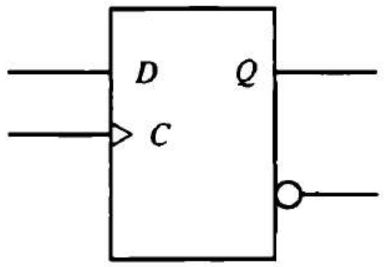
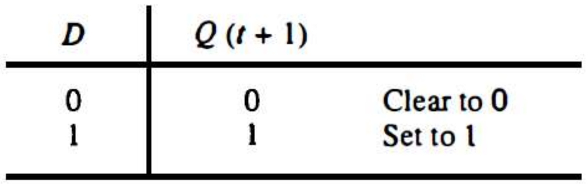
D-flipflop



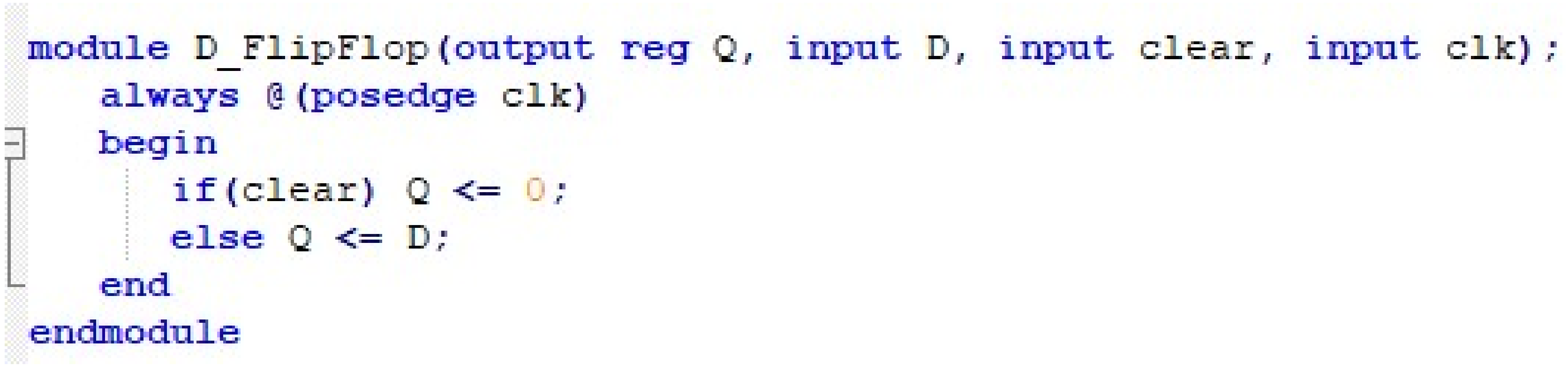
(a) Graphic symbol



D	$Q(t+1)$	
0	0	Clear to 0
1	1	Set to 1

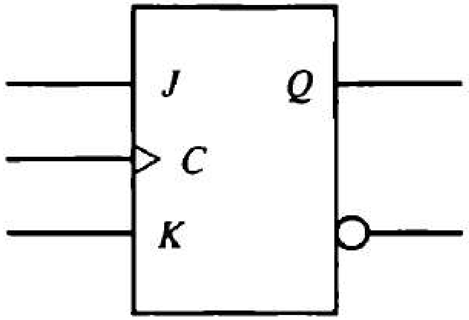
(b) Characteristic table

Figure 1-20 *D* flip-flop.



```
module D_FlipFlop(output reg Q, input D, input clear, input clk);
   always @(posedge clk)
   begin
      if(clear) Q <= 0;
      else Q <= D;
   end
endmodule
```

JK-flipflop



(a) Graphic symbol

J	K	$Q(t+1)$	
0	0	$Q(t)$	No change
0	1	0	Clear to 0
1	0	1	Set to 1
1	1	$Q'(t)$	Complement

(b) Characteristic table

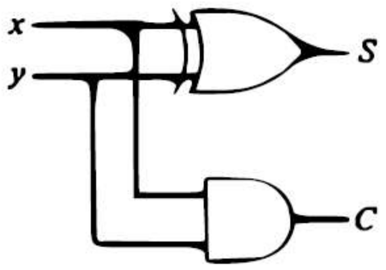
Figure 1-21 *JK* flip-flop.

```
module JK_FlipFlop(output reg Q, input J, input K, input clear, input clk);
   always @(posedge clk)
   begin
      if(clear) Q <= 0;
      else if (J ^ K) Q <= J;
      else if (J & K) Q <= ~Q;
   end
endmodule
```

Half Adder

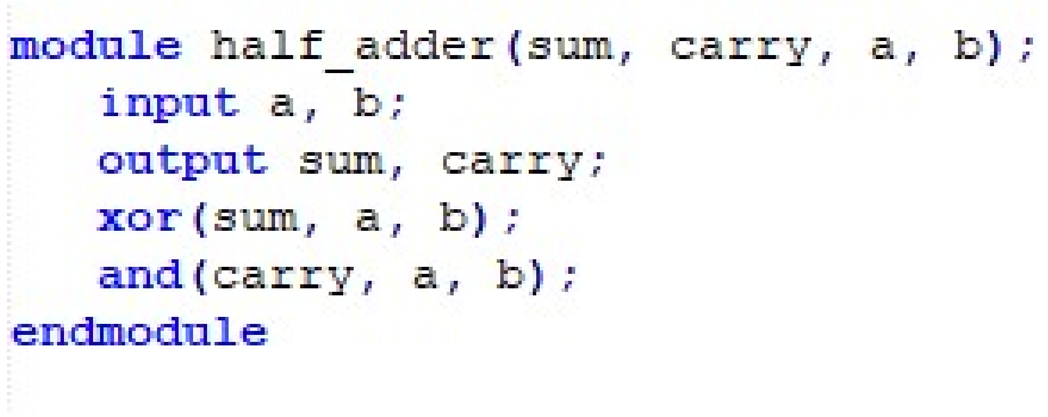
x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

(a) **Truth** table



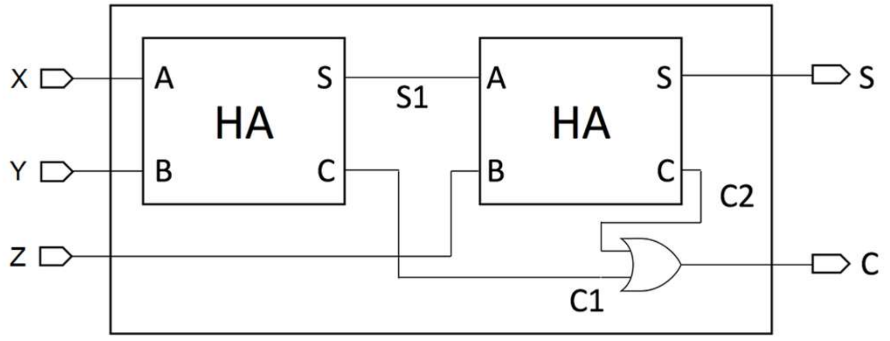
(b) Logic diagram

Figure 1-16 Half-adder.



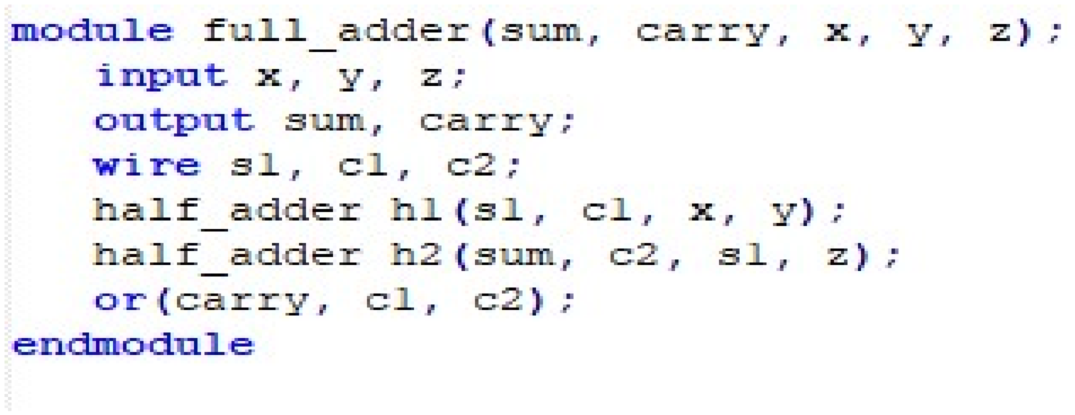
```
module half_adder(sum, carry, a, b);
   input a, b;
   output sum, carry;
   xor(sum, a, b);
   and(carry, a, b);
endmodule
```

Full Adder

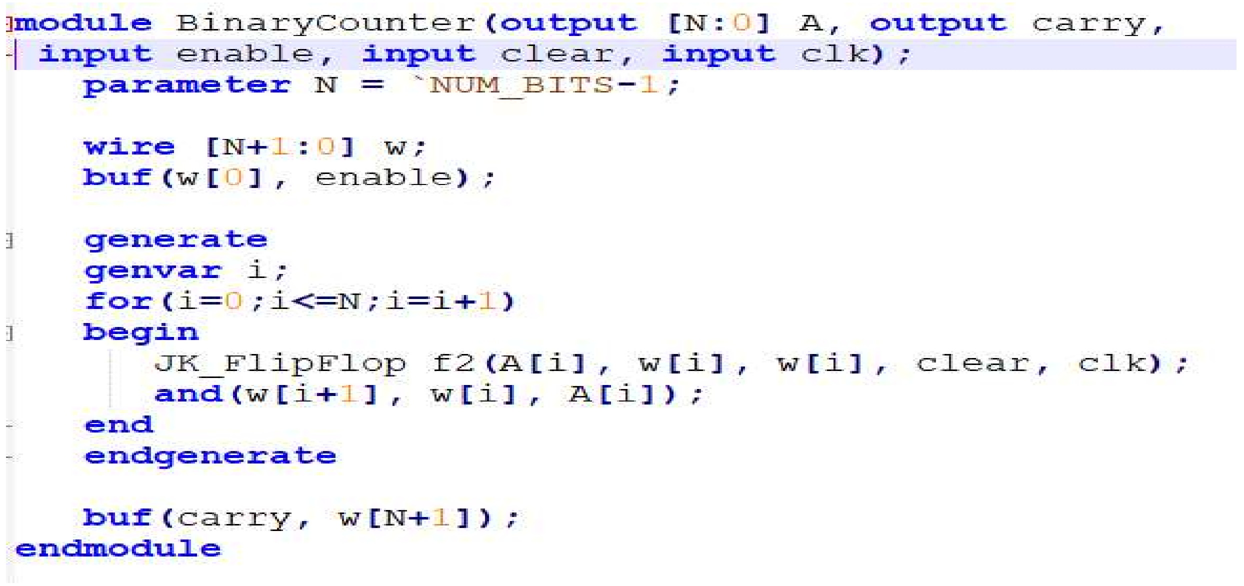


Full-adder using two half-adders

Inputs			Outputs	
A	B	C_{in}	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



```
module full_adder(sum, carry, x, y, z);
    input x, y, z;
    output sum, carry;
    wire s1, c1, c2;
    half_adder h1(s1, c1, x, y);
    half_adder h2(sum, c2, s1, z);
    or(carry, c1, c2);
endmodule
```



```verilog
module BinaryCounter(output [N:0] A, output carry,
 input enable, input clear, input clk);
   parameter N = `NUM_BITS-1;

   wire [N+1:0] w;
   buf(w[0], enable);

   generate
   genvar i;
   for(i=0;i<=N;i=i+1)
   begin
      JK_FlipFlop f2(A[i], w[i], w[i], clear, clk);
      and(w[i+1], w[i], A[i]);
   end
   endgenerate

   buf(carry, w[N+1]);
endmodule
```

A3	A2	A1	A0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

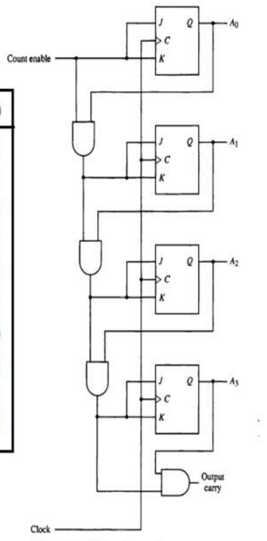
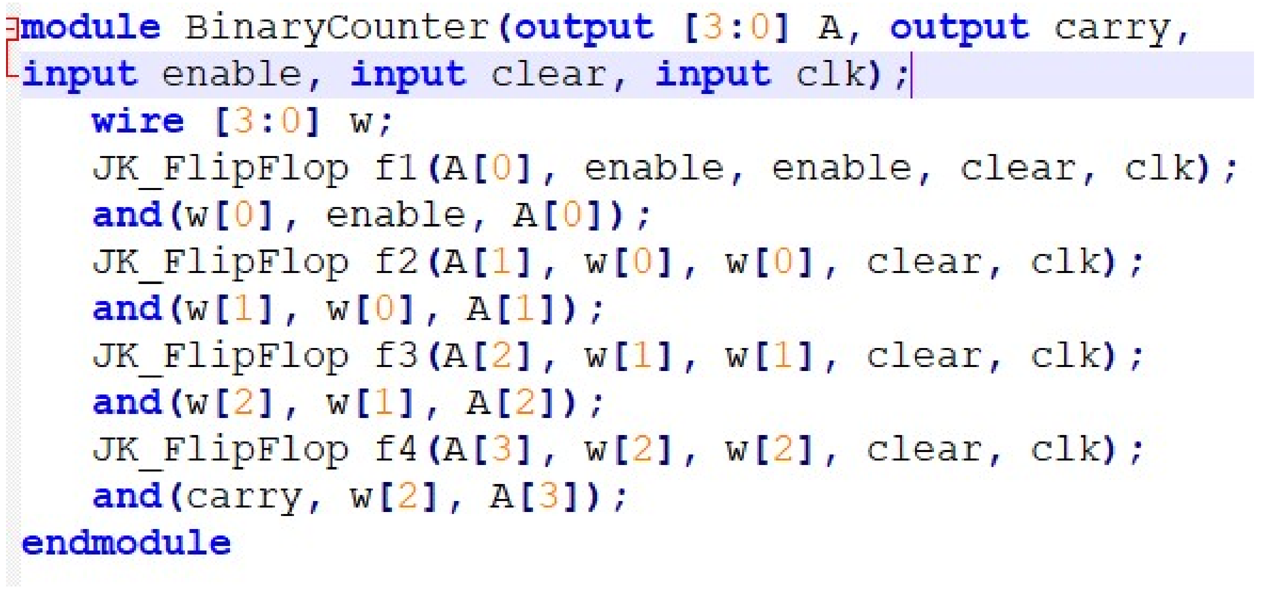


Figure 2-10 4-bit synchronous binary counter.

J	K	Q (t + 1)	
0	0	Q (t)	No change
0	1	0	Clear to 0
1	0	1	Set to 1
1	1	Q′ (t)	Complement



```verilog
module BinaryCounter(output [3:0] A, output carry,
input enable, input clear, input clk);
   wire [3:0] w;
   JK_FlipFlop f1(A[0], enable, enable, clear, clk);
   and(w[0], enable, A[0]);
   JK_FlipFlop f2(A[1], w[0], w[0], clear, clk);
   and(w[1], w[0], A[1]);
   JK_FlipFlop f3(A[2], w[1], w[1], clear, clk);
   and(w[2], w[1], A[2]);
   JK_FlipFlop f4(A[3], w[2], w[2], clear, clk);
   and(carry, w[2], A[3]);
endmodule
```