

King Saud University

FALL 2022

College of Computer & Information Sciences

CEN 415

Department of Computer Engineering

Dr. KHURSHEED AURANGZEB

Student Name: Muhannad Funissan Alanizy.	Section: 32693	Student ID: 439101758.
Student Name: Rami Khaled Almutiry.	Section: 32693	Student ID: 439102284.
Student Name: Sultan Alotaibi.	Section: 32693	Student ID: 439101605.

CEN 415_Project

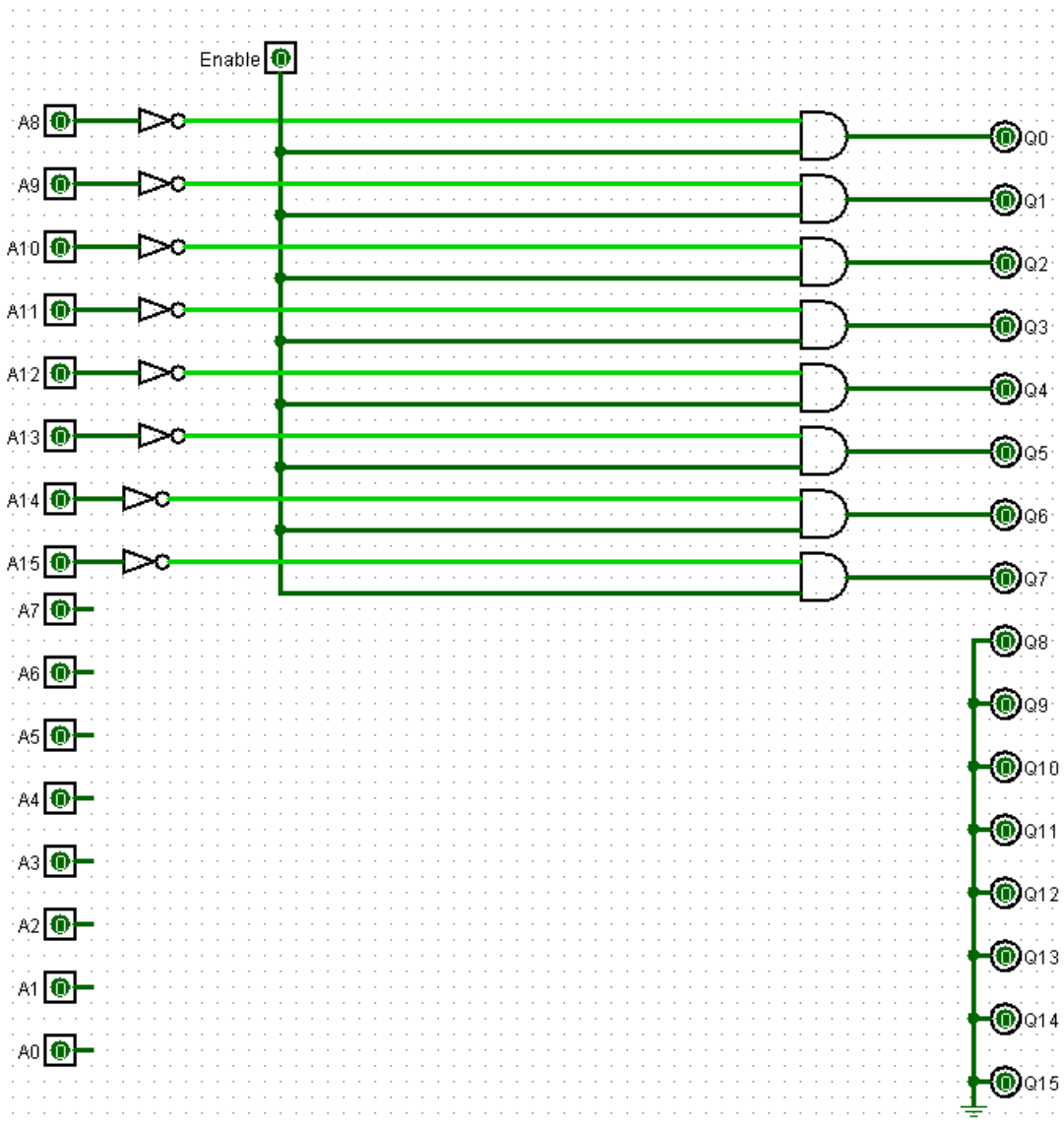
Phase (1)

Introduction:

In this phase we will design and implement a circuit that has two 8-bit inputs and one 16-bit output to perform the four cases that we have.

Case 00:

This the circuit design in Logisim that generate the complement of the first input and showing the result in the 8 least significant bits of the output $\rightarrow Q = A'$.



Verilog Code:

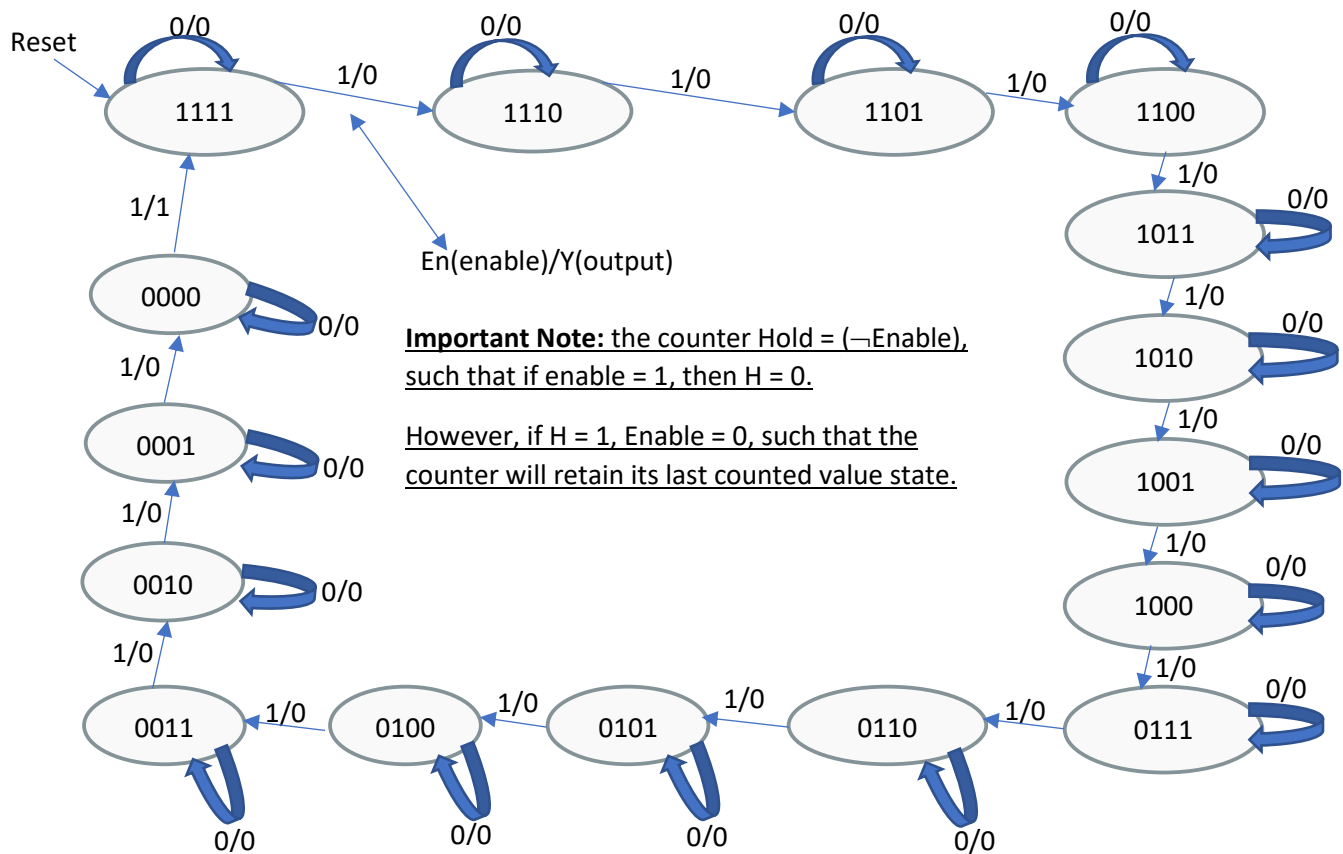
```
1 module inverter(a,b,q);
2
3     input [7:0]a;
4     input [7:0]b;
5     output [15:0]q;
6
7     for(genvar i = 0; i<16; i=i+1)begin
8         if (i<8)
9             assign q[i]= ~a[i];
10        else
11            assign q[i] = 0;
12        end
13    endmodule
```

Case 01:

- 1) Firstly, we started constructing the State Table and drawing the state diagram as shown below:

Number#	Q _A Q _B Q _C Q _D	Q _A ⁺ Q _B ⁺ Q _C ⁺ Q _D ⁺	Y
15	1111	1110	0
14	1110	1101	0
13	1101	1100	0
12	1100	1011	0
11	1011	1010	0
10	1010	1001	0
9	1001	1000	0
8	1000	0111	0
7	0111	0110	0
6	0110	0101	0
5	0101	0100	0
4	0100	0011	0
3	0011	0010	0
2	0010	0001	0
1	0001	0000	0
0	0000	1111	1

Table 1.1



state diagram 1.1

- 2) Secondly, we moved on to develop K-maps for extracting Minimum next states Equations, as you can see, we used only 3 k-maps because the Q_d and $y(\text{output})$ are easy to deduce by observing the state table.

D_A AB / CD	00	01	11	10
00	1	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	1	1	1

Table 1.2

$$D_A = A'B'C'D' + AB + AC + AD$$

D_B AB / CD	00	01	11	10
00	1	0	0	0
01	0	1	1	1
11	0	1	1	1
10	1	0	0	0

Table 1.3

$$D_B = B'C'D' + BC + BD$$

D _C AB / CD	00	01	11	10
00	1	0	1	0
01	1	0	1	0
11	1	0	1	0
10	1	0	1	0

Table 1.4

$$D_C = C'D' + CD, \quad D_0 = D'$$

$$Y = ABCD$$

3) Lastly, we used Logisim to build our equivalent sequential circuit (modulo 16 down-counter).

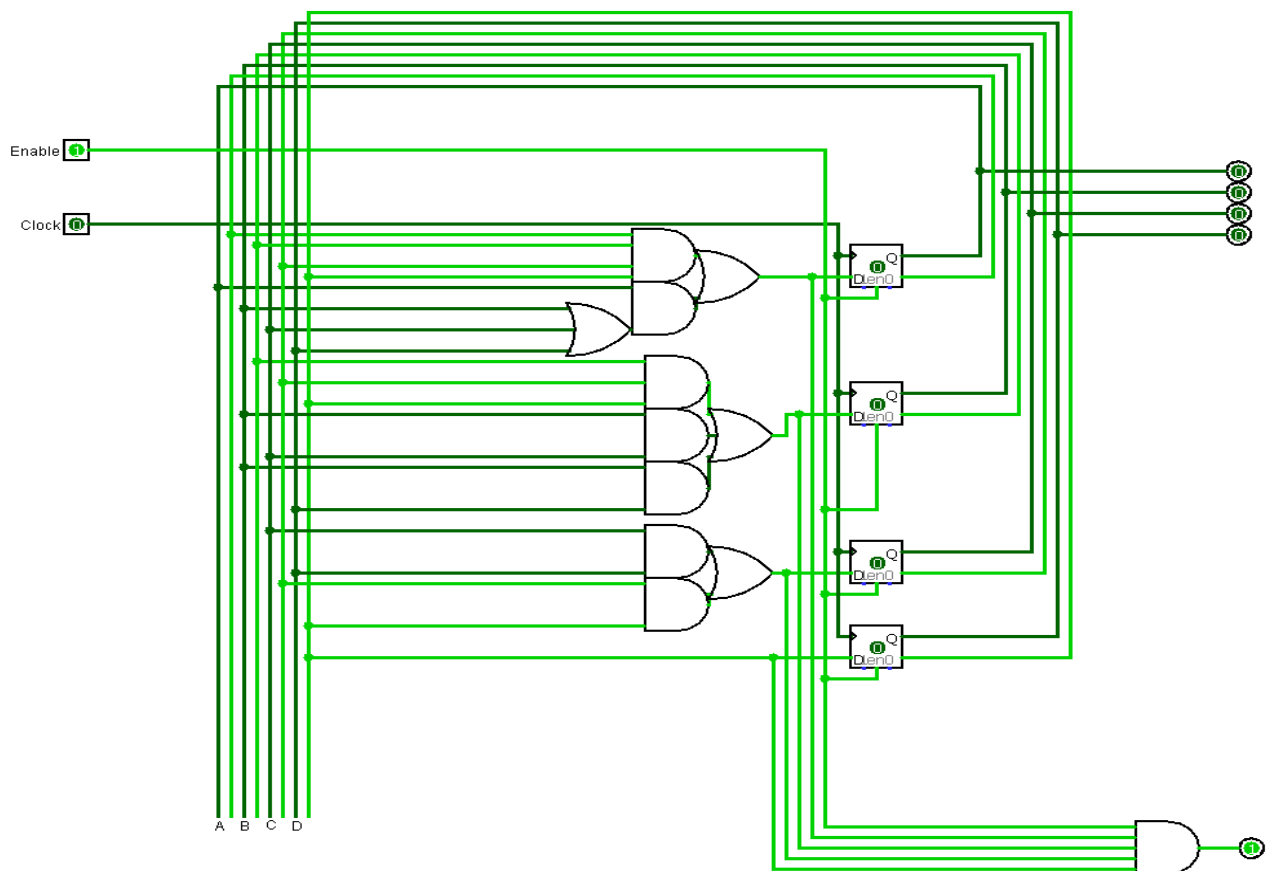


Figure 1.1

As you can see, we used **Divide and conquer approach** to this modulo 256 down-counter since we cannot humanly write the whole state table of a full modulo 256 down-counter and its equivalent state diagram and k-maps.

But we could approach it by stating that modulo 256 = $2^8 = 2^4 * 2^4$ = modulo 16 down-counter * modulo 16 down-counter, Which intern made the design process easier to understand and implement.

And resulted in the following circuit:

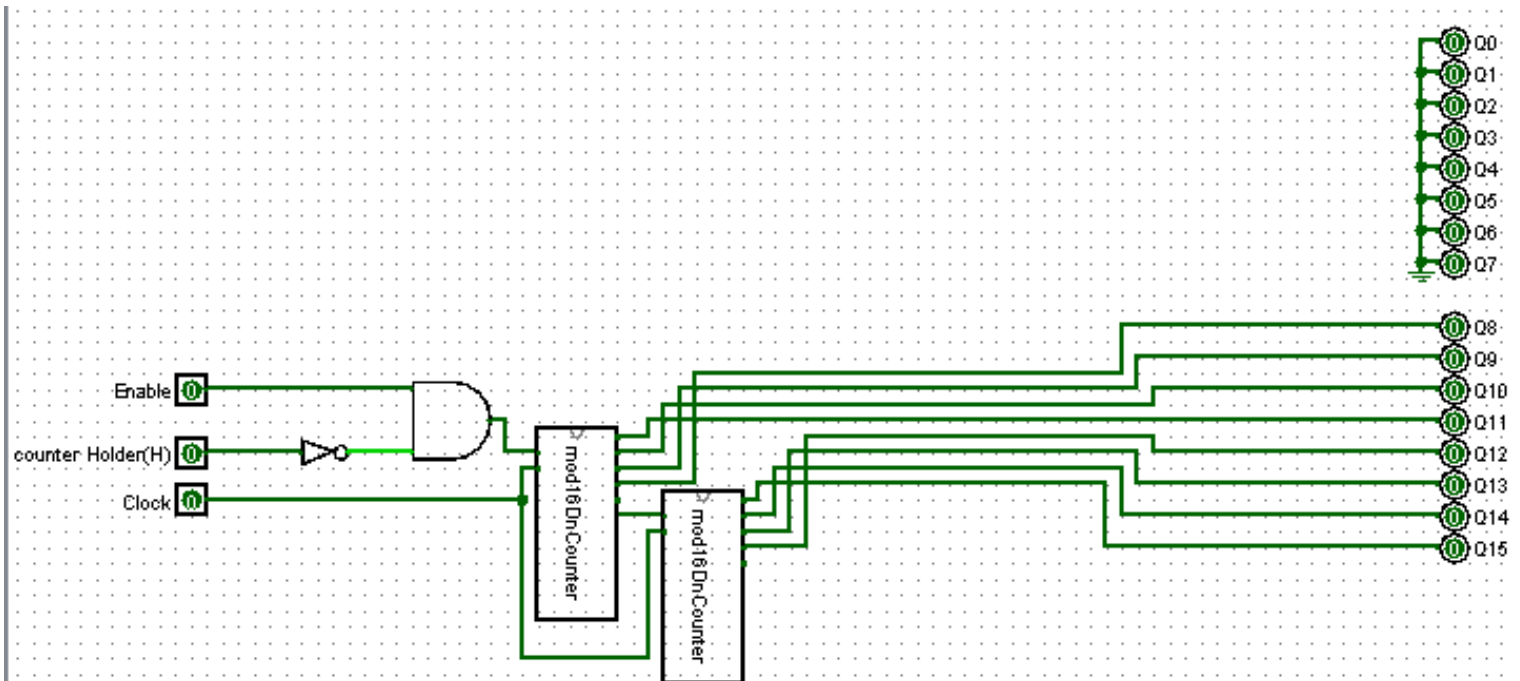


Figure 1.1.2

Verilog Code:

```

1  module counter3(clk,hold,counter2);
2      input clk,hold;
3      reg [7:0] count;
4      output [15:0]counter2;
5      initial count =8'b11111111 ;
6
7      always@(posedge clk)
8      begin
9          if(hold)
10             count = count;
11          else
12             count = count - 1;
13      end
14      assign counter2 = count;
15
16  endmodule

```

Case 10:

The design circuit that it will give us a multiplication of both input using shift and add method:

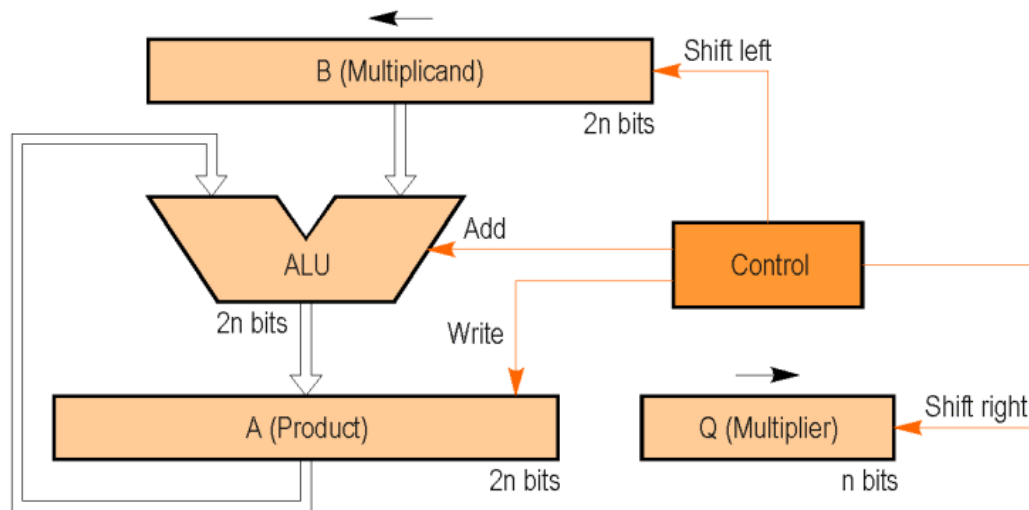


Figure 2.1

Multiplicand circuit:

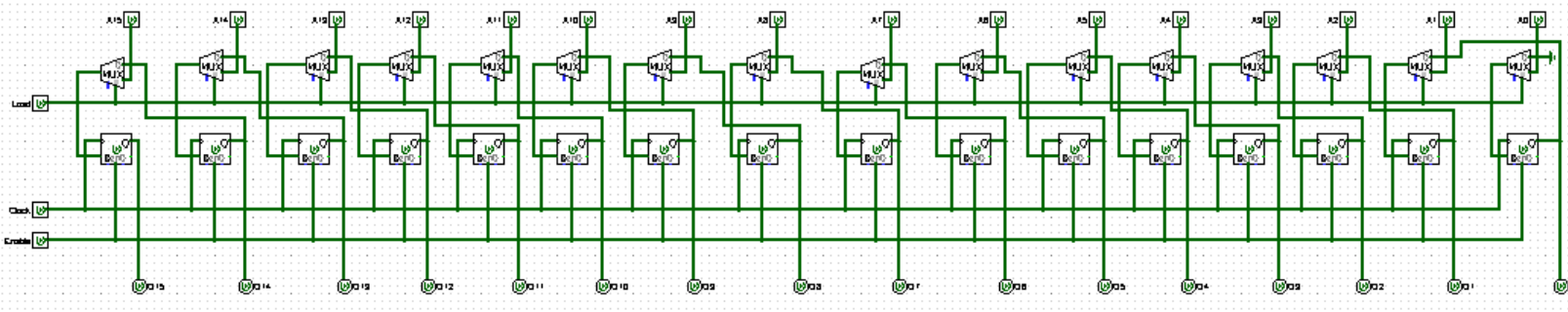


Figure 2.2

ALU circuit:

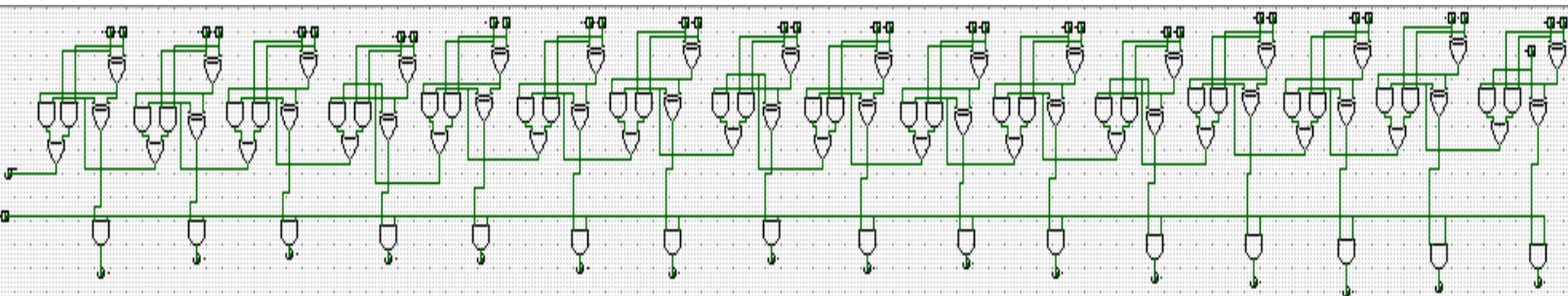


Figure 2.3

Product circuit:

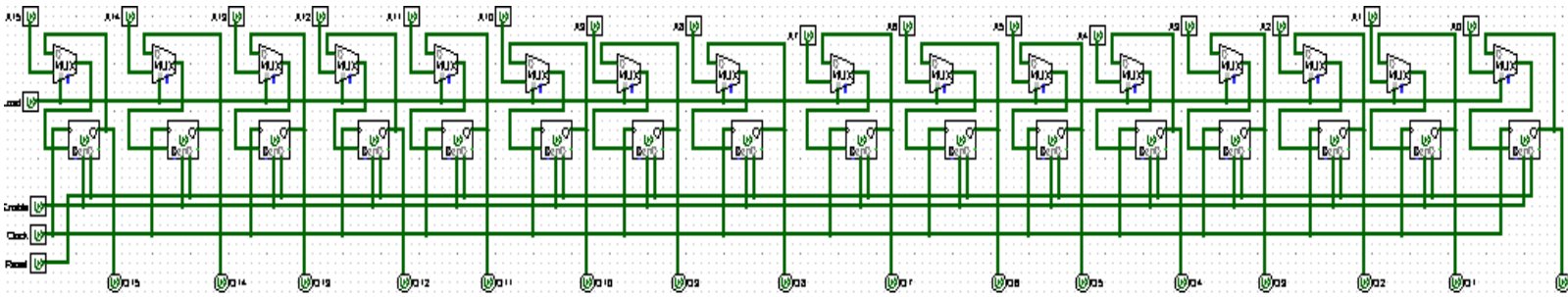


Figure 2.4

Multiplier circuit:

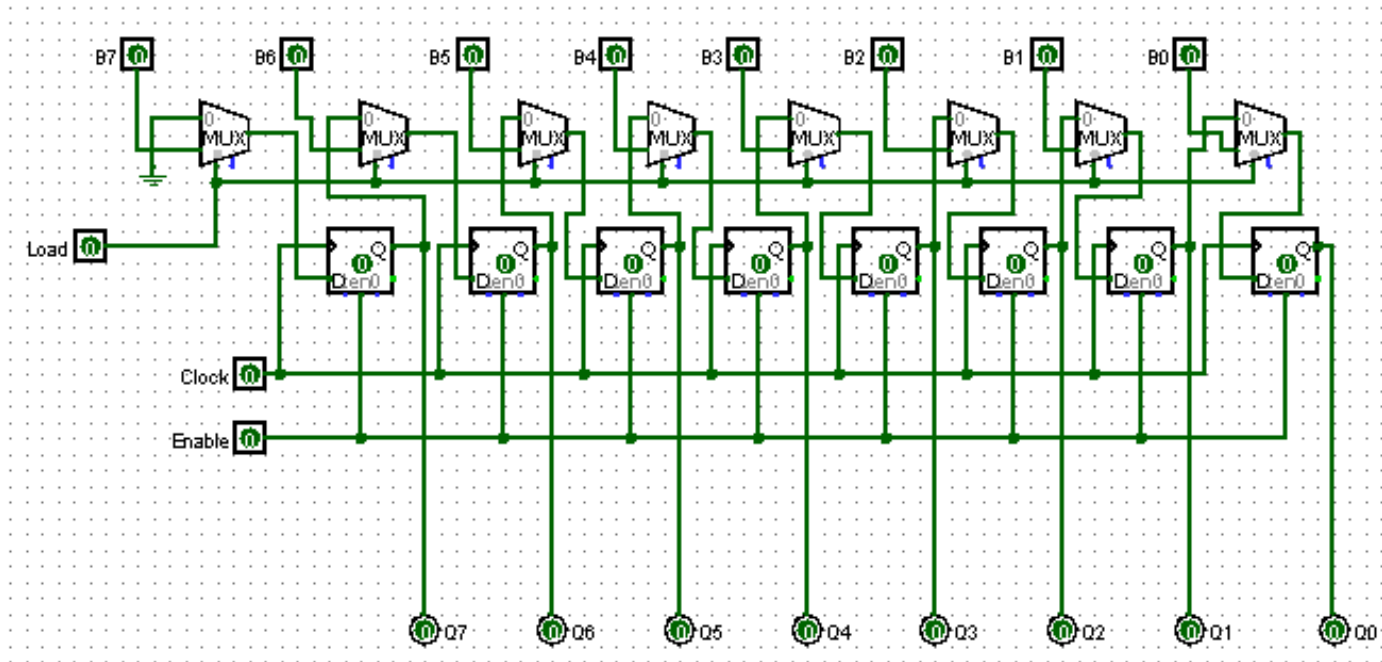


Figure 2.5

This is the final design circuit of multiply both input using shift and add method:

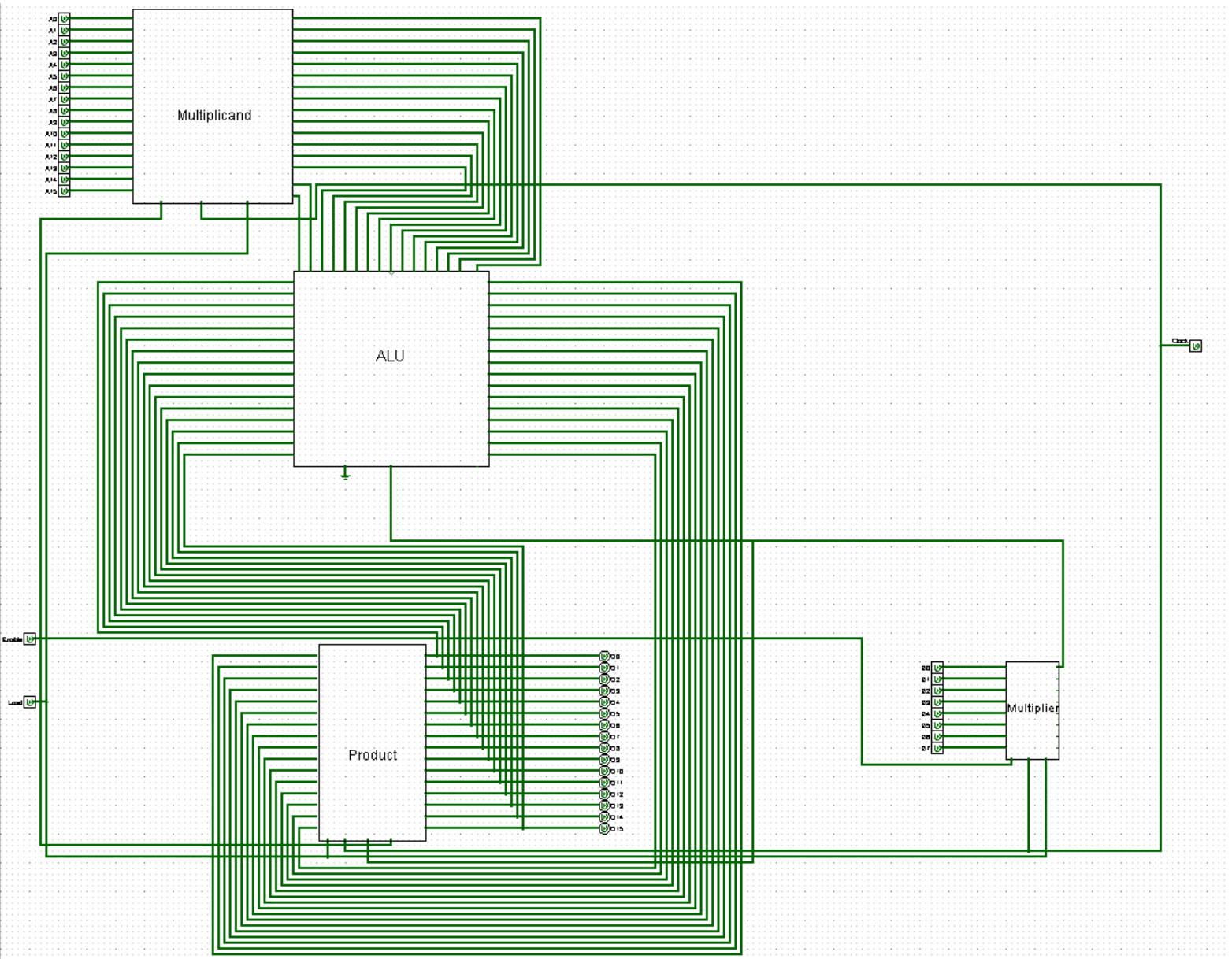


Figure 2.6

Verilog Code:

```
1  module shiftANDadd(product,multiplier,multiplicand);
2
3      input [7:0] multiplier, multiplicand;
4      output product;
5      reg [15:0] product;
6      reg c;
7      reg [7:0] m;
8      integer i;
9      always @( multiplier or multiplicand )
10         begin
11             product[15:8] = 8'd0;
12             product[7:0] = multiplier;
13             m = multiplicand;
14             c = 1'd0;
15             for(i=0; i<8; i=i+1)
16                 begin
17                     if(product[0])
18                         begin
19                             {c,product[15:8]} = product[15:8] + m ;
20                             //shift
21                             product[15:0] = {c,product[15:1]};
22                             c = 0;
23                         end
24                     else
25                         begin
26                             product[15:0] = {c,product[15:1]};
27                             c = 0;
28                         end
29                 end
30         end
31
32     end
33
34 endmodule
35
```

Case 11:

Even parity												
Number	N1	N2	N3	N4	O7	O6	O5	O4	O3	O2	O1	O0
1	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	1	0	0	0	0	0	0	1
3	0	1	0	0	0	1	0	0	0	0	0	1
4	1	1	0	0	1	1	0	0	0	0	0	0
5	0	0	1	0	0	0	1	0	0	0	0	1
6	1	0	1	0	1	0	1	0	0	0	0	0
7	0	1	1	0	0	1	1	0	0	0	0	0
8	1	1	1	0	1	1	1	0	0	0	0	1
9	0	0	0	1	0	0	0	1	0	0	0	1
10	1	0	0	1	1	0	0	1	0	0	0	0
11	0	1	0	1	0	1	0	1	0	0	0	0
12	1	1	0	1	1	1	0	1	0	0	0	1
13	0	0	1	1	0	0	1	1	0	0	0	0
14	1	0	1	1	1	0	1	1	0	0	0	1
15	0	1	1	1	0	1	1	1	0	0	0	1
16	1	1	1	1	1	1	1	1	0	0	0	0

Table 3.1

Design in Logisim:

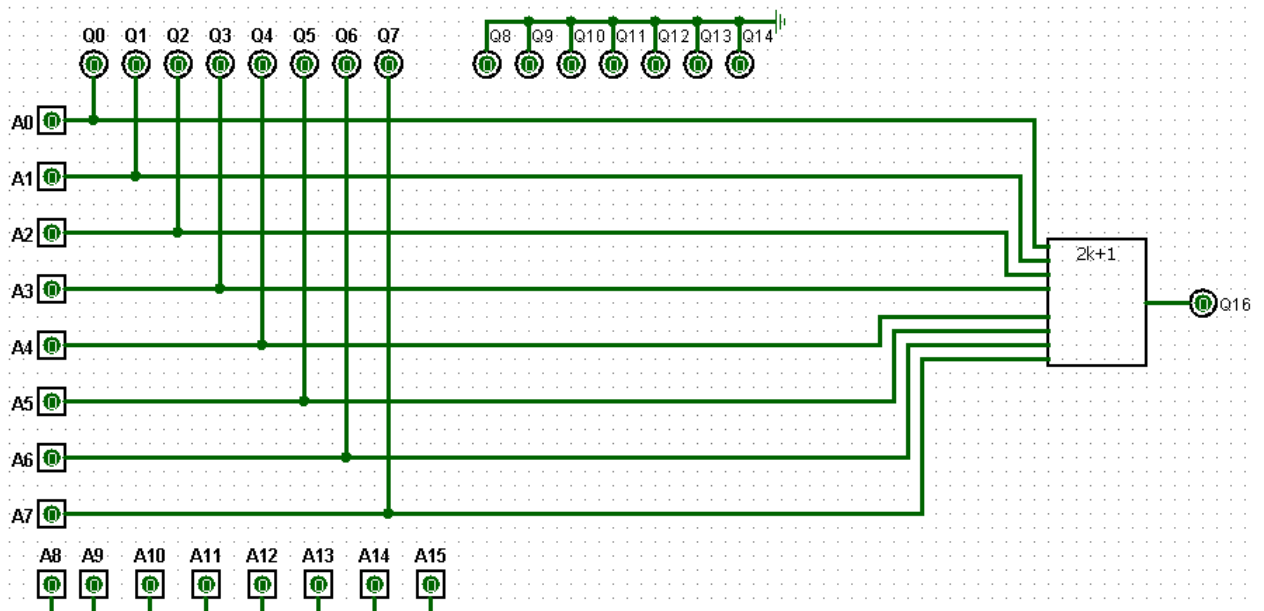


Figure 3.1

Block details:

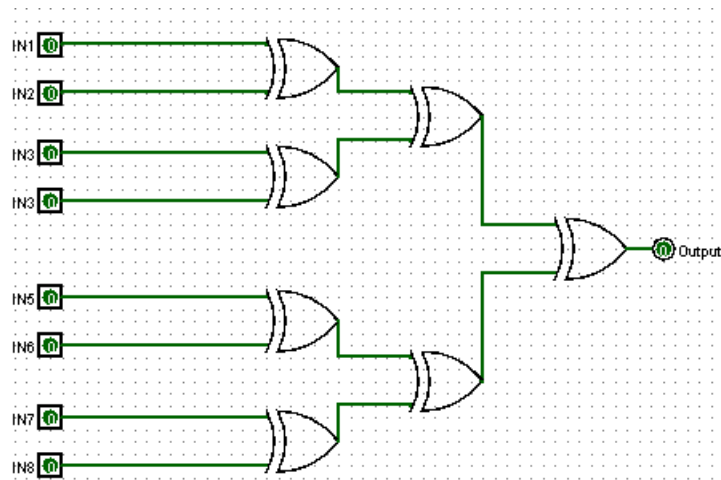


Figure 3.1.1

K-map:

N3N2 / N1N0	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	0	1	1	0
10	0	1	1	0

Table 3.2

$$O0 = N1 N0 + N1' N0 = N0 (N1 + N1') = N0$$

N3N2 / N1N0	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	0	0	1	1
10	0	0	1	1

Table 3.3

$$O1 = N1 N0 + N1 N0' = N1 (N0 + N0') = N1$$

N3N2 / N1N0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	1	1	1	1
10	0	0	0	0

Table 3.4

$$O2 = N3' N2 + N3 N2 = N2 (N3 + N3') = N2$$

N3N2 / N1N0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

Table 3.5

$$O3 = N3 N2 + N3 N2' = N3 (N2 + N2') = N3, \quad O4 - O6 = 0$$

N3N2 / N1N0	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

Table 3.6

$$O7 = N0 \otimes N1 \otimes N2 \otimes N3$$

Verilog Code:

```

1  module CASE11 (N,0) ;
2      input  N [7:0] ;
3      output O [15:0] ;
4      wire  C [6:0] ;
5
6      assign O[0] = N[0] ;
7      assign O[1] = N[1] ;
8      assign O[2] = N[2] ;
9      assign O[3] = N[3] ;
10     assign O[4] = N[4] ;
11     assign O[5] = N[5] ;
12     assign O[6] = N[6] ;
13     assign O[7] = N[7] ;
14
15     assign O[8] = 0 ;
16     assign O[9] = 0 ;
17     assign O[10] = 0 ;
18     assign O[11] = 0 ;
19     assign O[12] = 0 ;
20     assign O[13] = 0 ;
21     assign O[14] = 0 ;
22
23     assign C [0] = N[0]^N[1] ;
24     assign C [1] = N[2]^N[3] ;
25     assign C [2] = N[4]^N[5] ;
26     assign C [3] = N[6]^N[7] ;
27     assign C [4] = C[0]^C[1] ;
28     assign C [5] = C[2]^C[3] ;
29     assign C [6] = C[4]^C[5] ;
30
31     assign O[15] = C[6] ;
32
33 endmodule

```

The final design circuit of all the cases:

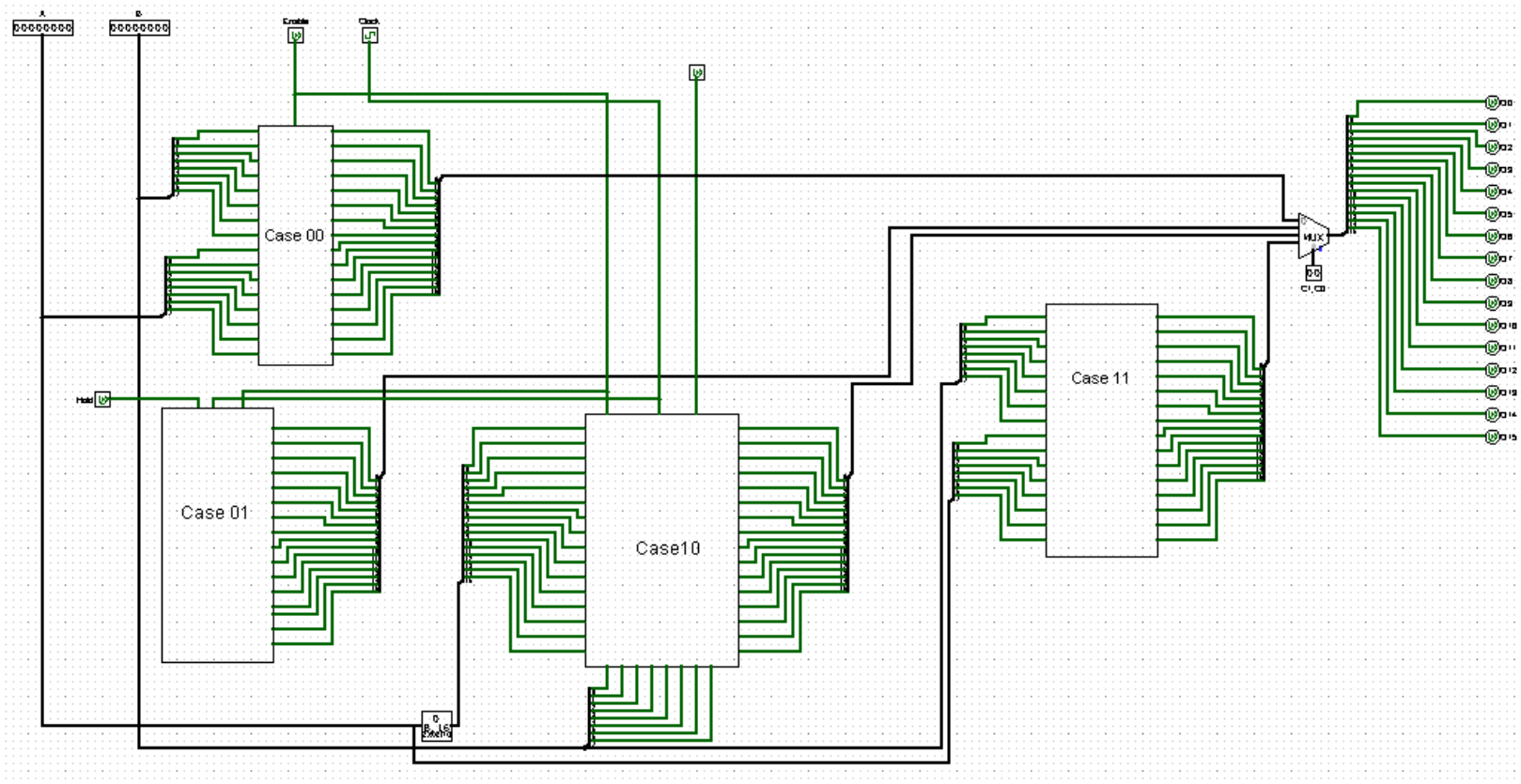


Figure 3.2

Verilog Code:

1)

```

1  //-----
2  //
3  // Title       : No Title
4  // Design      : project_vlsi
5  // Author      : Waheed alsaad
6  // Company     : Ksu
7  //
8  //-----
9  //
10 // File        : c:\My_Designs\project_vlsi\project_vlsi\compile\project_vlsi.v
11 // Generated    : Thu Mar 17 14:45:45 2022
12 // From        : c:\My_Designs\project_vlsi\project_vlsi\src\project_vlsi.bde
13 // By          : Bde2Verilog ver. 2.01
14 //
15 //-----
16 //
17 // Description :
18 //
19 //-----
20
21 `ifndef _VCP
22 `else
23 `define library(a,b)
24 `endif
25
26
27 // ----- Design Unit Header ----- //
28 `timescale 1ps / 1ps
29
30 module project_vlsi (c0,c1,hold,A,B,q,clk) ;
31
32 // ----- Port declarations ----- //
33 input c0;
34 wire c0;
35 input c1;
36 wire c1;
37 input hold;
38 wire hold;
39 input [7:0] A;
40 wire [7:0] A;
41 input [7:0] B;
42 wire [7:0] B;
43 output [15:0] q;

```

2)

```

44 wire [15:0] q;
45 input clk;
46 wire clk;
47
48 // ----- Signal declarations ----- //
49 wire [15:0] BUS110;
50 wire [15:0] BUS114;
51 wire [15:0] BUS705;
52 wire [15:0] BUS83;
53
54 // ----- Component instantiations -----//
55
56 counter3 U13
57 (
58     .clk(clk),
59     .hold(hold),
60     .counter2(BUS83)
61 );
62
63
64
65 mux4t01 U15
66 (
67     .a(BUS705),
68     .b(BUS83),
69     .c(BUS110),
70     .d(BUS114),
71     .s0(c0),
72     .s1(c1),
73     .out(q)
74 );
75
76
77
78 even_parity U3
79 (
80     .N(B),
81     .O(BUS114)
82 );
83
84
85

```

3)

```

86 inverter U4
87 (
88     .a(A),
89     .b(B),
90     .q(BUS705)
91 );
92
93
94
95 shiftANDadd U5
96 (
97     .product(BUS110),
98     .multiplier(A),
99     .multiplicand(B)
100 );
101
102
103
104 endmodule
105

```

4) Also, the Mux 4-1 code:

```

1 module mux4t01 (a, b, c, d,s0,s1 , out);
2     input [15:0]a ;
3     input [15:0]b ;
4     input [15:0]c ;
5     input [15:0]d ;
6     input s0 ,s1 ;
7     output [15:0]out ;
8     assign out[0] = s1 ? (s0 ? d[0] : c[0]) : (s0 ? b[0] : a[0]);
9     assign out[1] = s1 ? (s0 ? d[1] : c[1]) : (s0 ? b[1] : a[1]);
10    assign out[2] = s1 ? (s0 ? d[2] : c[2]) : (s0 ? b[2] : a[2]);
11    assign out[3] = s1 ? (s0 ? d[3] : c[3]) : (s0 ? b[3] : a[3]);
12    assign out[4] = s1 ? (s0 ? d[4] : c[4]) : (s0 ? b[4] : a[4]);
13    assign out[5] = s1 ? (s0 ? d[5] : c[5]) : (s0 ? b[5] : a[5]);
14    assign out[6] = s1 ? (s0 ? d[6] : c[6]) : (s0 ? b[6] : a[6]);
15    assign out[7] = s1 ? (s0 ? d[7] : c[7]) : (s0 ? b[7] : a[7]);
16    assign out[8] = s1 ? (s0 ? d[8] : c[8]) : (s0 ? b[8] : a[8]);
17    assign out[9] = s1 ? (s0 ? d[9] : c[9]) : (s0 ? b[9] : a[9]);
18    assign out[10] = s1 ? (s0 ? d[10] : c[10]) : (s0 ? b[10] : a[10]);
19    assign out[11] = s1 ? (s0 ? d[11] : c[11]) : (s0 ? b[11] : a[11]);
20    assign out[12] = s1 ? (s0 ? d[12] : c[12]) : (s0 ? b[12] : a[12]);
21    assign out[13] = s1 ? (s0 ? d[13] : c[13]) : (s0 ? b[13] : a[13]);
22    assign out[14] = s1 ? (s0 ? d[14] : c[14]) : (s0 ? b[14] : a[14]);
23    assign out[15] = s1 ? (s0 ? d[15] : c[15]) : (s0 ? b[15] : a[15]);
24 endmodule

```

Testing the designed circuit:

Case 00:

Input A, will give us an output $Q = A'$

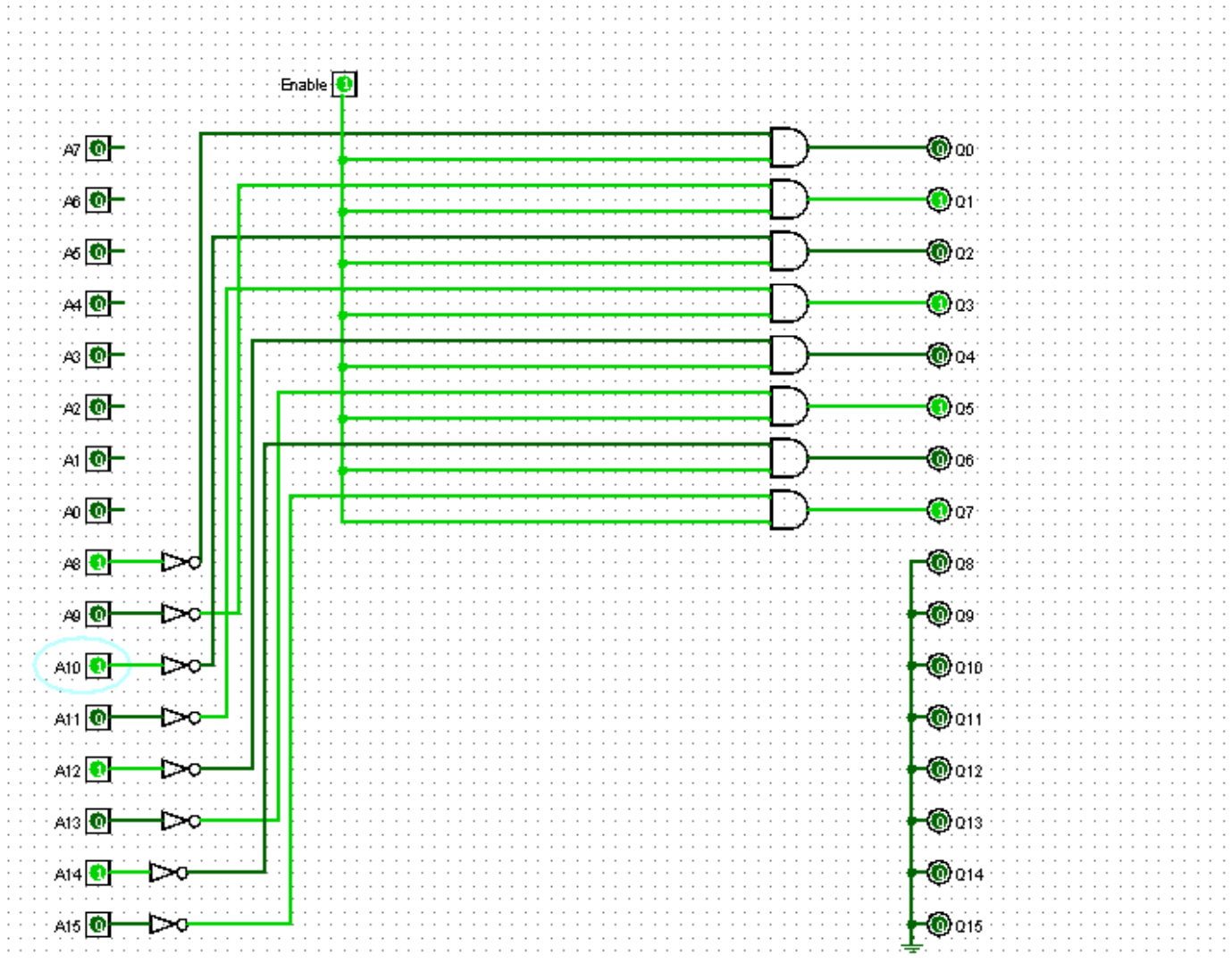


Figure 4.1

Case 01:

a 4-bit counter initially start with 256:

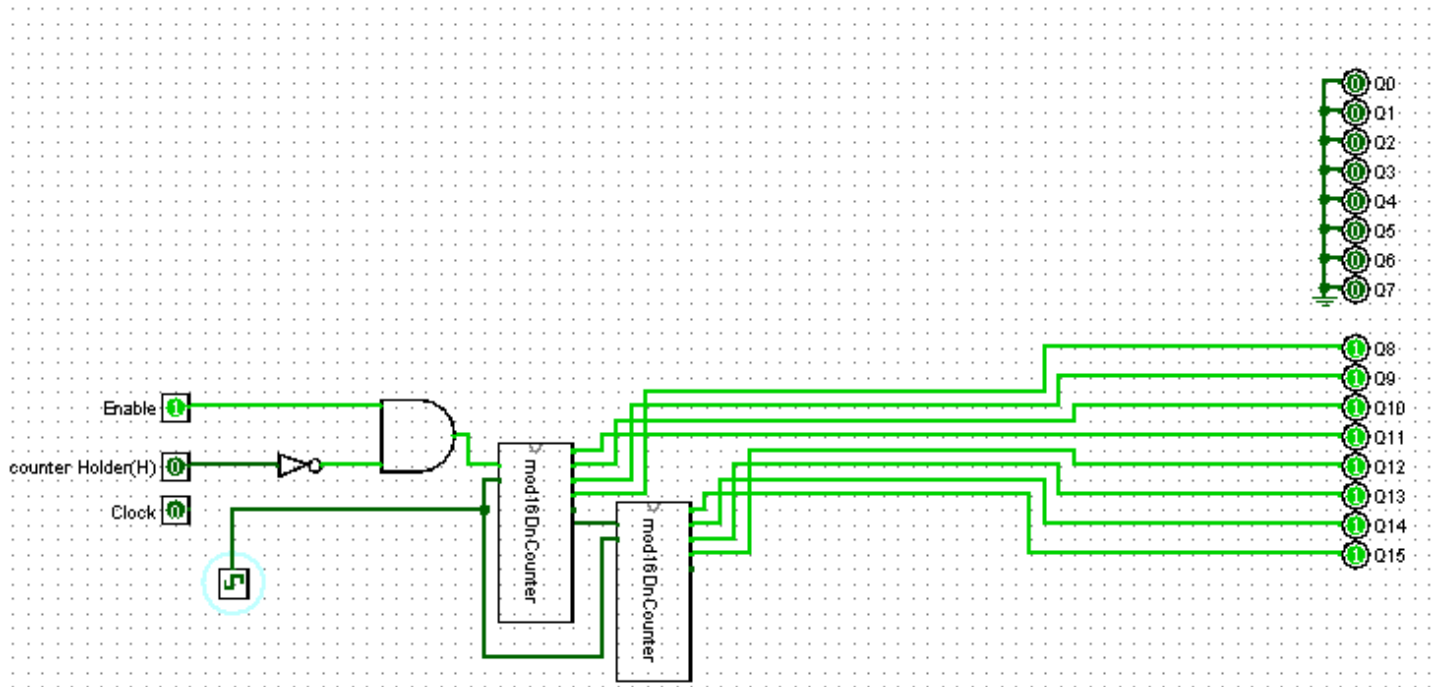


Figure 4.2

Now, if we pressed hold button without clock:

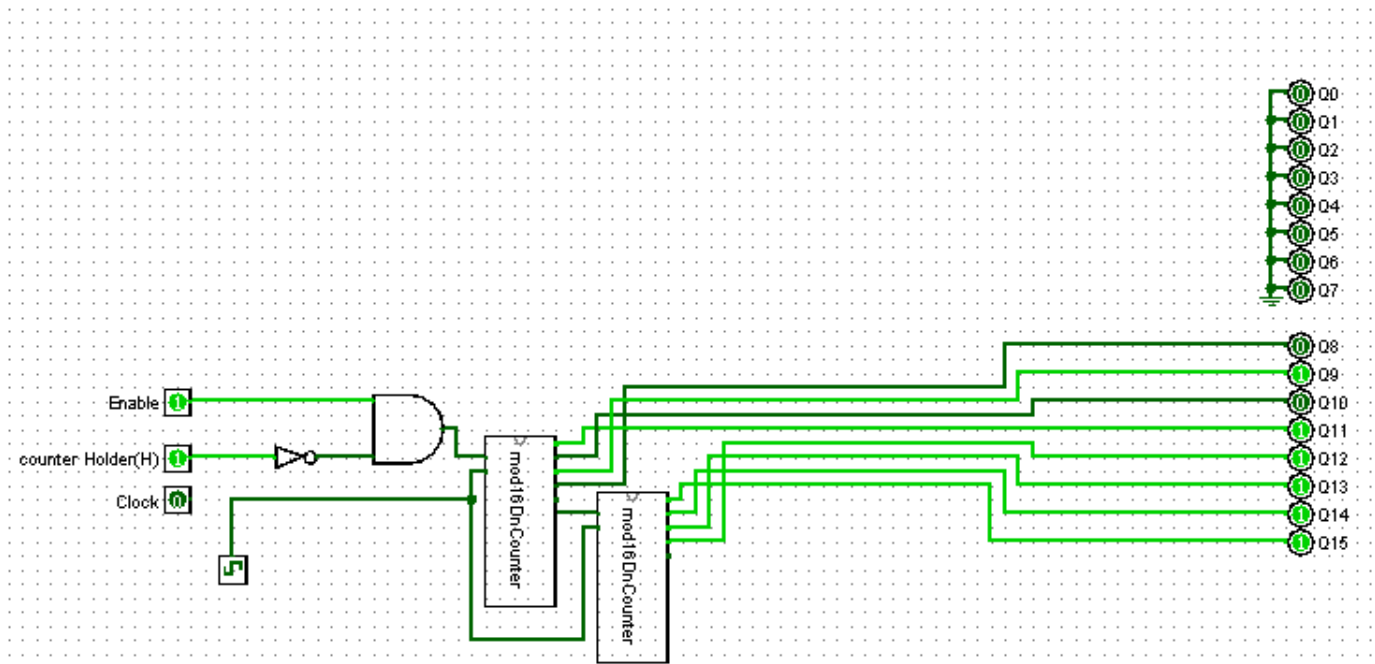


Figure 4.2.1

Now, the same case but with clock:

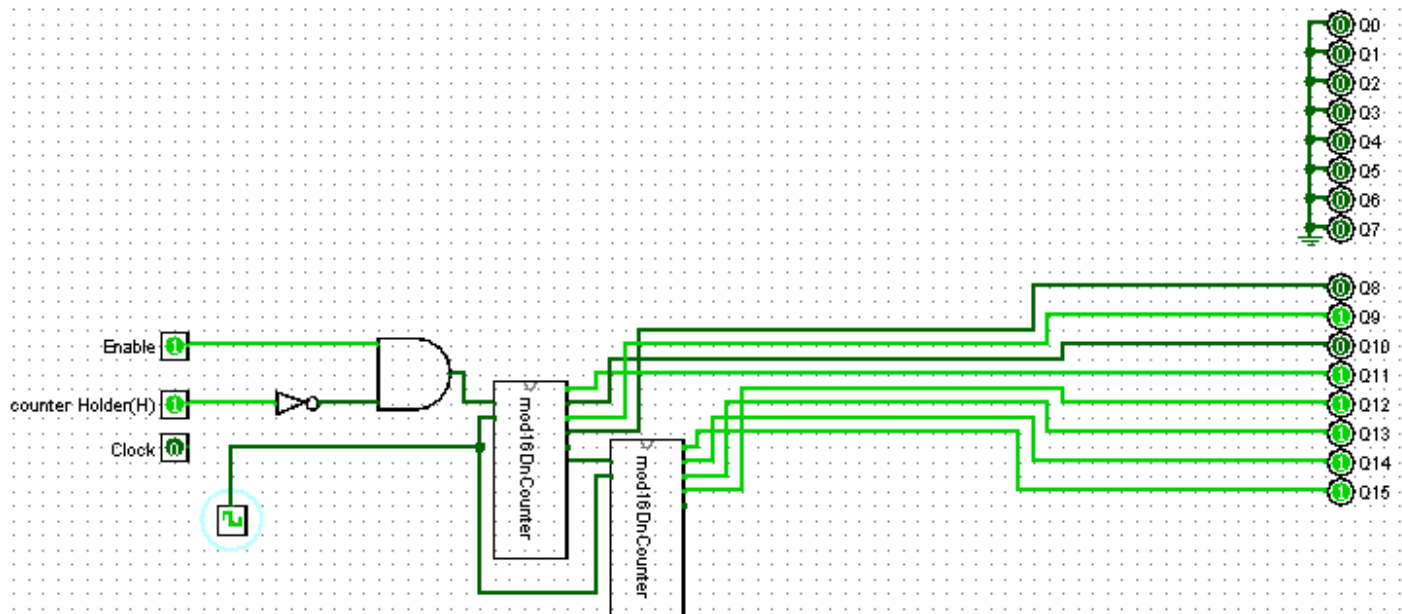


Figure 4.2.2

Case 10:

At $A = 101$, and $B = 100000$

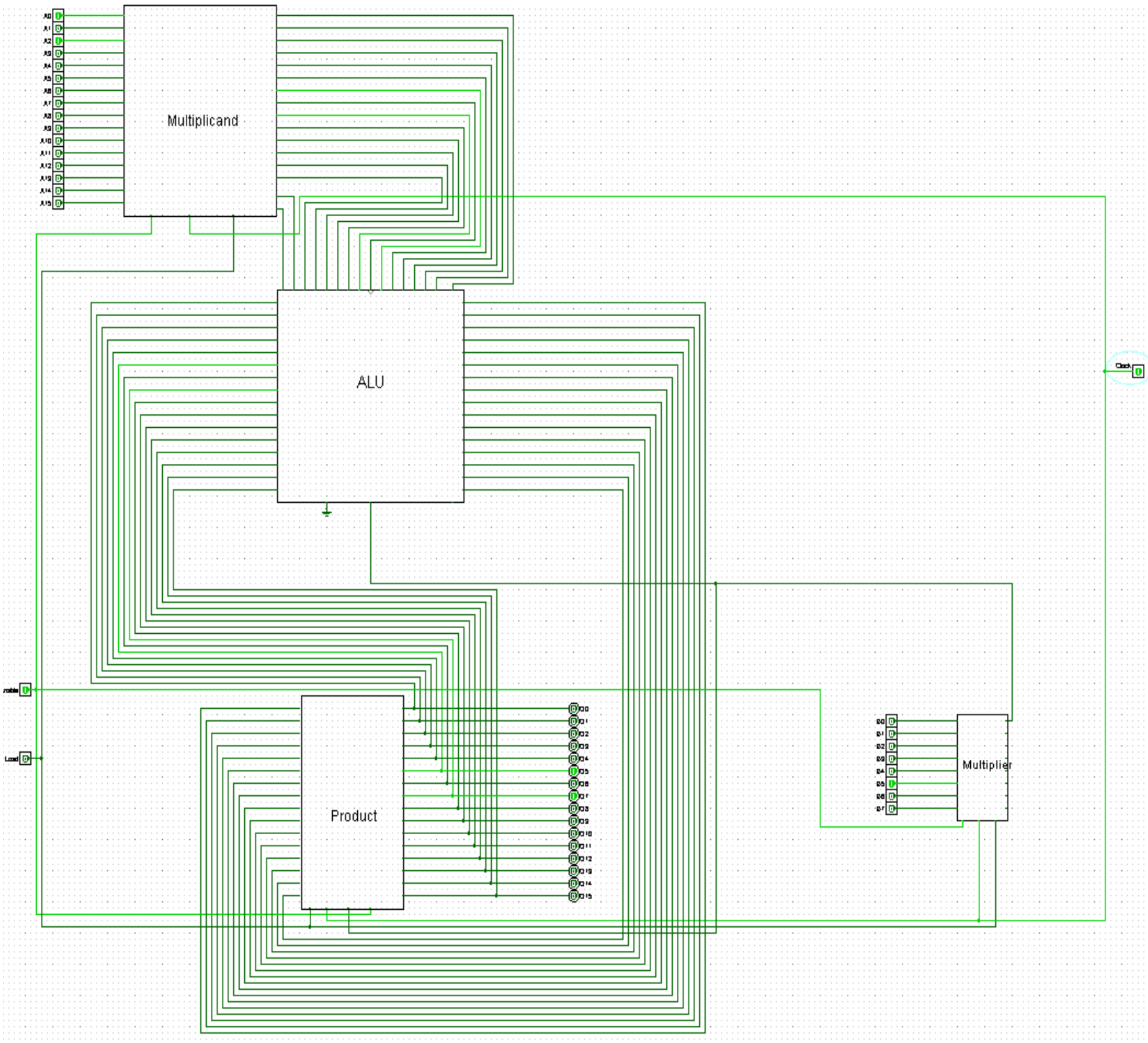


Figure 4.3

The output Q will be: 10100000

Case 11:

At $A = 00110101$, the output will be 0, because the input 1's is even:

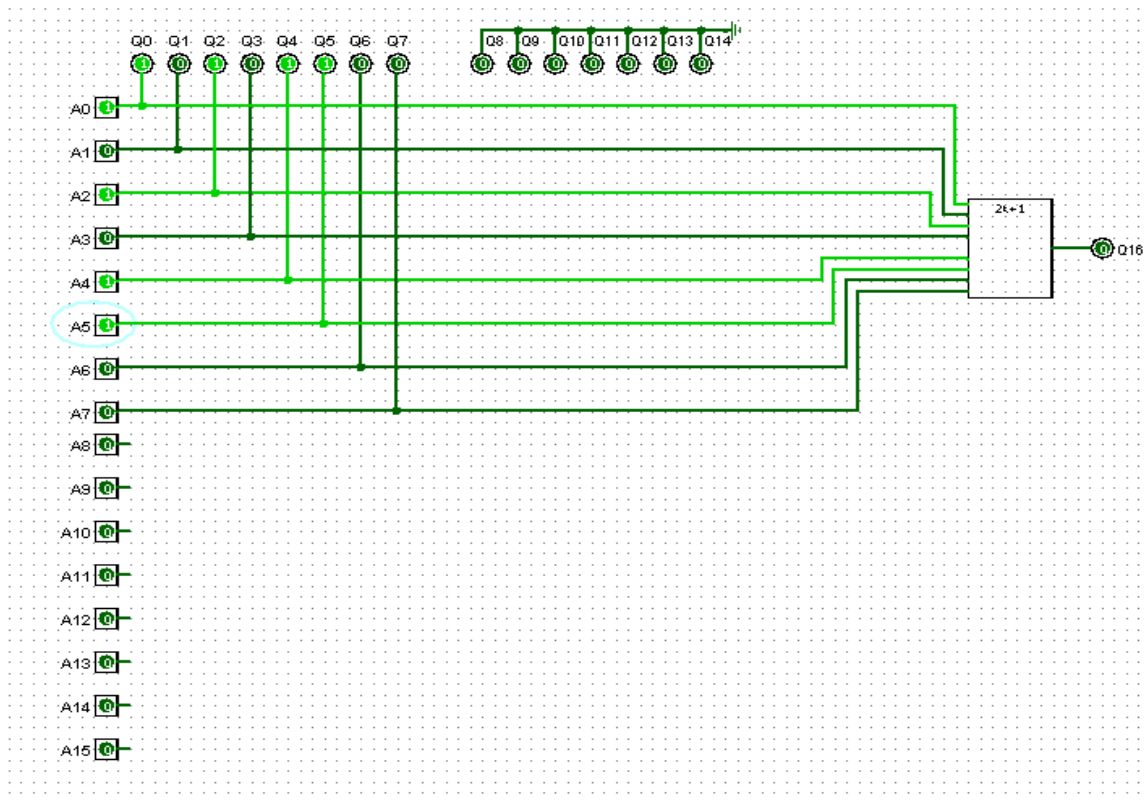


Figure 4.4

At $A = 00100101$, the output will be 1, because the input 1's is odd:

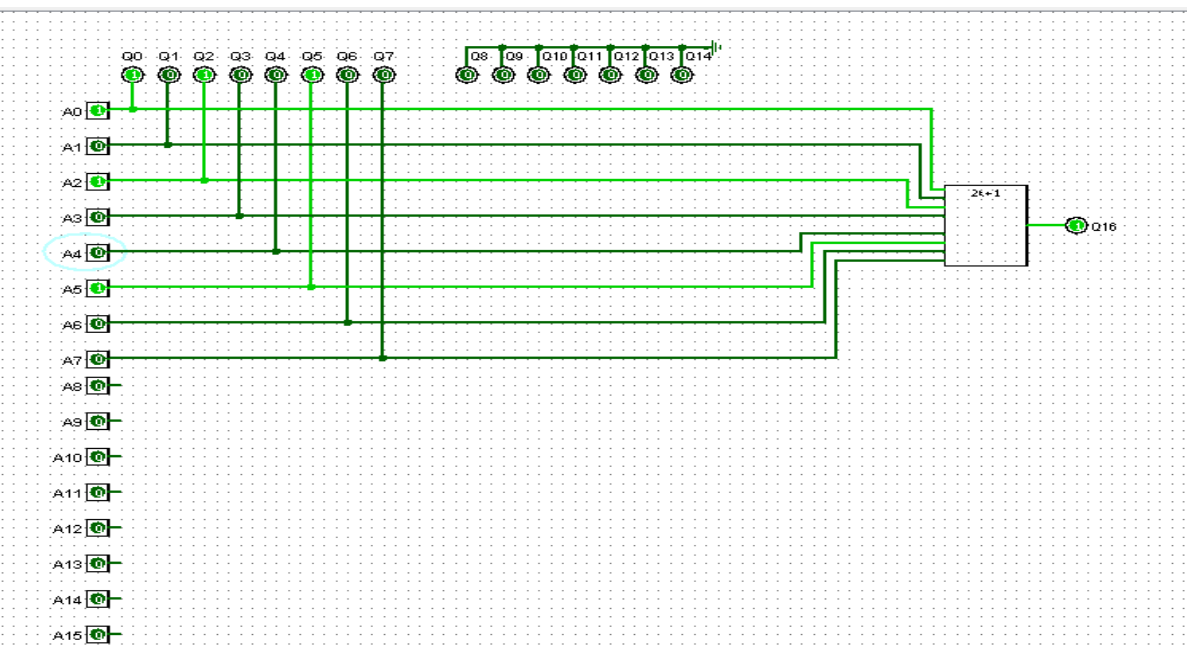


Figure 4.4.1

All cases:

Case 00: A = 01000010, and B = 00100000

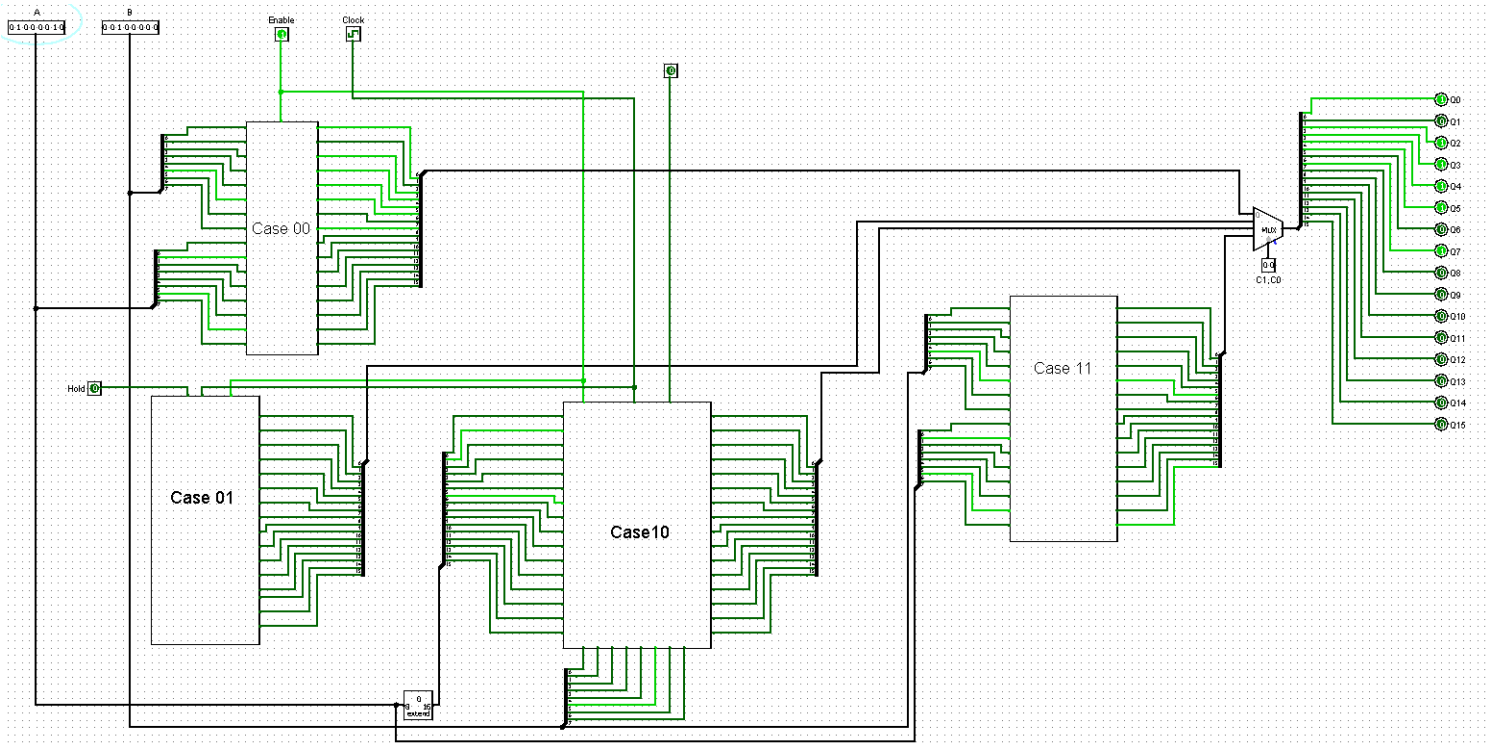


Figure 4.5

Case 01: A = 01000010, and B = 00100000

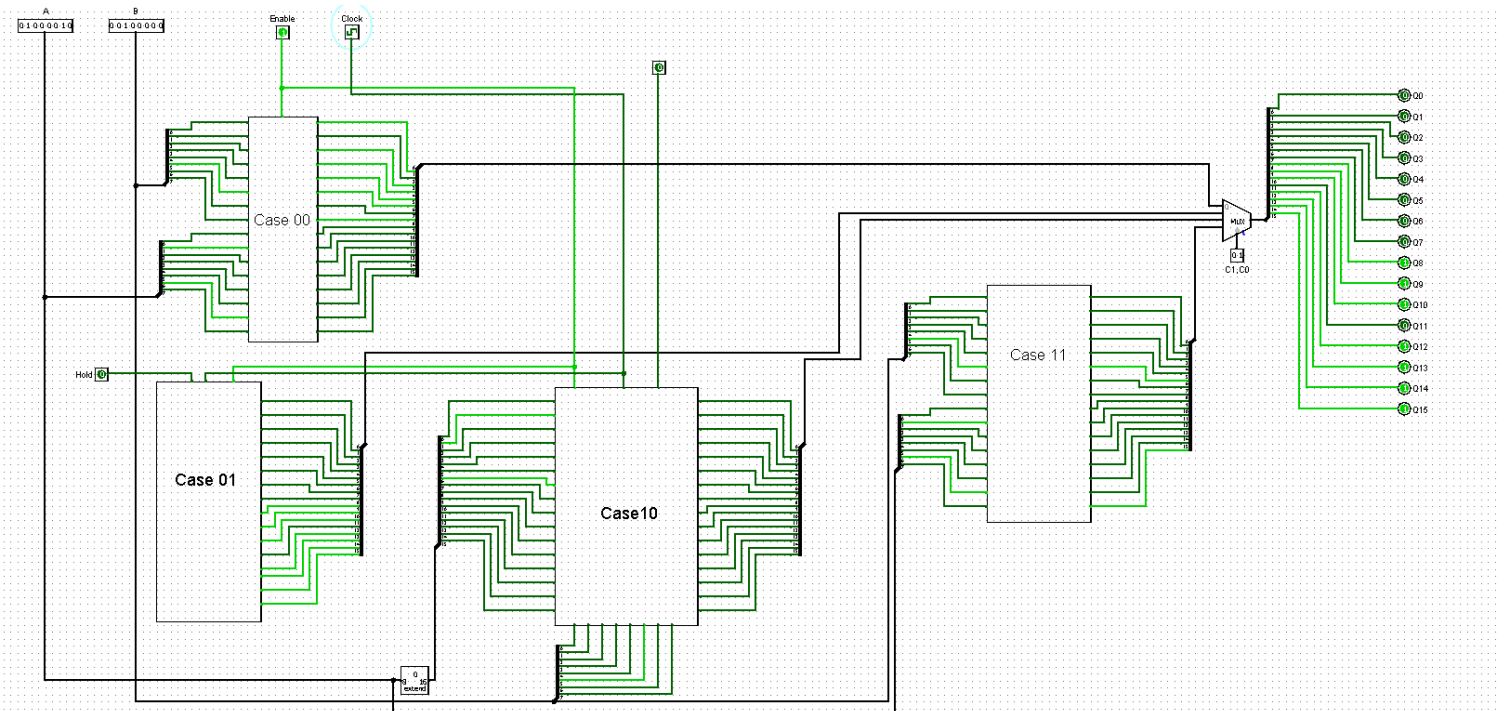


Figure 4.6

Case 10: A = 01000010, and B = 00100000

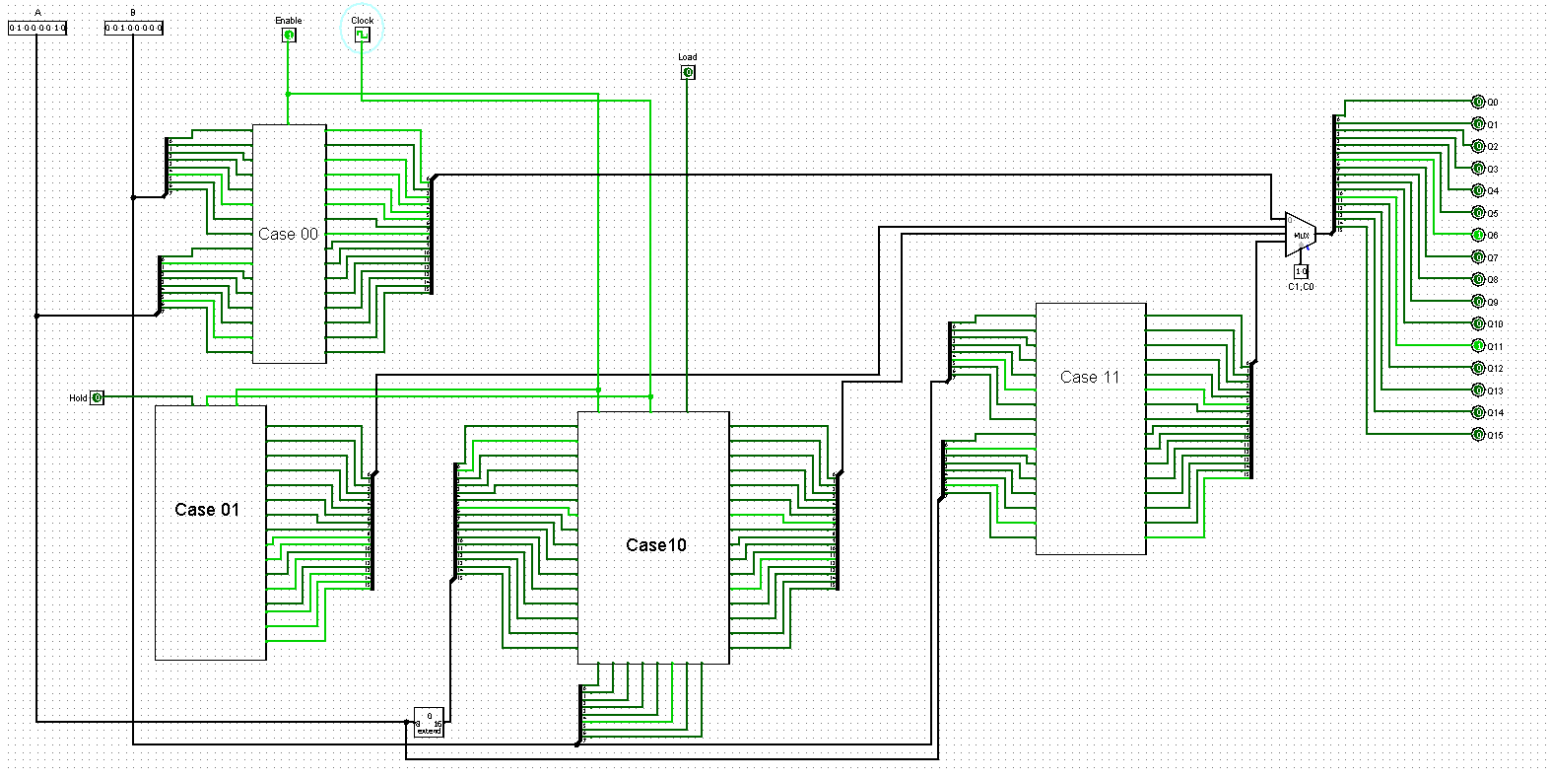


Figure 4.7

Case 11: A = 01000010, and B = 00100000

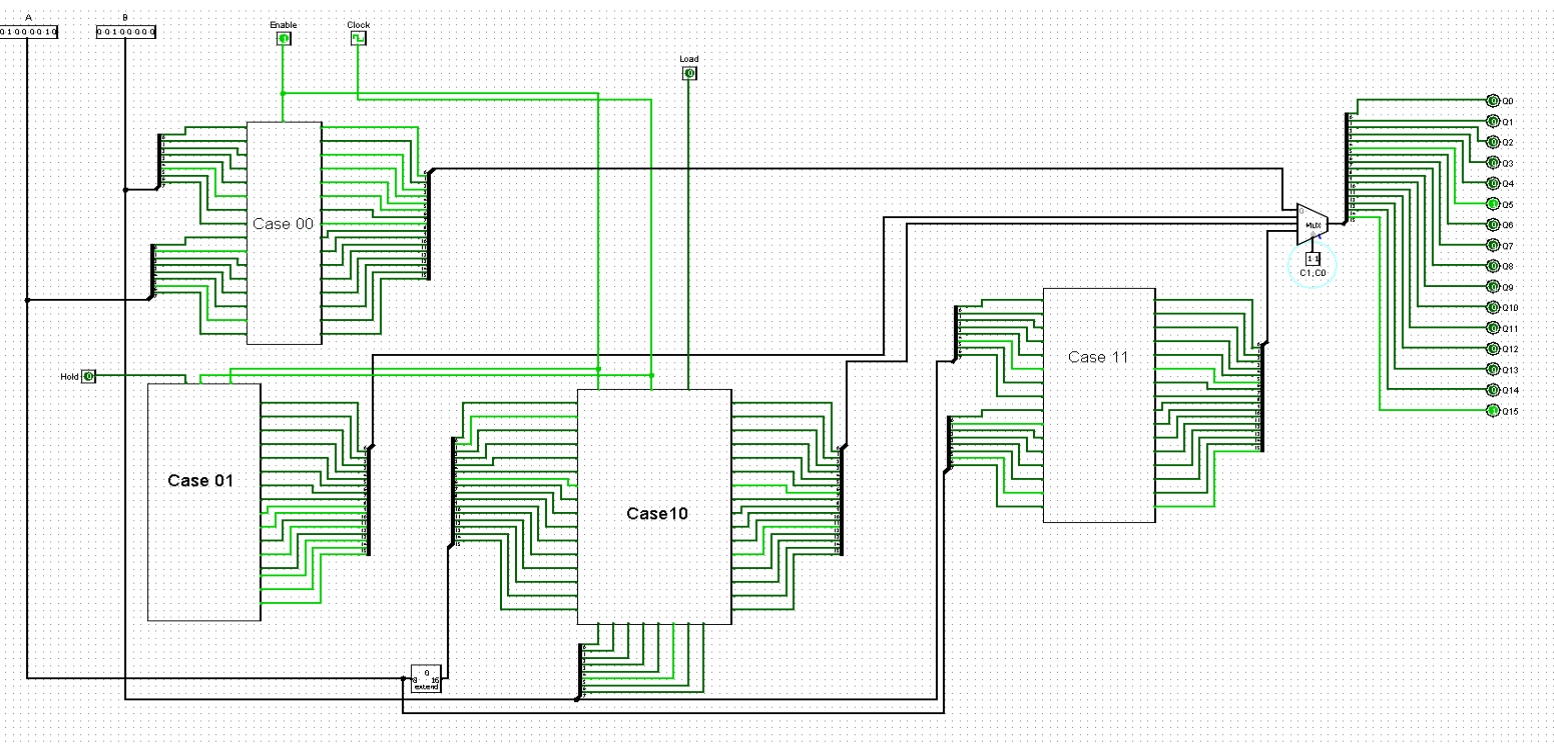


Figure 4.8

Testing the code of the circuit:

Case 00: inverting the input

1)

Signal name	Value	400
<input checked="" type="checkbox"/> a	1, 1, 1, 1, 1, 1, 1, 1	0 fs
<input checked="" type="checkbox"/> q	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0	

2)

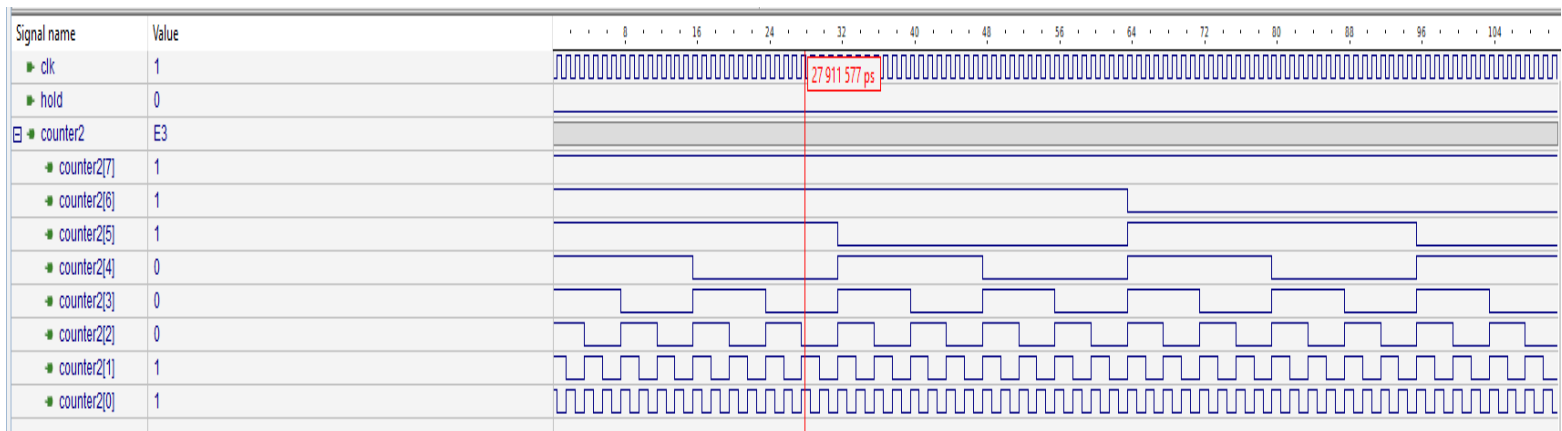
Signal name	Value	0 fs
<input checked="" type="checkbox"/> a	0, 1, 0, 1, 0, 1, 0, 1	
<input checked="" type="checkbox"/> q	0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0	

3)

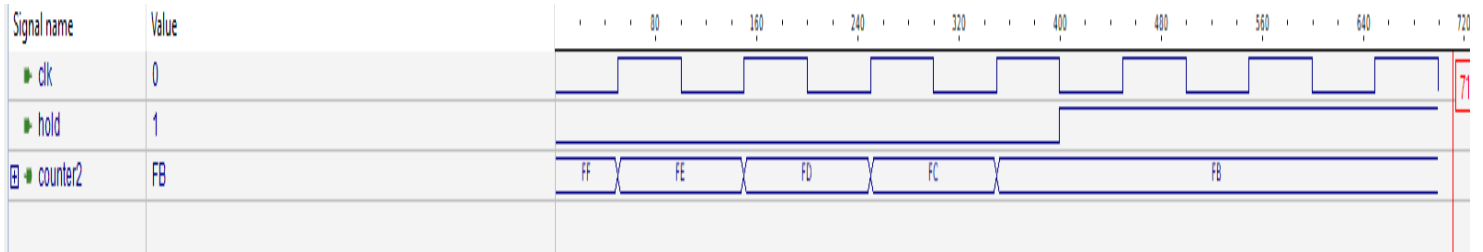
Signal name	Value	
<input checked="" type="checkbox"/> a	0, 0, 0, 0, 1, 1, 1, 1	
<input checked="" type="checkbox"/> q	0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0	

Case 01: Counter

Hold = 0



Hold = 1, after 400ns



Case 10: Shift, and add

1)

product	0E10	
product[15]	0	
product[14]	0	
product[13]	0	
product[12]	0	
product[11]	1	
product[10]	1	
product[9]	1	
product[8]	0	
product[7]	0	
product[6]	0	
product[5]	0	
product[4]	1	
product[3]	0	
product[2]	0	
product[1]	0	
product[0]	0	
multiplier	0F	
multiplicand	F0	

2)

product	FE01	0 fs
product[15]	1	
product[14]	1	
product[13]	1	
product[12]	1	
product[11]	1	
product[10]	1	
product[9]	1	
product[8]	0	
product[7]	0	
product[6]	0	
product[5]	0	
product[4]	0	
product[3]	0	
product[2]	0	
product[1]	0	
product[0]	1	
multiplier	FF	
multiplicand	FF	

3)

Signal name	Value
product	3872
product[15]	0
product[14]	0
product[13]	1
product[12]	1
product[11]	1
product[10]	0
product[9]	0
product[8]	0
product[7]	0
product[6]	1
product[5]	1
product[4]	1
product[3]	0
product[2]	0
product[1]	1
product[0]	0
multiplier	AA
multiplicand	55

Case 11: Even parity

1)

Signal name	Value
<input type="checkbox"/> N	1, 1, 1, 1, 0, 0, 0, 0
<input type="checkbox"/> O	0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0

2)

Signal name	Value
<input type="checkbox"/> N	1, 1, 0, 1, 0, 0, 0, 0
<input type="checkbox"/> O	1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0

3)

Signal name	Value
<input type="checkbox"/> N	1, 1, 1, 1, 1, 1, 1, 0
<input type="checkbox"/> O	1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0

Testing All the cases:

If C0 = 0, and C1 = 0

Signal name	Value	
➤ c0	0	0 fs
➤ c1	0	
☐ ➤ A	AA	
➤ A[7]	1	
➤ A[6]	0	
➤ A[5]	1	
➤ A[4]	0	
➤ A[3]	1	
➤ A[2]	0	
➤ A[1]	1	
➤ A[0]	0	
☐ ➤ B	zz	
➤ clk	z	
☐ ➤ q	0055	
➤ q[15]	0	
➤ q[14]	0	
➤ q[13]	0	
➤ q[12]	0	
➤ q[11]	0	
➤ q[10]	0	
➤ q[9]	0	
➤ q[8]	0	
➤ q[7]	0	
➤ q[6]	1	
➤ q[5]	0	
➤ q[4]	1	
➤ q[3]	0	
➤ q[2]	1	
➤ q[1]	0	
➤ q[0]	1	

If C0 = 1, and C1 = 0



























Signal name	Value	
➤ c0	1	0 fs
➤ c1	0	
☐ ➤ A	zz	
☐ ➤ B	zz	
☐ ➤ q	00FA	
➤ clk	0	
➤ hold	1	

If C0 = 0, and C1 = 1

[-] A	40	0 fs
[-] A[7]	0	
[-] A[6]	1	
[-] A[5]	0	
[-] A[4]	0	
[-] A[3]	0	
[-] A[2]	0	
[-] A[1]	0	
[-] A[0]	0	
[-] B	04	
[-] B[7]	0	
[-] B[6]	0	
[-] B[5]	0	
[-] B[4]	0	
[-] B[3]	0	
[-] B[2]	1	
[-] B[1]	0	
[-] B[0]	0	
[+] q	0100	

If C0 = 1, and C1 = 1

Signal name	Value
[-] c0	1
[-] c1	1
[+] A	zz
[-] B	F0
[-] B[7]	1
[-] B[6]	1
[-] B[5]	1
[-] B[4]	1
[-] B[3]	0
[-] B[2]	0
[-] B[1]	0
[-] B[0]	0
[-] q	00F0
[-] q[15]	0
[-] q[14]	0
[-] q[13]	0
[-] q[12]	0
[-] q[11]	0
[-] q[10]	0
[-] q[9]	0
[-] q[8]	0
[-] q[7]	1
[-] q[6]	1
[-] q[5]	1
[-] q[4]	1
[-] q[3]	0
[-] q[2]	0
[-] q[1]	0
[-] q[0]	0
[-] clk	z
[-] hold	z

 B	E0	
 B[7]	1	
 B[6]	1	
 B[5]	1	
 B[4]	0	
 B[3]	0	
 B[2]	0	
 B[1]	0	
 B[0]	0	
 q	80E0	
 q[15]	1	
 q[14]	0	
 q[13]	0	
 q[12]	0	
 q[11]	0	
 q[10]	0	
 q[9]	0	
 q[8]	0	
 q[7]	1	
 q[6]	1	
 q[5]	1	
 q[4]	0	
 q[3]	0	
 q[2]	0	
 q[1]	0	
 q[0]	0	