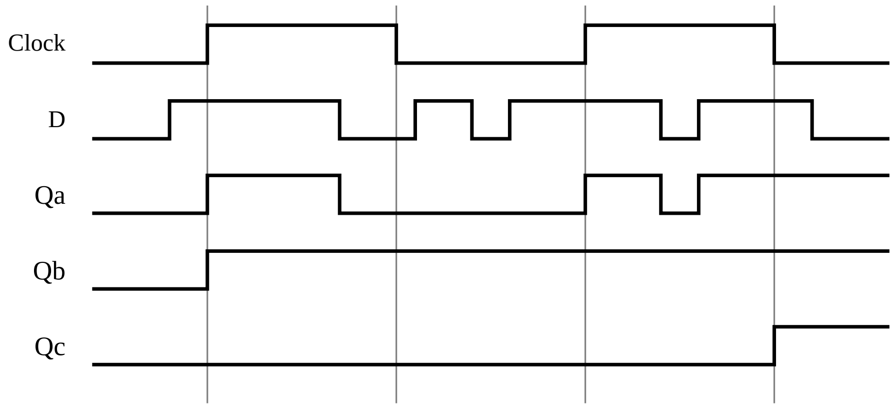
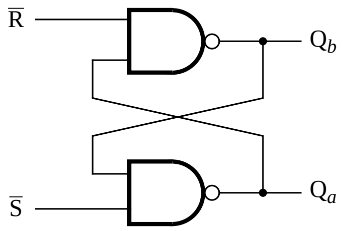
Chapter 5

5.1.

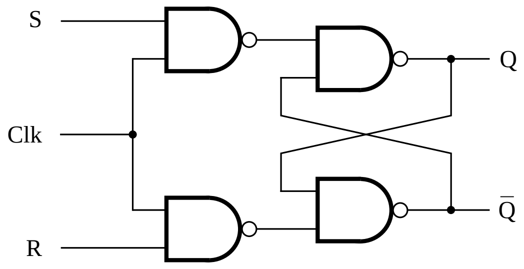


5.2.

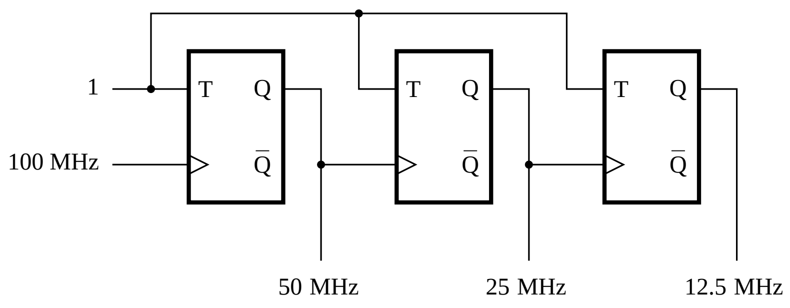


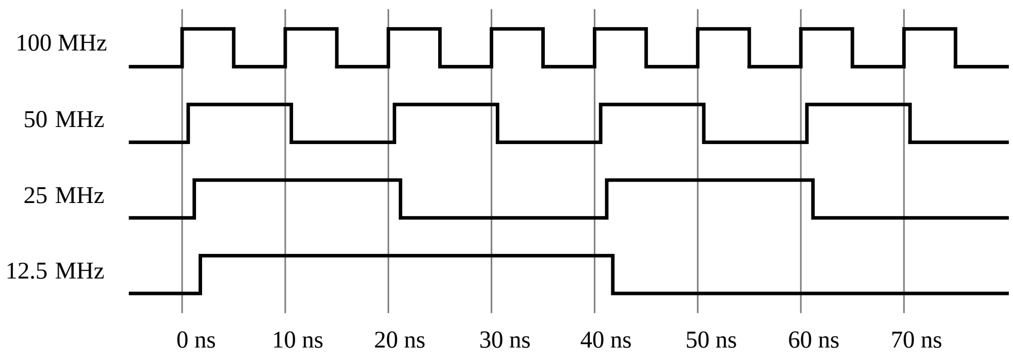
$\overline{S}$	$\overline{R}$	Q_a	Q_b	
1	1	0/1	1/0	(no change)
1	0	0	1	
0	1	1	0	
0	0	1	1	

5.3.

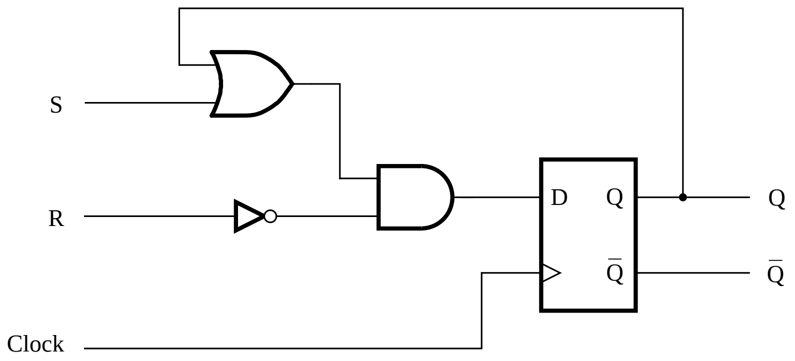


5.4.

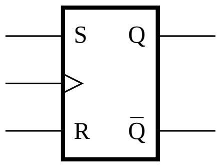




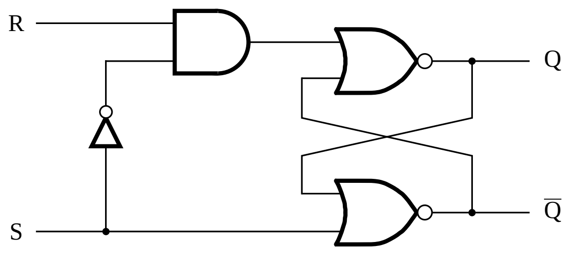
5.5.



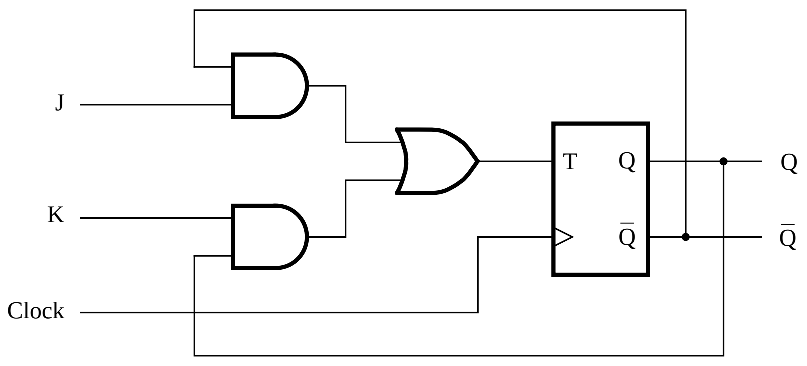
S	R	Q(*t* + 1)
0	0	Q(*t*)
0	1	0
1	0	1
1	1	0



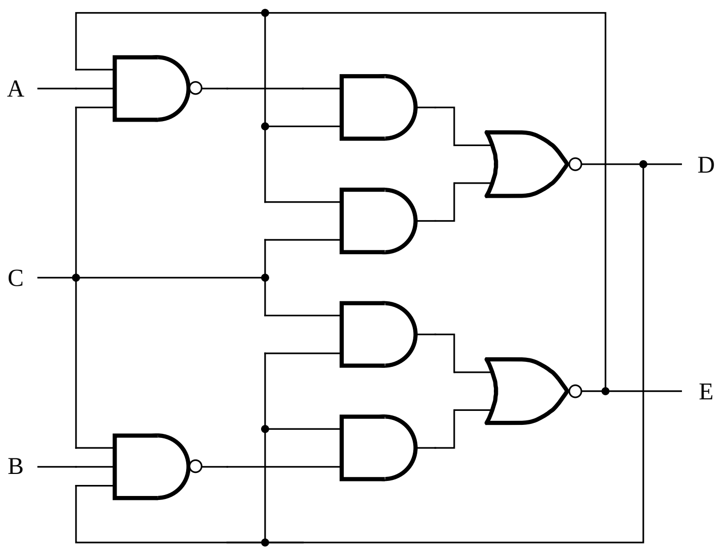
5.6.



5.7.



5.8. As the circuit in Figure P5.2 is drawn, it is not a useful flip-flop circuit, because setting $C = 0$ results in both of the circuit outputs being set to 0. Consider the slightly modified circuit shown below:



This modified circuit acts as a negative-edge-triggered JK flip-flop, in which $J = A, K = B$, *Clock* = C, Q $= D$, and $\overline{\text{Q}} = E$. This circuit is found in the standard chip called 74LS107A (plus a *Clear* input, which is not shown).

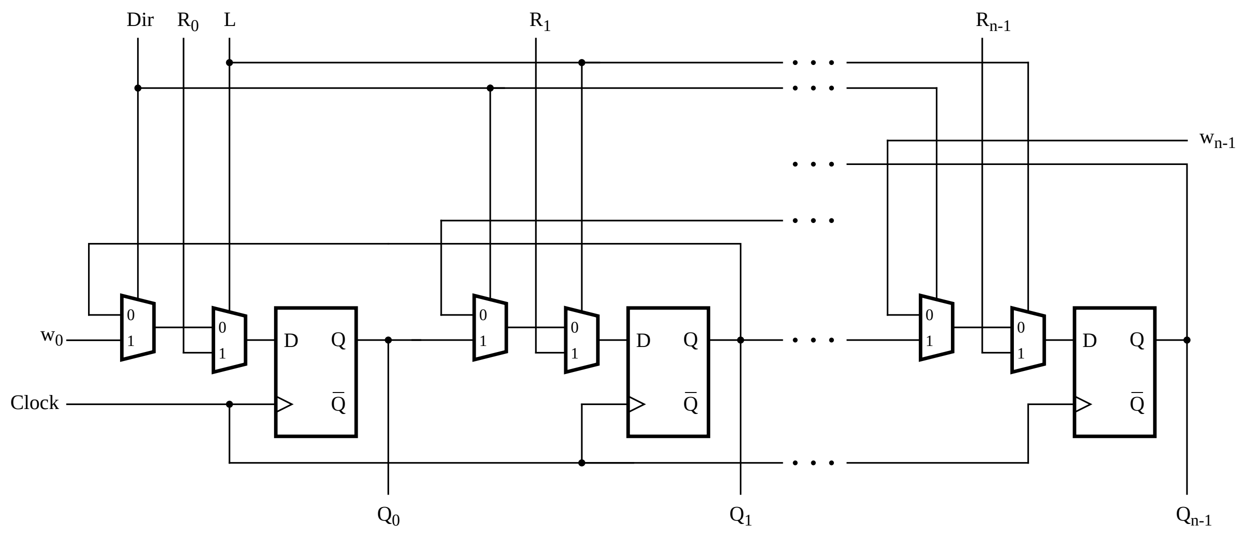
5.9.

```
module tflipflop (T, Clock, Resetn, Q);
   input T, Clock, Resetn;
   output reg Q;

   always @(negedge Resetn, posedge Clock)
      if (!Resetn)
         Q <= 0;
      else if (T)
         Q <=  Q;

endmodule
```

5.10.

```
module jkflipflop (J, K, Clock, Resetn, Q);
   input J, K, Clock, Resetn;
   output reg Q;

   always @(negedge Resetn, posedge Clock)
      if (!Resetn)
         Q <= 0;
      else
         case (J, K)
            1'b01:      Q <= 0;
            1'b10:      Q <= 1;
            1'b11:      Q <=  Q;
            default:    Q <= Q;
         endcase

endmodule
```

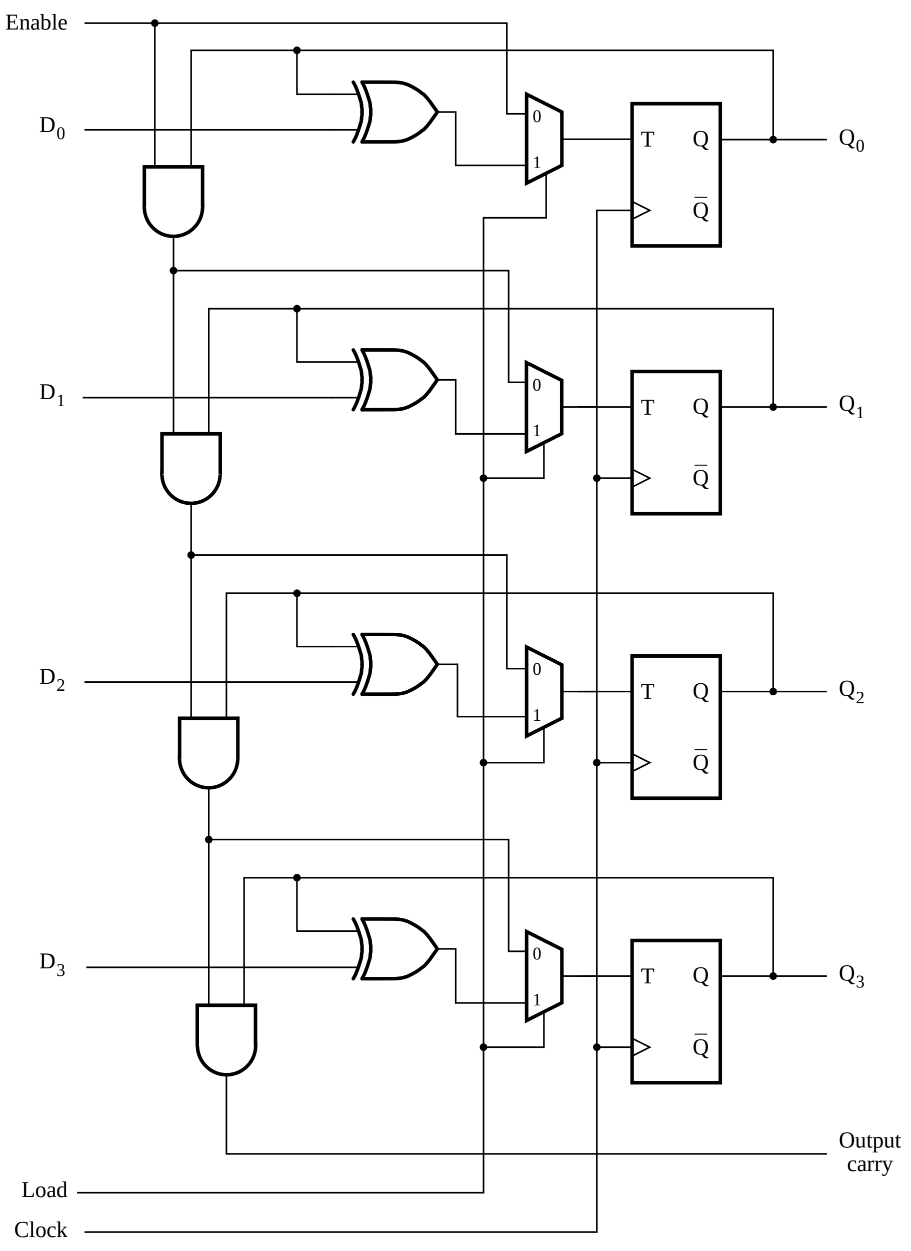
5.12. A possible circuit is



5.13.

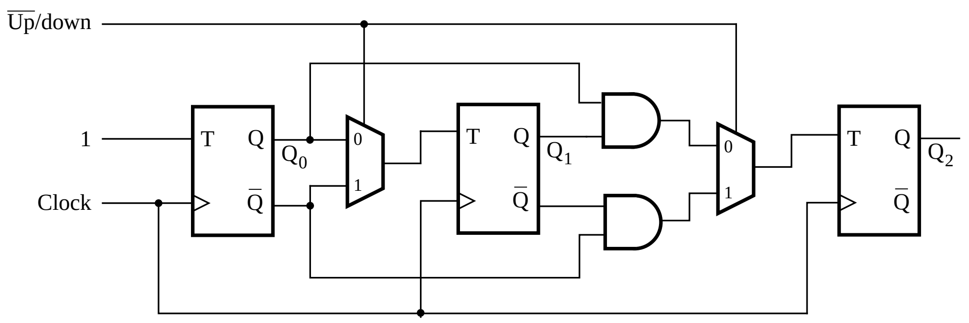
```
// Universal shift register. If Dir = 0 shifting is to the left.
module universaln (R, L, Dir, w0, w1, Clock, Q);
    parameter n = 4;
    input [n−1:0] R;
    input L, Dir, w0, w1, Clock;
    output reg [n−1:0] Q;
    integer k;

    always @(posedge Clock)
        if (L)
            Q <= R;
        else
        begin
            if (Dir)
            begin
                for (k = 0; k < n−1; k = k+1)
                    Q[k] <= Q[k+1];
                Q[n−1] <= w0;
            end
            else
            begin
                Q[0] <= w1;
                for (k = n−1; k > 0; k = k−1)
                    Q[k] <= Q[k−1];
            end
        end

endmodule
```

5.14.



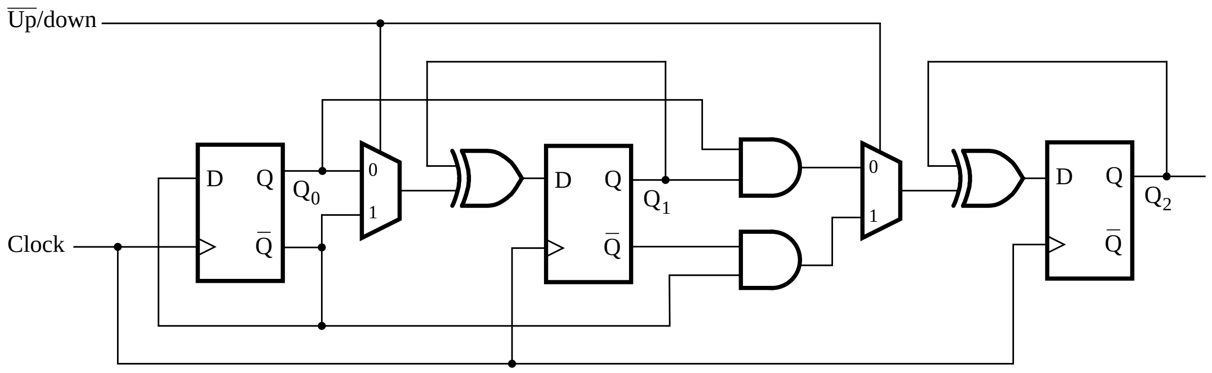
Enable
D_0
D_1
D_2
D_3
0
1
T
Q
$\overline{Q}$
Q_0
Q_1
Q_2
Q_3
Output carry
Load
Clock

5.15.



$\overline{Up}$/down
1
Clock
T
Q
$\overline{Q}$
Q_0
Q_1
Q_2
0
1

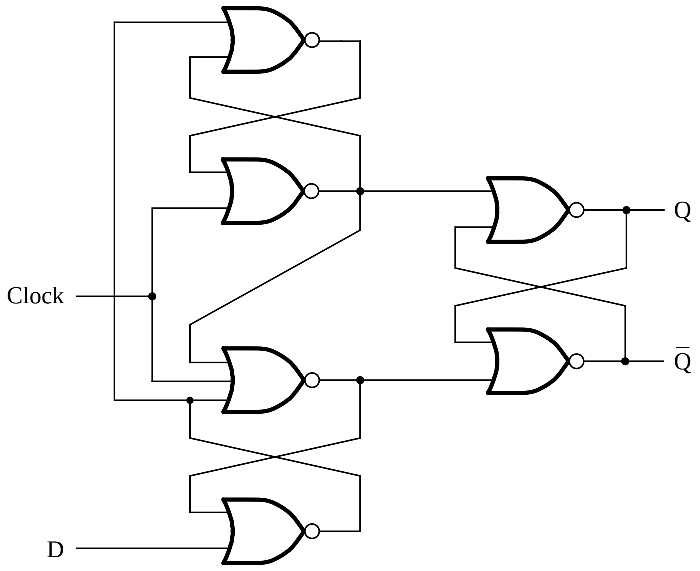
5.16.



5.17. The counting sequence is $000, 001, 010, 111$.

5.18. The circuit in Figure P5.4 is a master-slave JK flip-flop. It suffers from a problem sometimes called *ones-catching*. Consider the situation where the Q output is low, *Clock* = 0, and $J = K = 0$. Now let *Clock* remain stable at 0 while J change from 0 to 1 and then back to 0. The master stage is now set to 1 and this value will be incorrectly transferred into the slave stage when the clock changes to 1.

5.19. Repeated application of DeMorgan's theorem can be used to change the positive-edge triggered D flip-flop in Figure 5.11 into the negative-edge D triggered flip-flop:



5.20.
```
module upcount12 (Resetn, Clock, Q);
    input Resetn, Clock;
    output reg [3:0] Q;

    always @(posedge Clock)
        if (!Resetn)
            Q <= 0;
        else if (Q == 11)
            Q <= 0;
        else
            Q <= Q + 1;

endmodule
```

5.21. The longest delay in the circuit is the from the output of FF_0 to the input of FF_3. This delay totals 5 ns. Thus the minimum period for which the circuit will operate reliably is

$$T_{min} = 5 \text{ ns} + t_{su} = 8 \text{ ns}$$

The maximum frequency is

$$F_{max} = 1/T_{min} = 125 \text{ MHz}$$

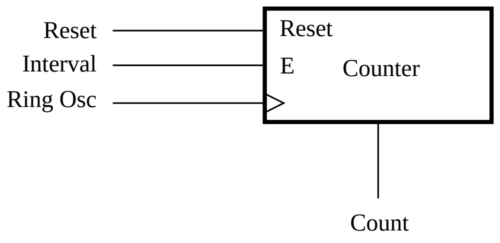
5.22.
```
module johnson8 (Resetn, Clock, Q);
    input Resetn, Clock;
    output reg [7:0] Q;
    reg [7:0] Q;

    always @(negedge Resetn, posedge Clock)
        if (!Resetn)
            Q <= 0;
        else
            Q <= {{Q[6:0]}, {~Q[7]}};

endmodule
```

5.23.

```
// Ring counter with synchronous reset
module ripplen (Resetn, Clock, Q);
    parameter n = 8;
    input Resetn, Clock;
    output reg [n−1:0] Q;

    always @(posedge Clock)
        if (!Resetn)
        begin
            Q[7:1] <= 0;
            Q[0] <= 1;
        end
        else
            Q <= {{Q[6:0]}, {Q[7]}};

endmodule
```

5.24. (a) $\text{Period} = 2 \times n \times t_p$

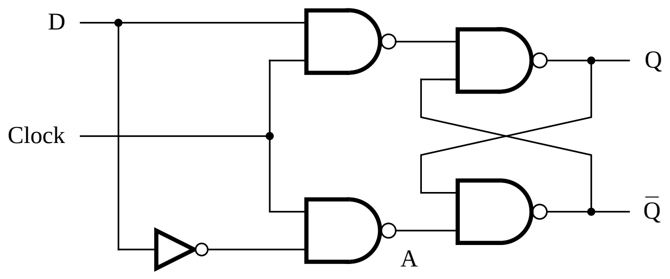
(b)

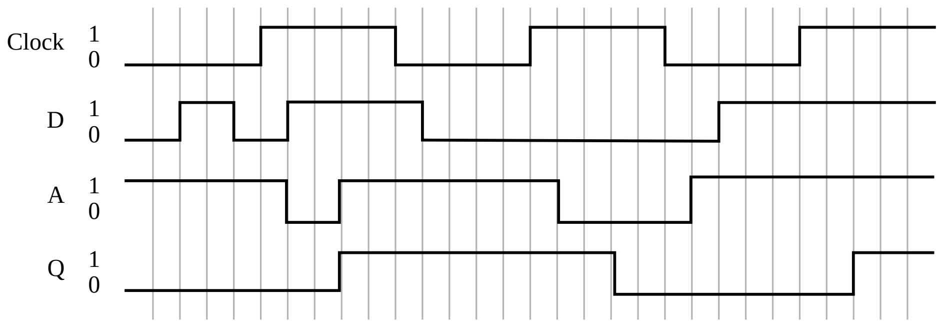


The counter tallies the number of pulses in the 100 ns time period. Thus

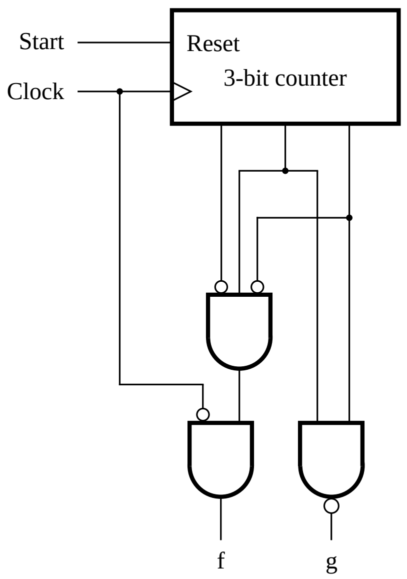
$$t_p = \frac{100\text{ ns}}{2 \times Count \times n}$$

5.25.



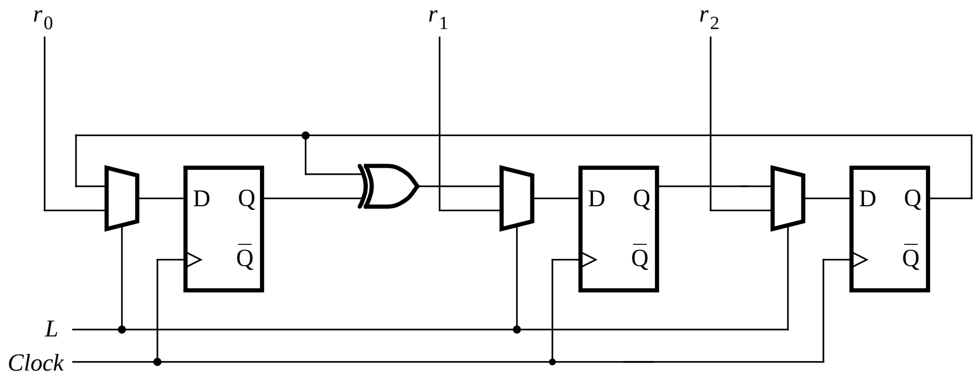


5.26.



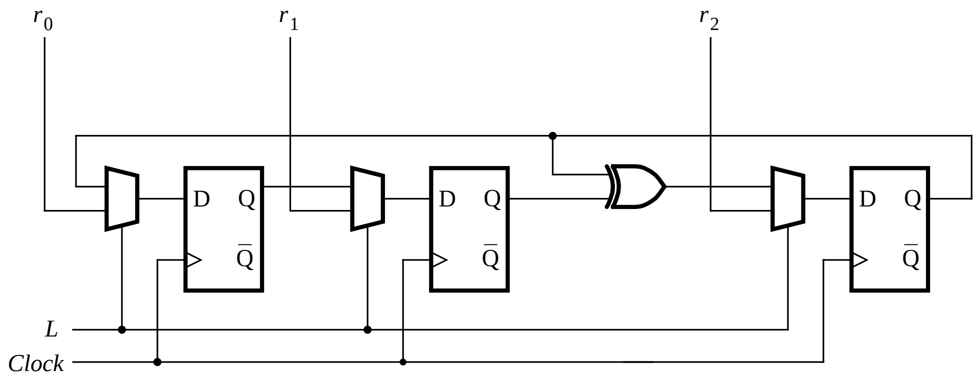
5.27. With non-blocking assignments, the result of the assignment f <= A[1] & A[0] is not seen by the successive assignments inside the **for** loop. Thus, f has an uninitialized value when the **for** loop is entered. Similarly, each **for** loop interation sees the unitialized value of f. The result of the code is the sequential circuit specified by f = f | A[n-1] A[n-2].

5.28.



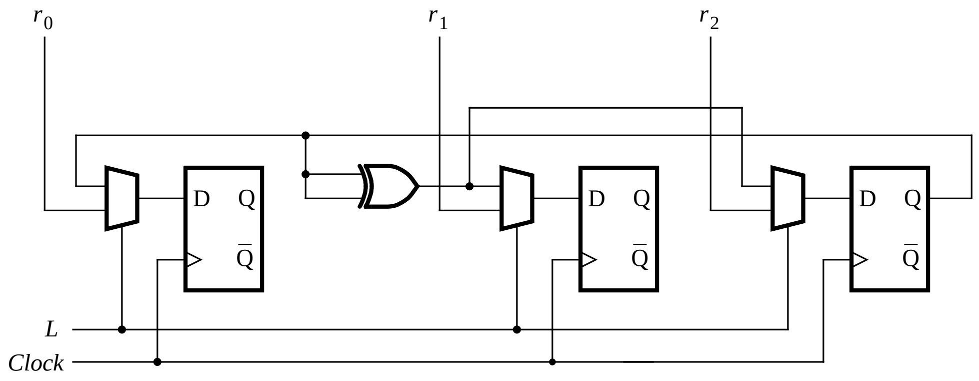
The counting sequence is: 001, 110, 011, 111, 101, 100, 010, 001

5.29.



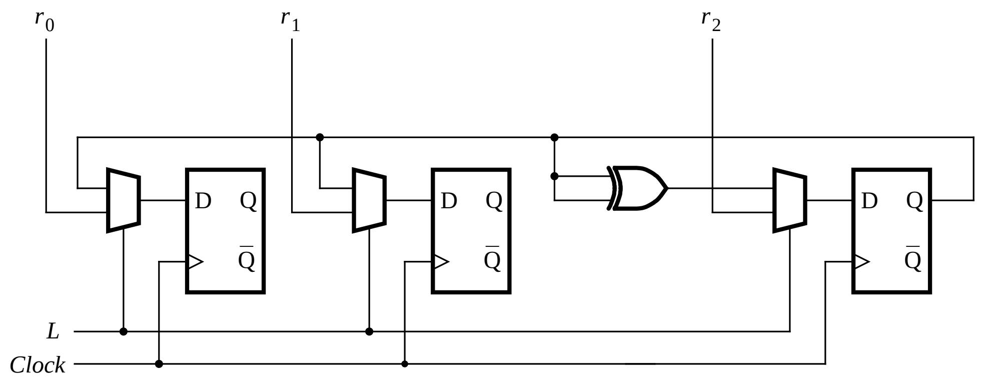
The counting sequence is: 001, 101, 111, 110, 011, 100, 010, 001

5.30.



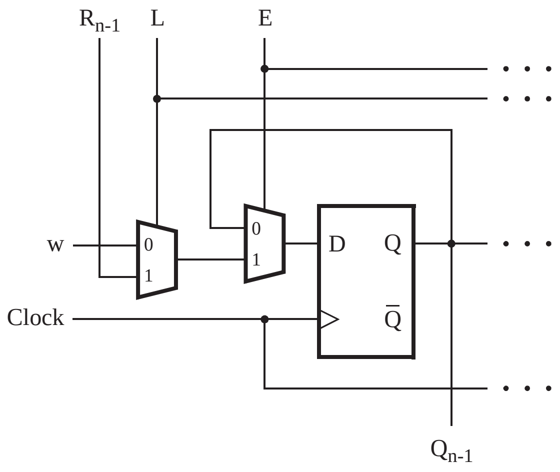
The counting sequence is: 001, 100, 000, 000, ...

5.31.



The counting sequence is: 001, 110, 000, 000, ...

5.32. The required shift register is the same as the one in Figure 5.59, except that the order of the two multiplexers that feed each flip-flop is reversed. The circuit structure is indicated in the figure below.



R_{n-1}
L
E
w
0
1
0
1
D Q
$\overline{Q}$
Clock
Q_{n-1}