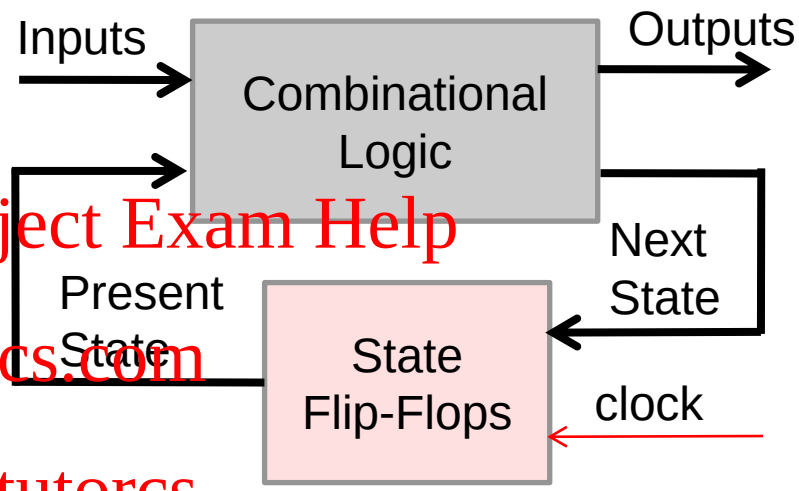
Digital System Design ELEC373/473

Algorithmic State Machines (ASMs)

(Implementation – Classical Method)

Controller Structure

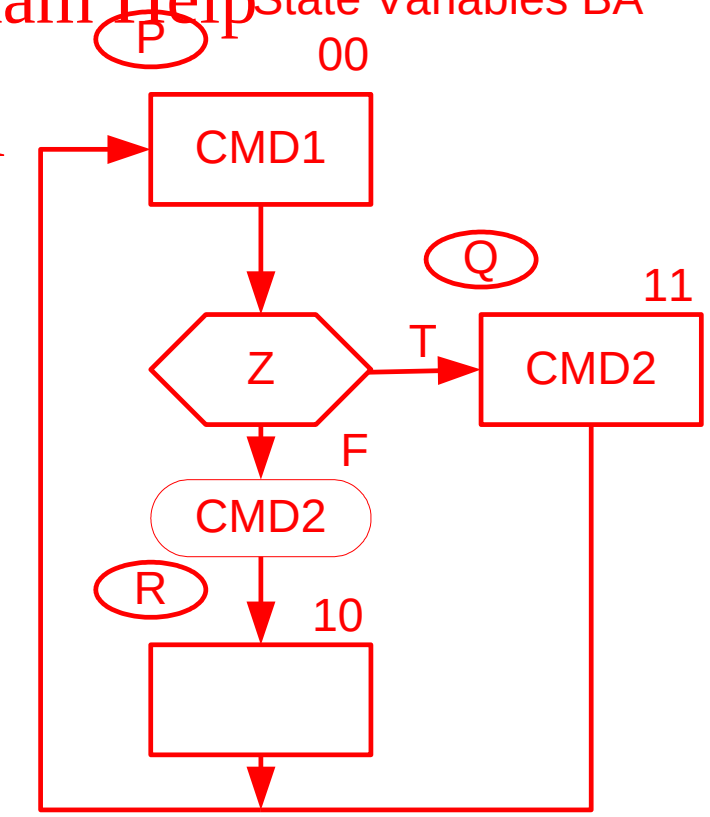
- All sequential circuits can be divided into a combinational block and a storage element block implemented by Flip-Flops.



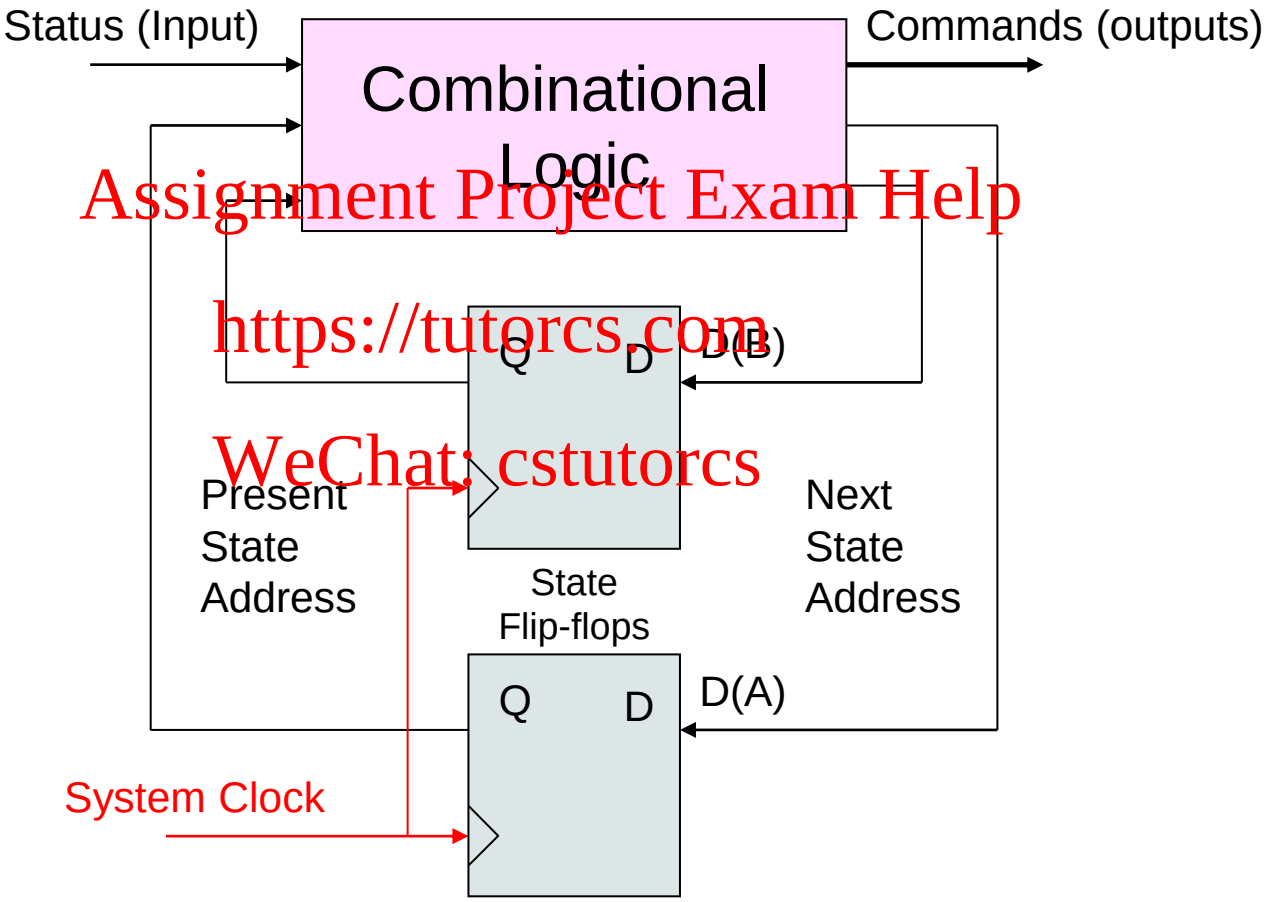
- There are two "classes" of state machines:
  - Moore type outputs are a combinational function of only "Present State" signals.
  - Mealy type outputs are a combinational function of both "Present State" and "Input" signals.

Traditional Design Implementation

- Use flip-flops as state memory (either D, T, or J-K types)
- There are two ways to represent the present state in flip-flop memory
  1. Assign a unique binary number to each state (Encoding Method).
  2. Assign one flip-flop to each state (One Hot Method).
- Using the Encoding method, two state variables are required for encoding 3 states.
- The state assignment is arbitrary.
- Given the present state we need to compute the new state code to load into the state flip flops.

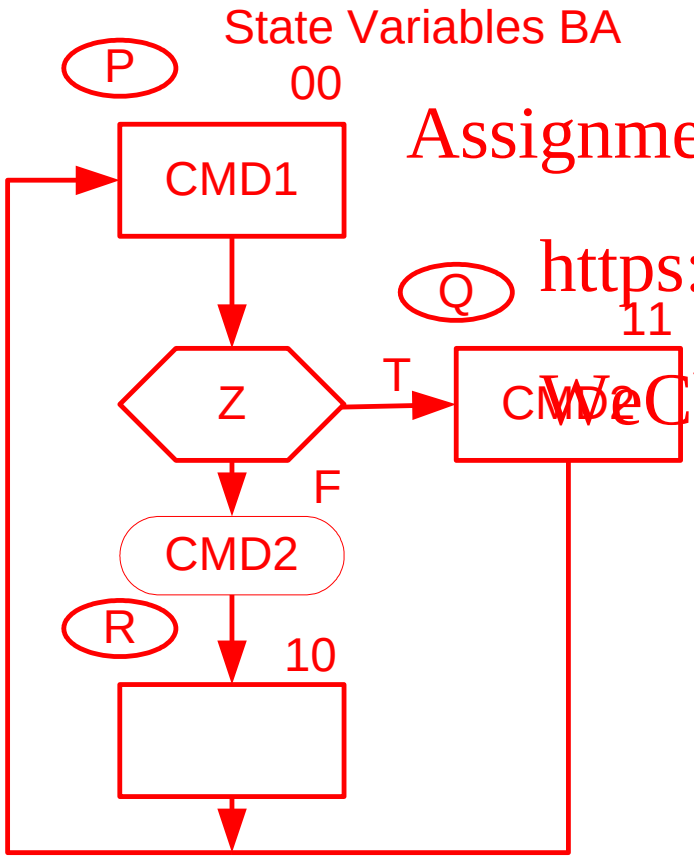


Process Model



4

State Transition Table



Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(R) 00				
(P) 00				
(Q) 11				
(R) 10				
(?) 01				

BA \ Z	0	1
00		
01		
11		
10		

CMD1 =

BA \ Z	0	1
00		
01		
11		
10		

CMD2 =

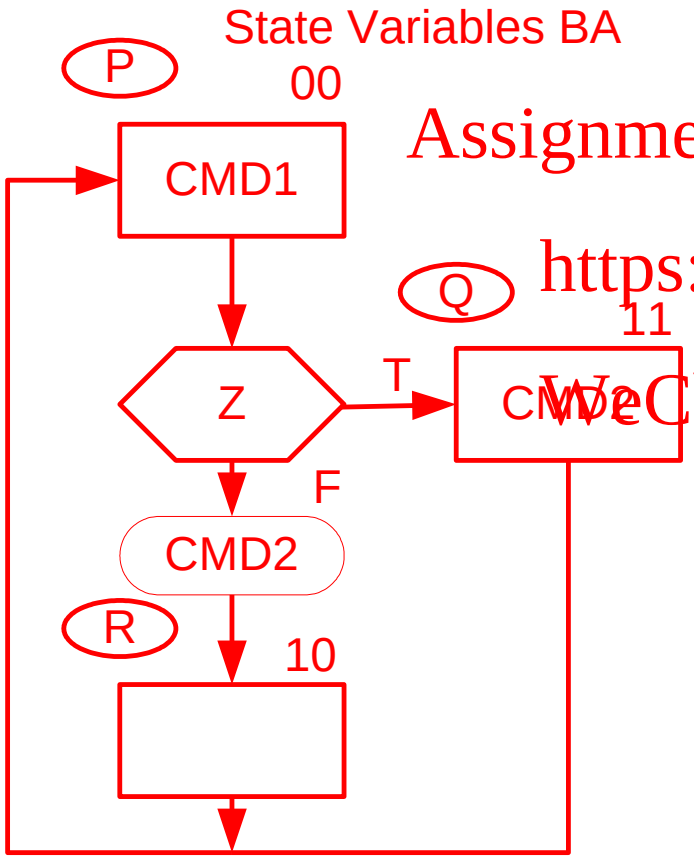
BA \ Z	0	1
00		
01		
11		
10		

D(B) =

BA \ Z	0	1
00		
01		
11		
10		

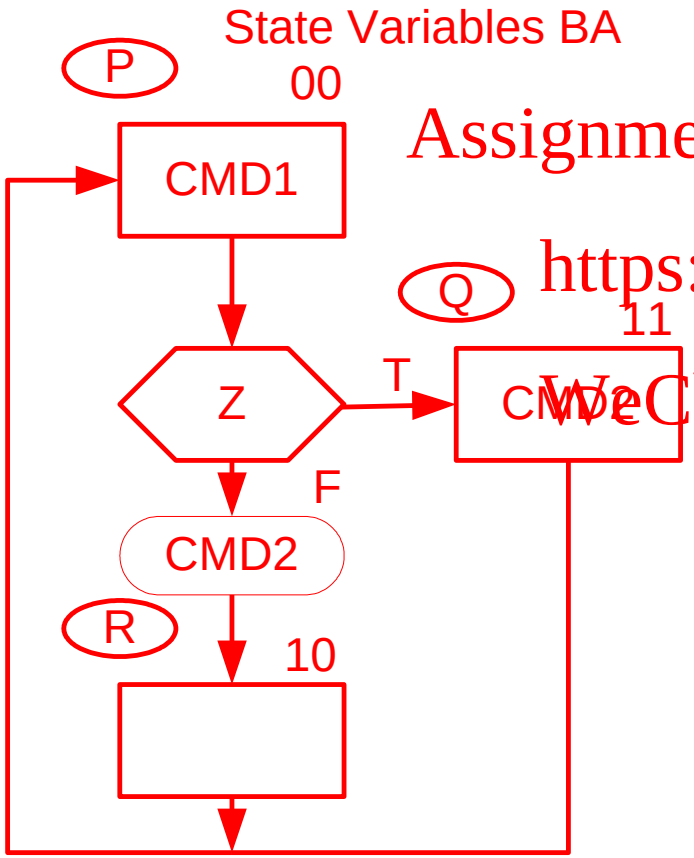
D(A) =

State Transition Table



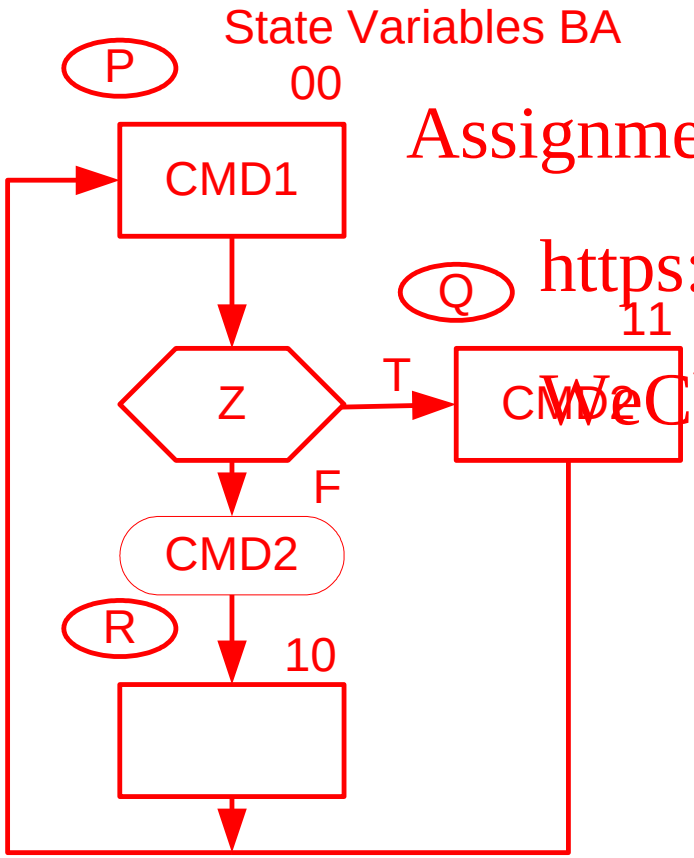
Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(P) 00	0			
(P) 00	1			
(Q) 11	-			
(R) 10	-			
(?) 01	-	-	-	-

State Transition Table



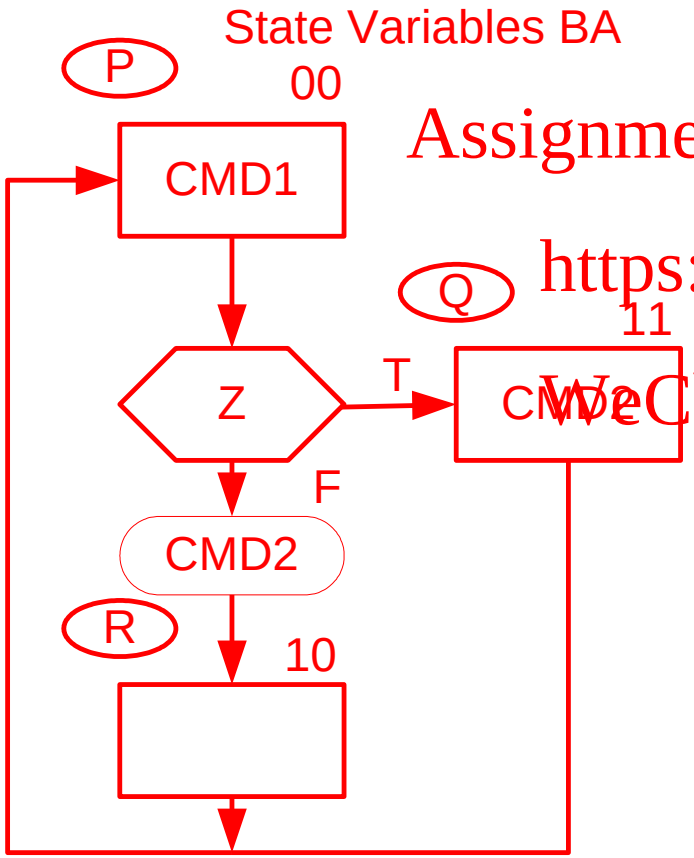
Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(R) 00	1	(R) 10		
(P) 00	1			
(Q) 11	-			
(R) 10	-			
(?) 01	-	-	-	-

State Transition Table



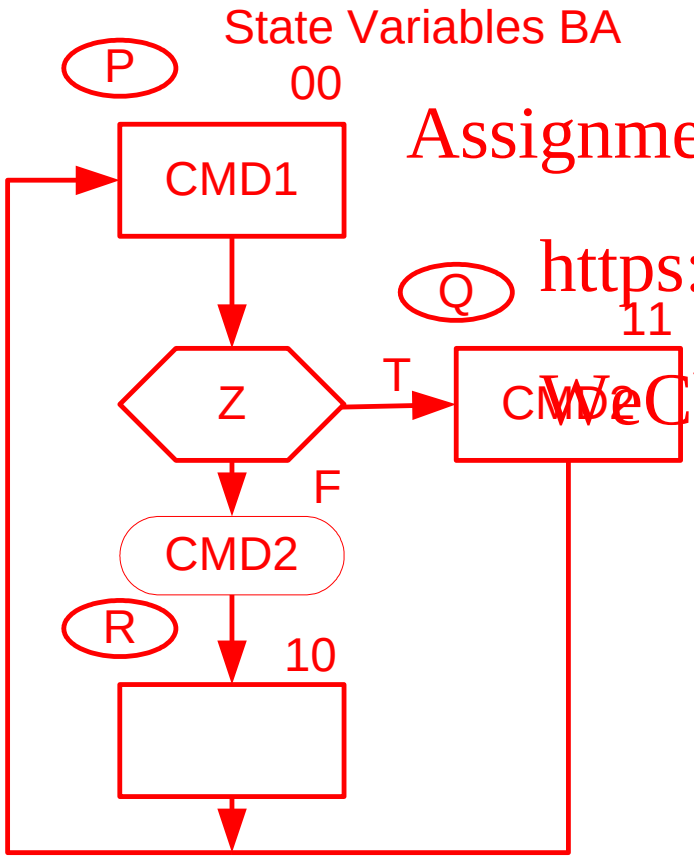
Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(P) 00	1	(R) 10	1	
(P) 00	1			
(Q) 11	-			
(R) 10	-			
(?) 01	-	-	-	-

State Transition Table



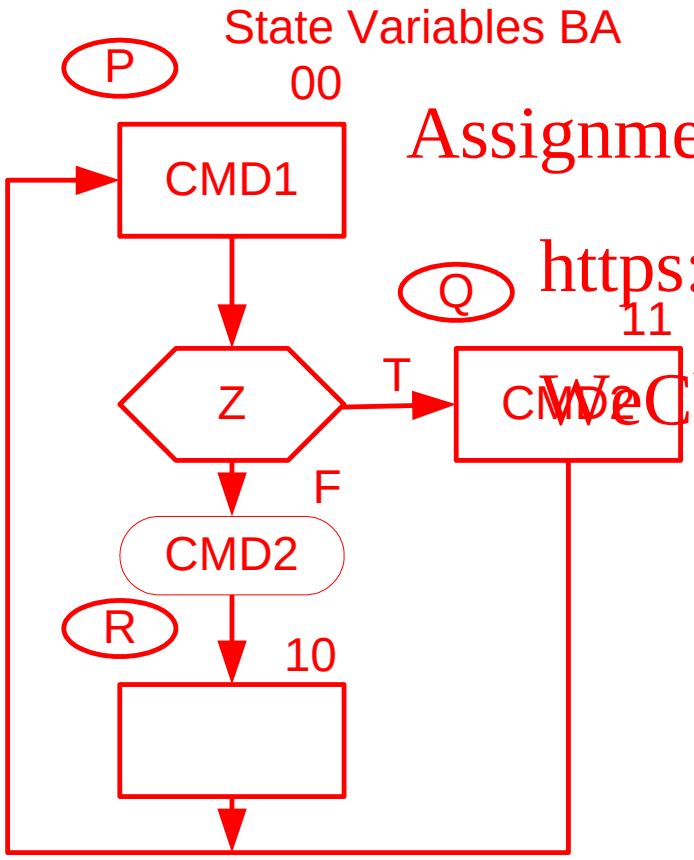
Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(P) 00	0	(R) 10	1	1
(P) 00	1			
(Q) 11	-			
(R) 10	-			
(?) 01	-	-	-	-

State Transition Table



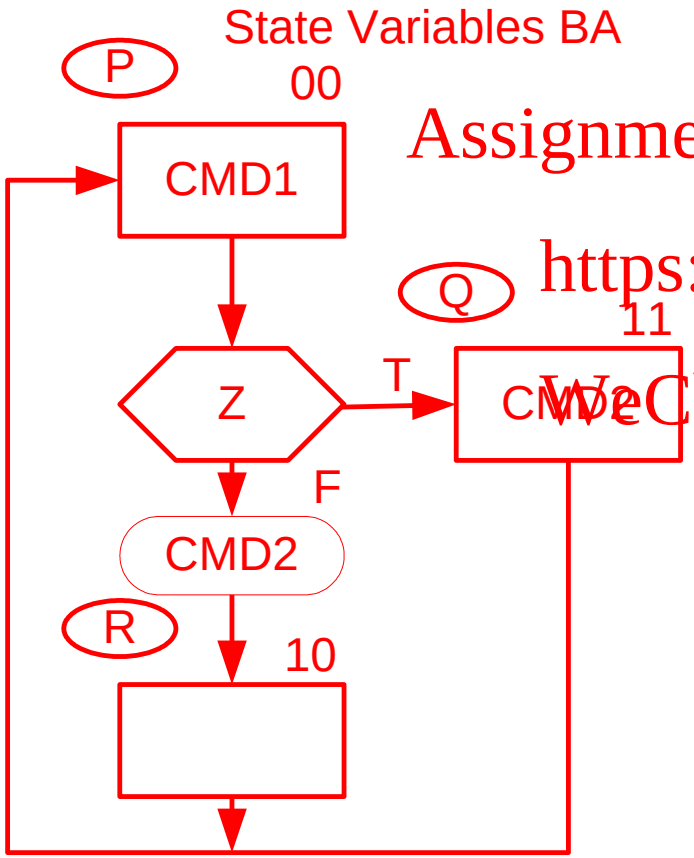
Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(P) 00	1	(R) 10	1	1
(P) 00	1	(Q) 11		
(Q) 11	-			
(R) 10	-			
(?) 01	-	-	-	-

State Transition Table



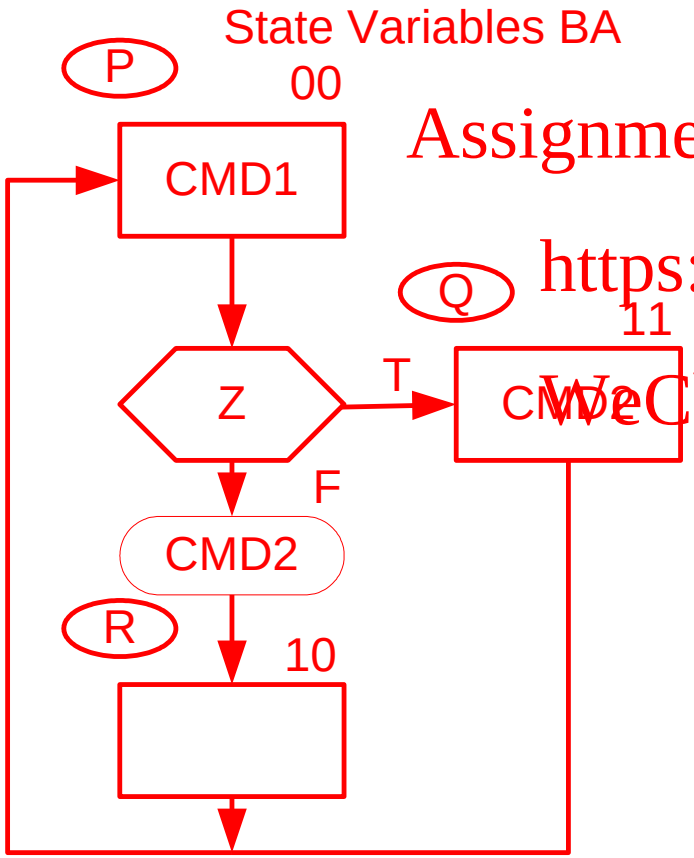
Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(P) 00	1	(R) 10	1	1
(P) 00	1	(Q) 11	1	
(Q) 11	-			
(R) 10	-			
(?) 01	-	-	-	-

State Transition Table



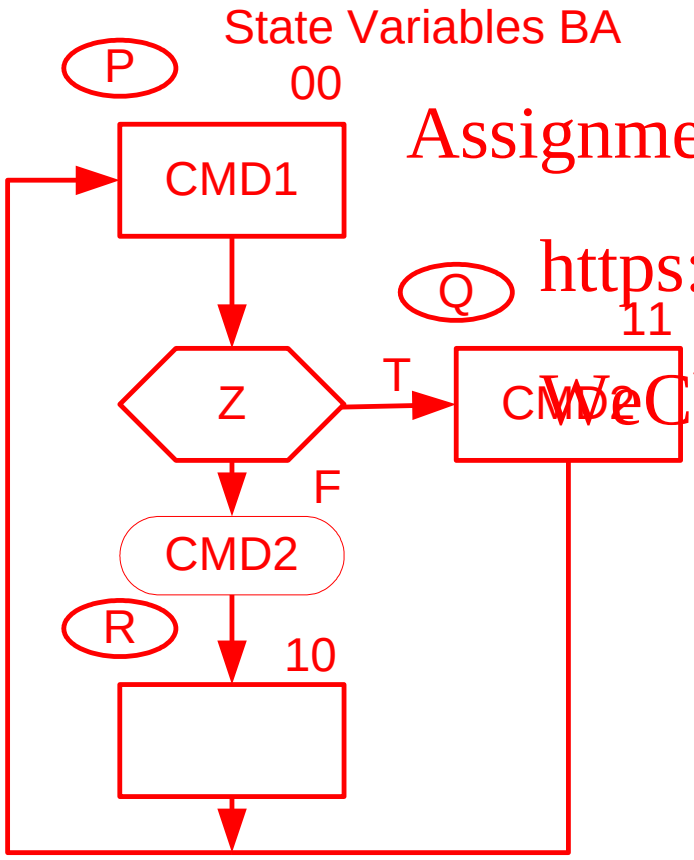
Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(R) 00	0	(R) 10	1	1
(P) 00	1	(Q) 11	1	0
(Q) 11	-			
(R) 10	-			
(?) 01	-	-	-	-

State Transition Table



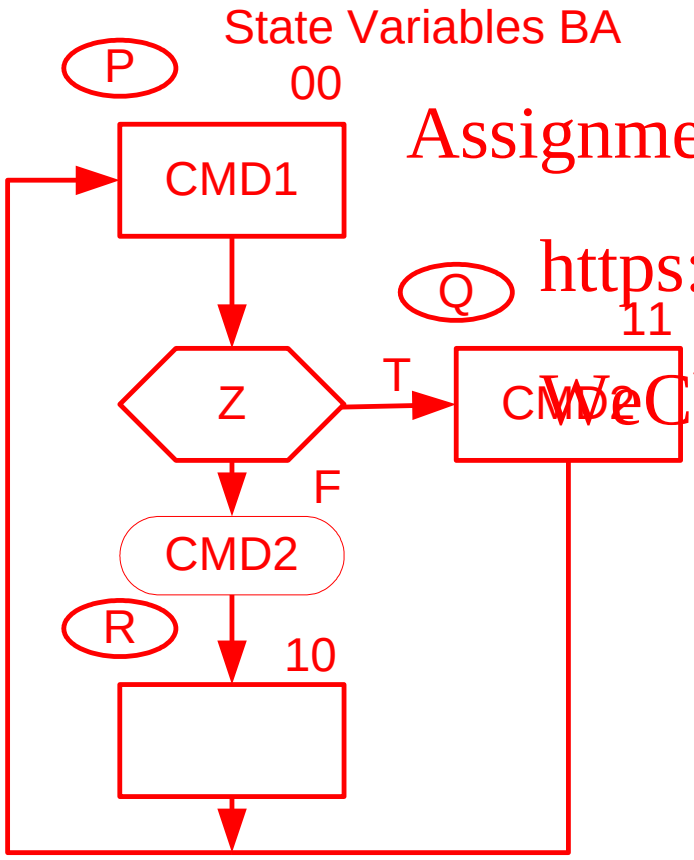
Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(R) 10	0	(R) 10	1	1
(P) 00	1	(Q) 11	1	0
(Q) 11	-	(P) 00		
(R) 10	-			
(?) 01	-	-	-	-

State Transition Table



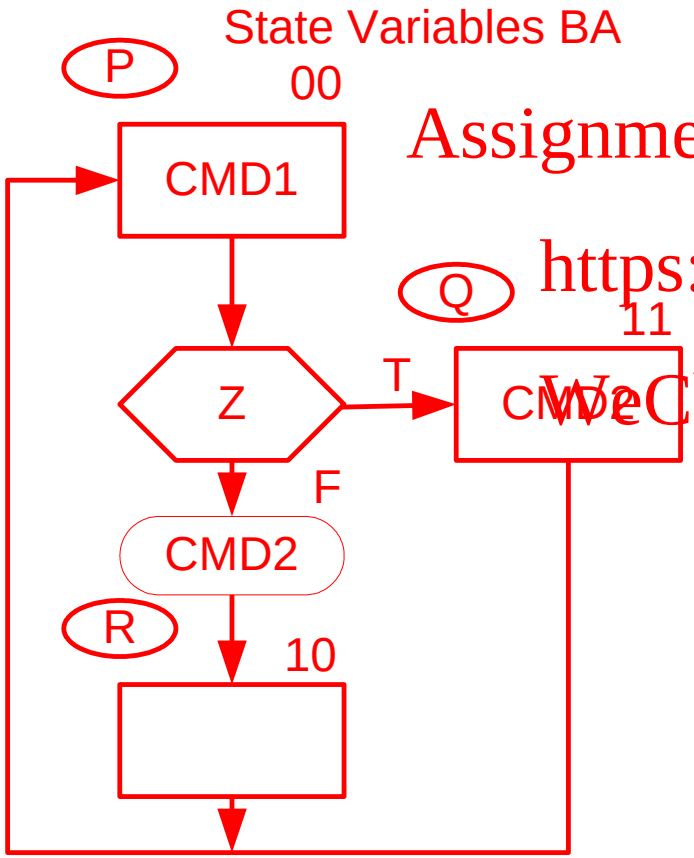
Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(R) 00	0	(R) 10	1	1
(P) 00	1	(Q) 11	1	0
(Q) 11	-	(P) 00	0	
(R) 10	-			
(?) 01	-	-	-	-

State Transition Table



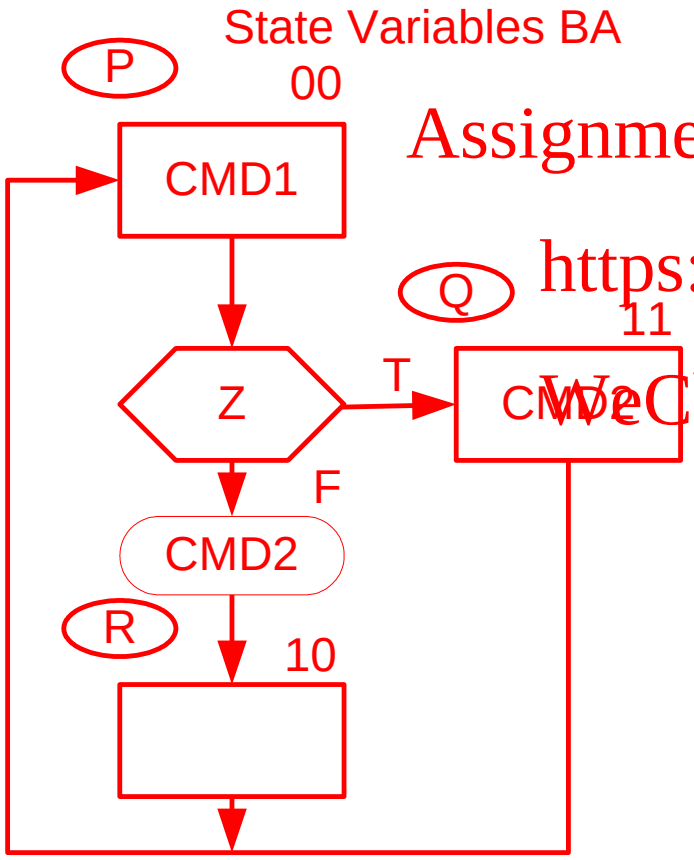
Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(P) 00	0	(R) 10	1	1
(P) 00	1	(Q) 11	1	0
(Q) 11	-	(P) 00	0	1
(R) 10	-			
(?) 01	-	-	-	-

State Transition Table



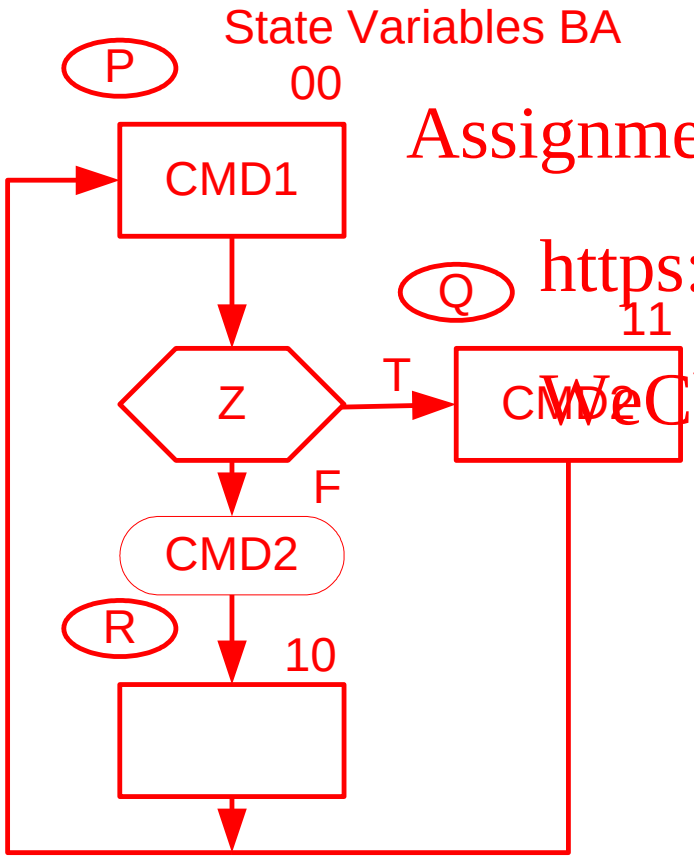
Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(P) 00	0	(R) 10	1	1
(P) 00	1	(Q) 11	1	0
(Q) 11	-	(P) 00	0	1
(R) 10	-	(P) 00		
(?) 01	-	-	-	-

State Transition Table



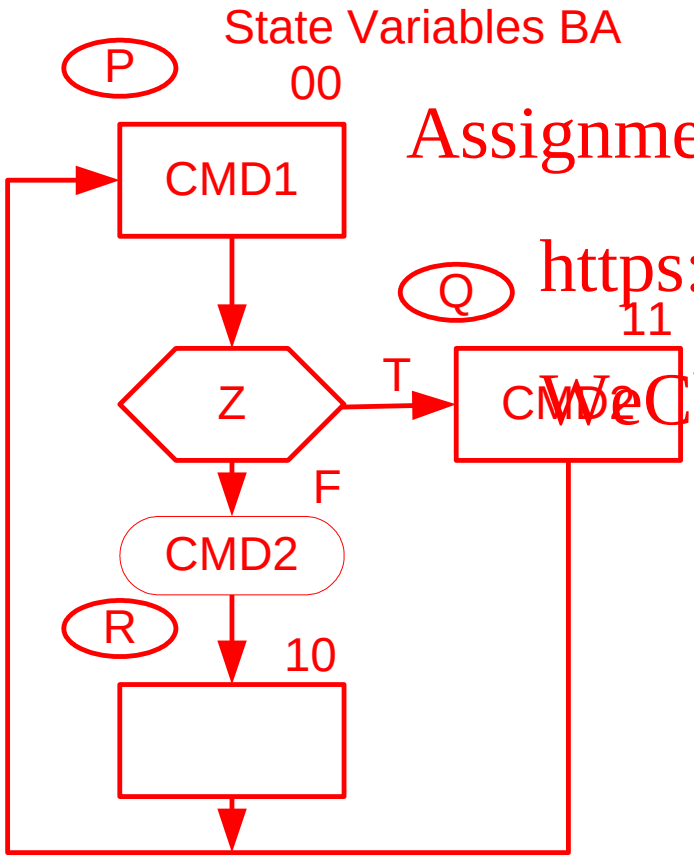
Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(P) 00	0	(R) 10	1	1
(P) 00	1	(Q) 11	1	0
(Q) 11	-	(P) 00	0	1
(R) 10	-	(P) 00	0	
(?) 01	-	-	-	-

State Transition Table



Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(P) 00	0	(R) 10	1	1
(P) 00	1	(Q) 11	1	0
(Q) 11	-	(P) 00	0	1
(R) 10	-	(P) 00	0	0
(?) 01	-	-	-	-

State Transition Table



Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(P) 00	0	(R) 10	1	1
(P) 00	1	(Q) 11	1	0
(Q) 11	-	(P) 00	0	1
(R) 10	-	(P) 00	0	0
(?) 01	-	-	-	-

BA \ Z	0	1
00		
01		
11		
10		

CMD1 =

BA \ Z	0	1
00		
01		
11		
10		

CMD2 =

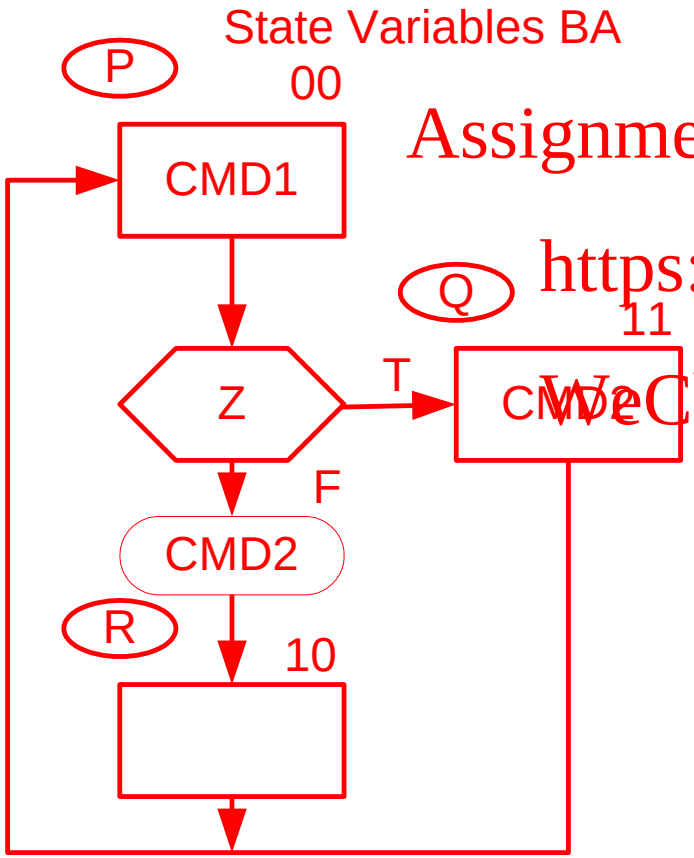
BA \ Z	0	1
00		
01		
11		
10		

D(B) =

BA \ Z	0	1
00		
01		
11		
10		

D(A) =

State Transition Table



Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(P) 00	0	(R) 10	1	1
(P) 00	1	(Q) 11	1	0
(Q) 11	-	(P) 00	0	1
(R) 10	-	(P) 00	0	0
(?) 01	-	-	-	-

BA \ Z	0	1
00		
01		
11		
10		

CMD1 =

BA \ Z	0	1
00		
01		
11		
10		

CMD2 =

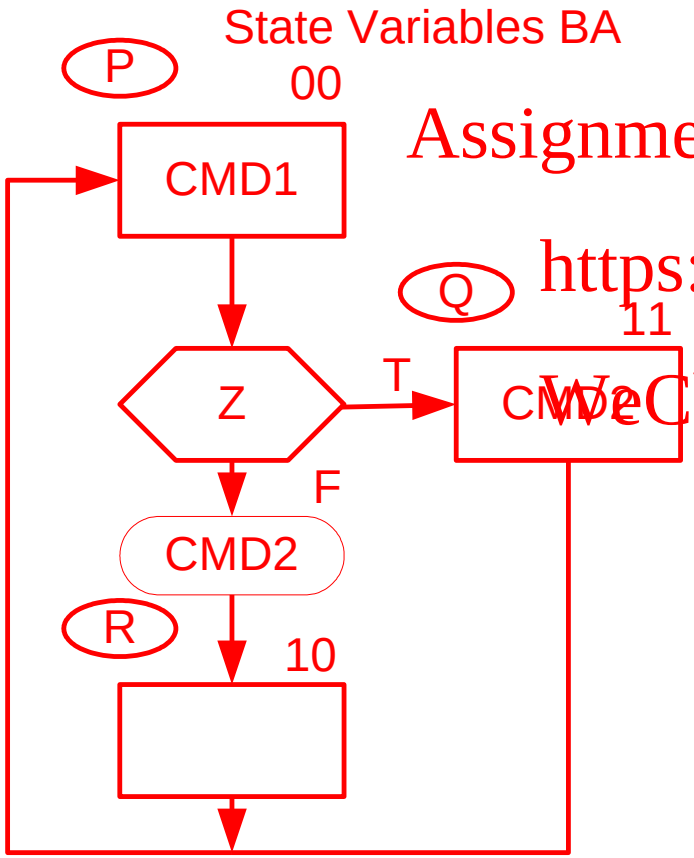
BA \ Z	0	1
00		
01		
11		
10		

D(B) =

BA \ Z	0	1
00	0	
01		
11		
10		

D(A) =

State Transition Table



Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(P) 00	0	(R) 10	1	1
(P) 00	1	(Q) 11	1	0
(Q) 11	-	(P) 00	0	1
(R) 10	-	(P) 00	0	0
(?) 01	-	-	-	-

BA \ Z	0	1
00		
01		
11		
10		

CMD1 =

BA \ Z	0	1
00		
01		
11		
10		

CMD2 =

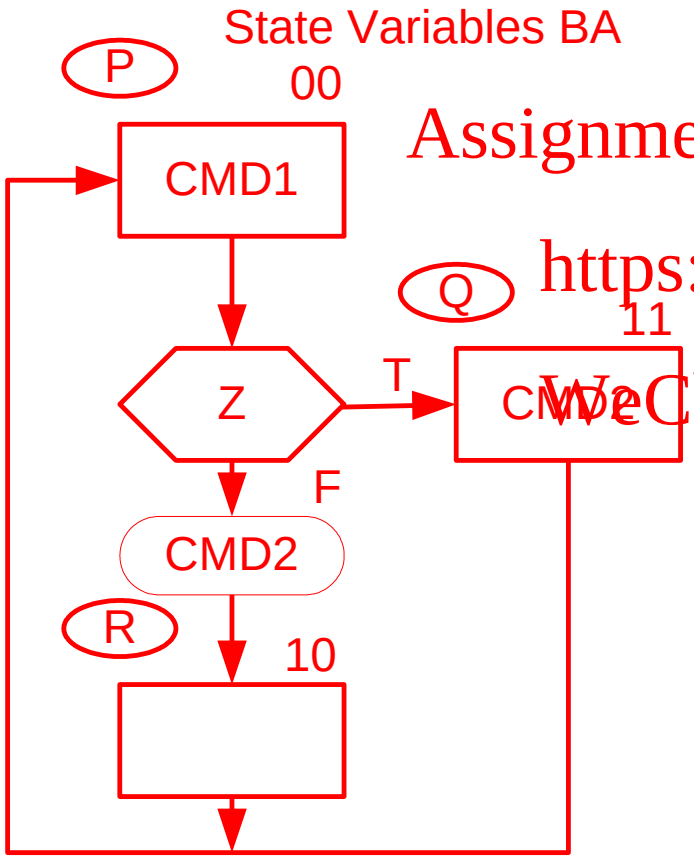
BA \ Z	0	1
00		
01		
11		
10		

D(B) =

BA \ Z	0	1
00	0	1
01		
11		
10		

D(A) =

State Transition Table



Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(P) 00	0	(R) 10	1	1
(P) 00	1	(Q) 11	1	0
(Q) 11	-	(P) 00	0	1
(R) 10	-	(P) 00	0	0
(?) 01	-	-	-	-

BA \ Z	0	1
00		
01		
11		
10		

CMD1 =

BA \ Z	0	1
00		
01		
11		
10		

CMD2 =

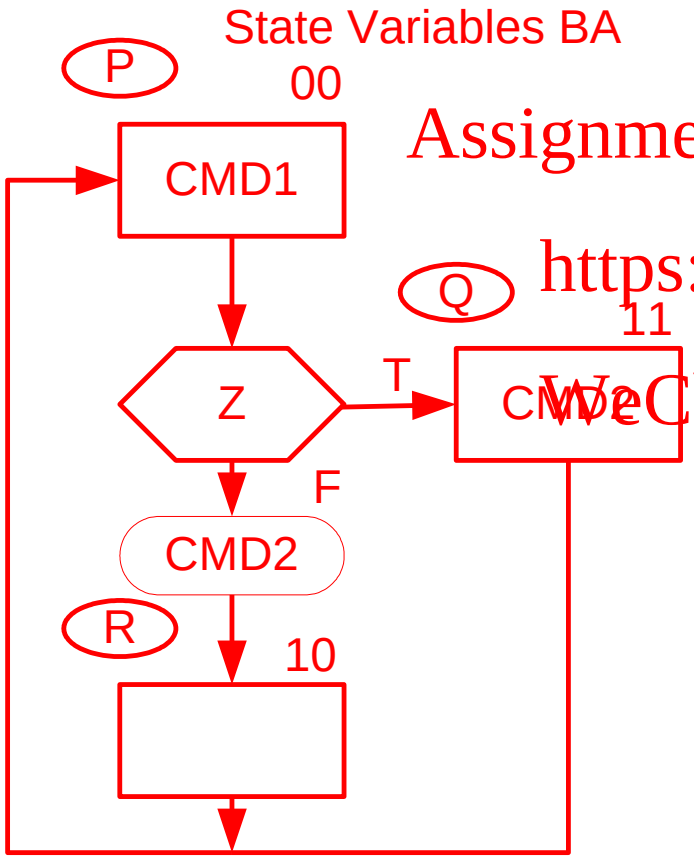
BA \ Z	0	1
00		
01		
11		
10		

D(B) =

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

D(A) =

State Transition Table



Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(P) 00	0	(R) 10	1	1
(P) 00	1	(Q) 11	1	0
(Q) 11	-	(P) 00	0	1
(R) 10	-	(P) 00	0	0
(?) 01	-	-	-	-

BA \ Z	0	1
00		
01		
11		
10		

CMD1 =

BA \ Z	0	1
00		
01		
11		
10		

CMD2 =

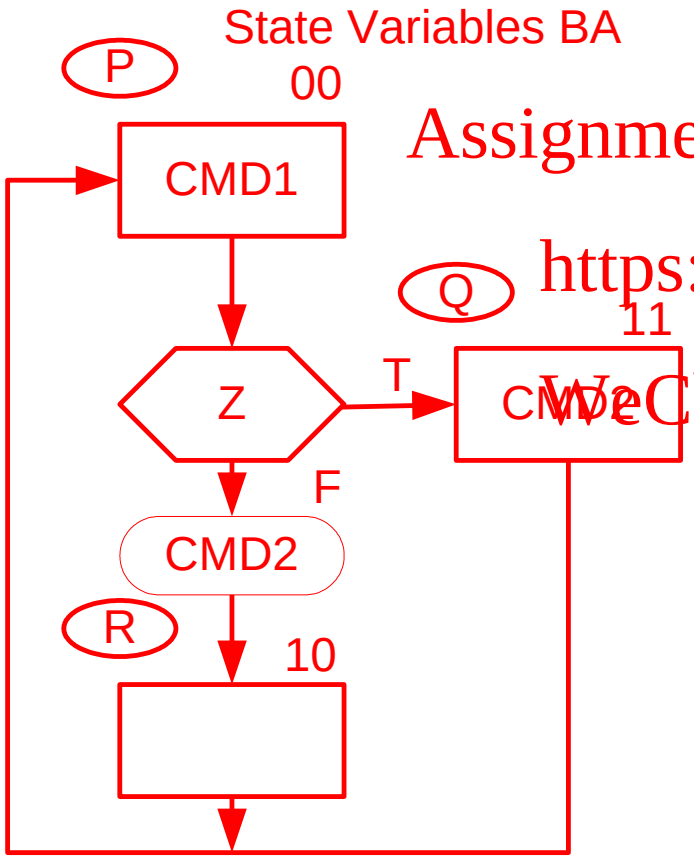
BA \ Z	0	1
00		
01		
11		
10		

D(B) =

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

D(A) =

State Transition Table



Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(P) 00	0	(R) 10	1	1
(P) 00	1	(Q) 11	1	0
(Q) 11	-	(P) 00	0	1
(R) 10	-	(P) 00	0	0
(?) 01	-	-	-	-

CMD1 =

BA \ Z	0	1
00		
01		
11		
10		

CMD2 =

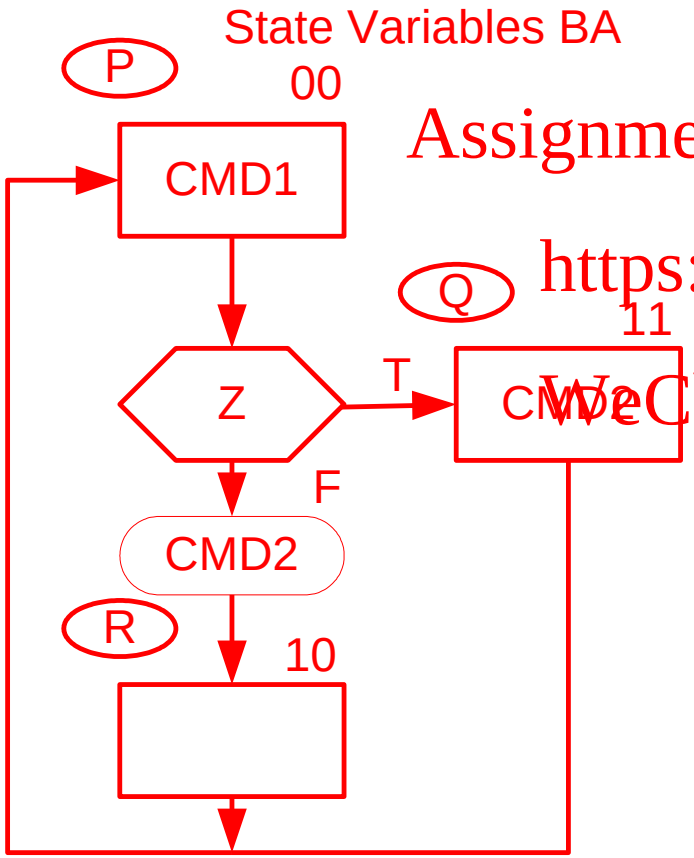
BA \ Z	0	1
00		
01		
11		
10		

D(B) =

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

D(A) =

State Transition Table



Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(P) 00	0	(R) 10	1	1
(P) 00	1	(Q) 11	1	0
(Q) 11	-	(P) 00	0	1
(R) 10	-	(P) 00	0	0
(?) 01	-	-	-	-

BA \ Z	0	1
00		
01		
11		
10		

CMD1 =

BA \ Z	0	1
00		
01		
11		
10		

CMD2 =

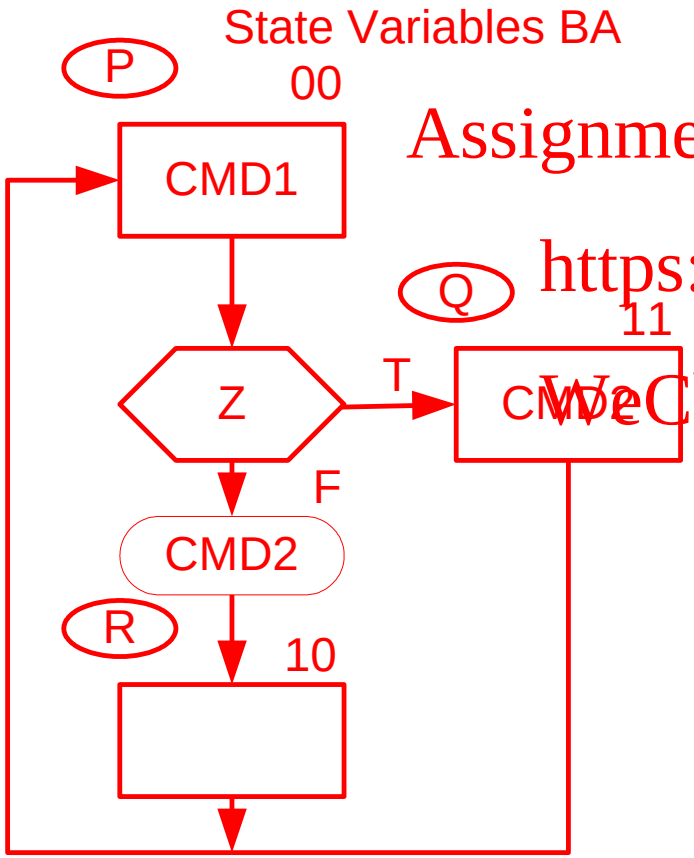
BA \ Z	0	1
00		
01		
11		
10		

D(B) =

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

D(A) = B’Z

State Transition Table



Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(P) 00	0	(R) 10	1	1
(P) 00	1	(Q) 11	1	0
(Q) 11	-	(P) 00	0	1
(R) 10	-	(P) 00	0	0
(?) 01	-	-	-	-

BA \ Z	0	1
00		
01		
11		
10		

CMD1 =

BA \ Z	0	1
00	1	1
01		
11		
10		

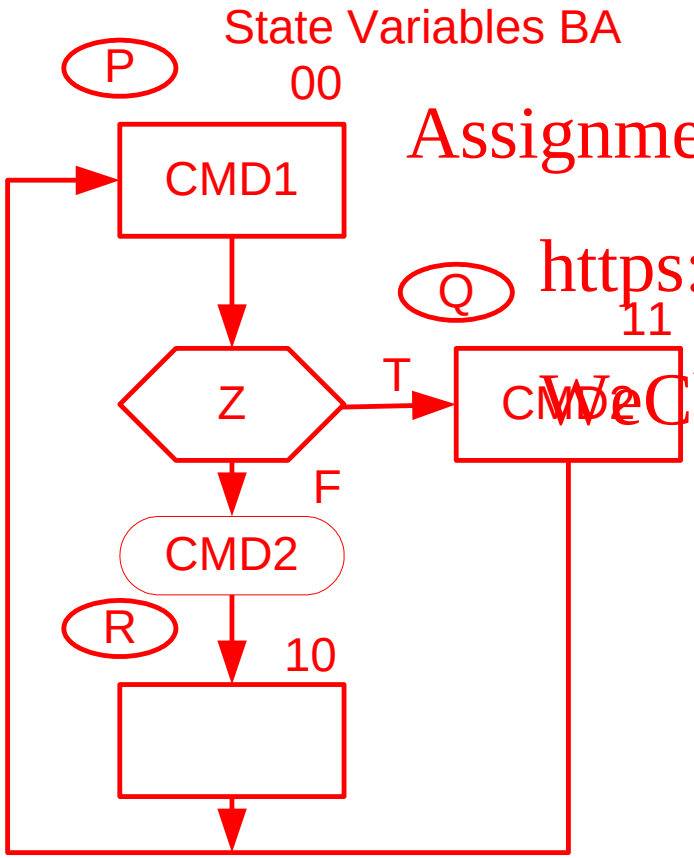
CMD2 =

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

D(B) =

D(A) = B'Z

State Transition Table



Present State (state)BA	Input Z	Next State D(B)D(A)	Outputs CMD1	Outputs CMD2
(R) 00	0	(R) 10	1	1
(P) 00	1	(Q) 11	1	0
(Q) 11	-	(P) 00	0	1
(R) 10	-	(P) 00	0	0
(?) 01	-	-	-	-

CMD1 =

BA \ Z	0	1
00		
01		
11		
10		

CMD2 =

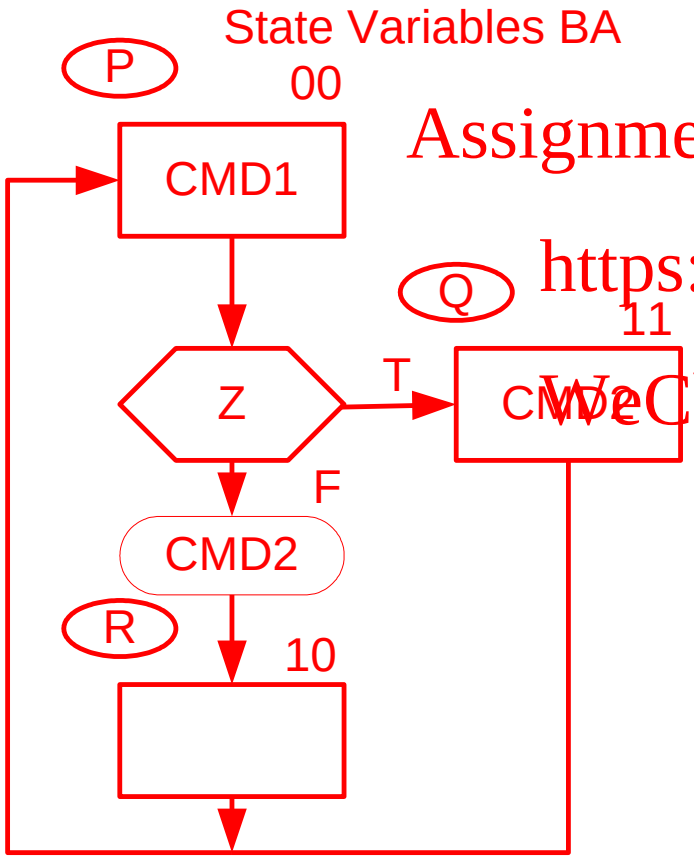
BA \ Z	0	1
00	1	1
01	X	X
11		
10		

D(B) =

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

D(A) = B'Z

State Transition Table



Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(P) 00	0	(R) 10	1	1
(P) 00	1	(Q) 11	1	0
(Q) 11	-	(P) 00	0	1
(R) 10	-	(P) 00	0	0
(?) 01	-	-	-	-

BA \ Z	0	1
00		
01		
11		
10		

CMD1 =

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

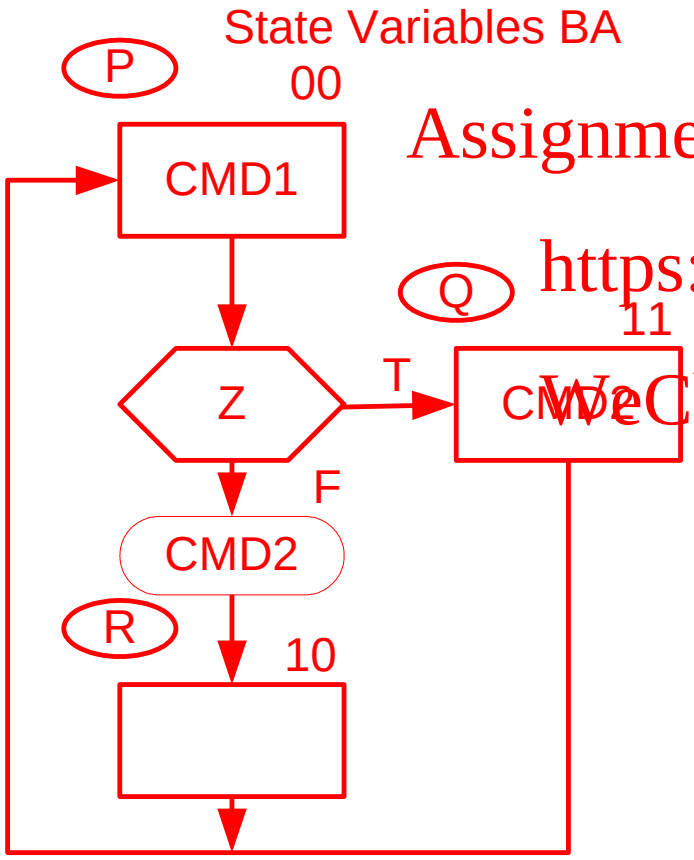
CMD2 =

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

D(B) =

D(A) = B’Z

State Transition Table



Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(P) 00	0	(R) 10	1	1
(P) 00	1	(Q) 11	1	0
(Q) 11	-	(P) 00	0	1
(R) 10	-	(P) 00	0	0
(?) 01	-	-	-	-

BA \ Z	0	1
00		
01		
11		
10		

CMD1 =

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

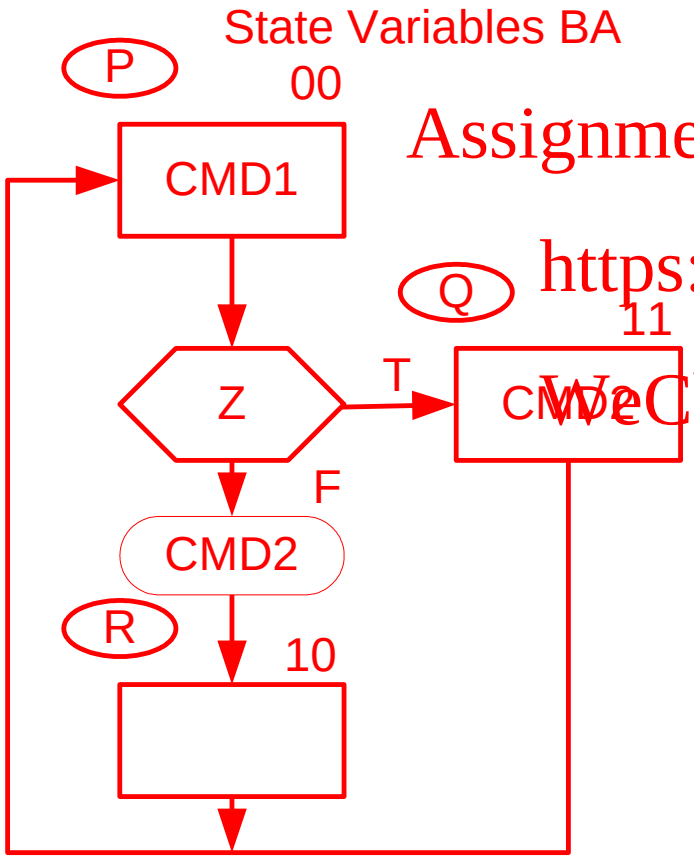
CMD2 =

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

D(B) =

D(A) = B'Z

State Transition Table



Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(P) 00	0	(R) 10	1	1
(P) 00	1	(Q) 11	1	0
(Q) 11	-	(P) 00	0	1
(R) 10	-	(P) 00	0	0
(?) 01	-	-	-	-

CMD1 =

BA \ Z	0	1
00		
01		
11		
10		

CMD2 =

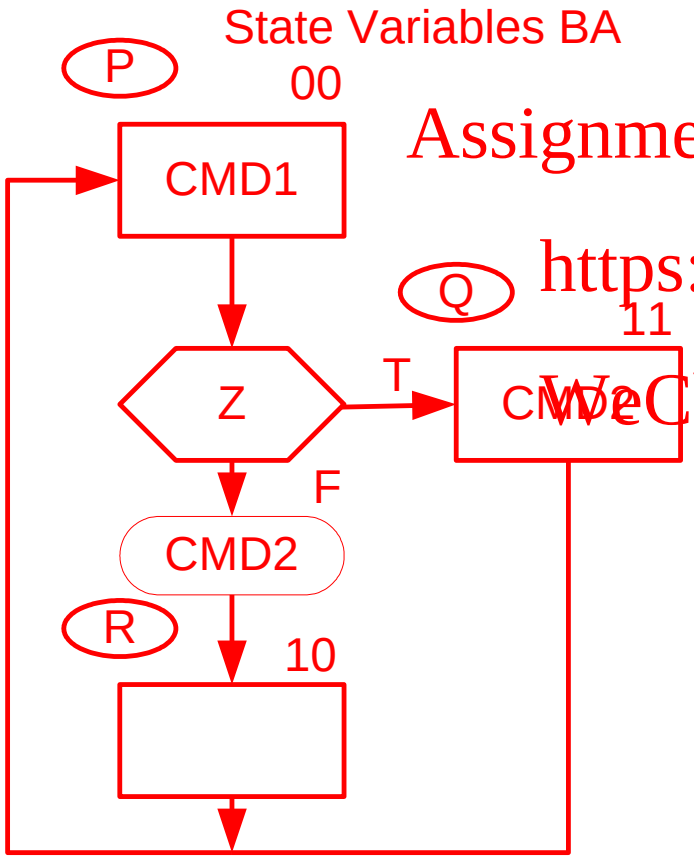
BA \ Z	0	1
00	1	1
01	X	X
11	0	0
10	0	0

D(B) = B’

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

D(A) = B’Z

State Transition Table



Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(P) 00	0	(R) 10	1	1
(P) 00	1	(Q) 11	1	0
(Q) 11	-	(P) 00	0	1
(R) 10	-	(P) 00	0	0
(?) 01	-	-	-	-

BA \ Z	0	1
00		
01		
11		
10		

CMD1 = B’

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

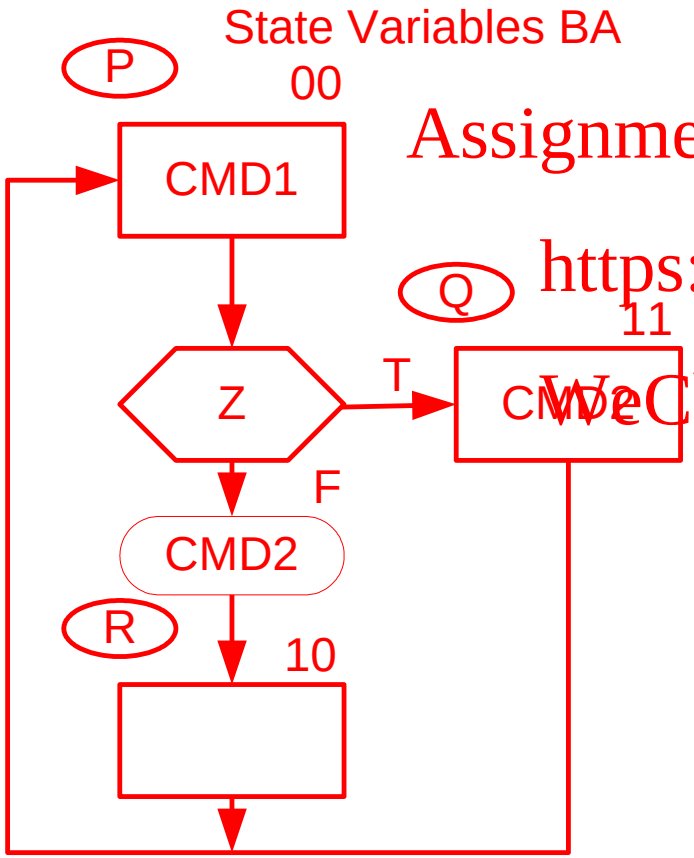
CMD2 =

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

D(B) = B’

D(A) = B’Z

State Transition Table



Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(P) 00	0	(R) 10	1	1
(P) 00	1	(Q) 11	1	0
(Q) 11	-	(P) 00	0	1
(R) 10	-	(P) 00	0	0
(?) 01	-	-	-	-

BA \ Z	0	1
00	1	
01		
11		
10		

CMD1 = B'

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

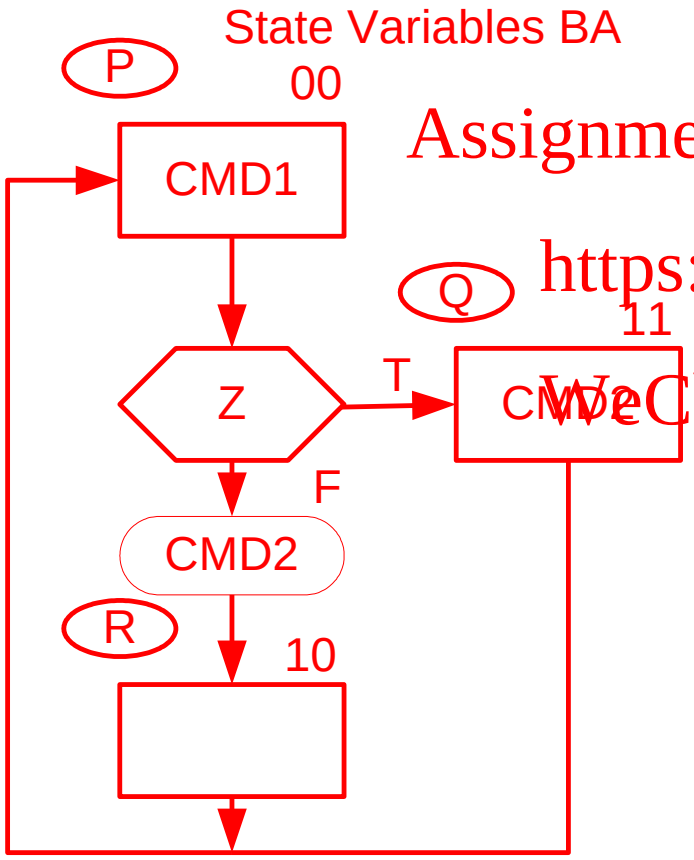
CMD2 =

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

D(B) = B'

D(A) = B'Z

State Transition Table



Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(R) 00	0	(R) 10	1	1
(P) 00	1	(Q) 11	1	0
(Q) 11	-	(P) 00	0	1
(R) 10	-	(P) 00	0	0
(?) 01	-	-	-	-

BA \ Z	0	1
00	1	0
01		
11		
10		

CMD1 = B'

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

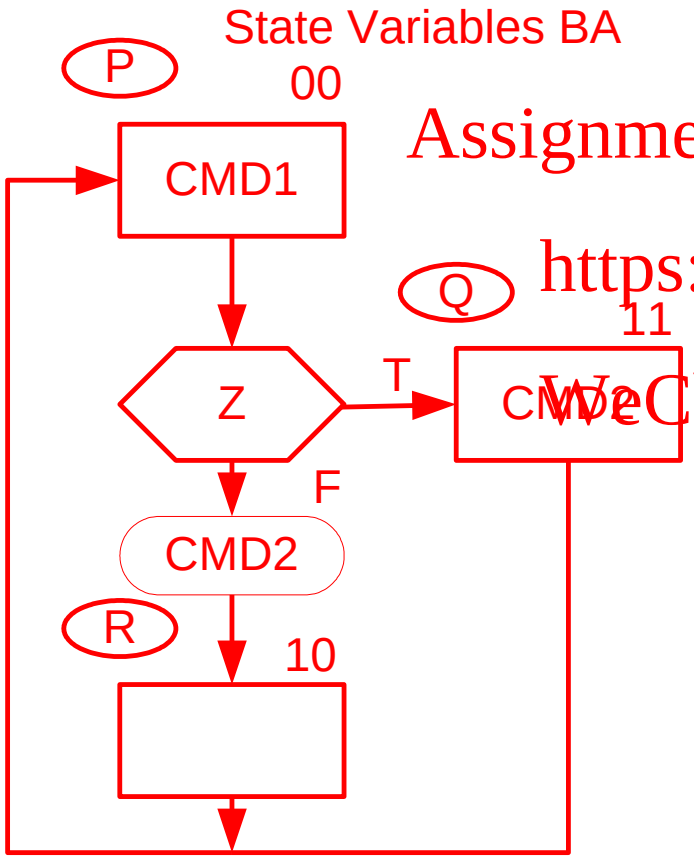
CMD2 =

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

D(B) = B'

D(A) = B'Z

State Transition Table



Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(P) 00	0	(R) 10	1	1
(P) 00	1	(Q) 11	1	0
(Q) 11	-	(P) 00	0	1
(R) 10	-	(P) 00	0	0
(?) 01	-	-	-	-

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

CMD1 = B'

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

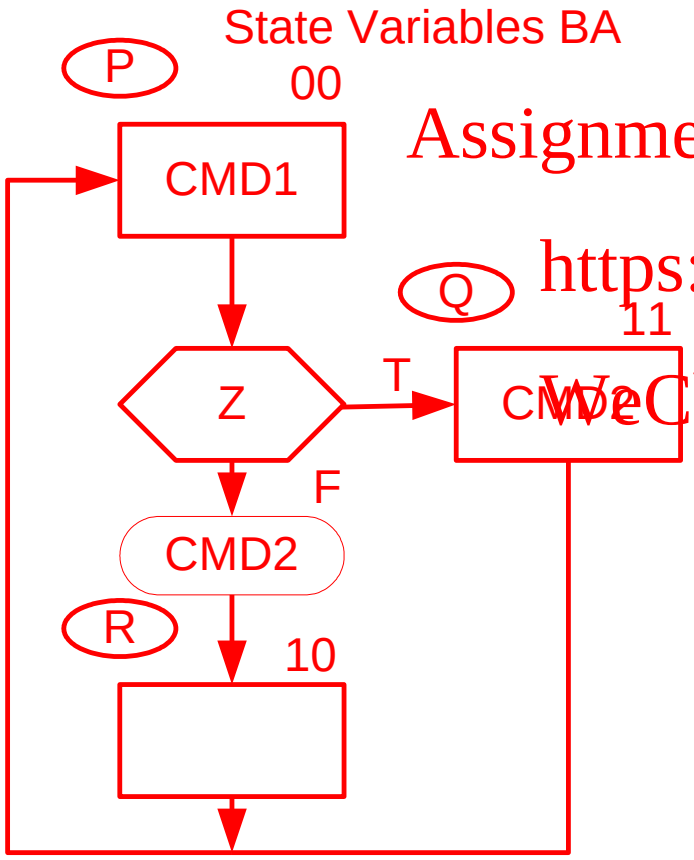
CMD2 =

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

D(B) = B'

D(A) = B'Z

State Transition Table



Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(R) 00	0	(R) 10	1	1
(P) 00	1	(Q) 11	1	0
(Q) 11	-	(P) 00	0	1
(R) 10	-	(P) 00	0	0
(?) 01	-	-	-	-

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

CMD1 = B’

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

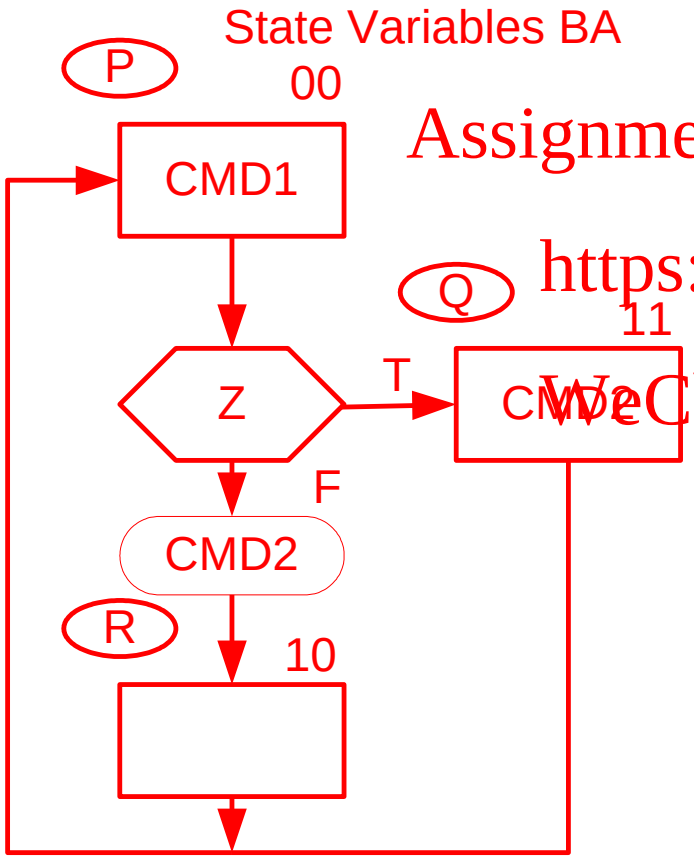
CMD2 =

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

D(B) = B’

D(A) = B’Z

State Transition Table



Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(P) 00	0	(R) 10	1	1
(P) 00	1	(Q) 11	1	0
(Q) 11	-	(P) 00	0	1
(R) 10	-	(P) 00	0	0
(?) 01	-	-	-	-

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

CMD1 = B’

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

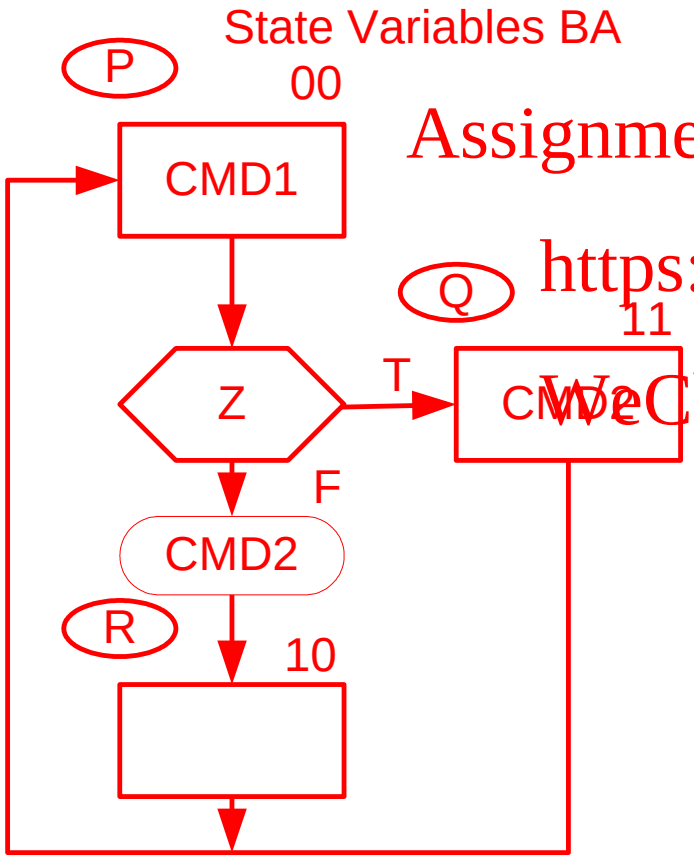
CMD2 =

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

D(B) = B’

D(A) = B’Z

State Transition Table



Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(P) 00	0	(R) 10	1	1
(P) 00	1	(Q) 11	1	0
(Q) 11	-	(P) 00	0	1
(R) 10	-	(P) 00	0	0
(?) 01	-	-	-	-

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

CMD1 = B’

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

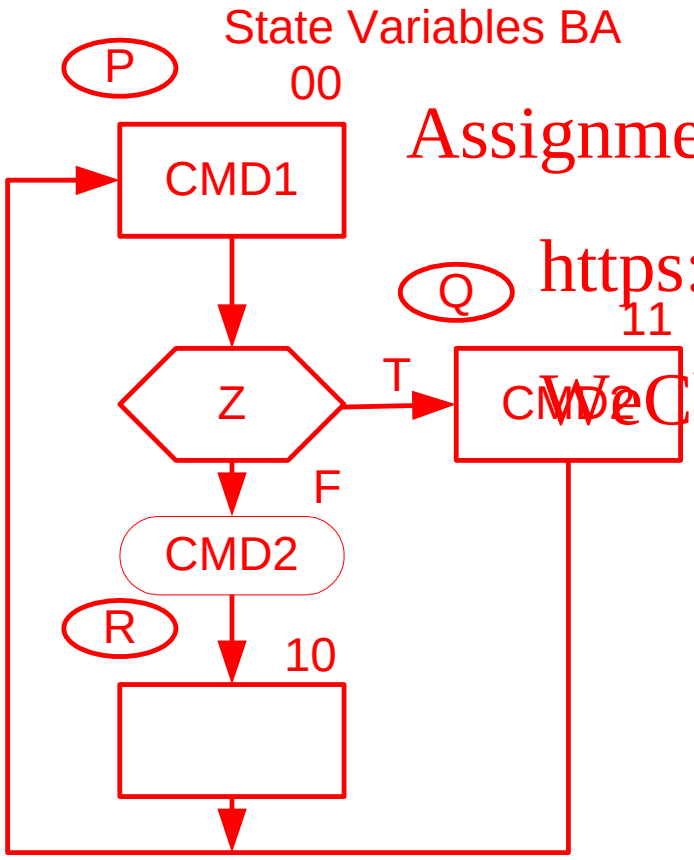
CMD2 =

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

D(B) = B’

D(A) = B’Z

State Transition Table



Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(P) 00	0	(R) 10	1	1
(P) 00	1	(Q) 11	1	0
(Q) 11	-	(P) 00	0	1
(R) 10	-	(P) 00	0	0
(?) 01	-	-	-	-

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

CMD1 = B’

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

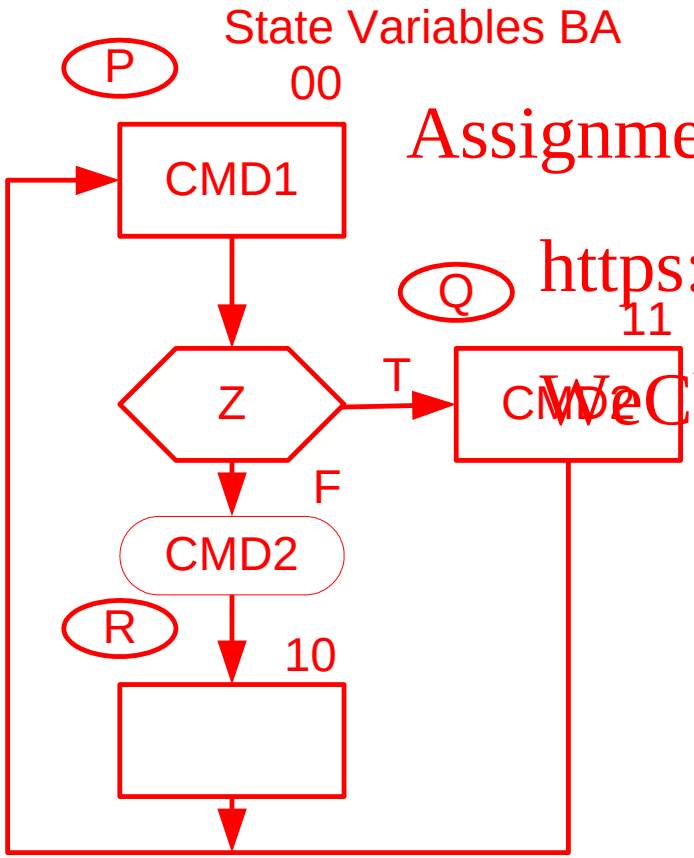
CMD2 =

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

D(B) = B’

D(A) = B’Z

State Transition Table



Present State	Input	Next State	Outputs	
(state)BA	Z	D(B)D(A)	CMD1	CMD2
(P) 00	0	(R) 10	1	1
(P) 00	1	(Q) 11	1	0
(Q) 11	-	(P) 00	0	1
(R) 10	-	(P) 00	0	0
(?) 01	-	-	-	-

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

CMD1 = B’

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

CMD2 =A+B’Z’

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

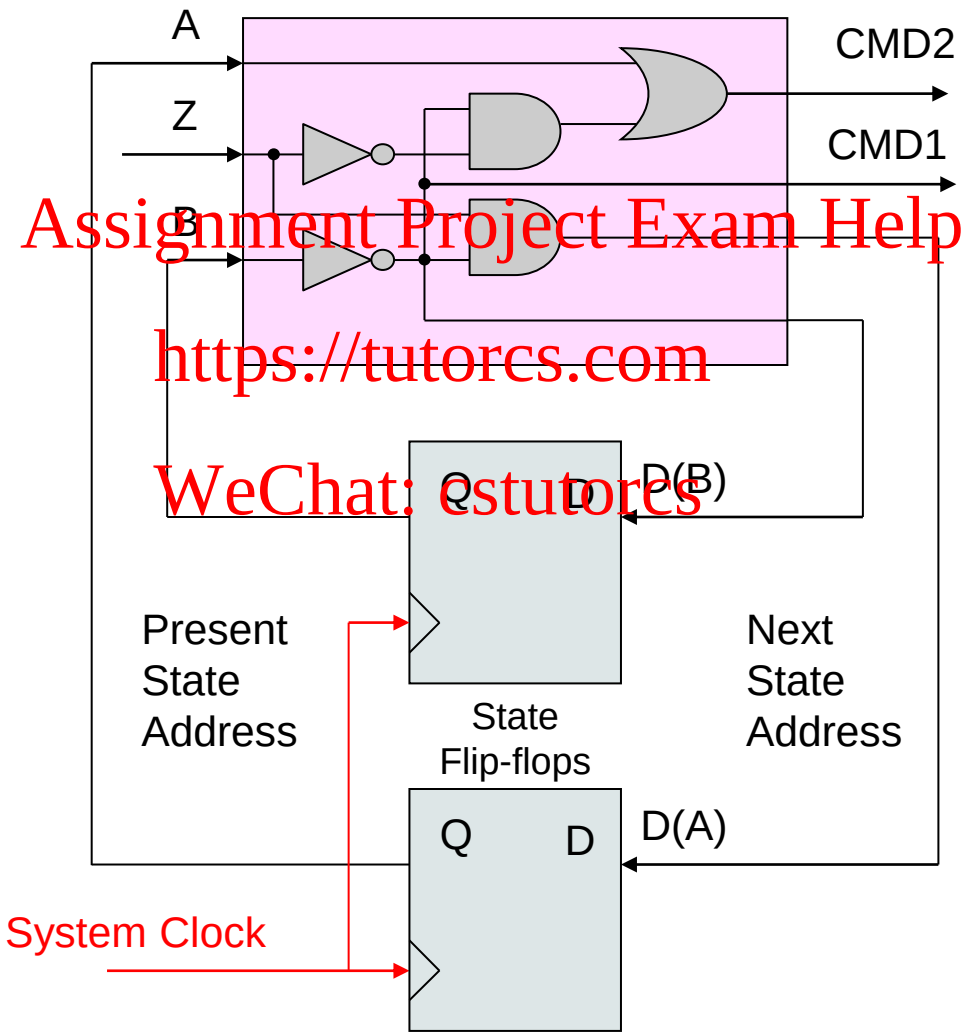
D(B) = B’

D(A) = B’Z

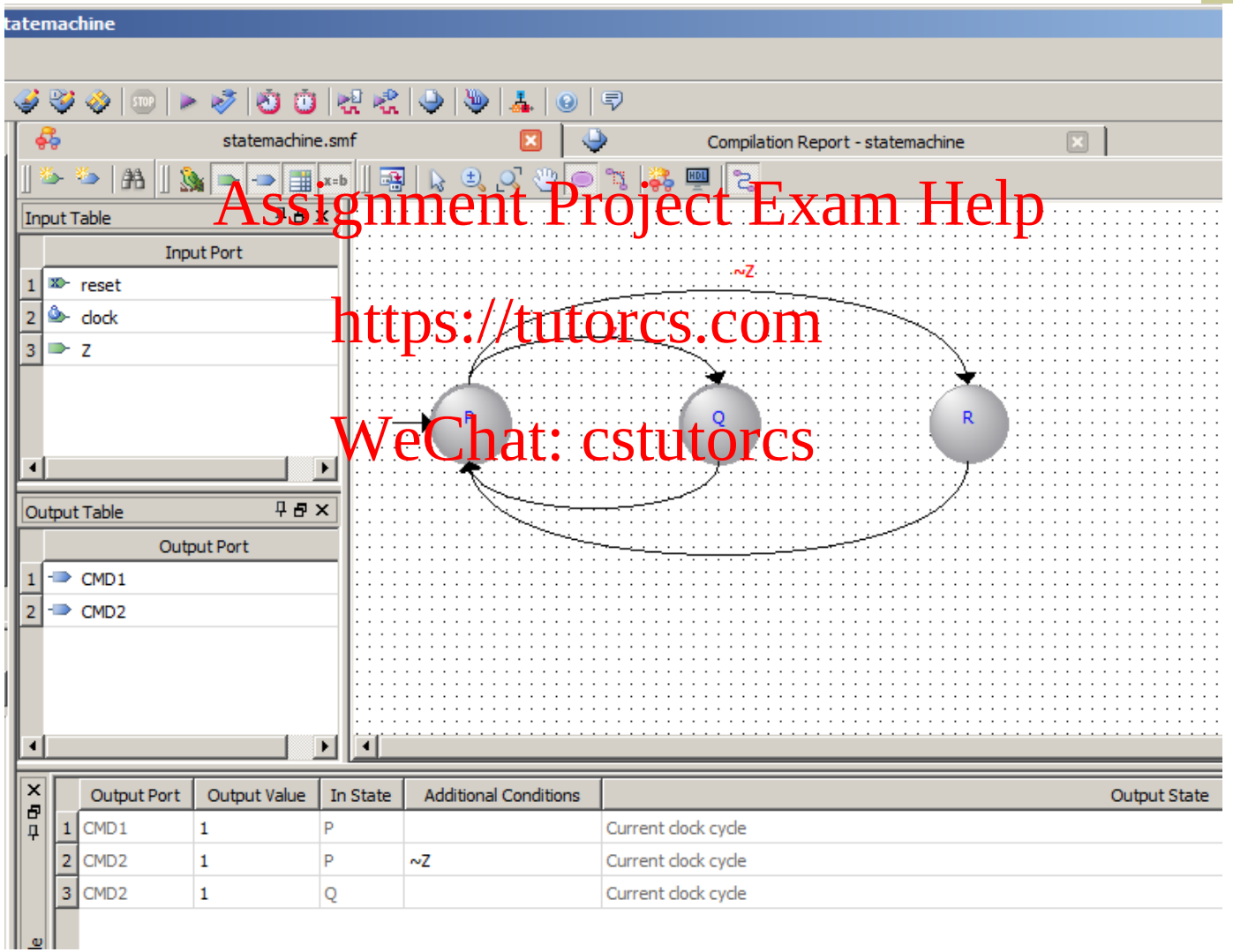
Check for unused states

- In the last example we didn't design for the unused state BA=01
- It is good practice to check what would happen if the system booted or "jumped" into the unused state.
- As D(A) = B'Z
  - If Z = 1 then D(A) = 1
  - If Z = 0 then D(A) = 0
- As D(B) = B'
  - D(B) = 1
- Therefore the next state(s) are
  - If Z = 1 BA=11 (State Q)
  - If Z = 0 BA=10 (State R)
- Hence even if the system enters the unused state (BA = 01) it will exit it.

ASM Implementation



Altera State Machine Implementation



Verilog Generated by State Machine Wizard

```
22  module statemachine (
23      reset,clock,Z,
24      CMD1,CMD2);
25
26      input reset;
27      input clock;
28      input Z;
29      tri0 reset;
30      tri0 Z;
31      output CMD1;
32      output CMD2;
33      reg CMD1;
34      reg CMD2;
35      reg [2:0] fstate;
36      reg [2:0] reg_fstate;
37      parameter P=0,Q=1,R=2;
38
39      always @(posedge clock)
40      begin
41          if (clock) begin
42              fstate <= reg_fstate;
43          end
44      end
45
```

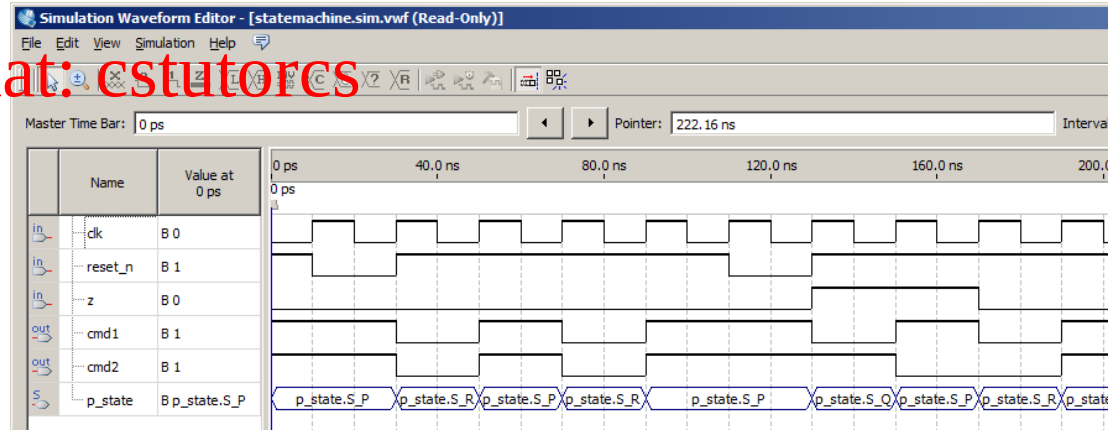


```
46      always @(fstate or reset or Z)
47      begin
48          if (reset) begin
49              reg_fstate <= P;
50              CMD1 <= 1'b0;
51              CMD2 <= 1'b0;
52          end
53          else begin
54              CMD1 <= 1'b0;
55              CMD2 <= 1'b0;
56              case (fstate)
57                  P: begin
58                      if (Z)
59                          reg_fstate <= Q;
60                      else if (~(Z))
61                          reg_fstate <= R;
62                      // Inserting 'else' block to prevent latch inference
63                      else
64                          reg_fstate <= P;
65
66                      if (~(Z))
67                          CMD2 <= 1'b1;
68                      // Inserting 'else' block to prevent latch inference
69                      else
70                          CMD2 <= 1'b0;
71
72                      CMD1 <= 1'b1;
73                  end
74                  Q: begin
75                      reg_fstate <= P;
76
77                      CMD2 <= 1'b1;
78                  end
79                  R: begin
80                      reg_fstate <= P;
81                  end
82                  default: begin
83                      CMD1 <= 1'bx;
84                      CMD2 <= 1'bx;
85                      $display ("Reach undefined state");
86                  end
87              endcase
88          end
89      end
90  endmodule // statemachine
91
```

Better Verilog

```
1  module statemachine( cmd1, cmd2, clk, reset_n, z);
2     input    clk;
3     input    reset_n;
4     input    z;
5     output   cmd1, cmd2;
6
7     reg [1:0] p_state, n_state;
8     reg      cmd1, cmd2;
9
10    parameter S_P=2'b00, S_Q=2'b01, S_R=2'b11;
11
12    always @ (posedge clk, negedge reset_n)
13       if (reset_n ==0) p_state <= S_P;
14          else p_state <= n_state;
15
16    always @ (p_state, z)
17       begin
18          cmd1 = 1'b0;
19          cmd2 = 1'b0;
20          n_state = p_state;
21          case (p_state)
22             S_P: begin
23                cmd1 = 1'b1;
24                if (z)
25                   n_state = S_Q;
26                else
27                   begin
28                      cmd2 = 1'b1;
29                      n_state = S_R;
30                   end
31                end
32             S_R: begin
33                   n_state = S_P;
34                end
35             S_Q: begin
36                   cmd2 = 1'b1;
37                   n_state = S_P;
38                end
39             endcase
40          end
41 endmodule
```

p_state is the output of the state flip-flops
n_state is what will be loaded into the state flip-flops on the next active clock edge



Note simulator only shows states when "Quartus II Simulator" selected under "Simulation/Options"