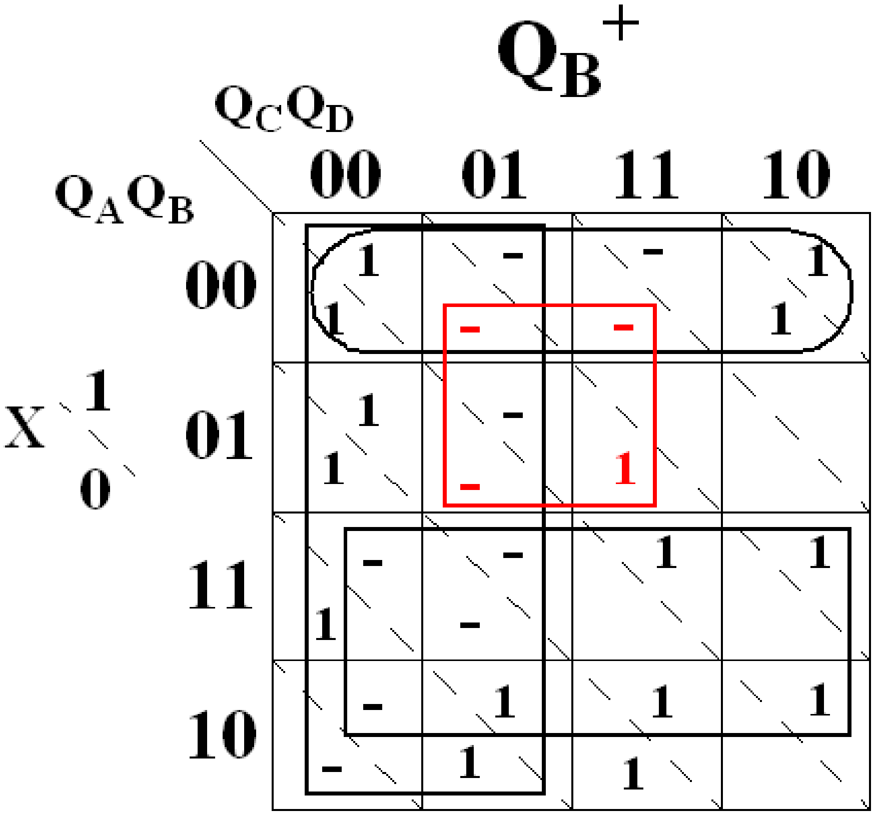
Q_B^+

X: 1 / 0

Q_AQ_B \ Q_CQ_D	00	01	11	10
00	1 1	- -	- -	1 1
01	1 1	- -	1	
11	- 1	- -	1	1
10	- -	1 1	1 1	1

$$Q_B^+ = Q_C' + Q_A'Q_B' + Q_AX$$

Flip-flop Q_B

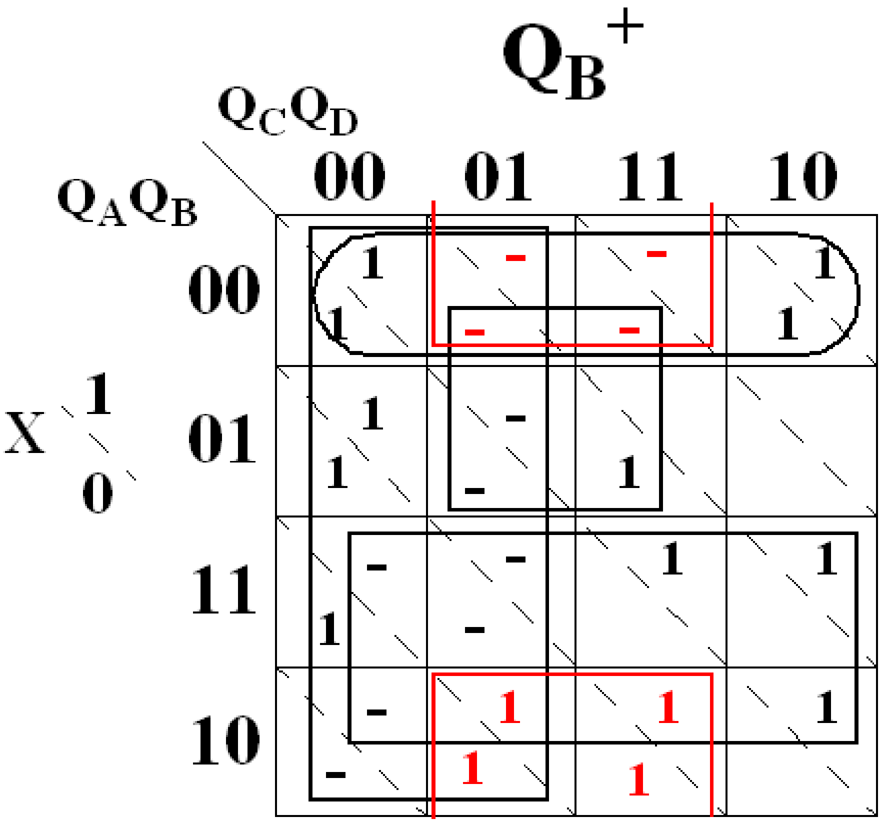


Q_AQ_B \ Q_CQ_D	00	01	11	10
00	1 1	- -	- -	1 1
01	1 1	- -	 1	
11	- 1	- -	1	1
10	- -	1 1	1 1	1

$$Q_B^+ = Q_C' + Q_A'Q_B' + Q_AX +$$

$$Q_A'Q_DX'$$

Flip-flop Q_B

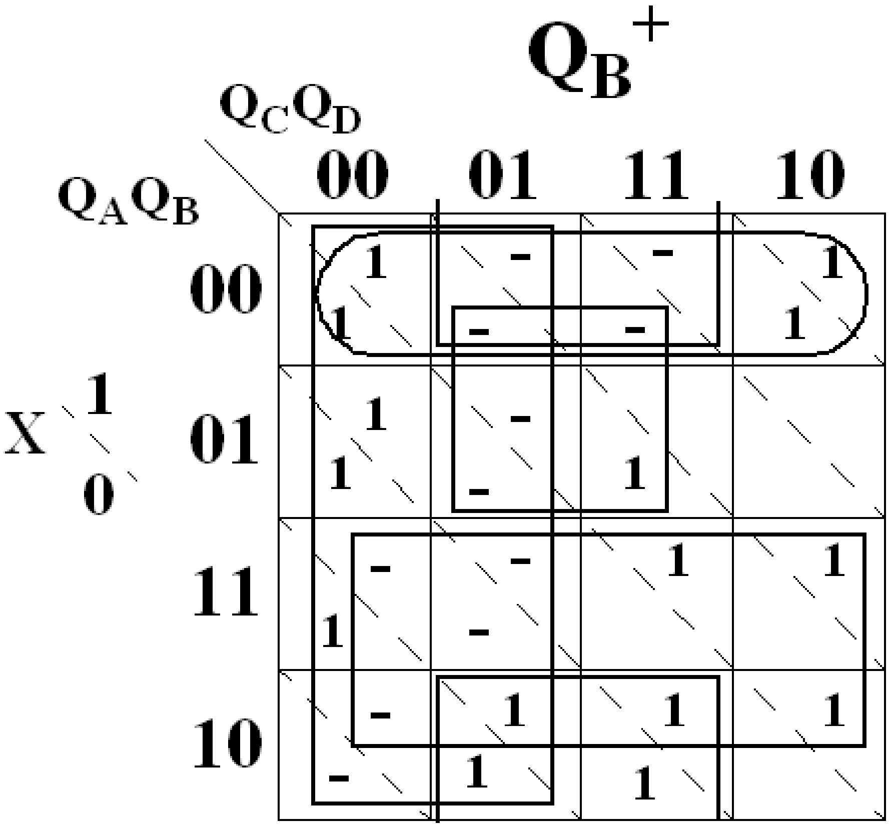


Q_AQ_B \ Q_CQ_D	00	01	11	10
00	1 1	- -	- -	1 1
01	1 1	- -	 1	
11	- 1	- -	1 	1
10	- -	1 1	1 1	1

$$Q_B^+ = Q_C' + Q_A'Q_B' + Q_AX +$$

$$Q_A'Q_DX' + Q_B'Q_D$$

Flip-flop Q_B



Q_AQ_B \ Q_CQ_D	00	01	11	10
00	1 1	- -	- -	1 1
01	1 1	- -	 1	
11	- 1	- -	1 	1
10	- -	1 1	1 1	1

$$Q_B^+ = Q_C' + Q_A'Q_B' + Q_AX +$$

$$Q_A'Q_DX' + Q_B'Q_D$$

Flip-flop Q_C

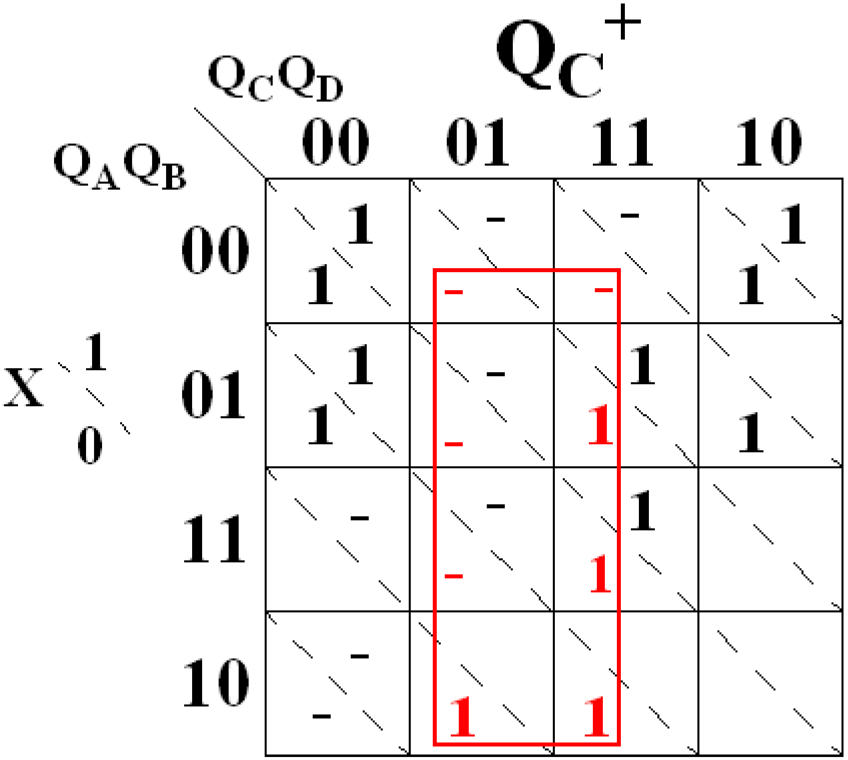
Q_C^+

X 1 / 0

Q_AQ_B \ Q_CQ_D	00	01	11	10
00	1 1	- -	- -	1 1
01	1 1	- -	1 1	 1
11	- 	- -	1 1	
10	- -	 1	 1	

$Q_C^+=$

Flip-flop Q_C



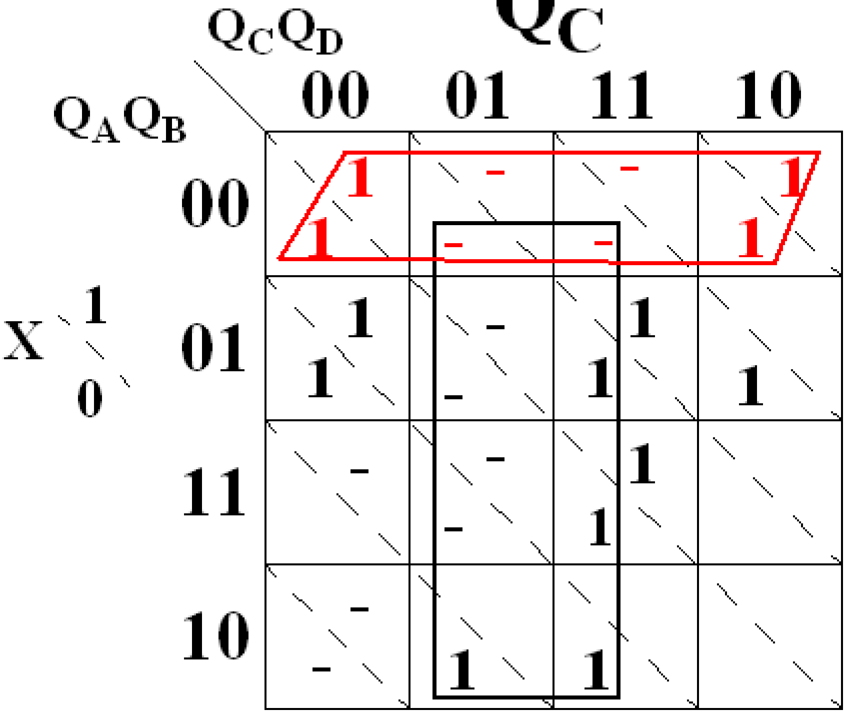
Q_C^+

Q_AQ_B \ Q_CQ_D	00	01	11	10
00	1 1	- -	- -	1 1
01	1 1	- -	1 1	 1
11	- 	- -	1 1	
10	- -	 1	 1	

$Q_C^+ = Q_DX'$

Flip-flop Q_C

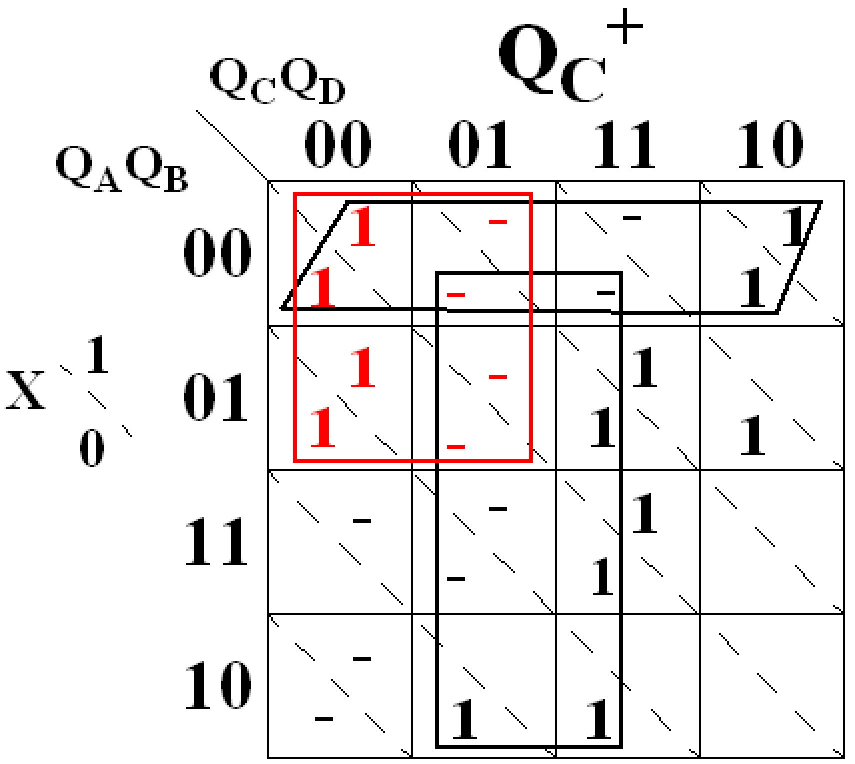
Q_C^+



Q_AQ_B \ Q_CQ_D	00	01	11	10
00	1 1	- -	- -	1 1
01	1 1	- -	1 1	 1
11	- 	- -	1 1	
10	- -	 1	 1	

$$Q_C^+ = Q_DX' + Q_A'Q_B'$$

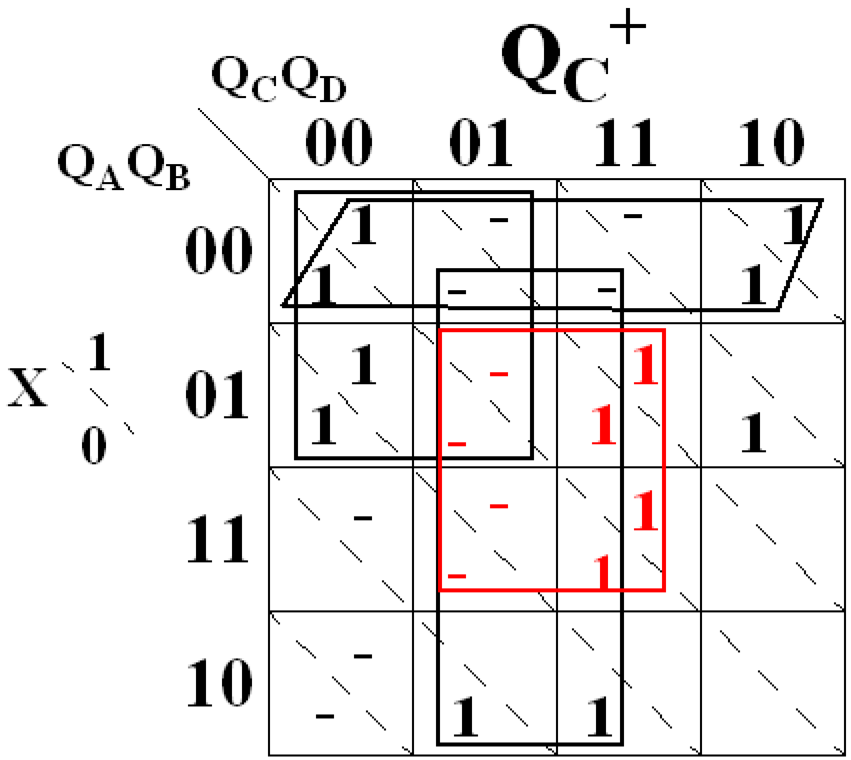
Flip-flop Q_C



Q_AQ_B \ Q_CQ_D	00	01	11	10
00	1 1	– –	– –	1 1
01	1 1	– –	1 1	 1
11	– 	– –	1 1	
10	– –	 1	 1	

$$Q_C^+ = Q_DX' + Q_A'Q_B' + Q_A'Q_C'$$

Flip-flop Q_C



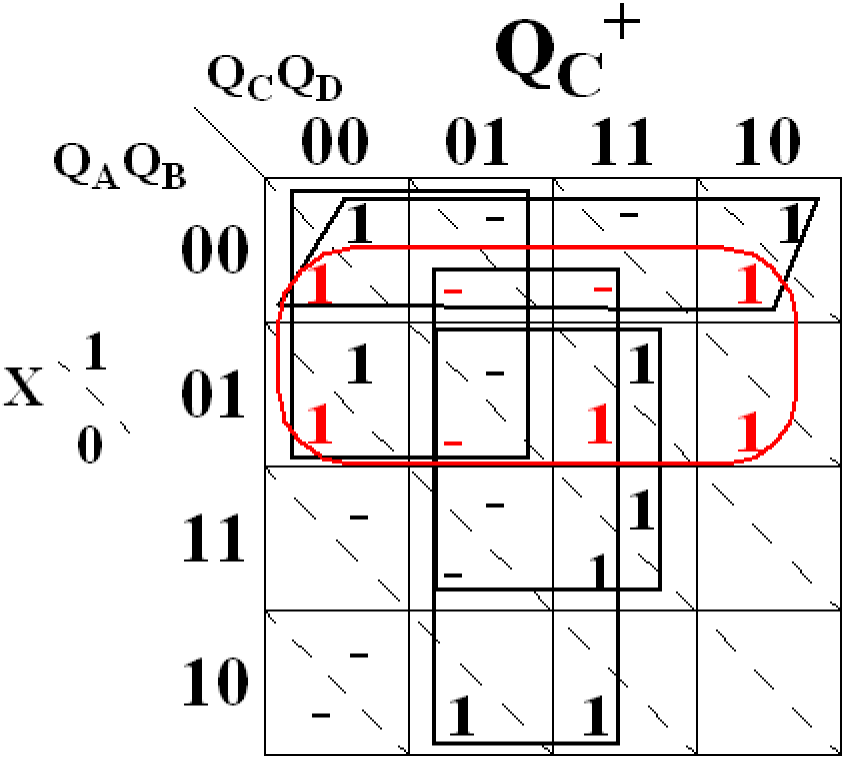
Q_C^+

Q_AQ_B \ Q_CQ_D	00	01	11	10
00	1 1	- -	- -	1 1
01	1 1	- -	1 1	 1
11	- 	- -	1 1	
10	- -	 1	 1	

$$Q_C^+ = Q_D X' + Q_A' Q_B' + Q_A' Q_C' +$$

$$Q_B Q_D$$

Flip-flop Q_C



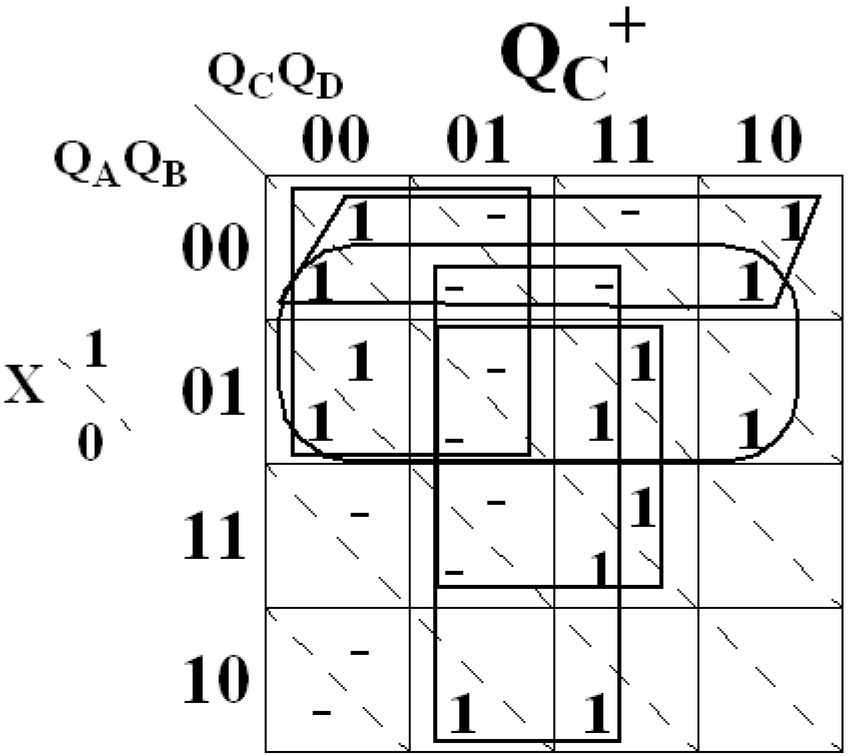
Q_C^+

X 1 / 0

Q_AQ_B \ Q_CQ_D	00	01	11	10
00	1 1	- -	- -	1 1
01	1 1	- -	1 1	 1
11	- 	- -	1 1	
10	- -	 1	 1	

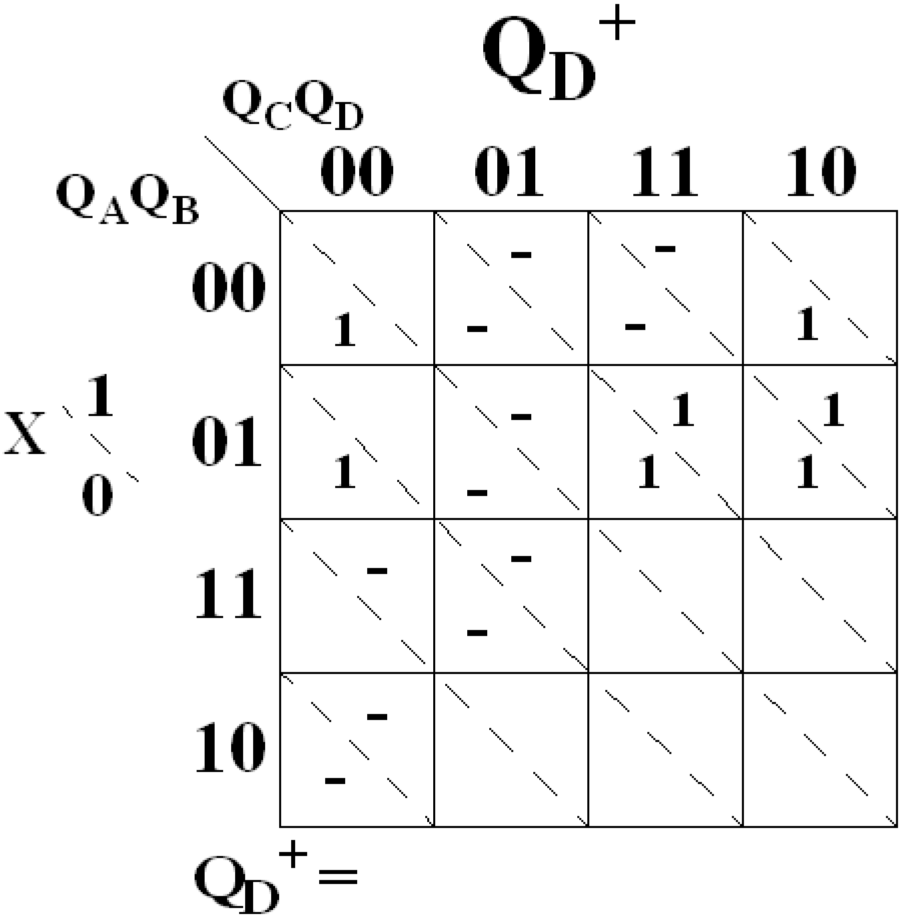
$$Q_C^+ = Q_DX' + Q_A'Q_B' + Q_A'Q_C' + Q_BQ_D + Q_A'X'$$

Flip-flop Q_C



$$Q_C^+ = Q_DX' + Q_A'Q_B' + Q_A'Q_C' +$$
$$Q_BQ_D + Q_A'X'$$

Flip-flop Q_D



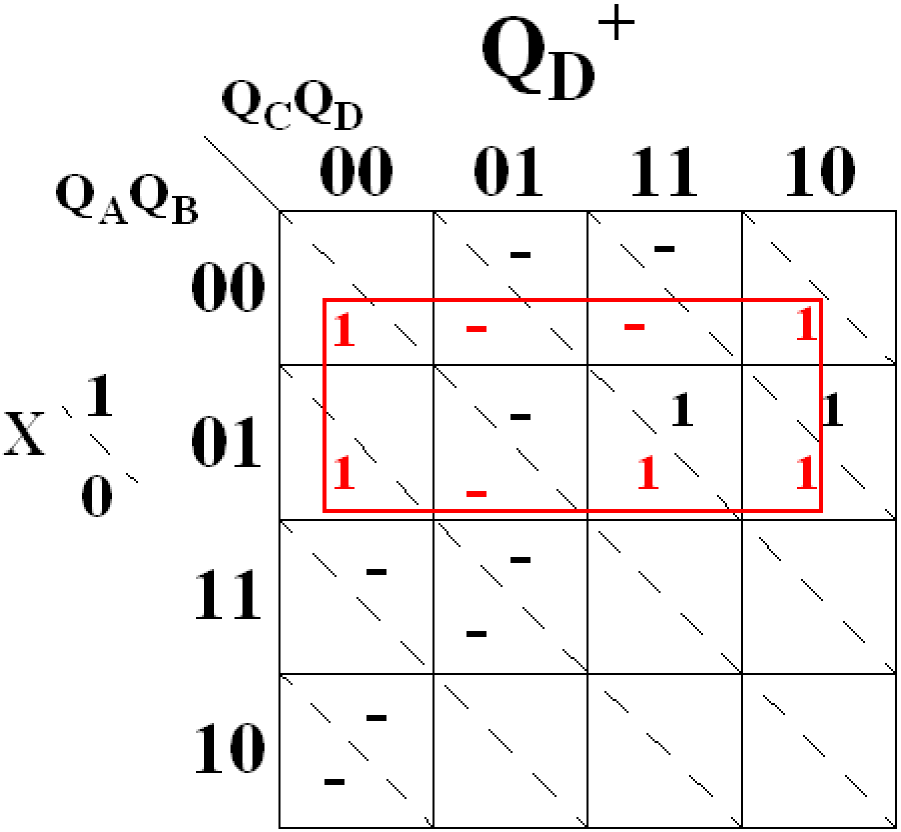
Q_D^+

X: 1 / 0

Q_AQ_B Q_CQ_D	00	01	11	10
00	 1	- -	- -	 1
01	 1	- -	1 1	1 1
11	- 	- -		
10	- -			

$Q_D^+ =$

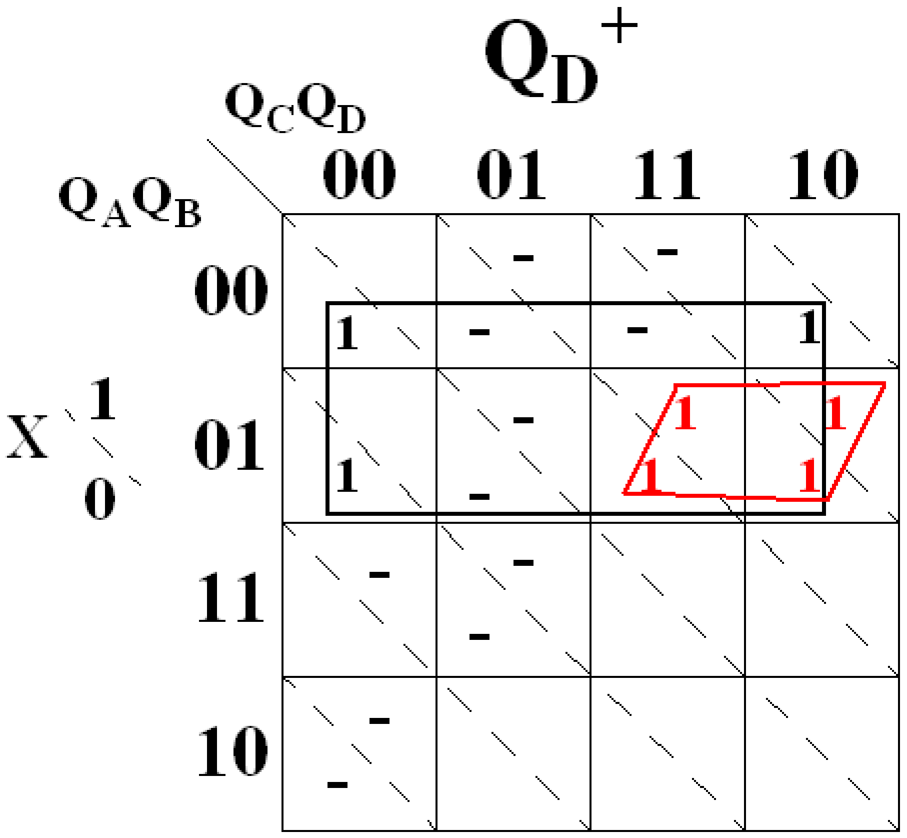
Flip-flop Q_D



Q_AQ_B \ Q_CQ_D	00	01	11	10
00	 1	- -	- -	 1
01	 1	- -	1 1	1 1
11	- 	- -		
10	- -			

$$Q_D^+ = Q_A'X'$$

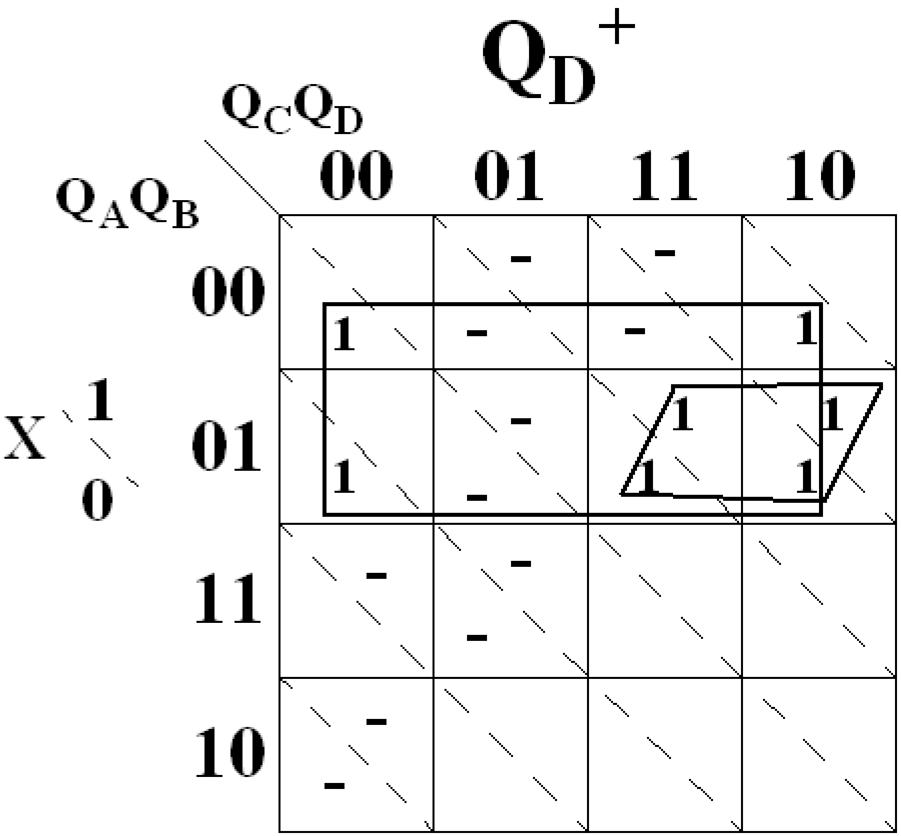
Flip-flop Q_D



Q_AQ_B \ Q_CQ_D	00	01	11	10
00	 1	- -	- -	 1
01	 1	- -	1 1	1 1
11	- 	- -		
10	- -			

$$Q_D^+ = Q_A'X' + Q_A'Q_BQ_C$$

Flip-flop Q_D



$Q_A Q_B$ \ $Q_C Q_D$	00	01	11	10
00	 1	- -	- -	 1
01	 1	- -	1 1	1 1
11	- 	- -		
10	- -			

$$Q_D^+ = Q_A'X' + Q_A'Q_BQ_C$$

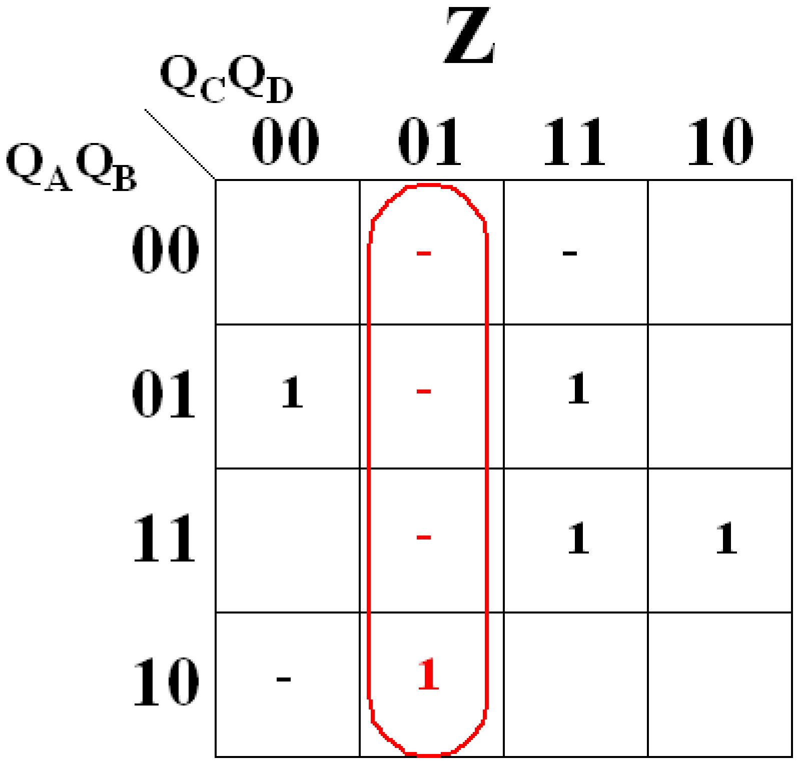
Output Z

Z

Q_AQ_B \ Q_CQ_D	00	01	11	10
00		-	-	
01	1	-	1	
11		-	1	1
10	-	1		

Z=

Output Z

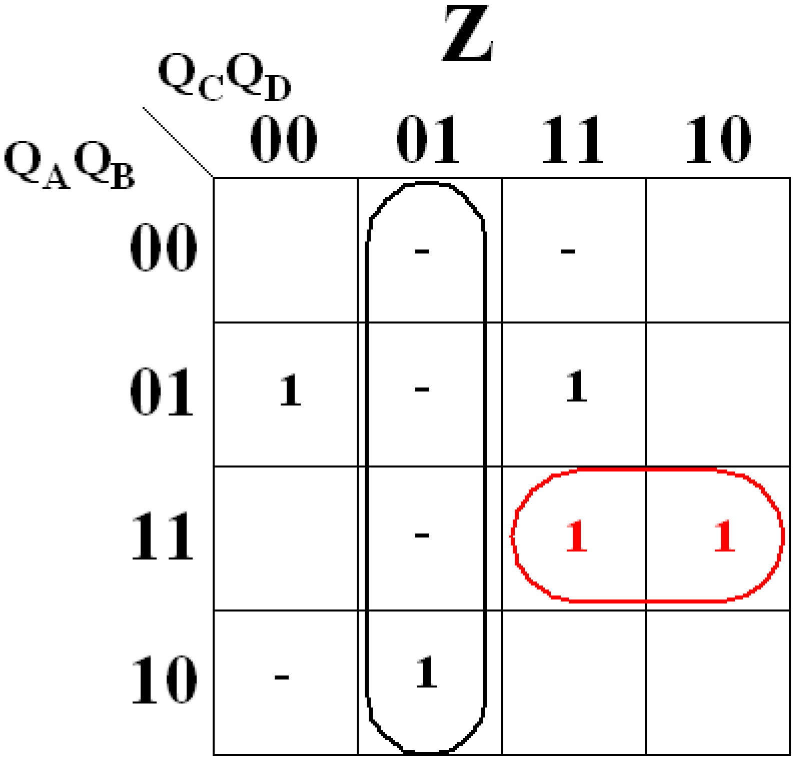


Z

Q_AQ_B \ Q_CQ_D	00	01	11	10
00		-	-	
01	1	-	1	
11		-	1	1
10	-	1		

Z= $Q_C'Q_D$

Output Z

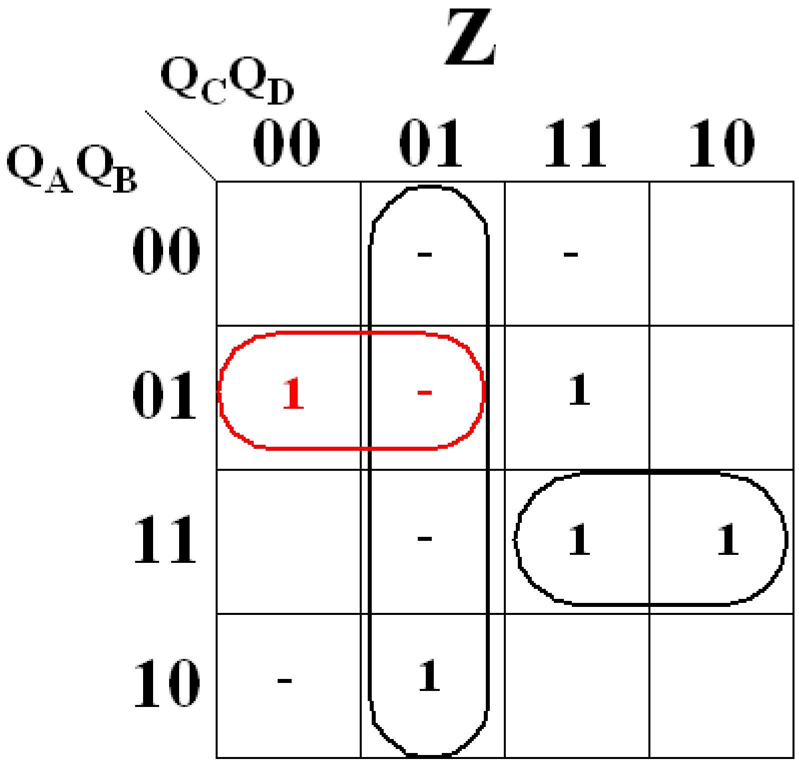


Z

Q_AQ_B \ Q_CQ_D	00	01	11	10
00		-	-	
01	1	-	1	
11		-	1	1
10	-	1		

$$Z = Q_C'Q_D + Q_AQ_BQ_C$$

Output Z



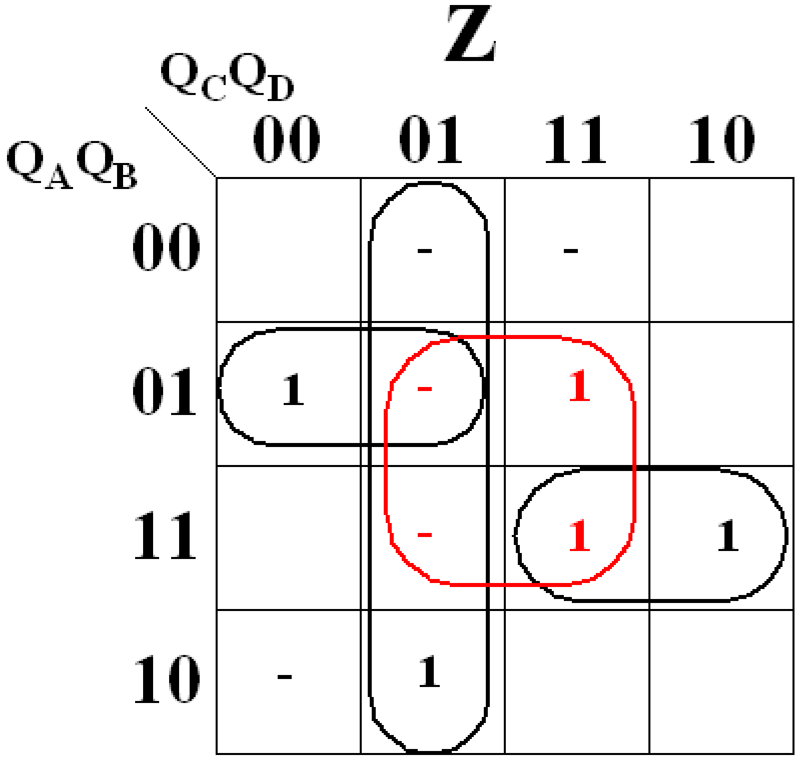
Z

Q_AQ_B \ Q_CQ_D	00	01	11	10
00		-	-	
01	1	-	1	
11		-	1	1
10	-	1		

$Z = Q_C'Q_D + Q_AQ_BQ_C +$

$Q_A'Q_BQ_C'$

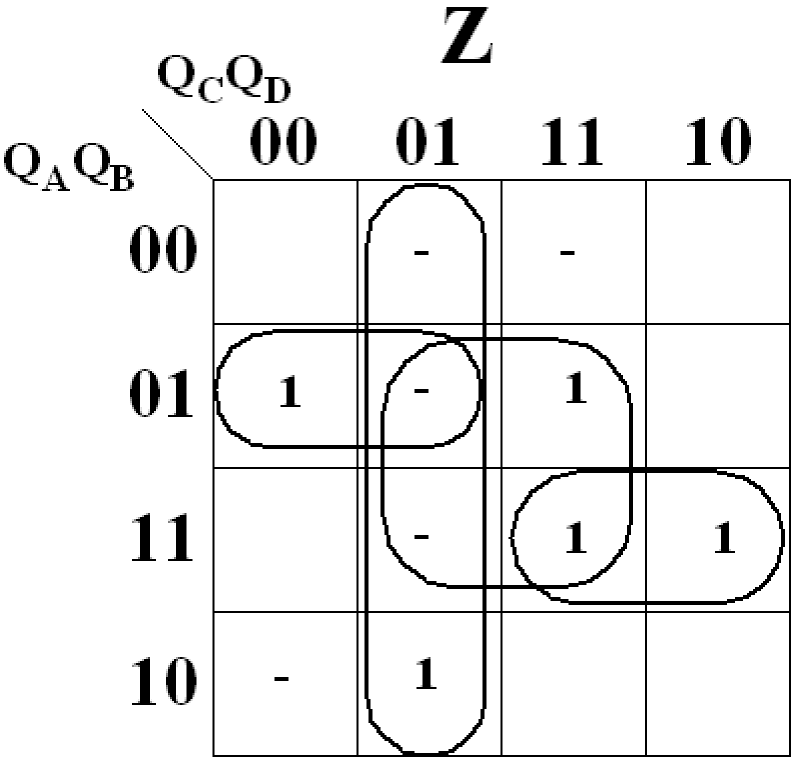
Output Z



$Q_AQ_B \backslash Q_CQ_D$	00	01	11	10
00		-	-	
01	1	-	1	
11		-	1	1
10	-	1		

$$Z = Q_C'Q_D + Q_AQ_BQ_C + Q_A'Q_BQ_C' + Q_BQ_D$$

Output Z

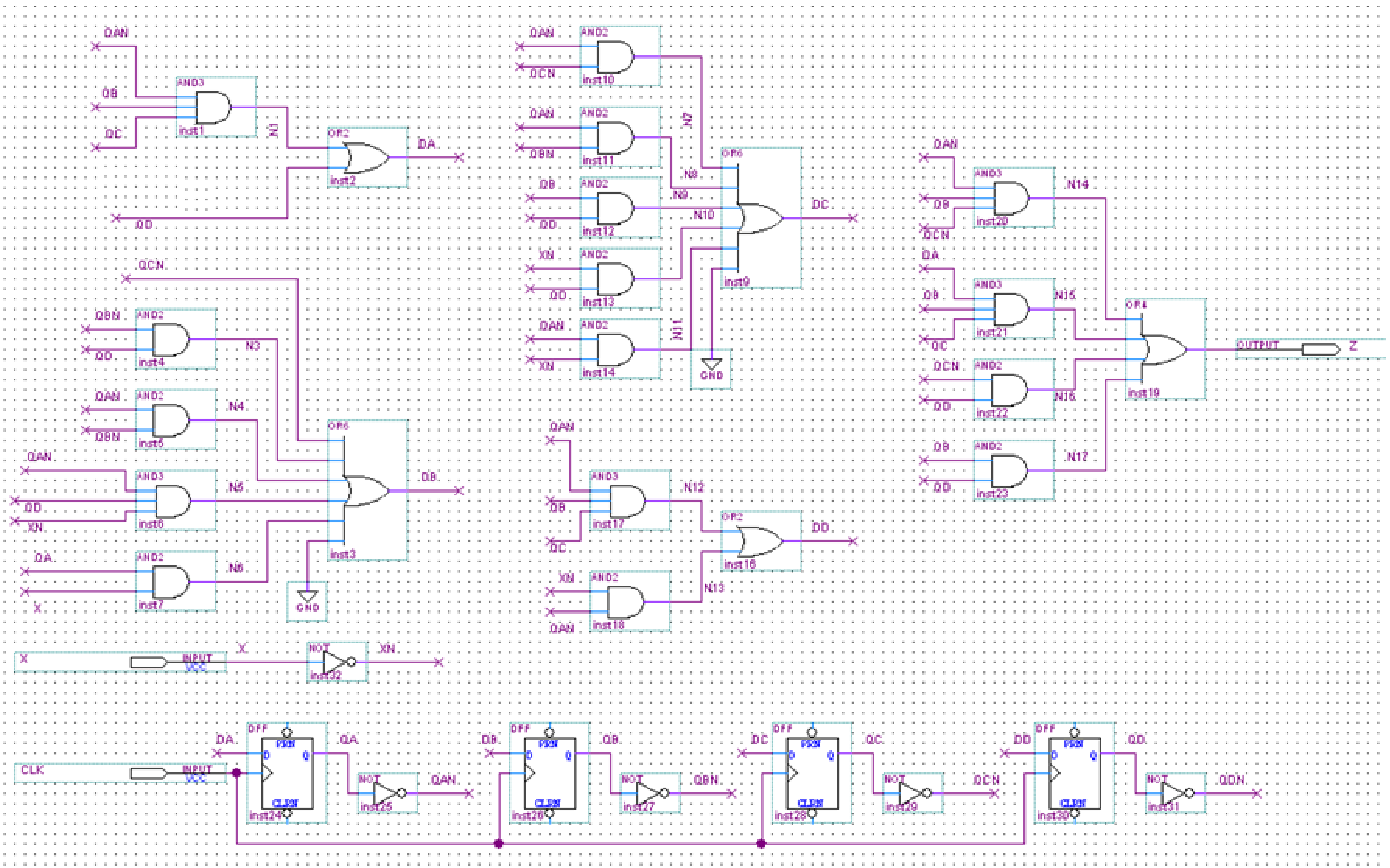


$Q_AQ_B \backslash Q_CQ_D$	00	01	11	10
00		-	-	
01	1	-	1	
11		-	1	1
10	-	1		

$$Z= Q_C'Q_D +Q_AQ_BQ_C+$$
$$Q_A'Q_BQ_C'+Q_BQ_D$$

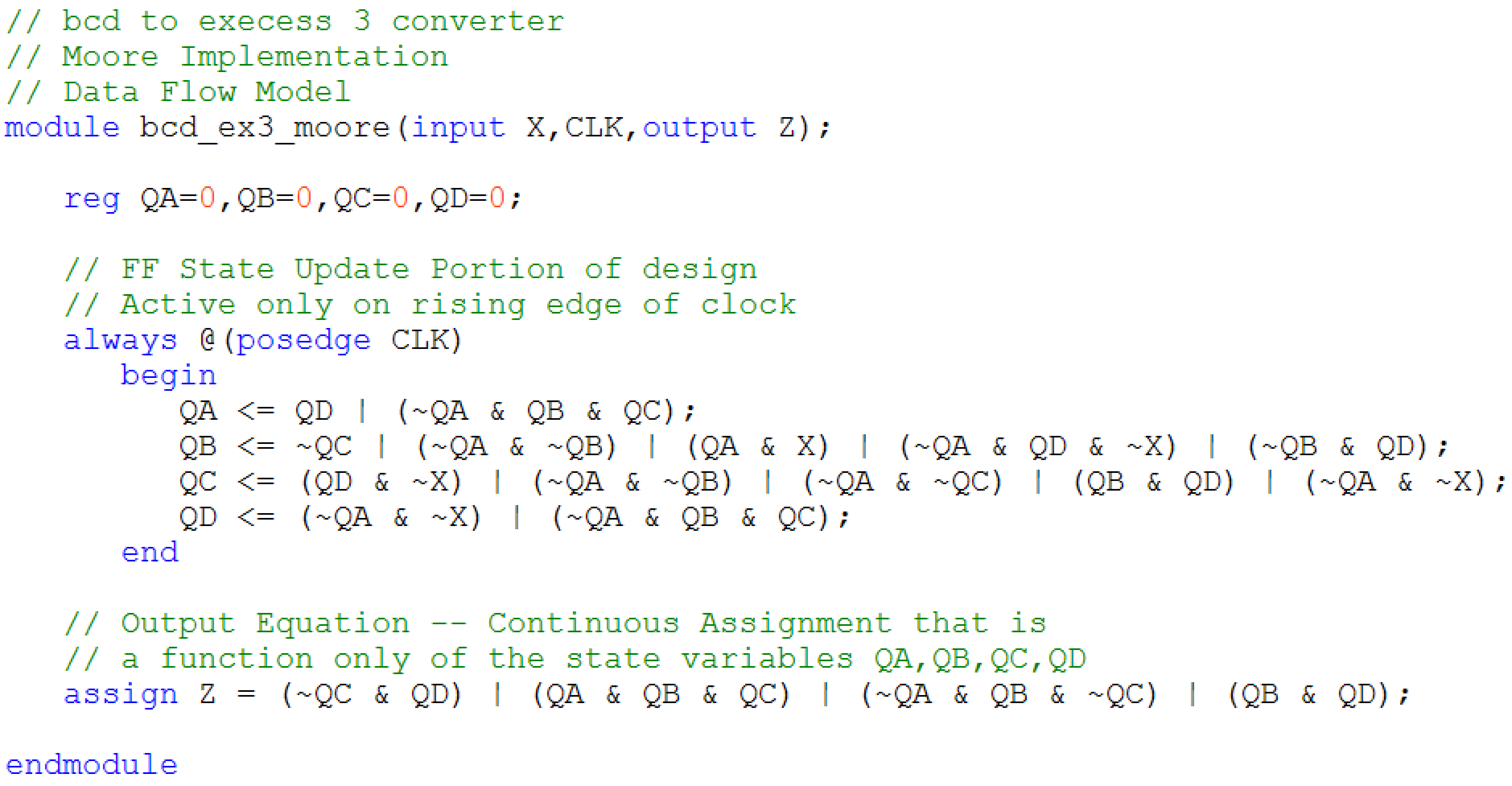
Quartus II Schematic Realization

8421 BCD to Excess3 Code Converter (Moore FSM)



Data Flow Representation Verilog HDL

8421 BCD to Excess3 Code Converter (Moore FSM)



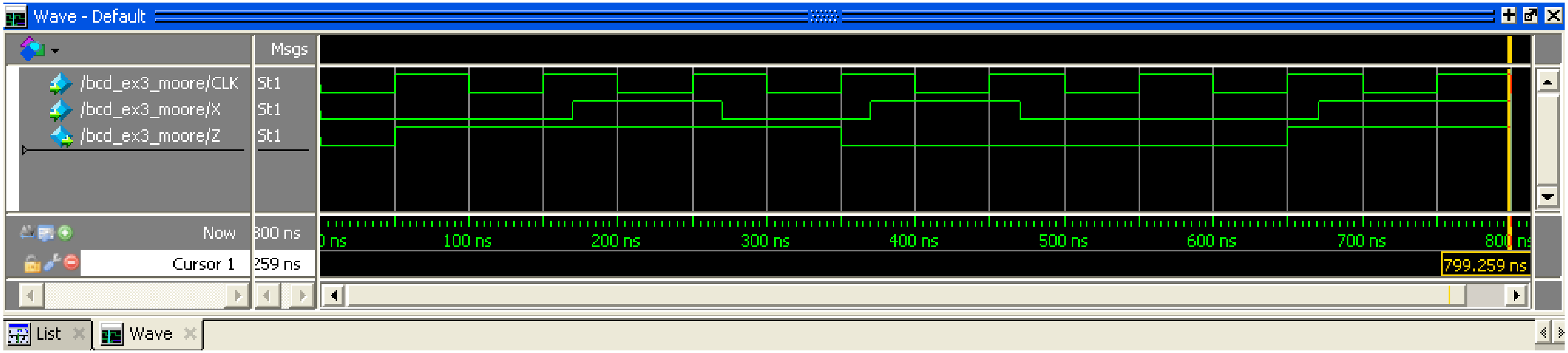
```
// bcd to execess 3 converter
// Moore Implementation
// Data Flow Model
module bcd_ex3_moore(input X,CLK,output Z);

   reg QA=0,QB=0,QC=0,QD=0;

   // FF State Update Portion of design
   // Active only on rising edge of clock
   always @(posedge CLK)
      begin
         QA <= QD | (~QA & QB & QC);
         QB <= ~QC | (~QA & ~QB) | (QA & X) | (~QA & QD & ~X) | (~QB & QD);
         QC <= (QD & ~X) | (~QA & ~QB) | (~QA & ~QC) | (QB & QD) | (~QA & ~X);
         QD <= (~QA & ~X) | (~QA & QB & QC);
      end

   // Output Equation -- Continuous Assignment that is
   // a function only of the state variables QA,QB,QC,QD
   assign Z = (~QC & QD) | (QA & QB & QC) | (~QA & QB & ~QC) | (QB & QD);

endmodule
```

ModelSim™ Waveform

Data Flow 8421 BCD to Excess3 Code Converter (Moore FSM)

- **add wave CLK X Z**
- **force CLK 0 0 ns, 1 50 ns -r 100 ns**
- **force X 0 0 ns, 1 170 ns, 0 270 ns, 1 370 ns, 0 470 ns, 1 670 ns**
- **run 800 ns**

- X = 0010_1001 => Z = 1110_0011



Wave - Default

	Msgs
/bcd_ex3_moore/CLK	St1
/bcd_ex3_moore/X	St1
/bcd_ex3_moore/Z	St1
Now	800 ns
Cursor 1	259 ns