ECE/CS 3700: Fundamentals of Digital System Design

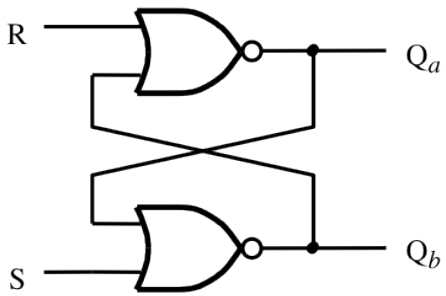
Chris J. Myers

Midterm Review 2

Chapters 5 and 6: Sequential Circuits

- *Combinational* - output depends only on the input.
- *Sequential* - output depends on input and past behavior.
  - Requires use of storage elements.
  - Contents of storage elements is called *state*.
  - Circuit goes through sequence of states as a result of changes in inputs.

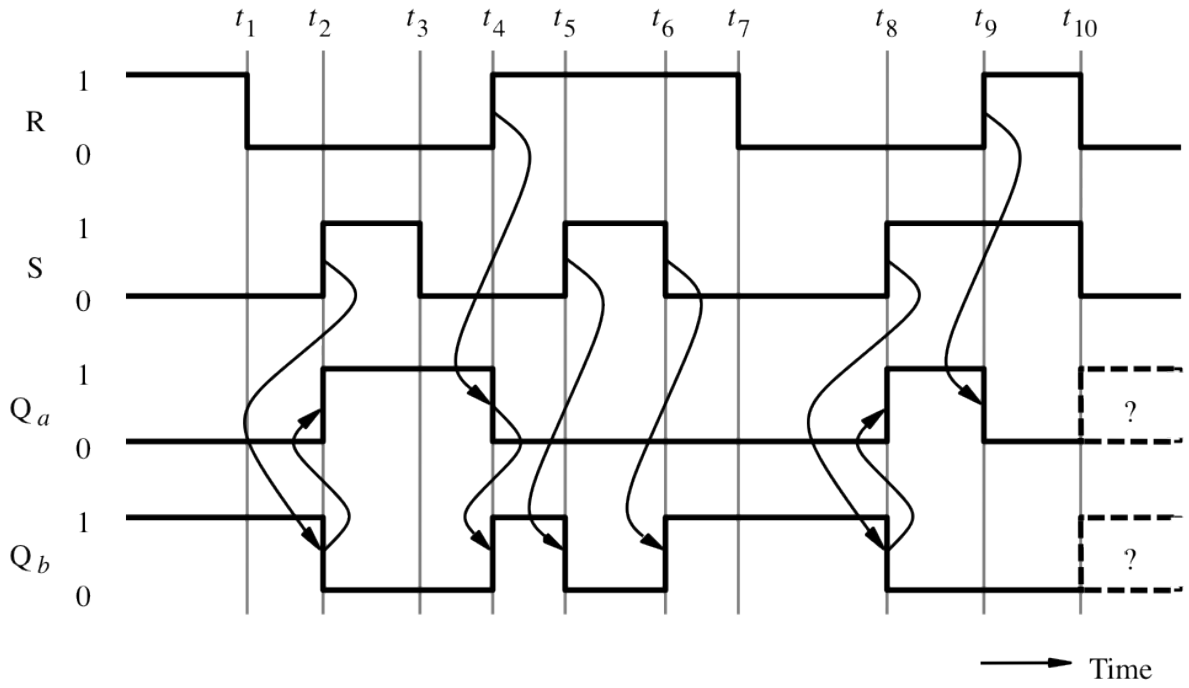
Chapter 5: Basic Latch



(a) Circuit

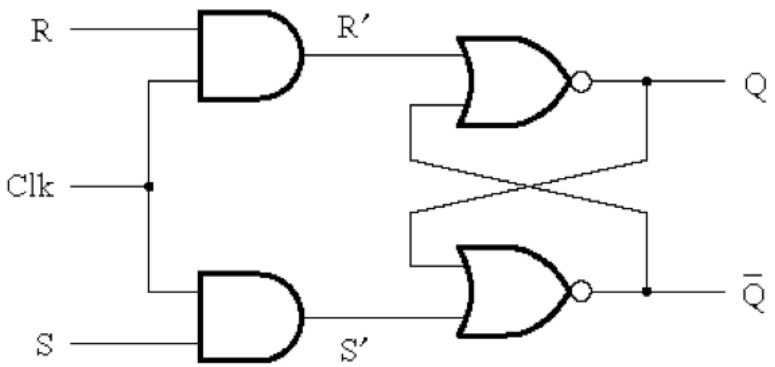
S	R	Q_a	Q_b	
0	0	0/1	1/0	(no change)
0	1	0	1	
1	0	1	0	
1	1	0	0	

(b) Truth table



(c) Timing diagram

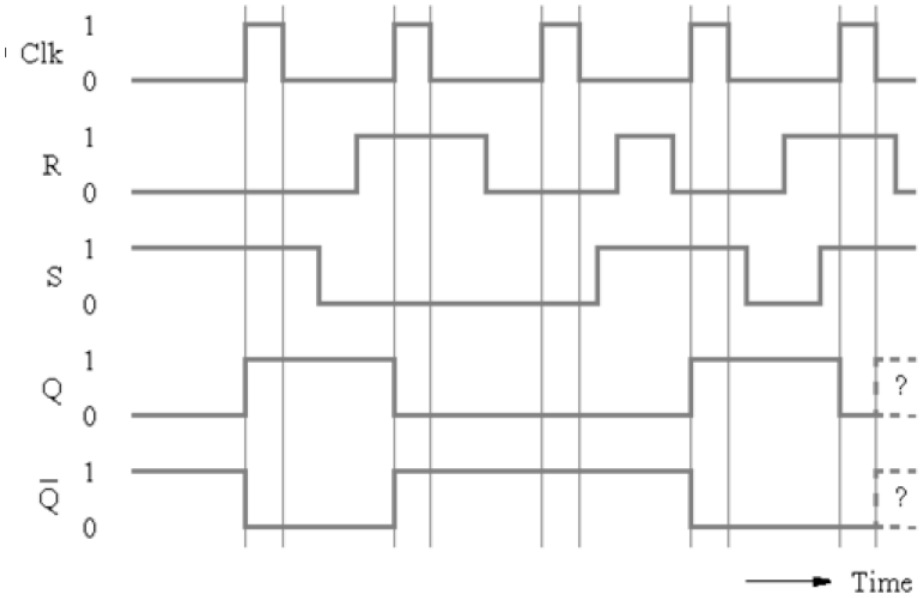
Chapter 5: Gated SR Latch



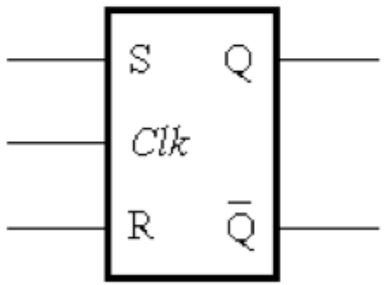
(a) Circuit

Clk	S	R	Q(t + 1)
0	x	x	Q(t) (no change)
1	0	0	Q(t) (no change)
1	0	1	0
1	1	0	1
1	1	1	x

(b) Characteristic table

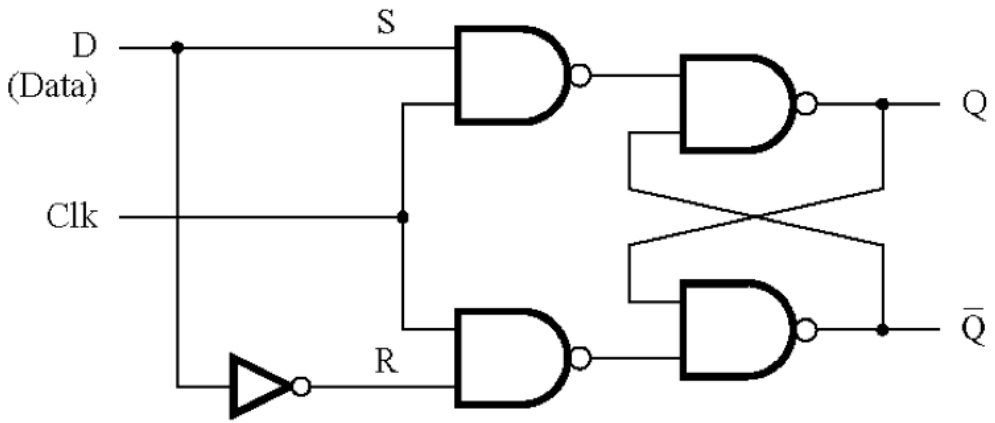


(c) Timing diagram



(d) Graphical symbol

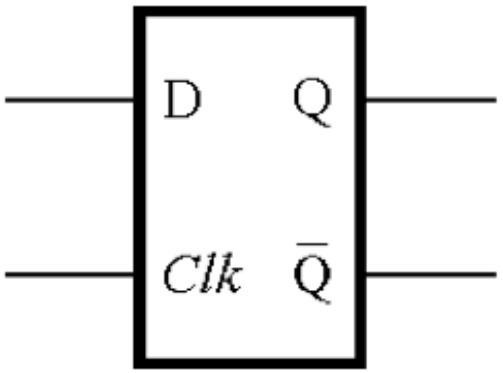
Chapter 5: Gated D Latch



(a) Circuit

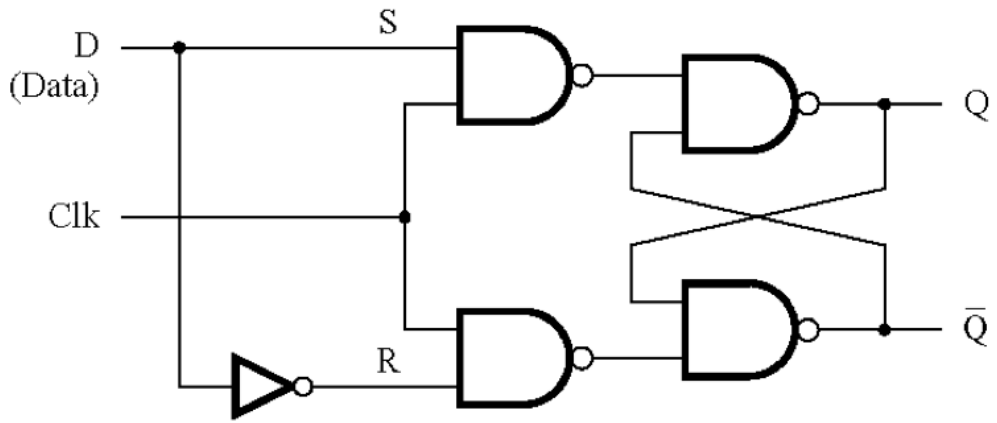
Clk	D	$Q(t+1)$
0	x	$Q(t)$
1	0	0
1	1	1

(b) Characteristic table

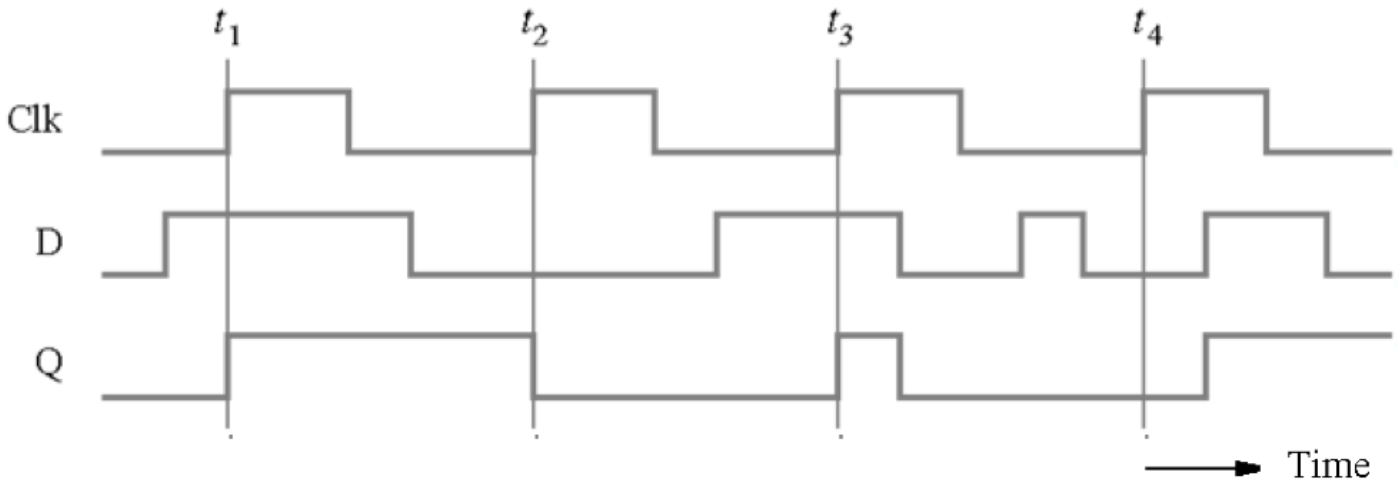


(c) Graphical symbol

Chapter 5: Gated D Latch

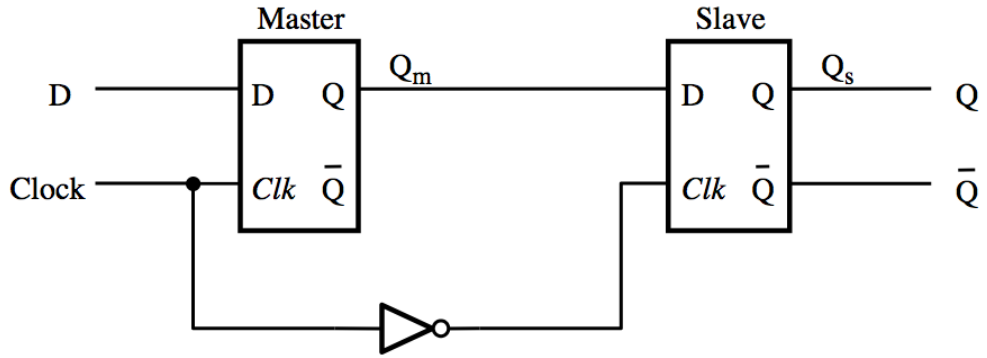


(a) Circuit

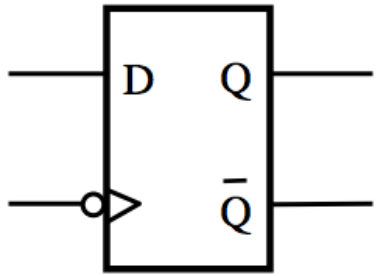


(d) Timing diagram

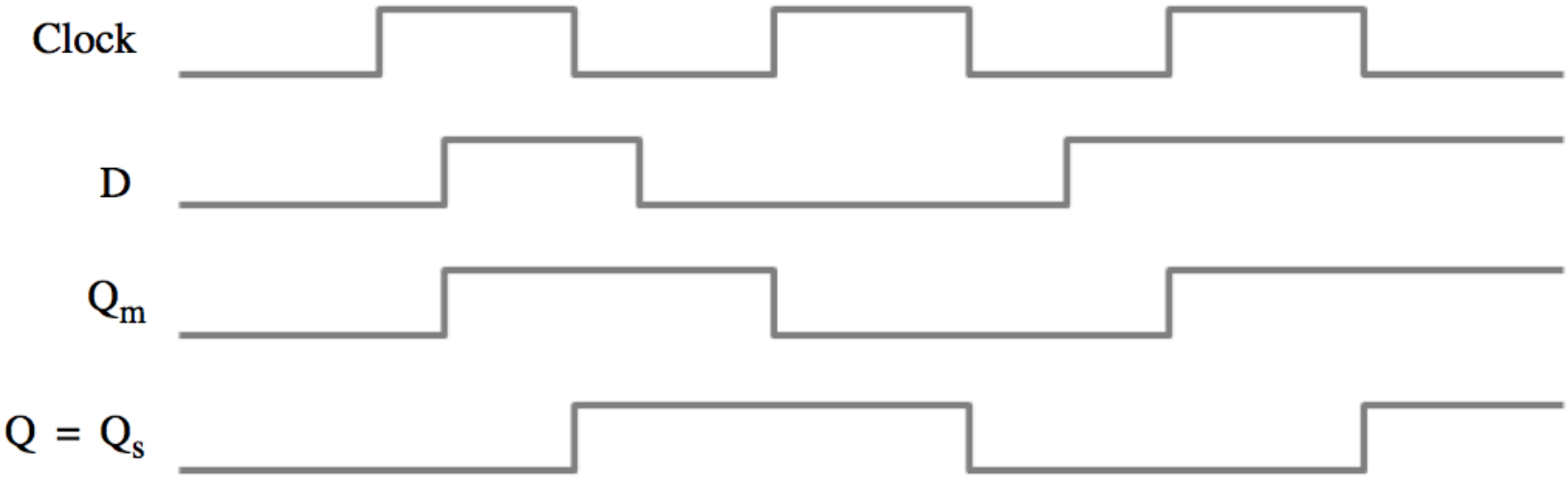
Chapter 5: Master-slave D Flip-flop



(a) Circuit

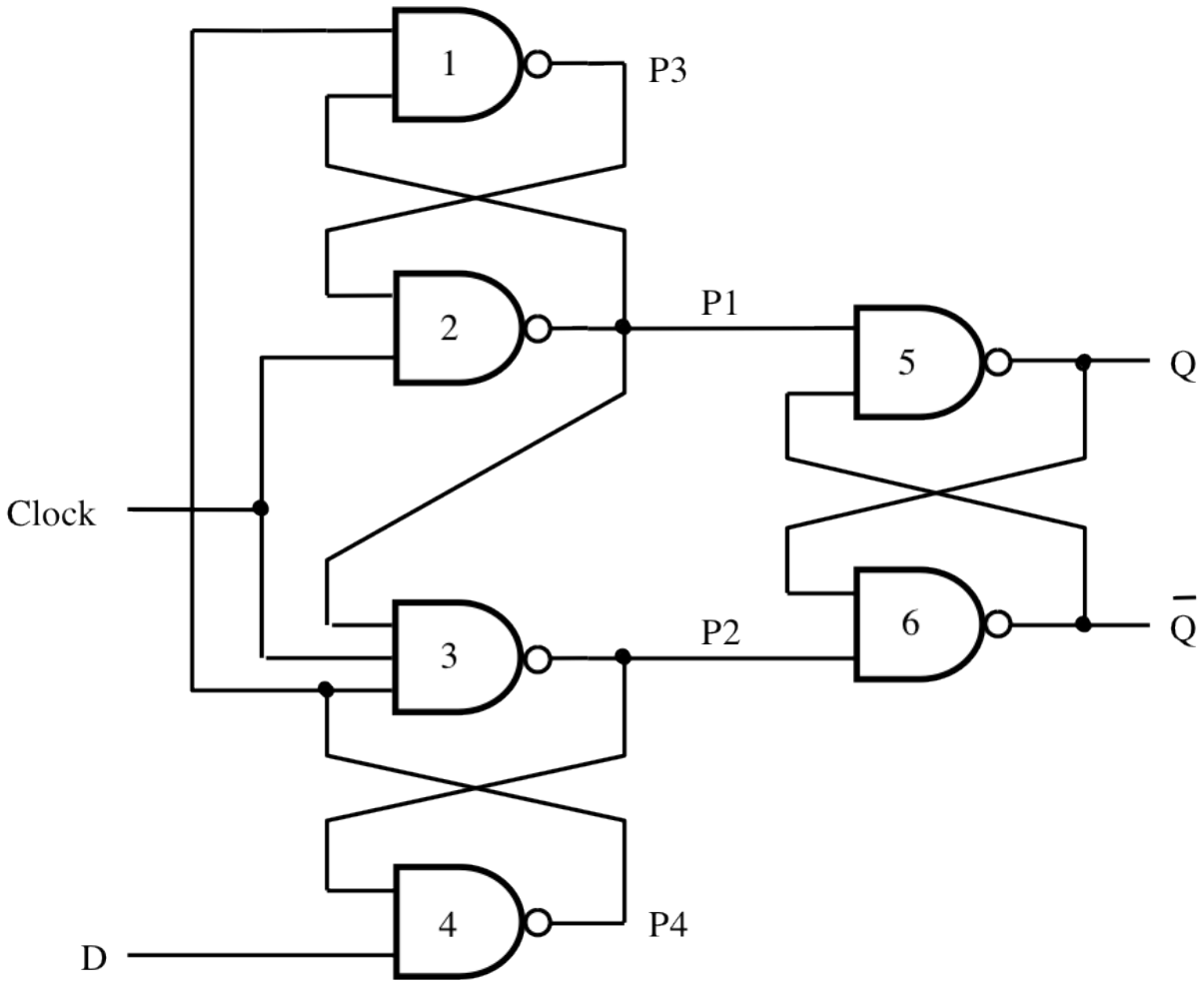


(c) Graphical symbol

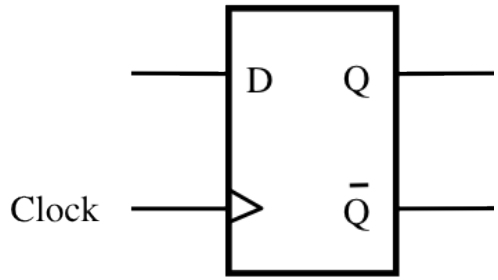


(b) Timing diagram

Chapter 5: Positive-edge-triggered D Flip-flop

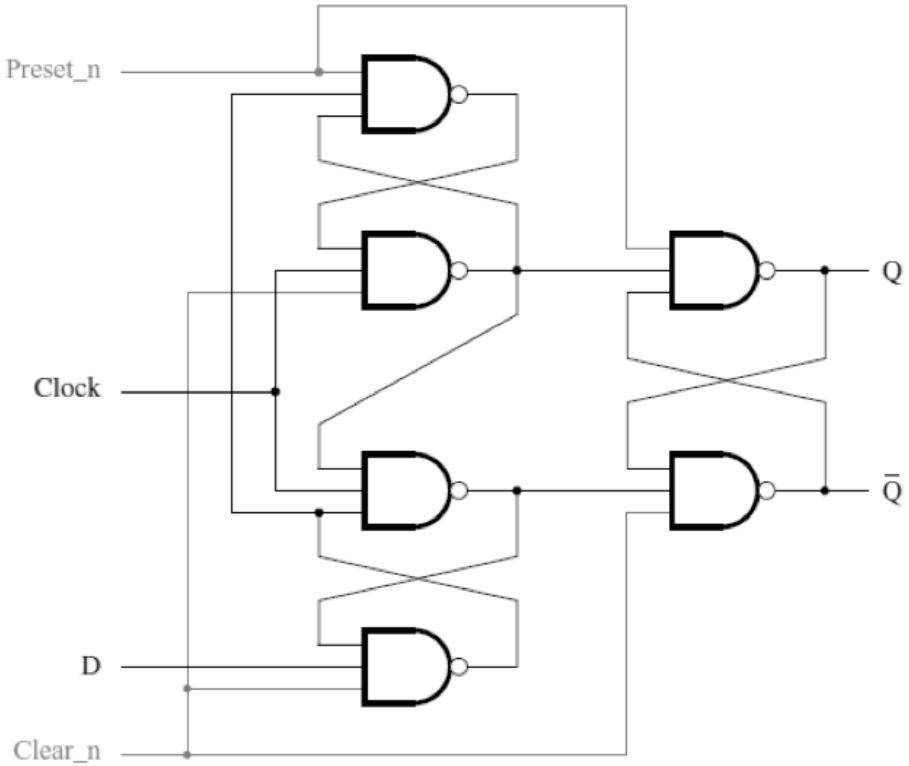


(a) Circuit

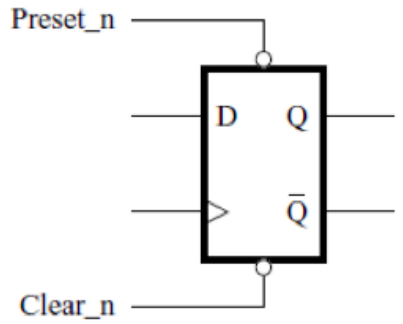


(b) Graphical symbol

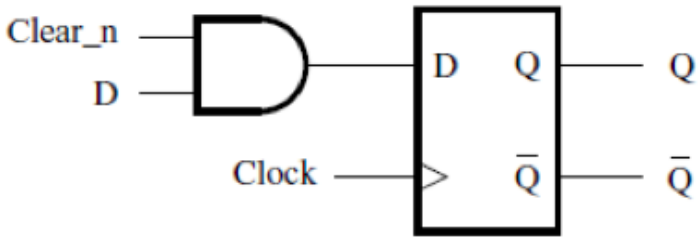
Chapter 5: Flip-flop with Clear and Preset



(a) Circuit

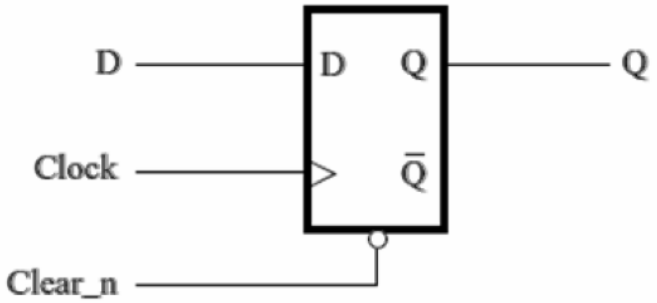


(b) Graphical symbol

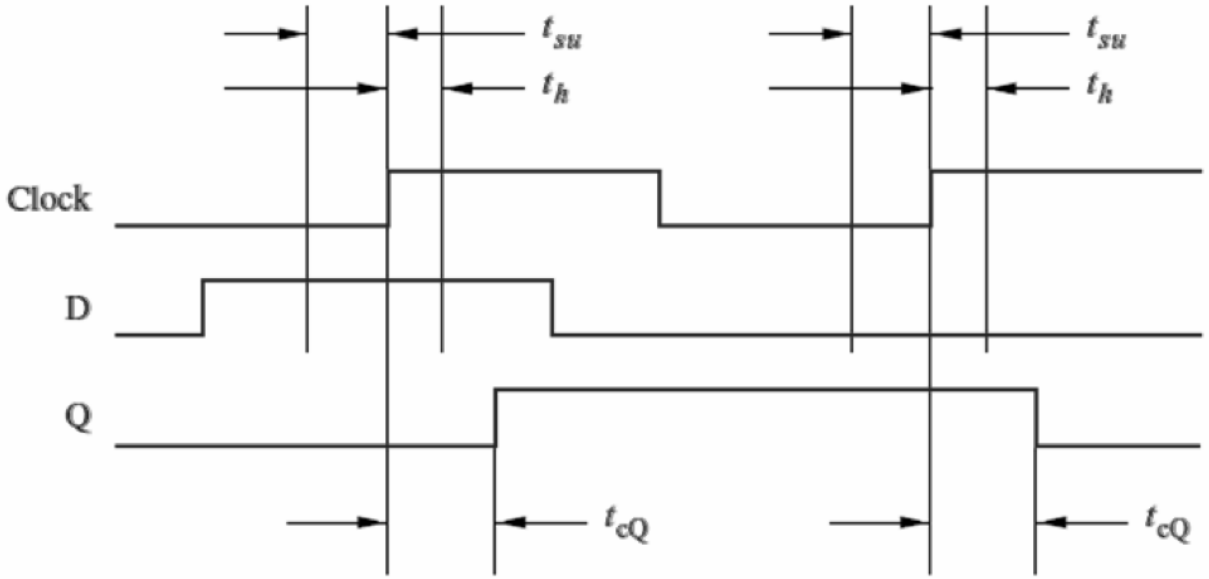


(c) Adding a synchronous clear

Chapter 5: Flip-flop Timing Parameters

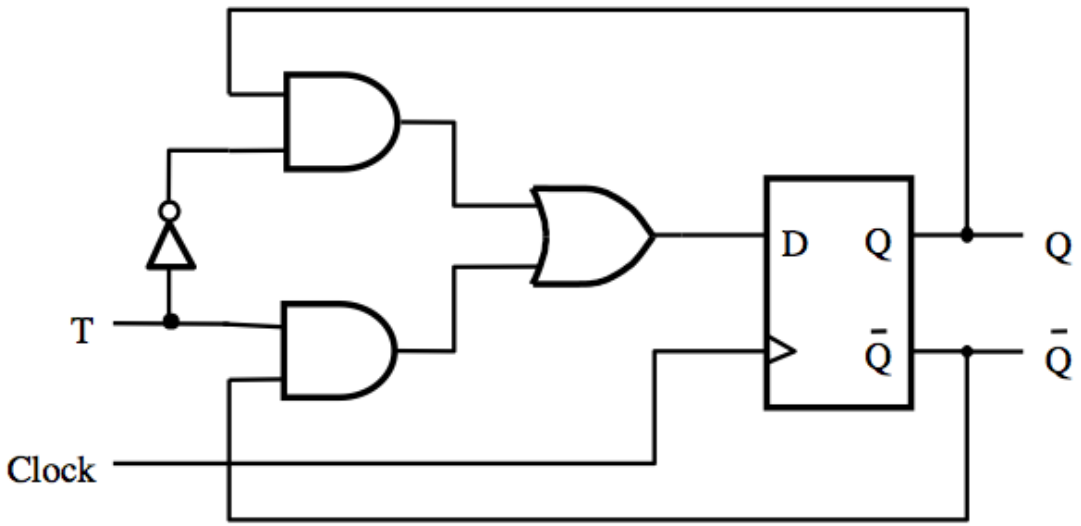


(a) D flip-flop with asynchronous clear



(b) Timing diagram

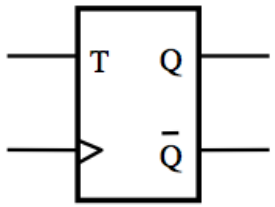
Chapter 5: T Flip-flop



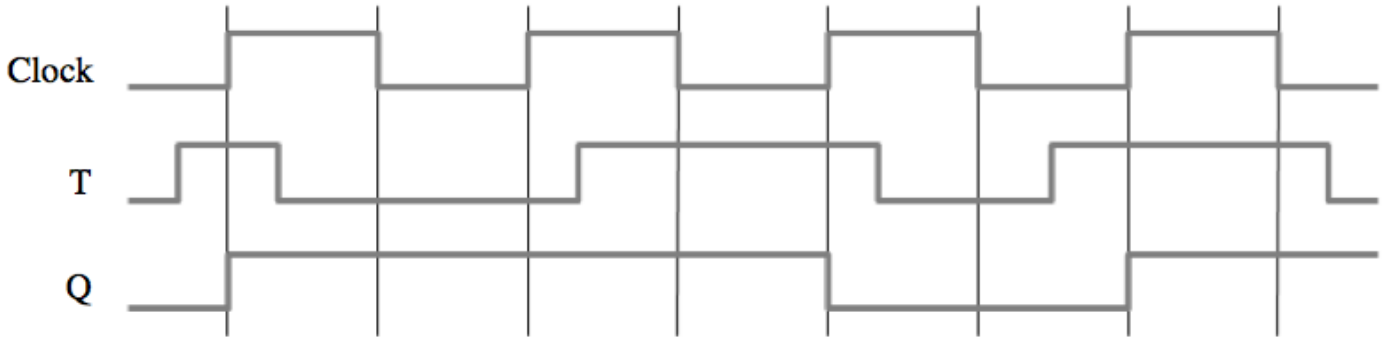
(a) Circuit

T	Q(t + 1)
0	Q(t)
1	$\bar{Q}(t)$

(b) Truth table

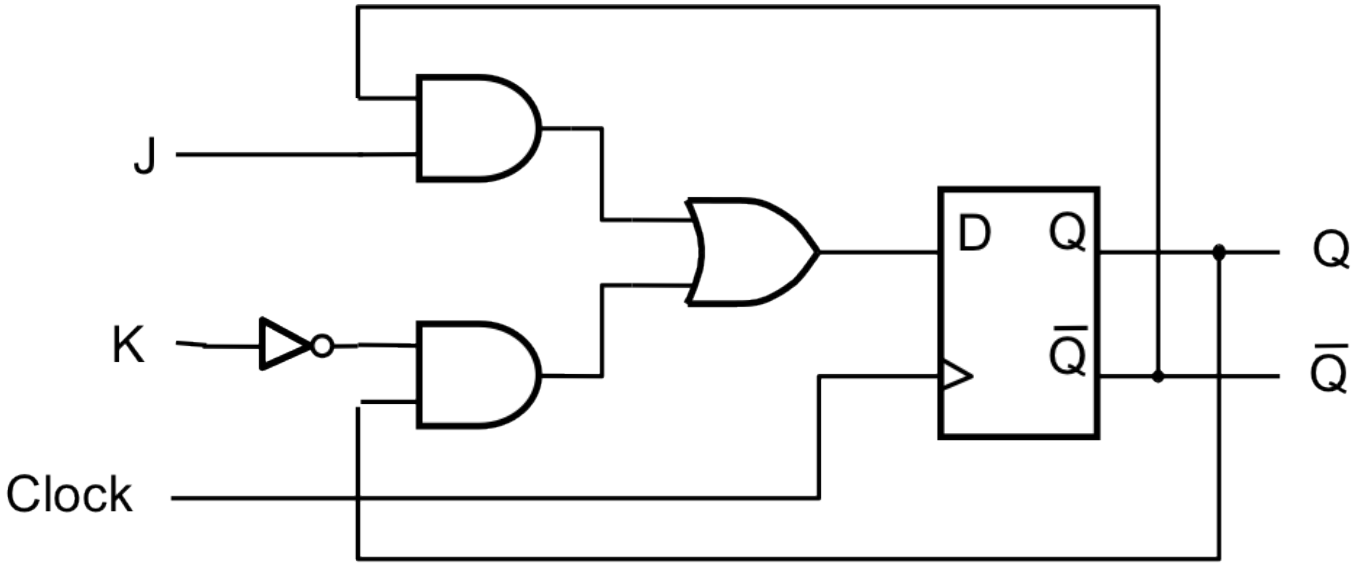


(c) Graphical symbol



(d) Timing diagram

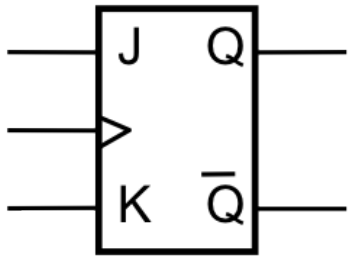
Chapter 5: JK Flip-flop



(a) Circuit

J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	$\overline{Q}$(t)

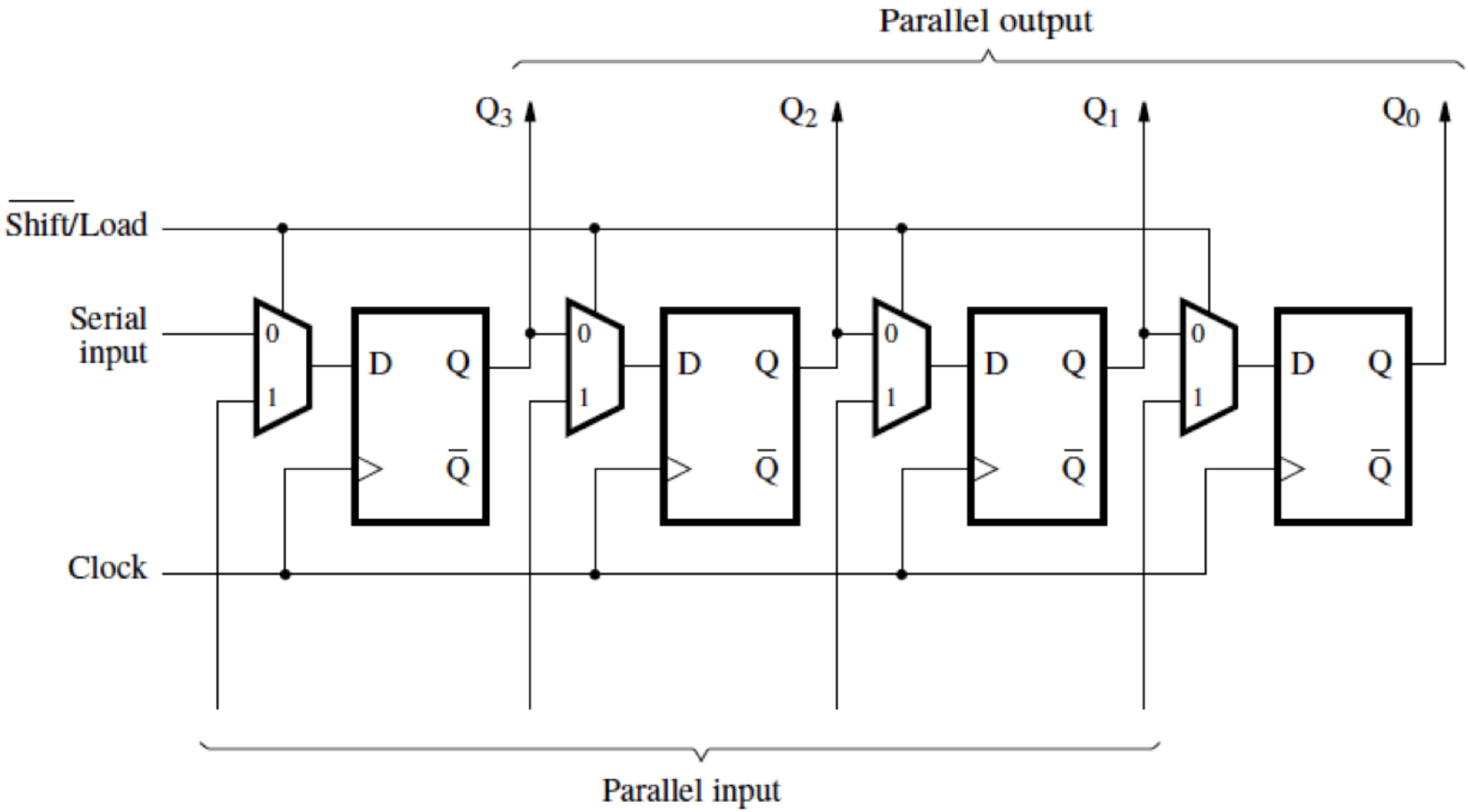
(b) Truth table

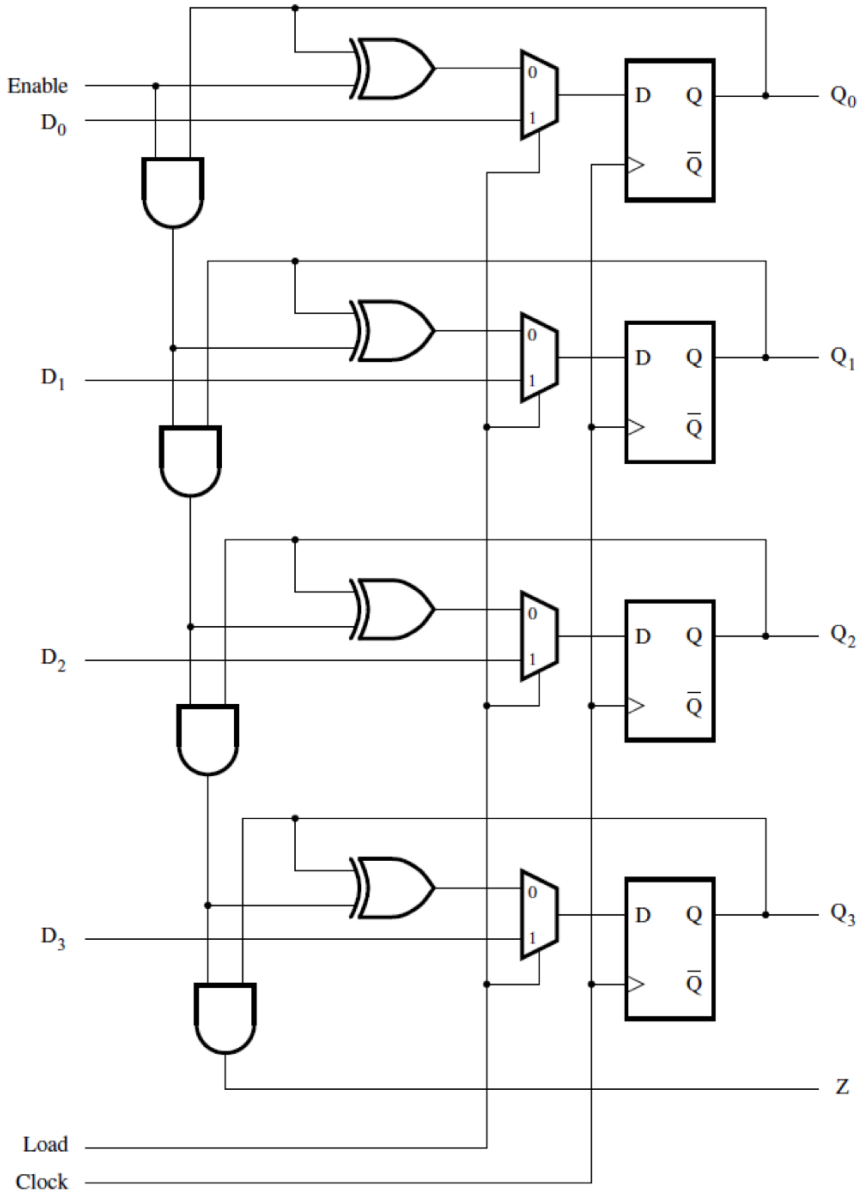


(c) Graphical symbol

Chapter 5: Summary of Terminology

- Basic latch - cross-coupled NAND/NOR
- Gated latch - output changes when clk = 1.
  - Gated SR latch
  - Gated D latch
- Flip-flop - output changes only on clk edge.
  - Edge-triggered flip-flop
  - Master-slave flip-flop


Parallel output
Q3
Q2
Q1
Q0
Shift/Load
Serial input
Clock
0
1
D
Q
Q̄
Parallel input


Enable
D0
D1
D2
D3
Load
Clock
0
1
D
Q
Q̄
Q0
Q1
Q2
Q3
Z

Chapter 5: Verilog for a Gated D Latch

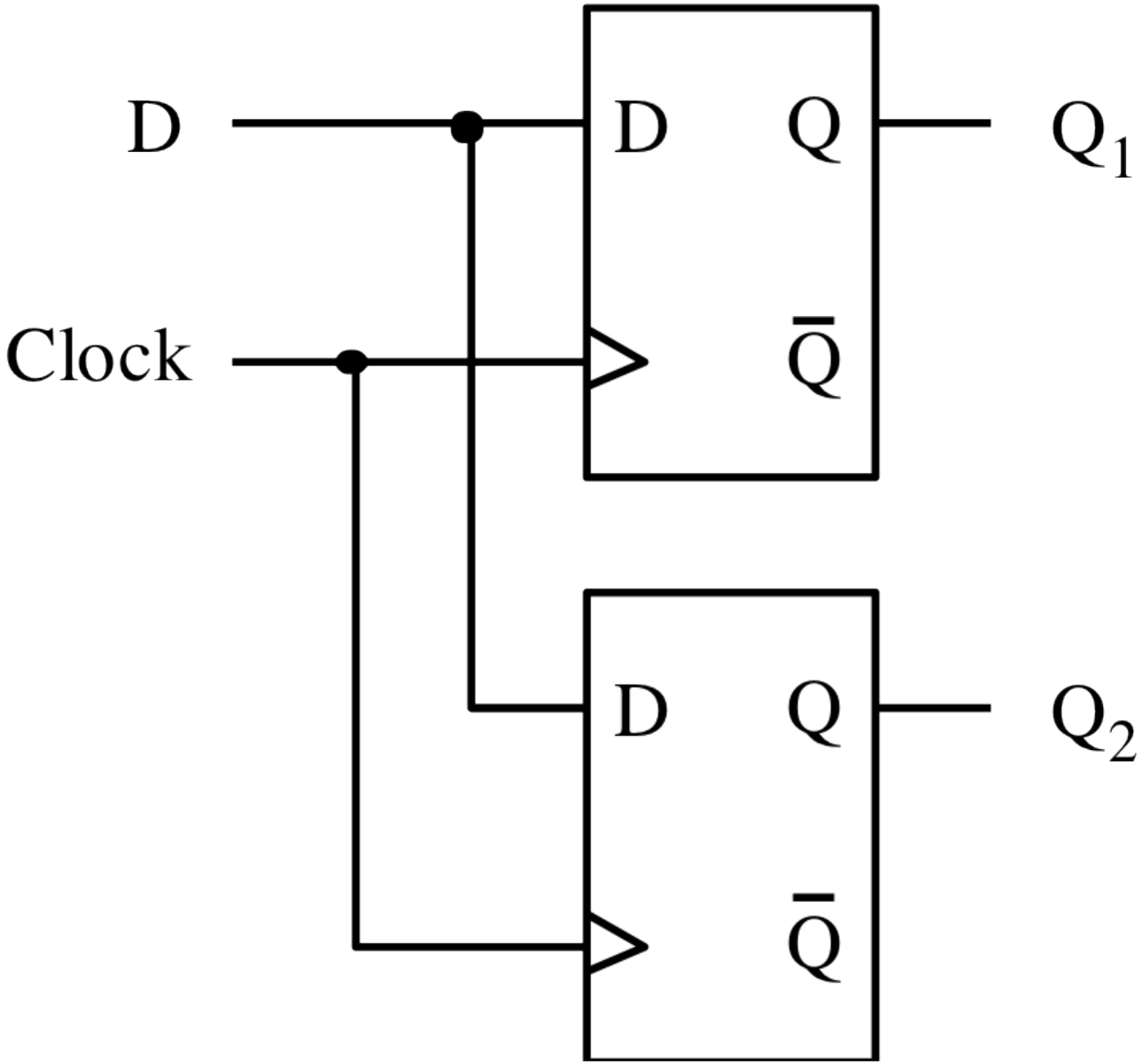
```
module D_latch (D, Clk, Q);
    input D, Clk;
    output reg Q;

    always @(D, Clk)
        if (Clk)
            Q = D;

endmodule
```

Chapter 5: Verilog for a D Flip-flop

```
module flipflop (D, Clock, Q);
    input D, Clock;
    output reg Q;

    always @(posedge Clock)
        Q = D;

endmodule
```

Chapter 5: Blocking vs. Non-blocking Assignments

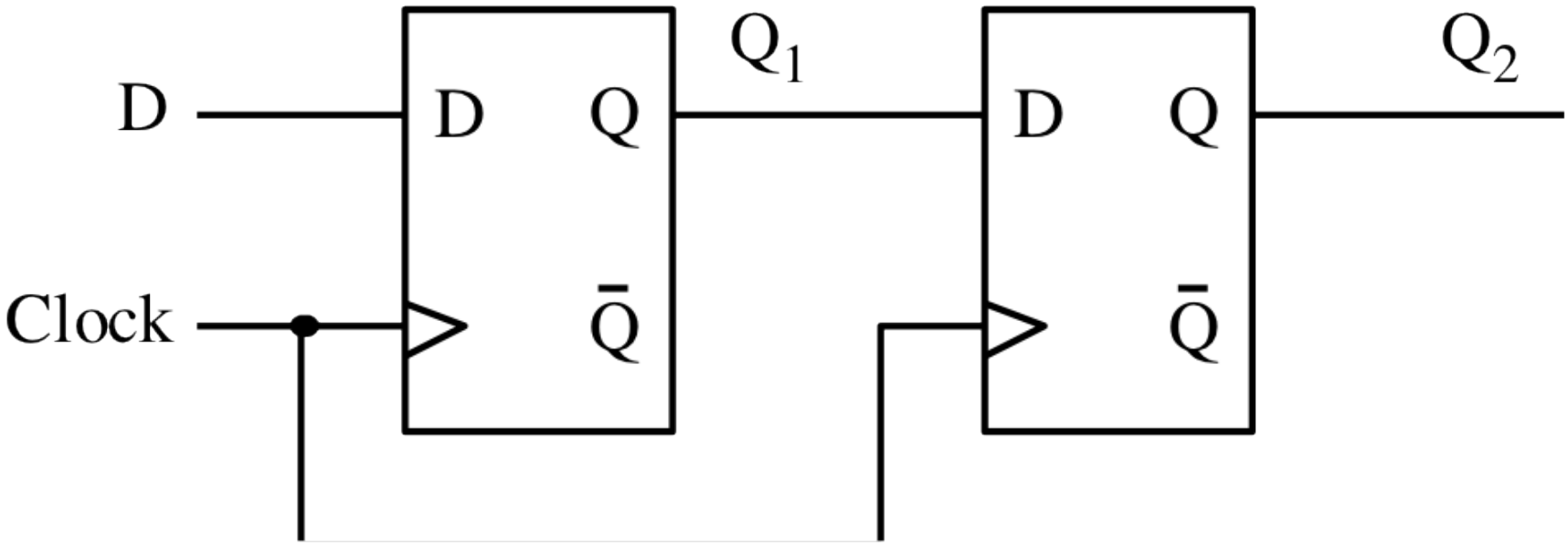
```
module example5_3 (D, Clock, Q1, Q2);
    input D, Clock;
    output reg Q1, Q2;

    always @(posedge Clock)
    begin
        Q1 = D;
        Q2 = Q1;
    end

endmodule
```


D
Clock
D
Q
$\bar{Q}$
Q_1
D
Q
$\bar{Q}$
Q_2

Chapter 5: Blocking vs. Non-blocking Assignments

```
module example5_4 (D, Clock, Q1, Q2);
    input D, Clock;
    output reg Q1, Q2;

    always @(posedge Clock)
    begin
        Q1 <= D;
        Q2 <= Q1;
    end

endmodule
```

Chapter 5: Blocking vs. Non-blocking Assignments



Chapter 5: Verilog for an *n*-bit Shift Register

```
module shiftn (R, L, w, Clock, Q);
    parameter n = 16;
    input [n-1:0] R;
    input L, w, Clock;
    output reg [n-1:0] Q;
    integer k;

    always @(posedge Clock)
        if (L)
          Q <= R;
        else
        begin
          for (k = 0; k < n-1; k = k+1)
              Q[k] <= Q[k+1];
          Q[n-1] <= w;
        end

endmodule
```

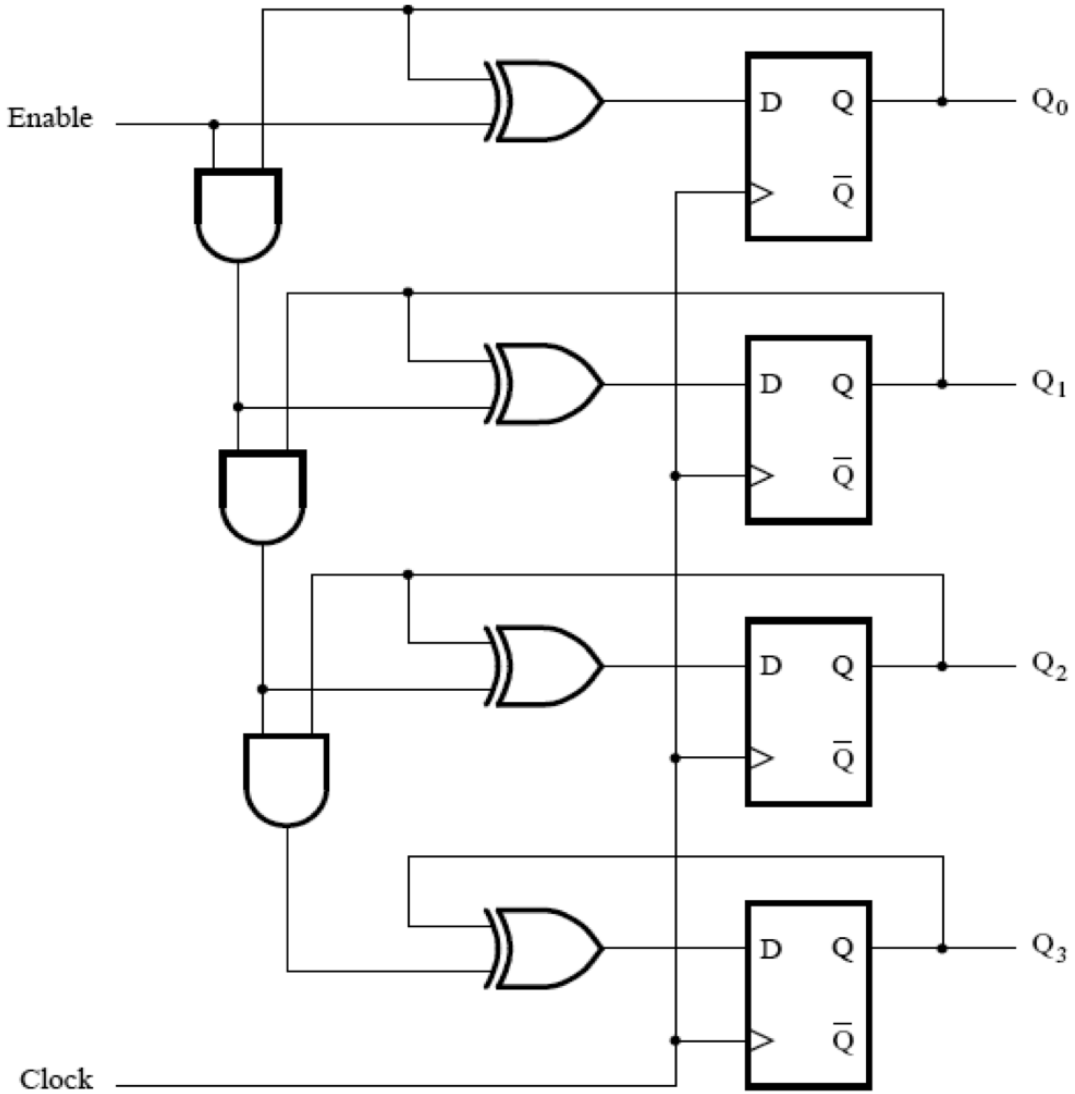
Chapter 5: Verilog for an Up/Down Counter

```
module updowncount (R, Clock, L, E, up_down, Q);
    parameter n = 8;
    input [n-1:0] R;
    input Clock, L, E, up_down;
    output reg [n-1:0] Q;
    always @(posedge Clock)
        if (L)
            Q <= R;
        else if (E)
            Q <= Q + (up_down ? 1 : -1);

endmodule
```

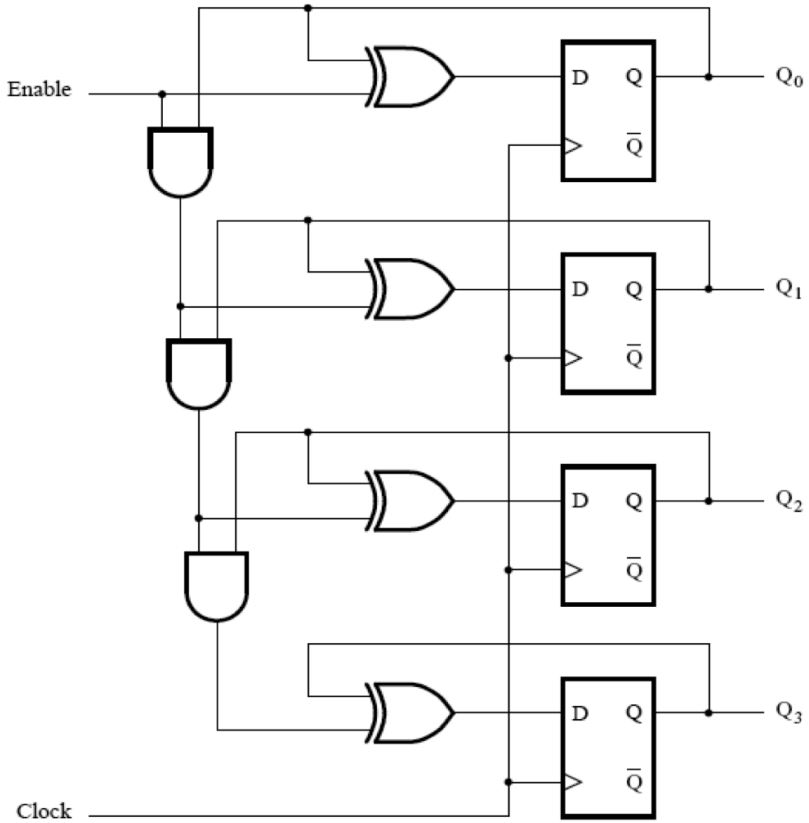
Chapter 5: Register Transfer Level (RTL) Code

- Can write behavioral code like a computer program.
- This *high-level* behavioral code can make it difficult to predict the circuit that will be synthesized.
- Better to create small design modules from which larger designs are created by connecting them together.
- This approach is known as *register-transfer level* (RTL) style.

Chapter 5: Timing Analysis for a 4-bit Counter

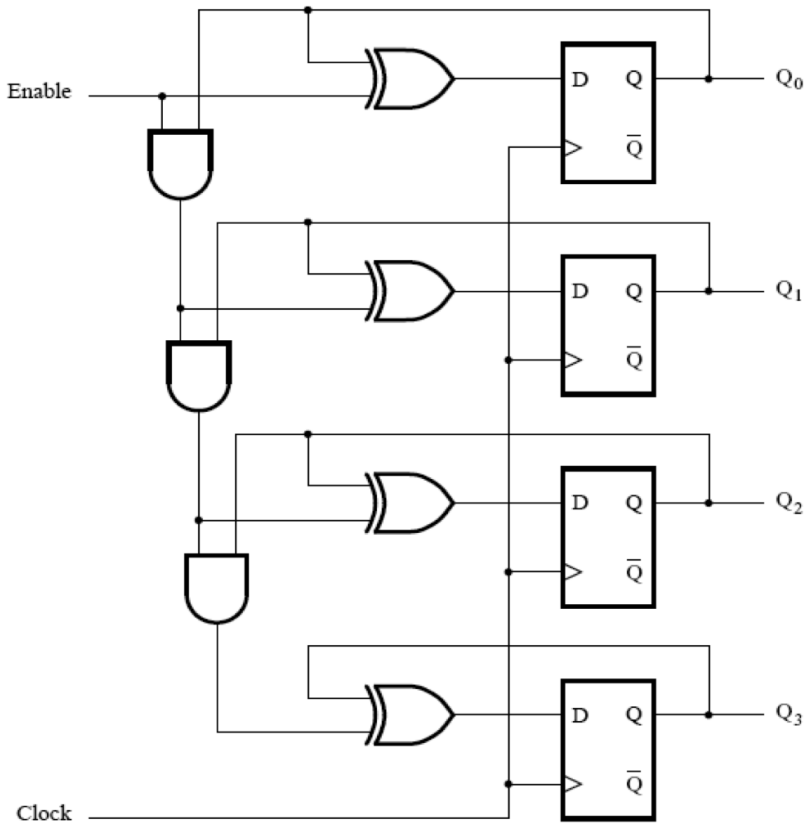


Chapter 5: Timing Analysis for a 4-bit Counter



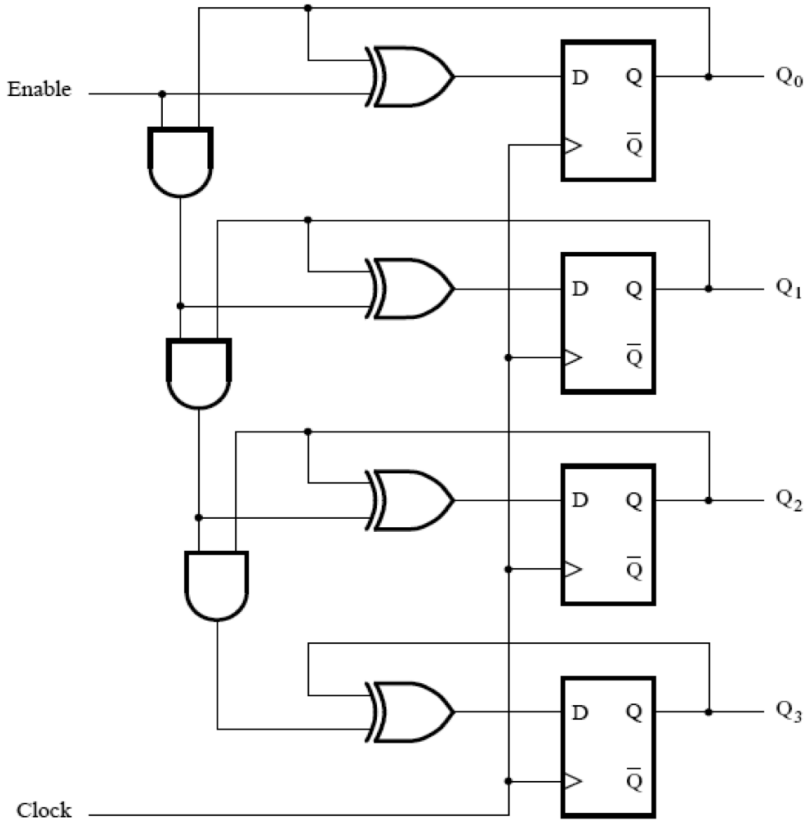
$$T_{min} = t_{cQ} + 3(t_{AND}) + t_{XOR} + t_{su}$$

Chapter 5: Timing Analysis for a 4-bit Counter



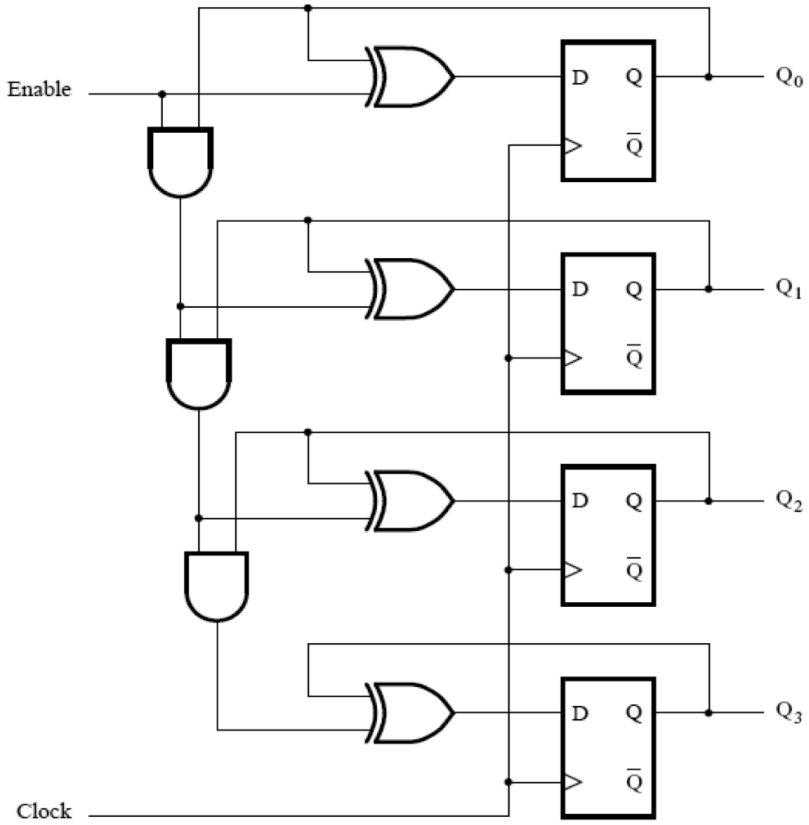
$$T_{min} = t_{cQ} + 3(t_{AND}) + t_{XOR} + t_{su}$$
$$T_{min} = 1.0 + 3(1.2) + 1.2 + 0.6 \text{ ns} = 6.4 \text{ ns}$$

Chapter 5: Timing Analysis for a 4-bit Counter

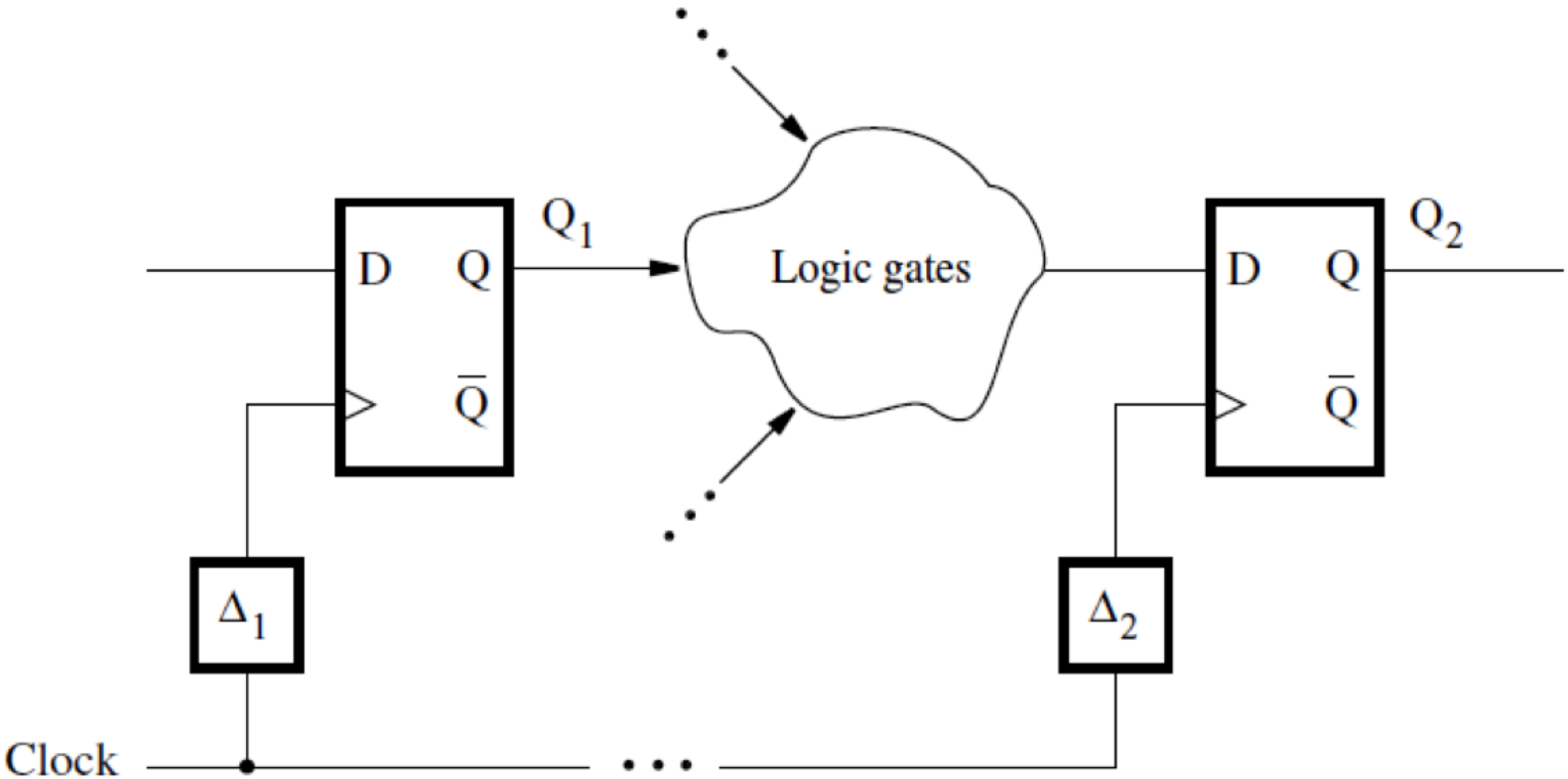


$$T_{min} = t_{cQ} + 3(t_{AND}) + t_{XOR} + t_{su}$$
$$T_{min} = 1.0 + 3(1.2) + 1.2 + 0.6 \text{ ns} = 6.4 \text{ ns}$$
$$F_{max} = 1/6.4 \text{ ns} = 156.25 \text{ MHz}$$

Chapter 5: Timing Analysis for a 4-bit Counter



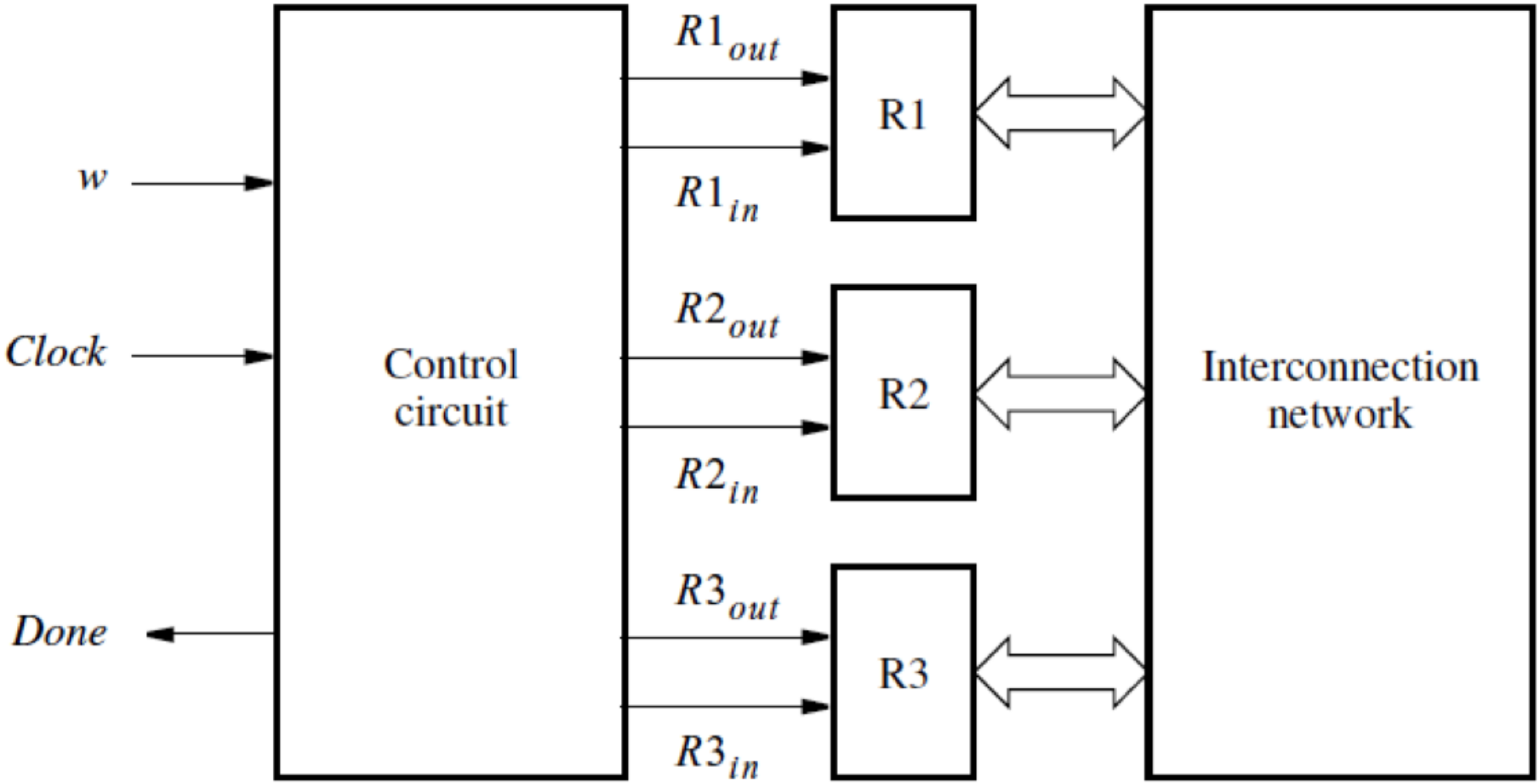
$$T_{min} = t_{cQ} + 3(t_{AND}) + t_{XOR} + t_{su}$$
$$T_{min} = 1.0 + 3(1.2) + 1.2 + 0.6 \text{ ns} = 6.4 \text{ ns}$$
$$F_{max} = 1/6.4 \text{ ns} = 156.25 \text{ MHz}$$
$$t_{cQ} + t_{XOR} = 0.8 + 1.2 = 2.0 \text{ ns} > t_h = 0.4 \text{ ns}$$

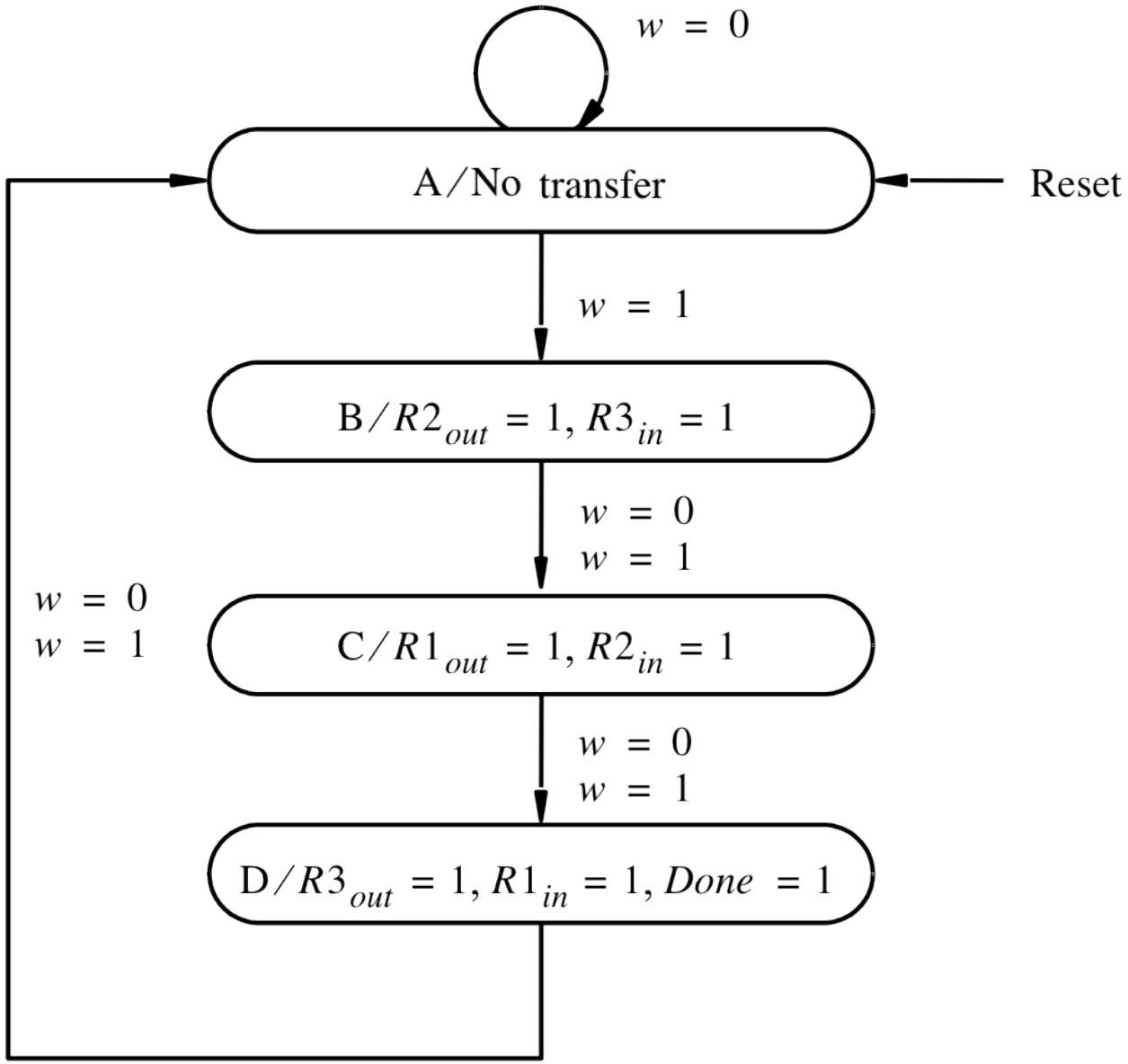


$$
\begin{aligned}
t_{skew} &= \Delta_2 - \Delta_1 \\
T_{min} &= t_{cQ} + t_L + t_{su} - t_{skew} \\
t_{cQ} + t_l &> t_h + t_{skew}
\end{aligned}
$$

Chapter 6: Basic Design Steps

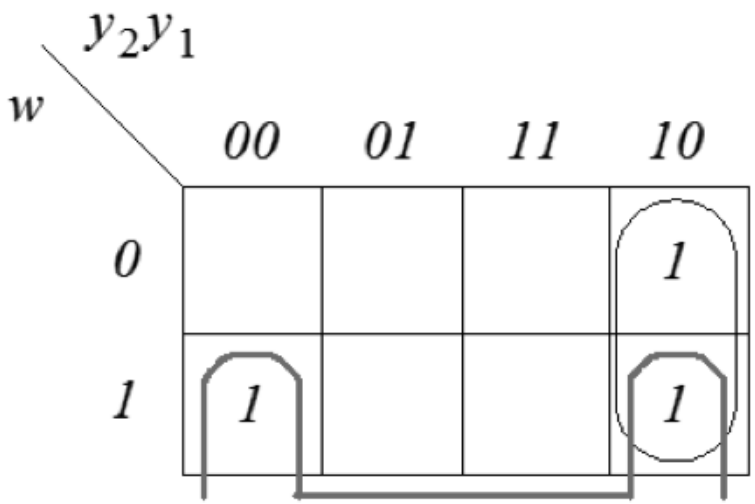
1. Obtain the specification of the desired circuit.
2. Derive a state diagram.
3. Derive the corresponding state table.
4. Reduce the number of states if possible.
5. Decide on the number of state variables.
6. Choose the type of flip-flops to be used.
7. Derive the logic expressions needed to implement the circuit.


w
Clock
Done
Control circuit
$R1_{out}$
$R1_{in}$
$R2_{out}$
$R2_{in}$
$R3_{out}$
$R3_{in}$
R1
R2
R3
Interconnection network

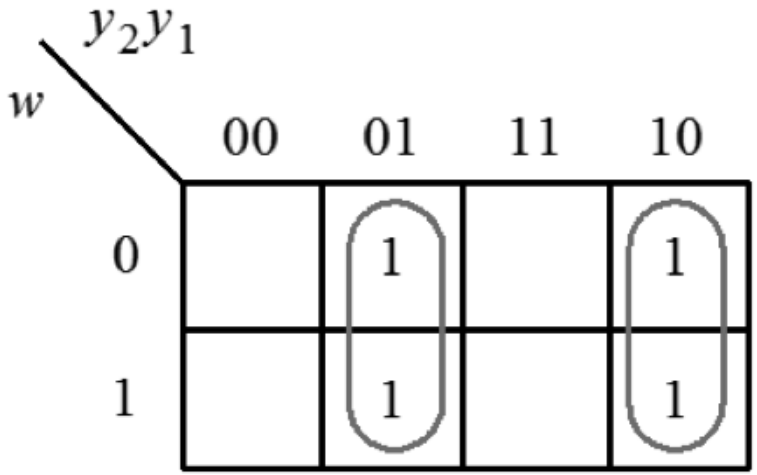

w = 0
A/No transfer
Reset
w = 1
B/R2out = 1, R3in = 1
w = 0
w = 1
w = 0
w = 1
C/R1out = 1, R2in = 1
w = 0
w = 1
D/R3out = 1, R1in = 1, Done = 1

Present state	Next state		Outputs						
	$w = 0$	$w = 1$	$R1_{out}$	$R1_{in}$	$R2_{out}$	$R2_{in}$	$R3_{out}$	$R3_{in}$	$Done$
A	A	B	0	0	0	0	0	0	0
B	C	C	0	0	1	0	0	1	0
C	D	D	1	0	0	1	0	0	0
D	A	A	0	1	0	0	1	0	1

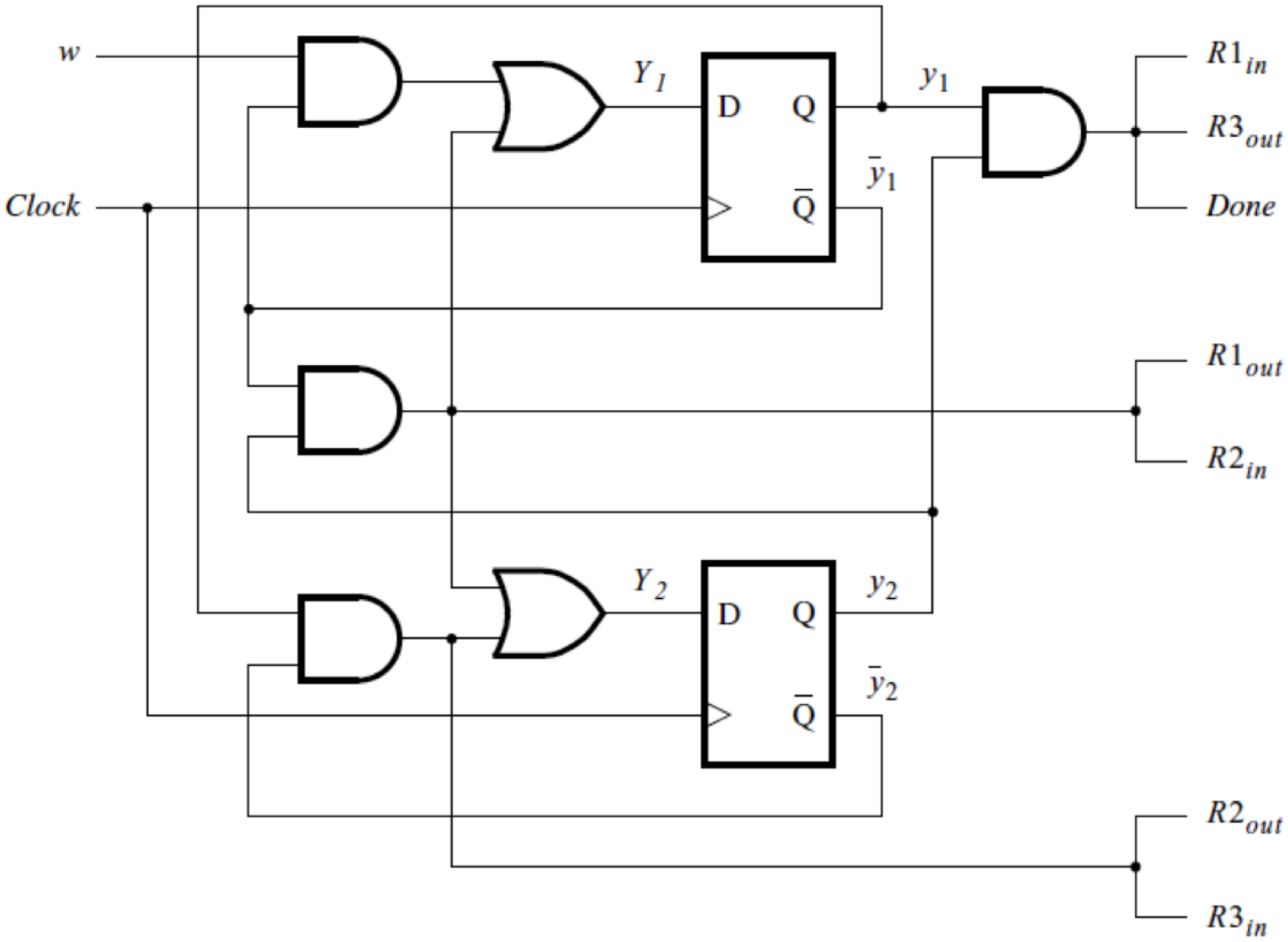
	Present state	Next state		Outputs						
		$w = 0$	$w = 1$							
	y_2y_1	Y_2Y_1	Y_2Y_1	$R1_{out}$	$R1_{in}$	$R2_{out}$	$R2_{in}$	$R3_{out}$	$R3_{in}$	$Done$
A	00	00	0 1	0	0	0	0	0	0	0
B	01	10	1 0	0	0	1	0	0	1	0
C	10	11	1 1	1	0	0	1	0	0	0
D	11	00	0 0	0	1	0	0	1	0	1



$$Y_1 = w\bar{y}_1 + \bar{y}_1 y_2$$



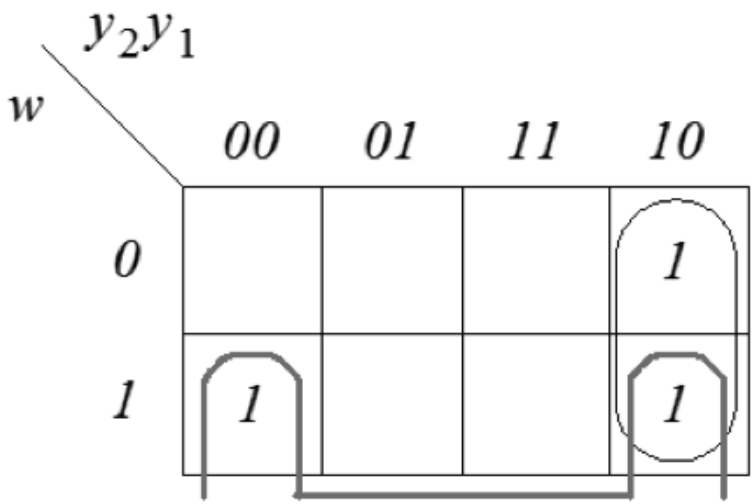
$$Y_2 = y_1\bar{y}_2 + \bar{y}_1 y_2$$


w
Clock
Y_1
D
Q
$\bar{Q}$
y_1
$\bar{y}_1$
$R1_{in}$
$R3_{out}$
Done
$R1_{out}$
$R2_{in}$
Y_2
D
Q
$\bar{Q}$
y_2
$\bar{y}_2$
$R2_{out}$
$R3_{in}$

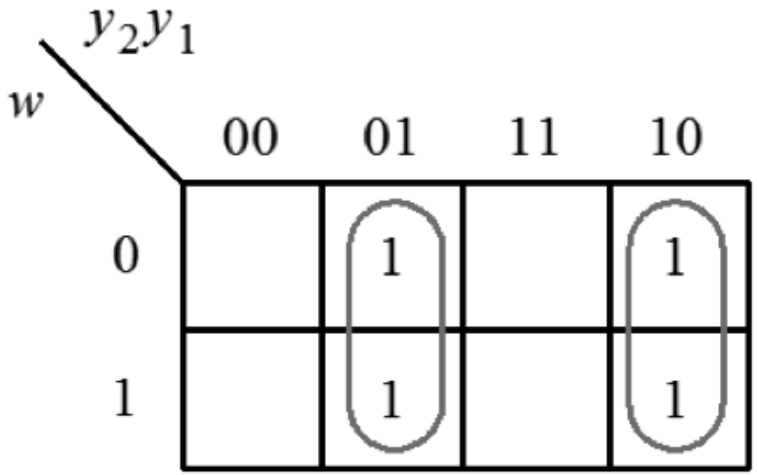
Chapter 6: State-assignment Problem

	Present state	Next state		Outputs						
		$w = 0$	$w = 1$							
	y_2y_1	Y_2Y_1	Y_2Y_1	$R1_{out}$	$R1_{in}$	$R2_{out}$	$R2_{in}$	$R3_{out}$	$R3_{in}$	$Done$
A	00	00	0 1	0	0	0	0	0	0	0
B	01	10	1 0	0	0	1	0	0	1	0
C	10	11	1 1	1	0	0	1	0	0	0
D	11	00	0 0	0	1	0	0	1	0	1

Chapter 6: State-assignment Problem



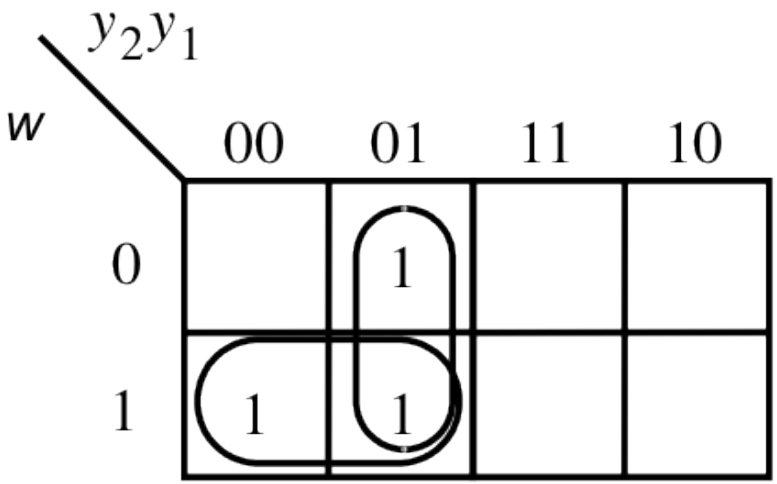
$$Y_1 = w\bar{y}_1 + \bar{y}_1 y_2$$



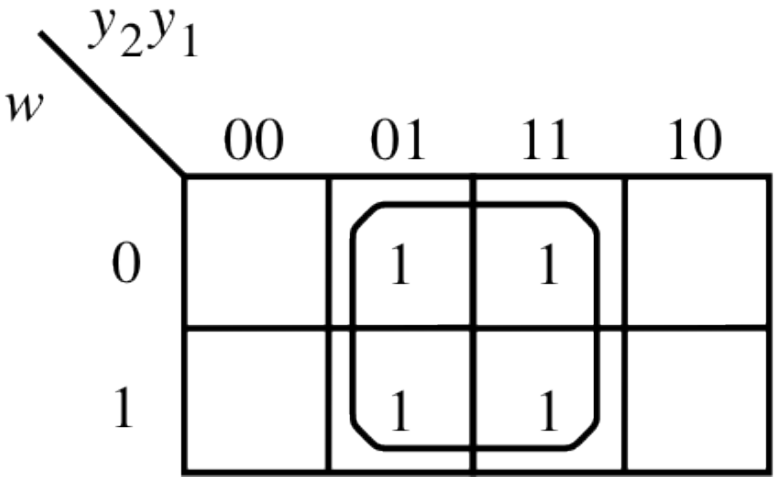
$$Y_2 = y_1\bar{y}_2 + \bar{y}_1 y_2$$

Chapter 6: State-assignment Problem

	Present state	Nextstate		Outputs						
		$w = 0$	$w = 1$							
	y_2y_1	Y_2Y_1	Y_2Y_1	$R1_{out}$	$R1_{in}$	$R2_{out}$	$R2_{in}$	$R3_{out}$	$R3_{in}$	$Done$
A	00	0 0	01	0	0	0	0	0	0	0
B	01	1 1	11	0	0	1	0	0	1	0
C	11	1 0	10	1	0	0	1	0	0	0
D	10	0 0	00	0	1	0	0	1	0	1

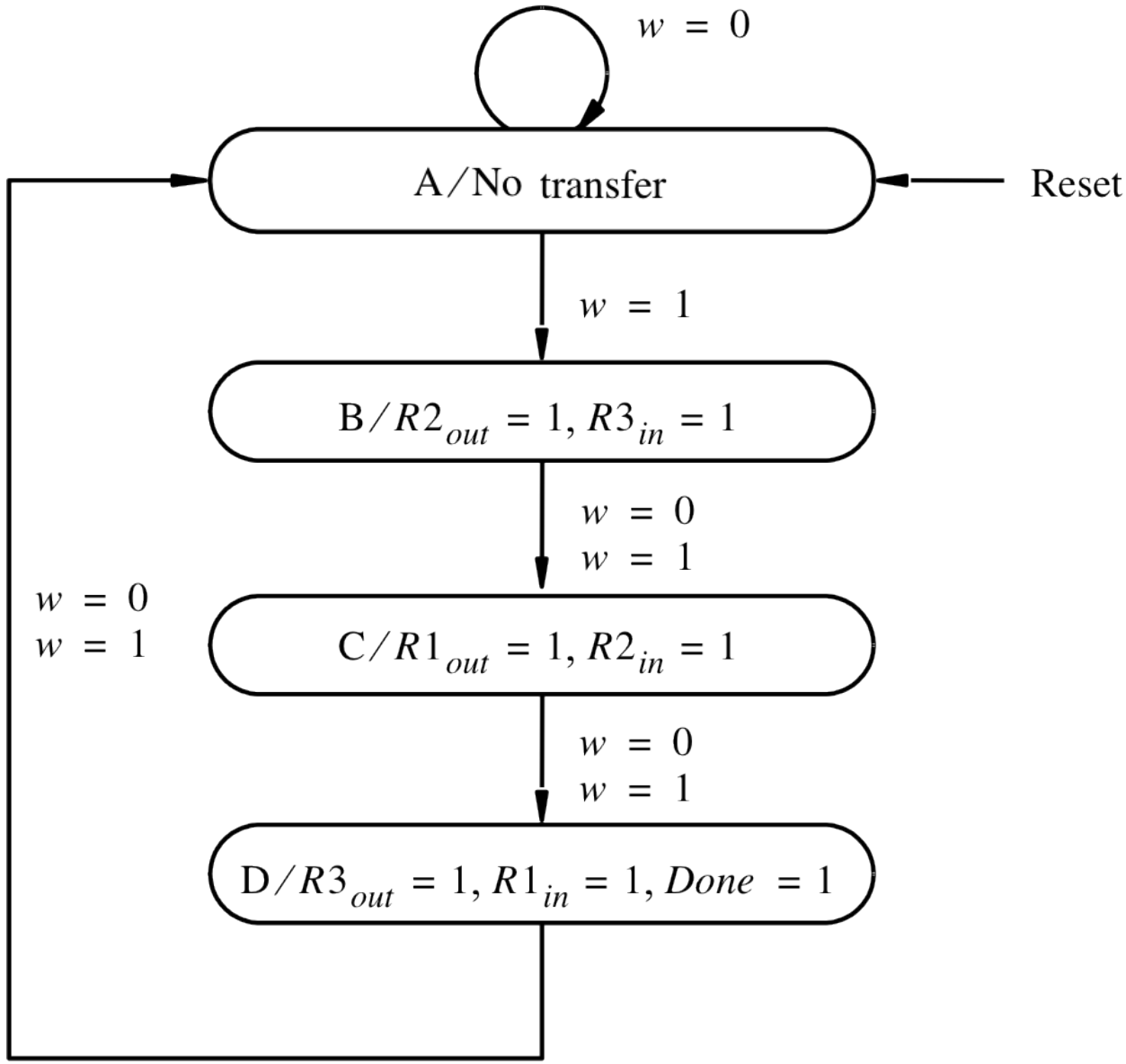


$$Y_1 = w\bar{y}_2 + y_1\bar{y}_2$$

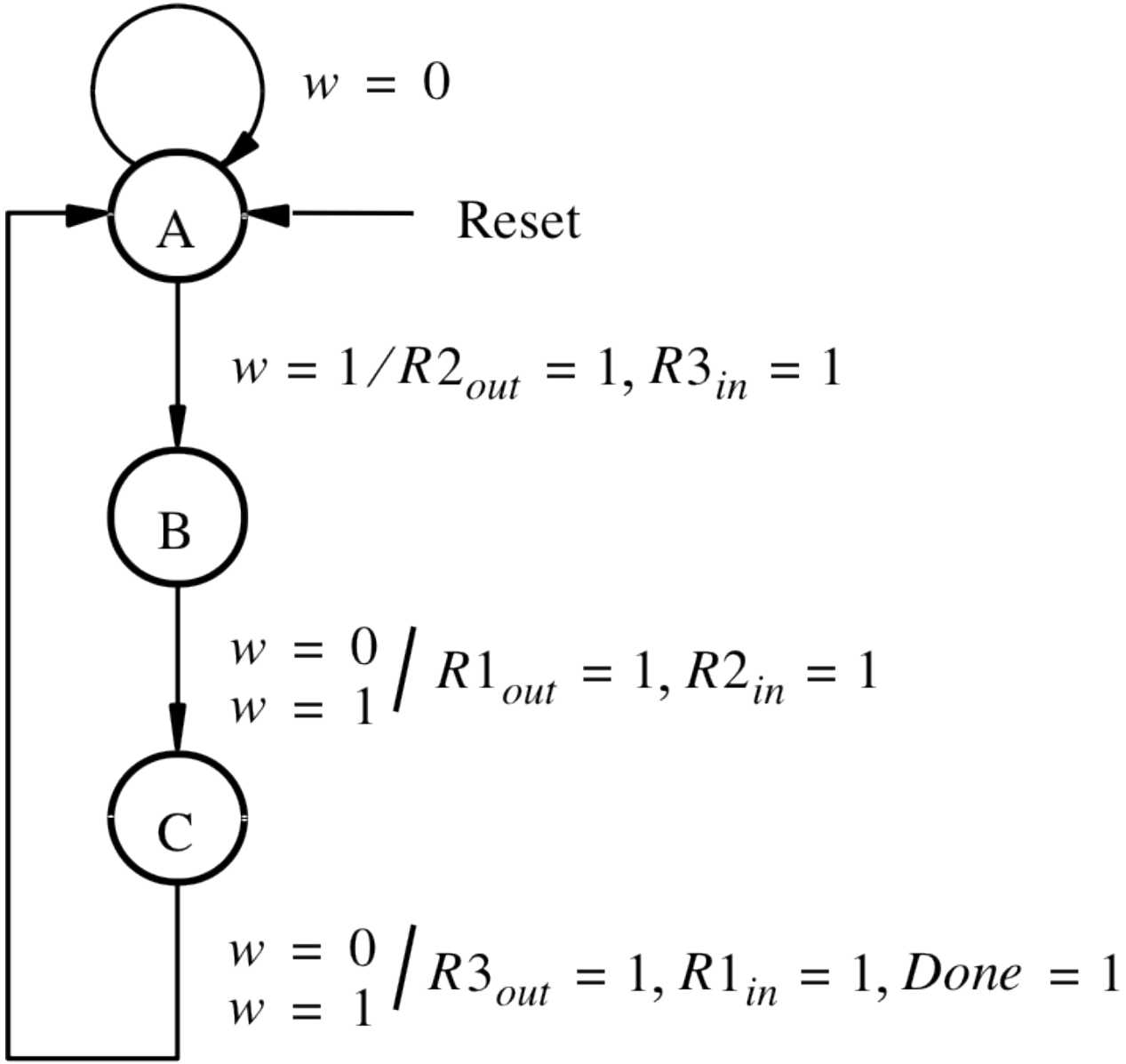


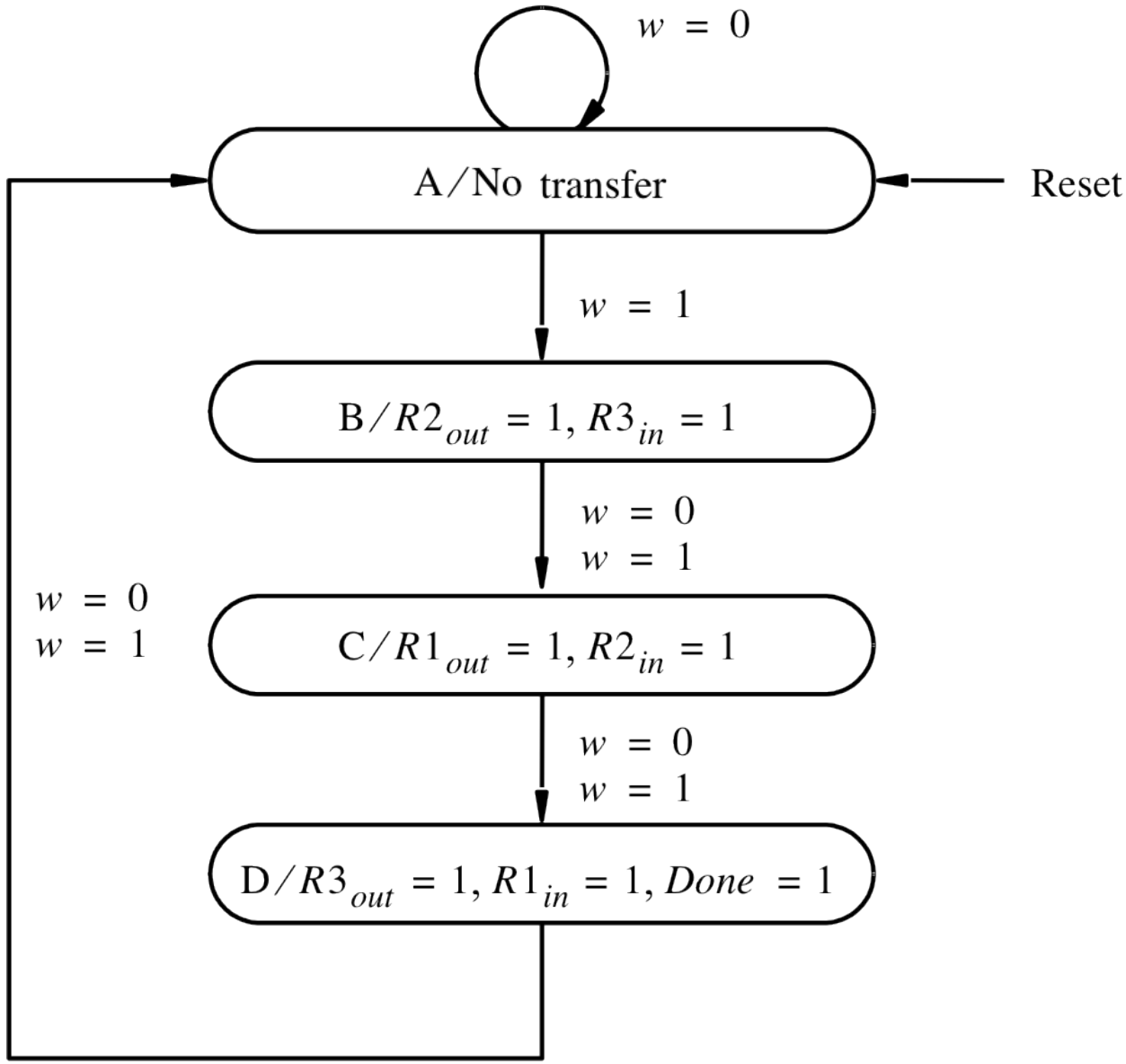
$$Y_2 = y_1$$

Chapter 6: Mealy State Model



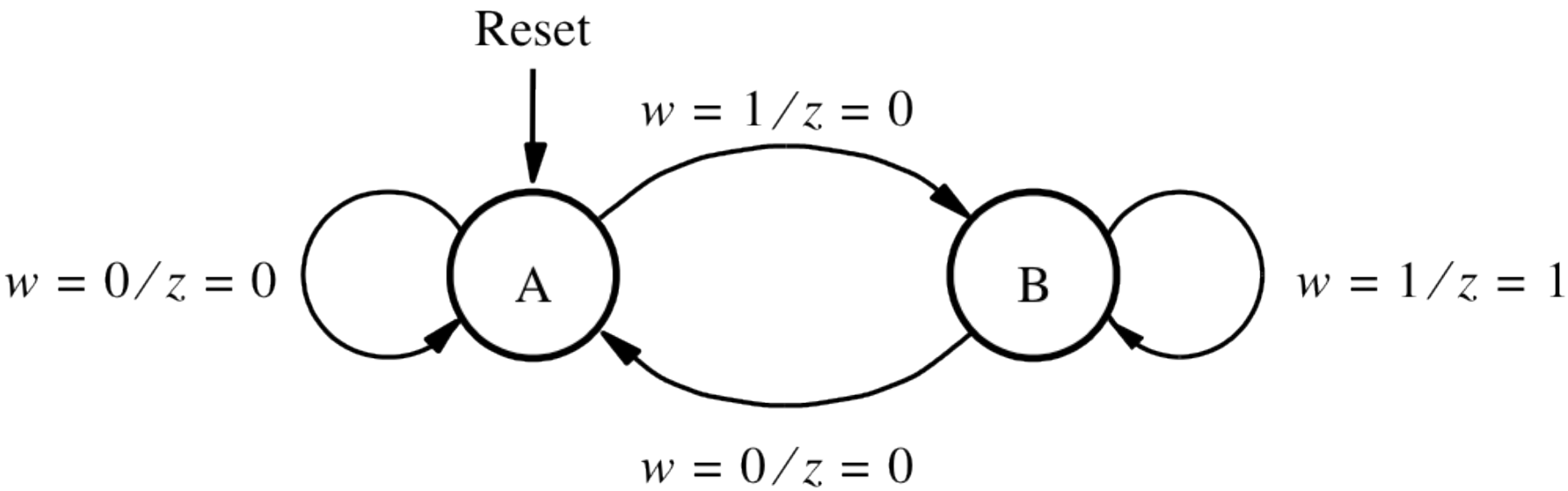
Chapter 6: Mealy State Model




w = 0
A/No transfer
Reset
w = 1
B/R2out = 1, R3in = 1
w = 0
w = 1
w = 0
w = 1
C/R1out = 1, R2in = 1
w = 0
w = 1
D/R3out = 1, R1in = 1, Done = 1

```
module control (Clock, Resetn, w, R1in, R1out, R2in, R2out, R3in,
R3out,Done);
    input Clock, Resetn, w;
    output R1in, R1out, R2in, R2out, R3in, R3out, Done;
    reg [2:1] y, Y;
    parameter [2:1] A = 2'b00, B = 2'b01, C = 2'b10, D = 2'b11;

    // Define the next state combinational circuit
    always @(w, y)
        case (y)
          A: if (w) Y = B;
              else Y = A;
          B: Y = C;
          C: Y = D;
          D: Y = A;
        endcase

    // Define the sequential block
    always @(negedge Resetn, posedge Clock)
        if (Resetn = = 0)   y <= A;
        else y <= Y;

    // Define outputs
    assign R2out = (y = = B);
    assign R3in = (y = = B);
    assign R1out = (y = = C);
    assign R2in = (y = = C);
    assign R3out = (y = = D);
    assign R1in = (y = = D);
    assign Done = (y = = D);

endmodule
```

Chapter 6: Verilog for a Mealy Machine



Present state	Next state		Output z	
	$w = 0$	$w = 1$	$w = 0$	$w = 1$
A	A	B	0	0
B	A	B	0	1

Chapter 6: Verilog for a Mealy Machine

```
module mealy (Clock, Resetn, w, z);
    input Clock, Resetn, w;
    output reg z;
    reg y, Y;
    parameter A = 1'b0, B = 1'b1;

    // Define the next state and output combinational circuits
    always @(w, y)
        case (y)
            A:  if (w)
                begin
                    z = 0;
                    Y = B;
                end
                else
                begin
                    z = 0;
                    Y = A;
                end
            B:  if (w)
                begin
                    z = 1;
                    Y = B;
                end
                else
                begin
                    z = 0;
                    Y = A;
                end
        endcase

    // Define the sequential block
    always @(negedge Resetn, posedge Clock)
        if (Resetn = = 0) y <= A;
        else       y <= Y;

endmodule
```

Present state	Next state		Outputz	
	$w = 0$	$w = 1$	$w = 0$	$w = 1$
A	B	C	0	0
B	D	–	0	–
C	F	E	0	1
D	B	G	0	0
E	F	C	0	1
F	E	D	0	1
G	F	–	0	–

Present state	Next state		Outputz	
	$w = 0$	$w = 1$	$w = 0$	$w = 1$
A	B	C	0	0
B	D	–	0	–
C	F	E	0	1
D	B	G	0	0
E	F	C	0	1
F	E	D	0	1
G	F	–	0	–

Assume unspecified outputs are 0:

$$P_1 = (ABCDEFG)$$

Present state	Next state		Outputz	
	$w = 0$	$w = 1$	$w = 0$	$w = 1$
A	B	C	0	0
B	D	–	0	–
C	F	E	0	1
D	B	G	0	0
E	F	C	0	1
F	E	D	0	1
G	F	–	0	–

Assume unspecified outputs are 0:

$$P_1 = (ABCDEFG)$$
$$P_2 = (ABDG)(CEF)$$

Present state	Next state		Outputz	
	$w = 0$	$w = 1$	$w = 0$	$w = 1$
A	B	C	0	0
B	D	–	0	–
C	F	E	0	1
D	B	G	0	0
E	F	C	0	1
F	E	D	0	1
G	F	–	0	–

Assume unspecified outputs are 0:

$$P_1 = (ABCDEFG)$$
$$P_2 = (ABDG)(CEF)$$
$$P_3 = (AB)(D)(G)(CE)(F)$$

Present state	Next state		Outputz	
	$w = 0$	$w = 1$	$w = 0$	$w = 1$
A	B	C	0	0
B	D	–	0	–
C	F	E	0	1
D	B	G	0	0
E	F	C	0	1
F	E	D	0	1
G	F	–	0	–

Assume unspecified outputs are 0:

$$P_1 = (ABCDEFG)$$
$$P_2 = (ABDG)(CEF)$$
$$P_3 = (AB)(D)(G)(CE)(F)$$
$$P_4 = (A)(B)(D)(G)(CE)(F)$$

Present state	Next state		Outputz	
	$w = 0$	$w = 1$	$w = 0$	$w = 1$
A	B	C	0	0
B	D	–	0	–
C	F	E	0	1
D	B	G	0	0
E	F	C	0	1
F	E	D	0	1
G	F	–	0	–

Assume unspecified outputs are 0:

$$P_1 = (ABCDEFG)$$
$$P_2 = (ABDG)(CEF)$$
$$P_3 = (AB)(D)(G)(CE)(F)$$
$$P_4 = (A)(B)(D)(G)(CE)(F)$$
$$P_5 = (A)(B)(D)(G)(CE)(F)$$

Present state	Next state		Outputz	
	$w = 0$	$w = 1$	$w = 0$	$w = 1$
A	B	C	0	0
B	D	–	0	–
C	F	E	0	1
D	B	G	0	0
E	F	C	0	1
F	E	D	0	1
G	F	–	0	–

Assume unspecified outputs are 1:

$$P_1 = (ABCDEFG)$$

Present state	Next state		Outputz	
	$w = 0$	$w = 1$	$w = 0$	$w = 1$
A	B	C	0	0
B	D	–	0	–
C	F	E	0	1
D	B	G	0	0
E	F	C	0	1
F	E	D	0	1
G	F	–	0	–

Assume unspecified outputs are 1:

$$P_1 = (ABCDEFG)$$
$$P_2 = (AD)(BCEFG)$$

Present state	Next state		Outputz	
	$w = 0$	$w = 1$	$w = 0$	$w = 1$
A	B	C	0	0
B	D	–	0	–
C	F	E	0	1
D	B	G	0	0
E	F	C	0	1
F	E	D	0	1
G	F	–	0	–

Assume unspecified outputs are 1:

$$P_1 = (ABCDEFG)$$
$$P_2 = (AD)(BCEFG)$$
$$P_3 = (AD)(B)(CEFG)$$

Present state	Next state		Outputz	
	$w = 0$	$w = 1$	$w = 0$	$w = 1$
A	B	C	0	0
B	D	–	0	–
C	F	E	0	1
D	B	G	0	0
E	F	C	0	1
F	E	D	0	1
G	F	–	0	–

Assume unspecified outputs are 1:

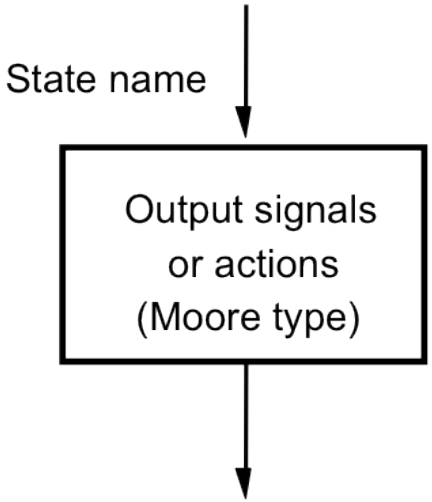
$$P_1 = (ABCDEFG)$$
$$P_2 = (AD)(BCEFG)$$
$$P_3 = (AD)(B)(CEFG)$$
$$P_4 = (AD)(B)(CEG)(F)$$

Present state	Next state		Outputz	
	$w = 0$	$w = 1$	$w = 0$	$w = 1$
A	B	C	0	0
B	D	–	0	–
C	F	E	0	1
D	B	G	0	0
E	F	C	0	1
F	E	D	0	1
G	F	–	0	–

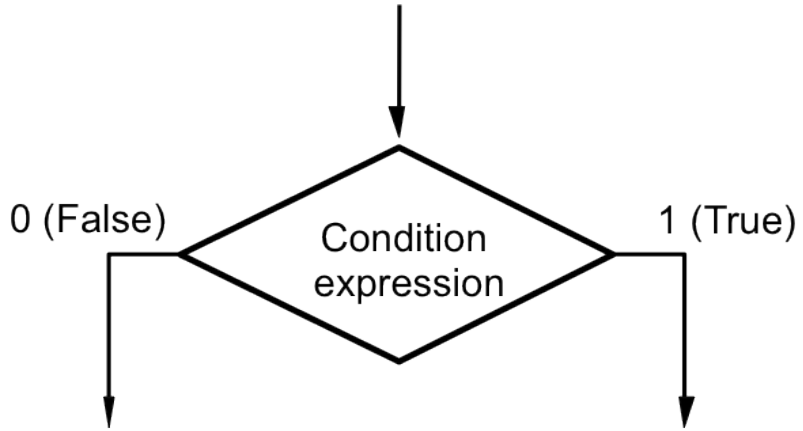
Assume unspecified outputs are 1:

$$P_1 = (ABCDEFG)$$
$$P_2 = (AD)(BCEFG)$$
$$P_3 = (AD)(B)(CEFG)$$
$$P_4 = (AD)(B)(CEG)(F)$$
$$P_5 = (AD)(B)(CEG)(F)$$

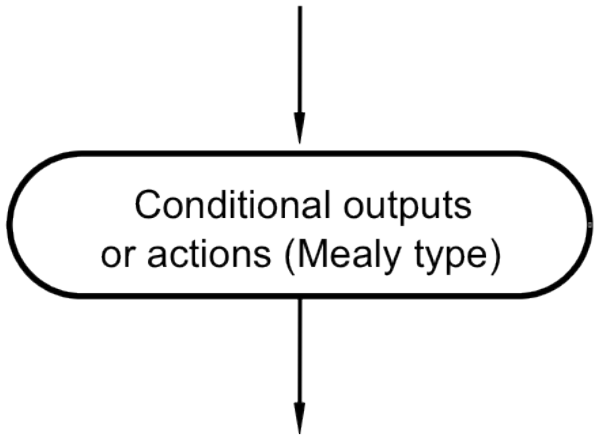
Chapter 6: ASM Charts



(a) State box

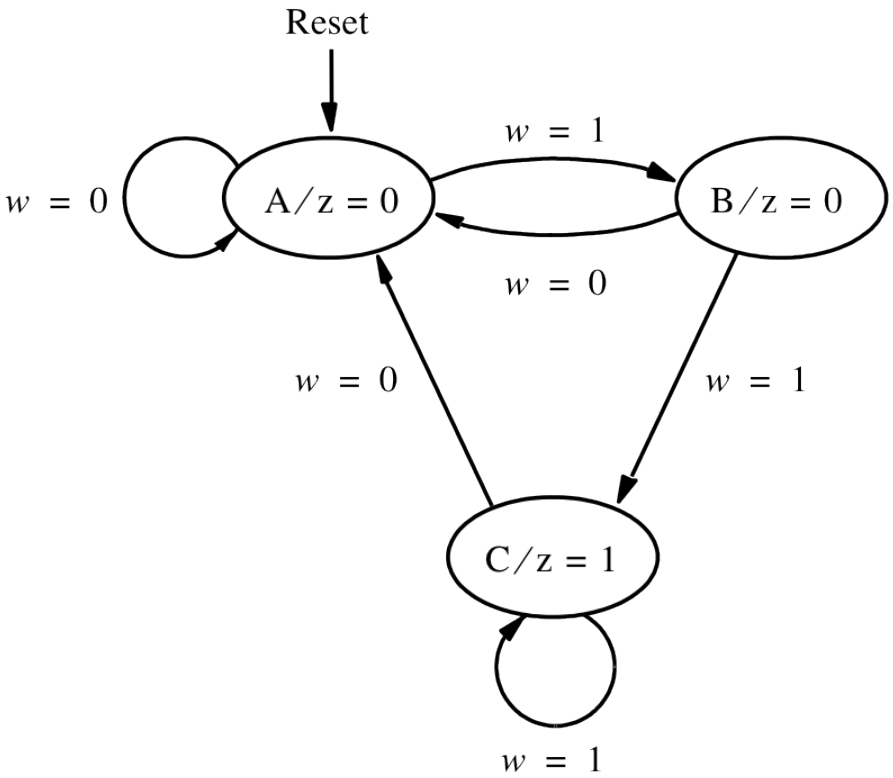


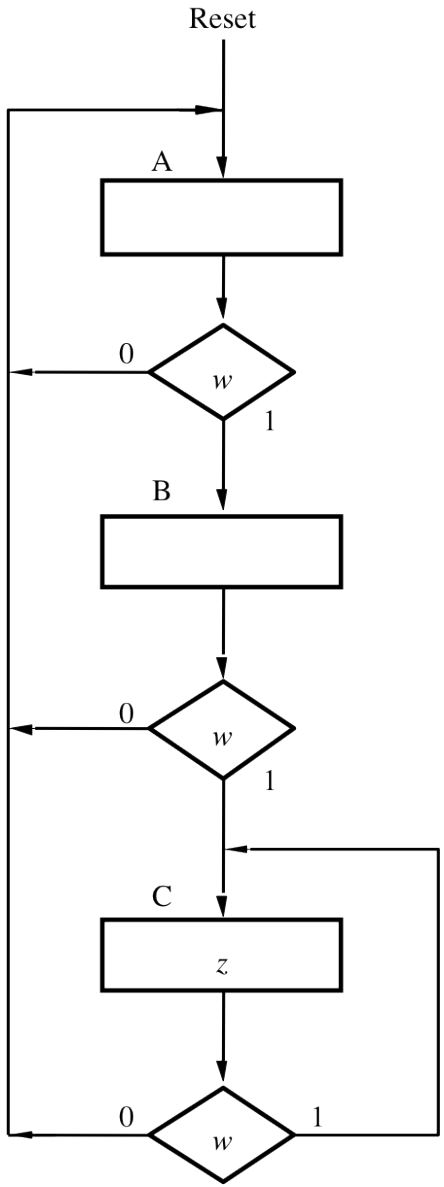
(b) Decision box



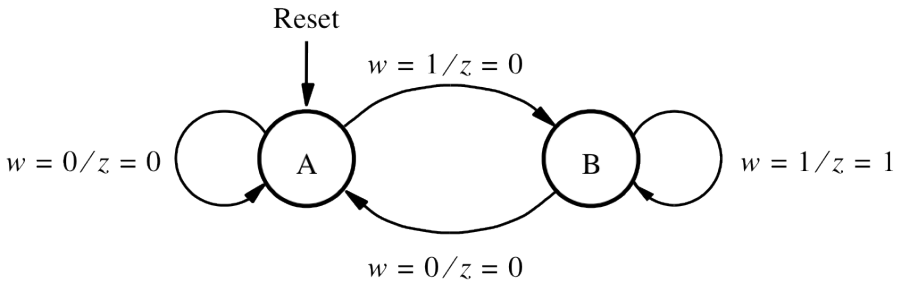
(c) Conditional output box

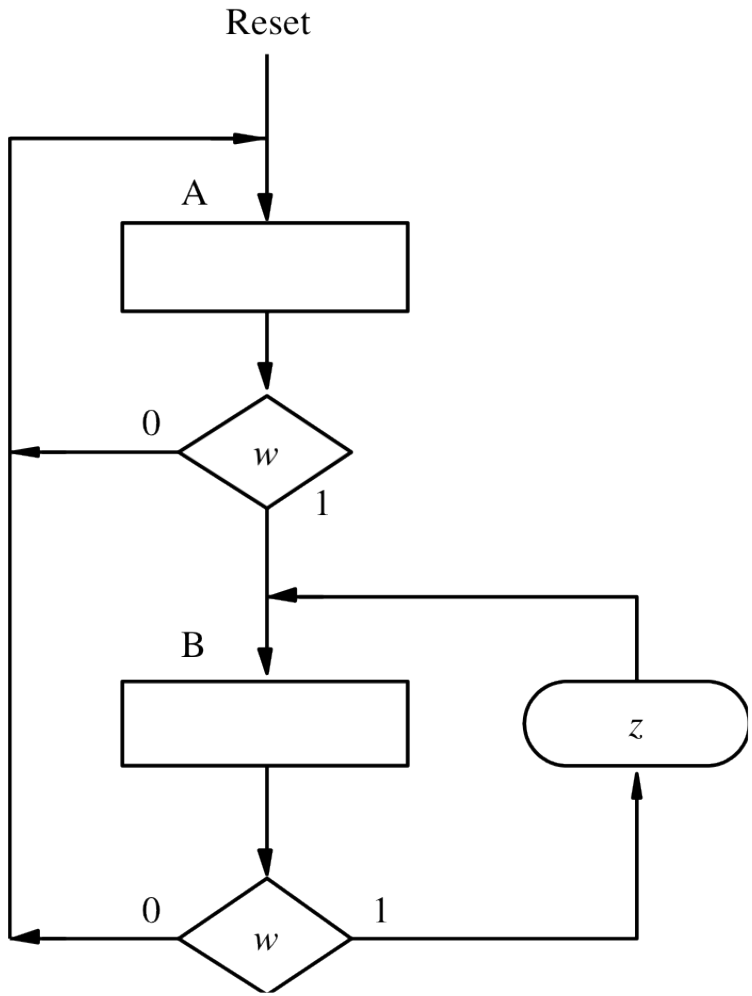
Chapter 6: ASM Charts



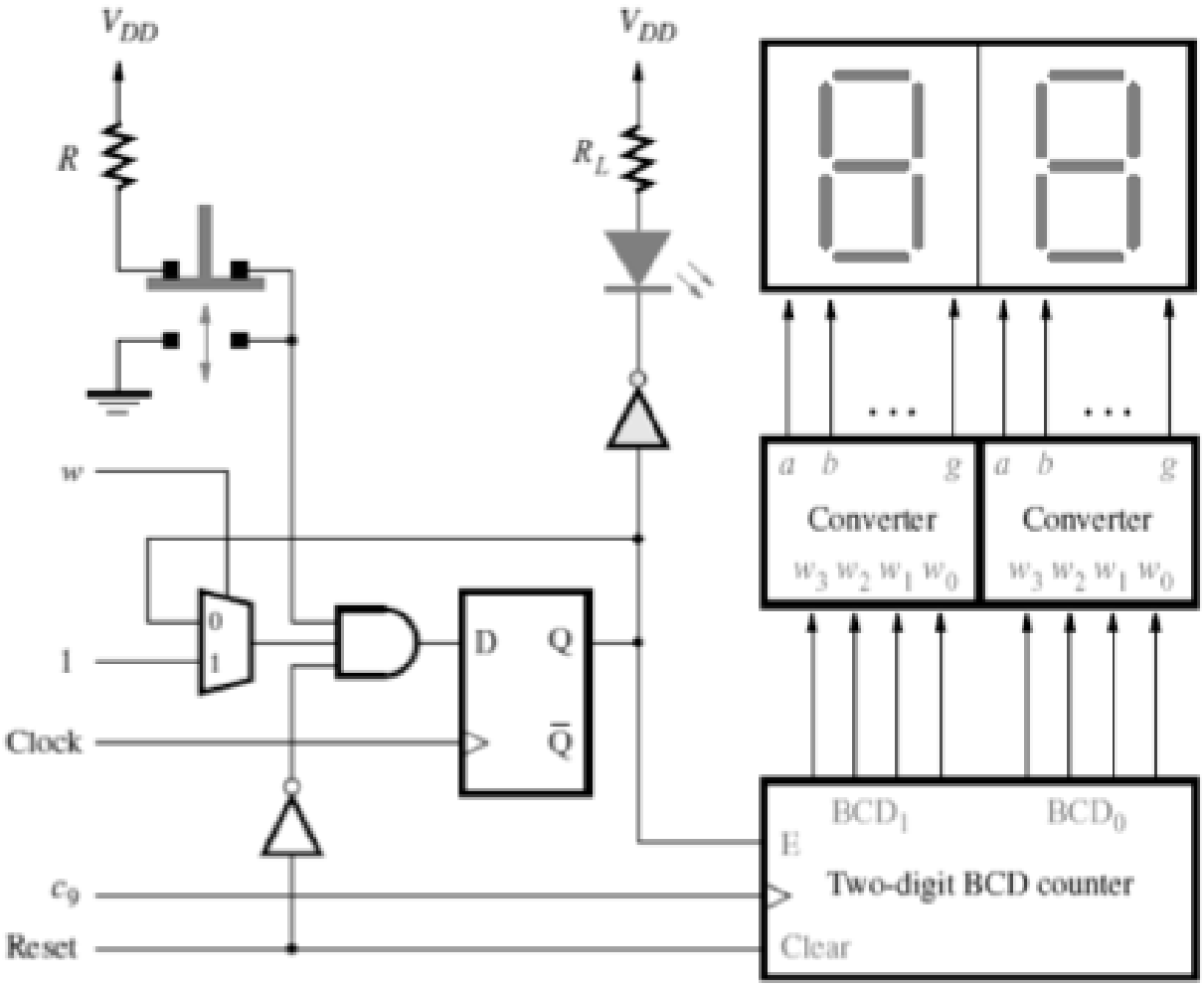


Chapter 6: ASM Charts



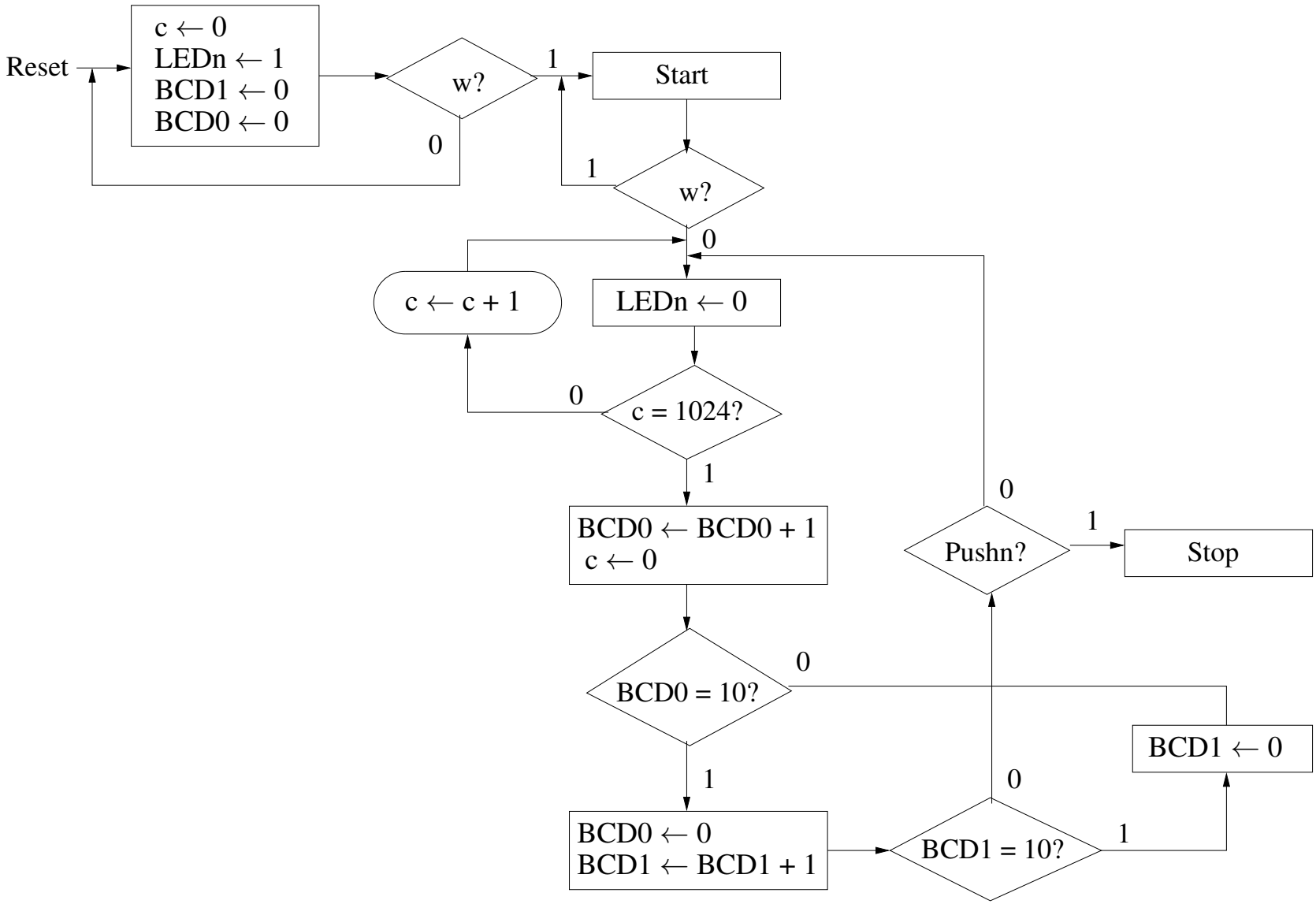


Reaction-timer Circuit



(c) Push-button switch, LED, and 7-segment displays

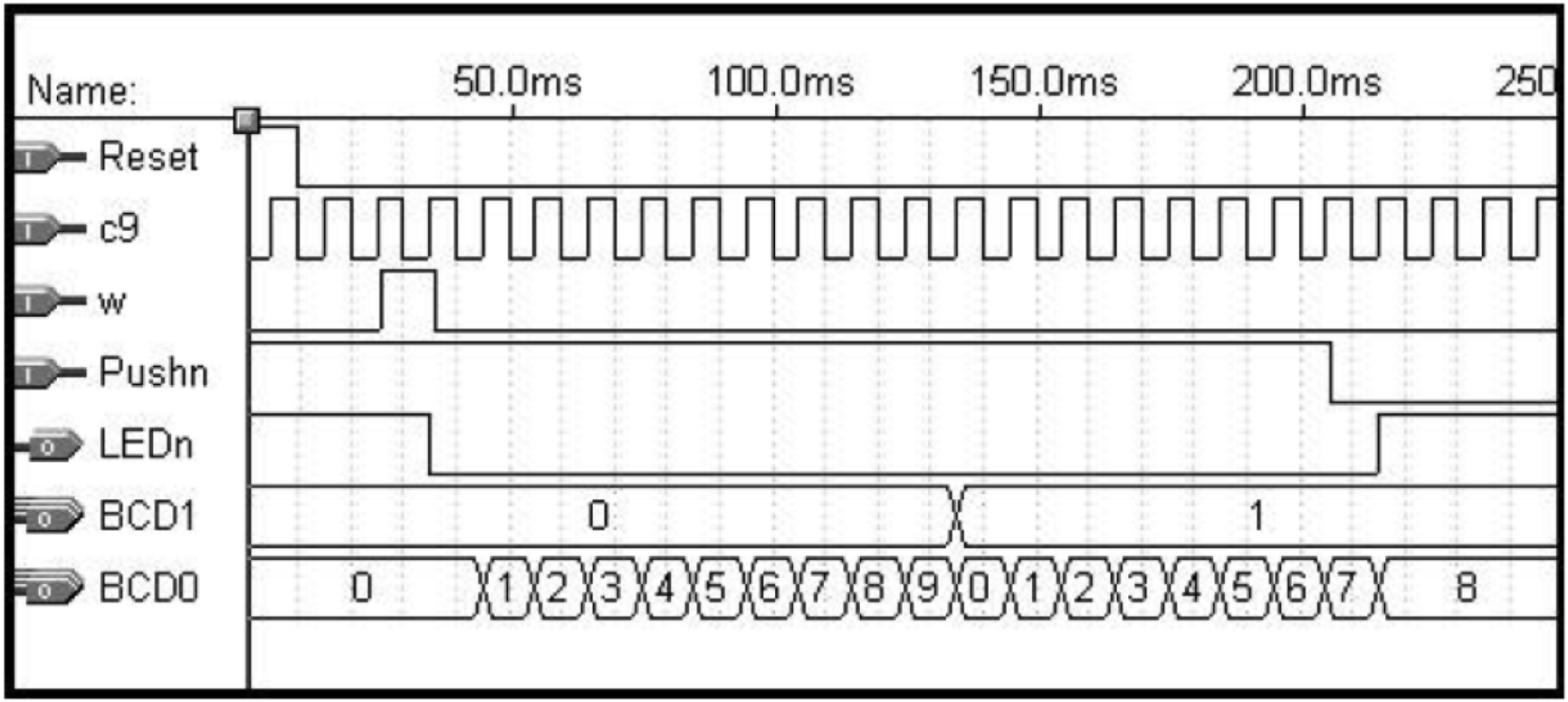
ASM Chart for the Reaction-timer Circuit



Verilog Testbenches (Sequential)

```
`timescale 1ms / 1ns
module test_reaction_timer;
  // Inputs
  reg Clock;
  reg Reset;
  reg w;
  reg Pushn;
  // Outputs
  wire LEDn;
  wire [3:0] BCD1, BCD0;
  // Instantiate the Unit Under Test (UUT)
  reaction uut (
    .Clock(Clock),
    .Reset(Reset),
    .w(w),
    .Pushn(Pushn),
    .LEDn(LEDn),
    .BCD1(BCD1),
    .BCD0(BCD0)
  );
```

Simulation for a Reaction-timer



Verilog Testbenches (Sequential)

```
initial begin
  $display("Simulations Begins...");
  Clock = 0;
  Reset = 1;
  w = 0;
  Pushn = 1;
  #10
  Reset = 0;
  #15
  w = 1;
  #10
  w = 0;
  #170
  Pushn = 0;
end
always
  #5 Clock <= ~Clock;
```

Verilog Testbenches (Sequential)

```
  always@(negedge Clock)
    if (Pushn == 0 && Ledn == 1)
    begin
      if (!(BCD1 == 1 && BCD0 == 8))
        $display("Wrong Count");
      $display("...  Simulations Ends");
    end
endmodule
```