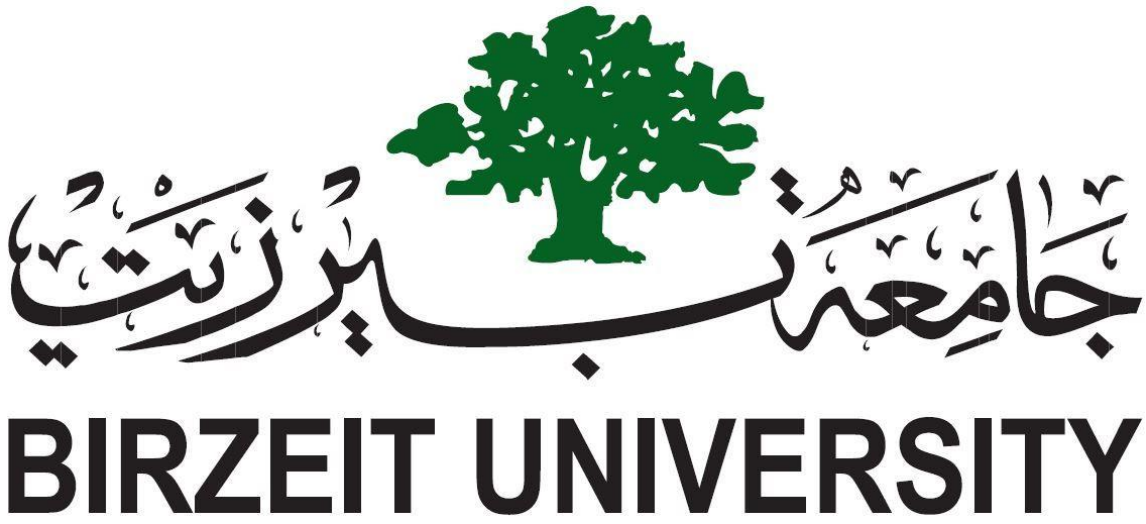


Summer2022

Digital Systems ENCS2340

Verilog HDL Project

Dr.Ayman Hroub

Section (3)

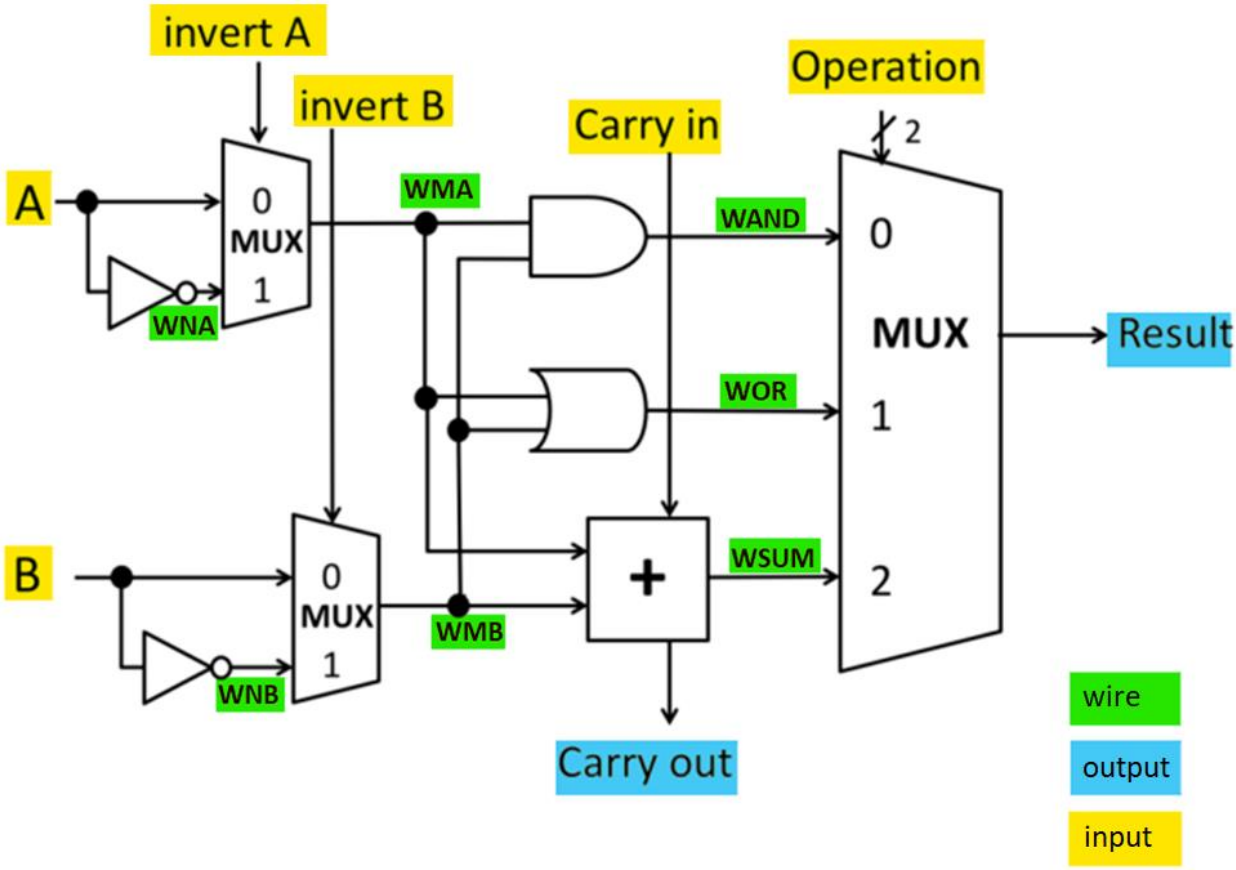
QOSSAY RIDA (1211553)

Contents:

Definition for 1-bit ALU :

(Arithmetic Logic Unit) ALU is a digital function that implements the micro-operation on the information stored in registers , ALU is a fundamental building block of the many varieties of computing circuits, including the central processing unit (CPU) of computers and graphics processing units (GPUs). one CPU or GPU may contain multiple ALUs.

Figure for 1-bit ALU circuit :



Component :

This circuit has a four component (1: Full Adder , 2: Mux2*1 , 1: Mux4*1) and has And gate , Or gate

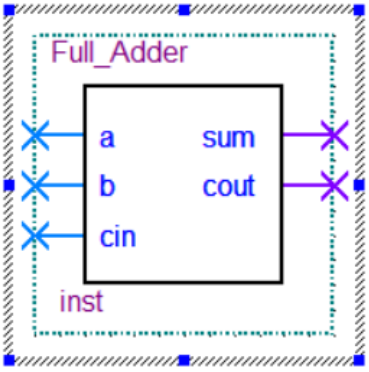
1- Full Adder :

A full adder is a digital circuit to adds 1-bit plus 1-bit plus 1-bit and produces sum and carry .

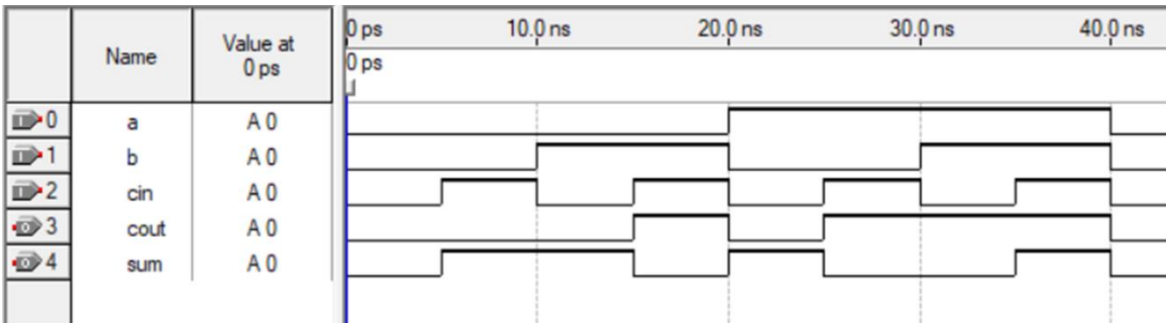
Code by Verilog for Full Adder :

```
1  module Full_Adder (a,b,cin,sum,cout);
2  input a,b,cin;
3  output sum ,cout;
4  assign {cout,sum}= a + b + cin ;
5  endmodule
```

The Block diagram for Full Adder :



The WaveForm for Full Adder :



The Truth table for Full Adder :

	a	b	cin	By WaveForm cout	By WaveForm F	By experience cout	By experience F
1	0	0	0	0	0	0	0
2	0	0	1	0	1	0	1
3	0	1	0	0	1	0	1
4	0	1	1	1	0	1	0
5	1	0	0	0	1	0	1
6	1	0	1	1	0	1	0
7	1	1	0	1	0	1	0
8	1	1	1	1	1	1	1

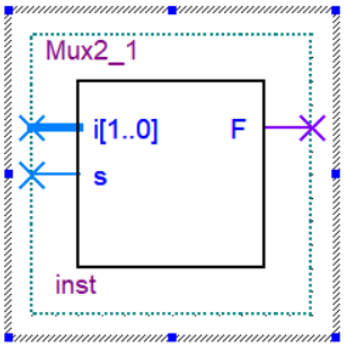
2- Mux 2-1 :

A 2-to-1 multiplexer consists of two inputs i0 and i1, one select input s and one output F. Depending on the select signal, the output is connected to either of the inputs , if s=0 then the output be i0 else the output be i1 .

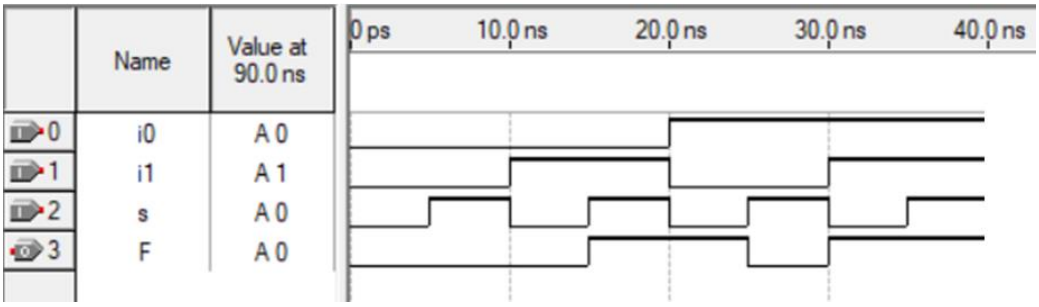
Code by Verilog for Mux 2-1 :

```
1  module Mux2_1 (i0,i1,s,F);
2  input i0,i1,s;
3  output F ;
4  assign F = (s == 0 )? i0 : i1;
5  endmodule
```

The Block diagram for Mux 2-1:



The WaveForm for Mux 2-1:



The Truth table for Mux 2-1:

By WaveForm

By experience

	i0	i1	s	F	F
1	0	x	0	0	0
2	1	x	0	1	1
3	x	0	1	0	0
4	x	1	1	1	1

3- Mux 4-1 :

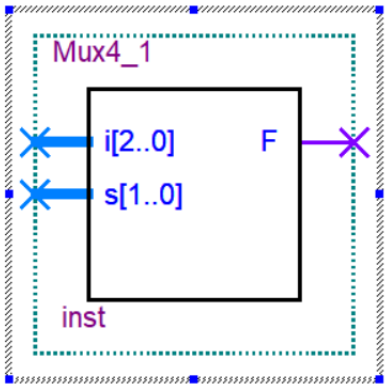
A 4-to-1 multiplexer consists four data input lines as i0 to i3, two select lines as s0 and s1 and a single output line F. The select lines S0 and S1 select one of the four input lines to connect the output line , if ({s1,s0}=00) the output be i0 , else if ({s1,s0}=01) the output be i1 , else if ({s1,s0}=10) the output be i2 , else the output be i3 .

NOTE : We will not use i3 in this circuit .

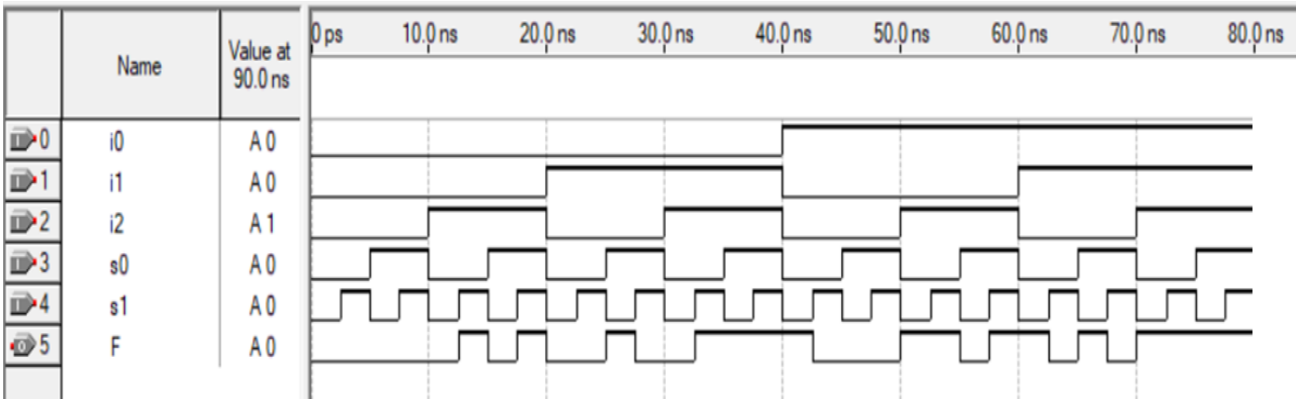
Code by Verilog for Mux 4-1 :

```
1  module Mux4_1 (i0,i1,i2,s0,s1,F);
2   input i0,i1,i2,s0,s1;
3   output reg F ;
4  always @(*) begin
5   if      ({s1,s0} == 'b00) F=i0;
6   else if ({s1,s0} == 'b01) F=i1;
7   else if ({s1,s0} == 'b10) F=i2;
8   end
9   endmodule
```

The Block diagram for Mux 4-1:



The WaveForm for Mux 4-1:



The Truth table for Mux 2-1:

By WaveForm

By experience

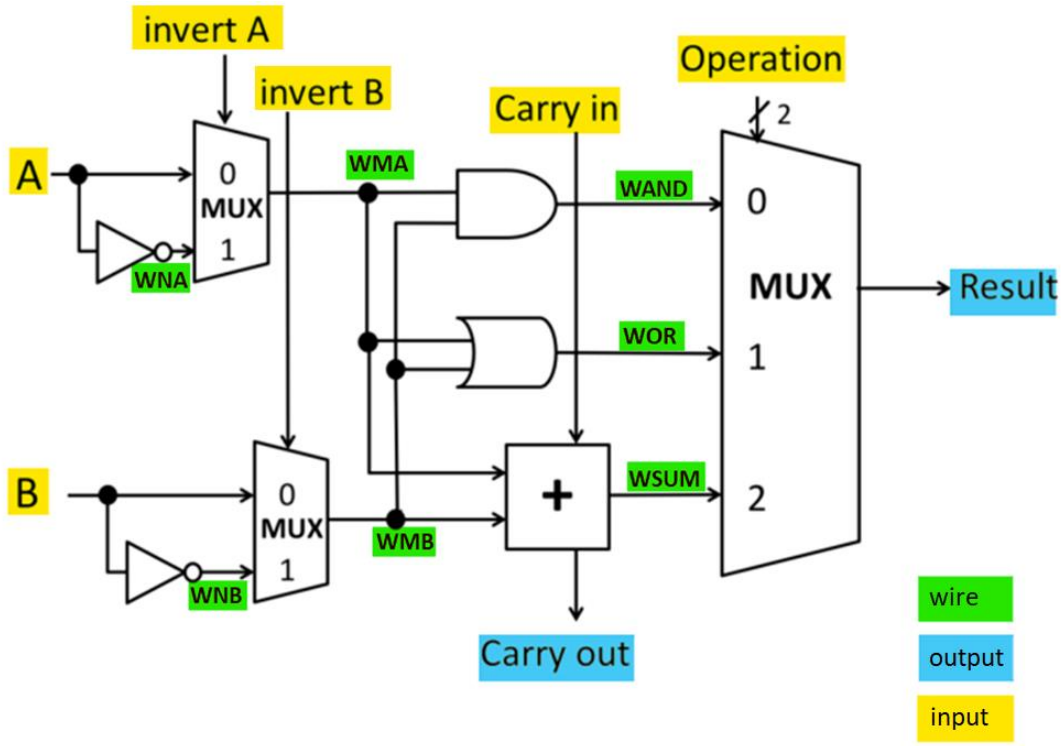
	i0	i1	i2	s0	s1	F	F
1	0	x	x	0	0	0	0
2	1	x	x	0	0	1	1
3	x	0	x	0	1	0	0
4	x	1	x	0	1	1	1
5	x	x	0	1	0	0	0
6	x	x	1	1	0	1	1
7	x	x	x	1	1	X	X

whole system:

This ALU will do Three operation (00, 01 , 10)

(11) operation is invalid operation

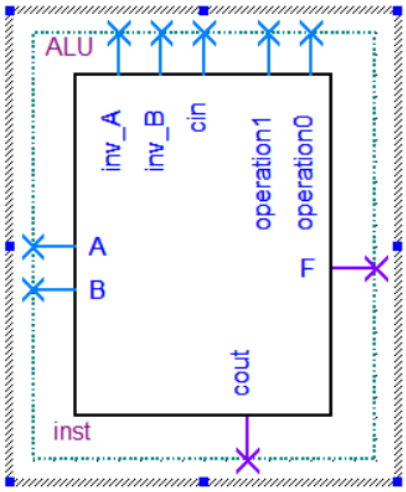
Figure for 1-bit ALU circuit:



The code for The 1-bit ALU:

```
1  module ALU (A,B,cin,inv_A,inv_B,operation0,operation1,cout,F);
2  input A,B,cin,inv_A,inv_B,operation0,operation1 ;
3  output cout,F ;
4  wire WNA,WNB,WMA,WMB,WAND,WOR,WSUM ;
5  not (WNA,A),
6      (WNB,B);
7  Mux2_1 (A,WNA,inv_A,WMA);
8  Mux2_1 (B,WNB,inv_B,WMB);
9  and (WAND,WMA,WMB);
10 or  (WOR,WMA,WMB);
11 Full_Adder (WMA,WMB,cin,WSUM,cout);
12 Mux4_1 (WAND,WOR,WSUM,operation0,operation1,F);
13 endmodule
```

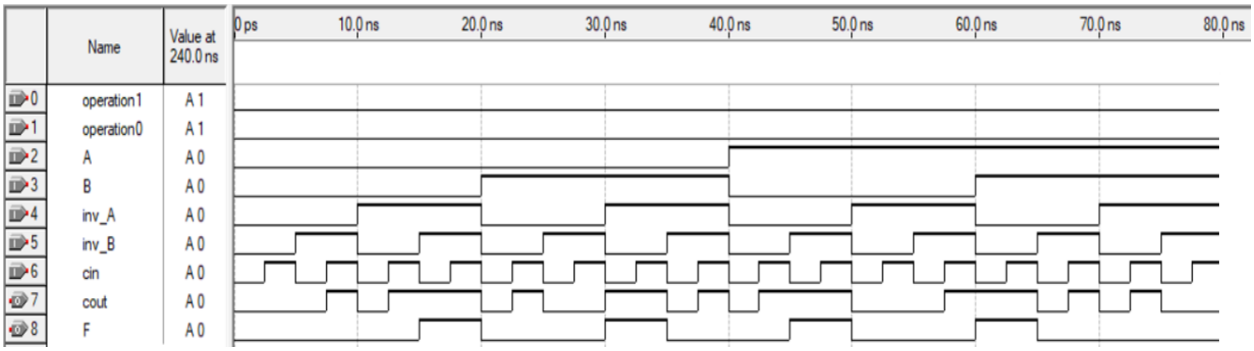
The block diagram for ALU:



This ALU has [1:0] operation :

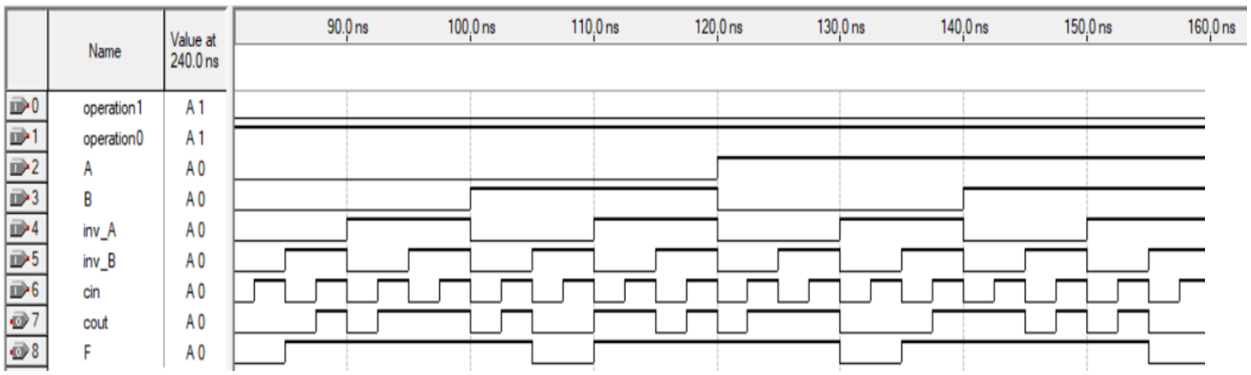
- **When the user enters the operation (00):**

WaveForm



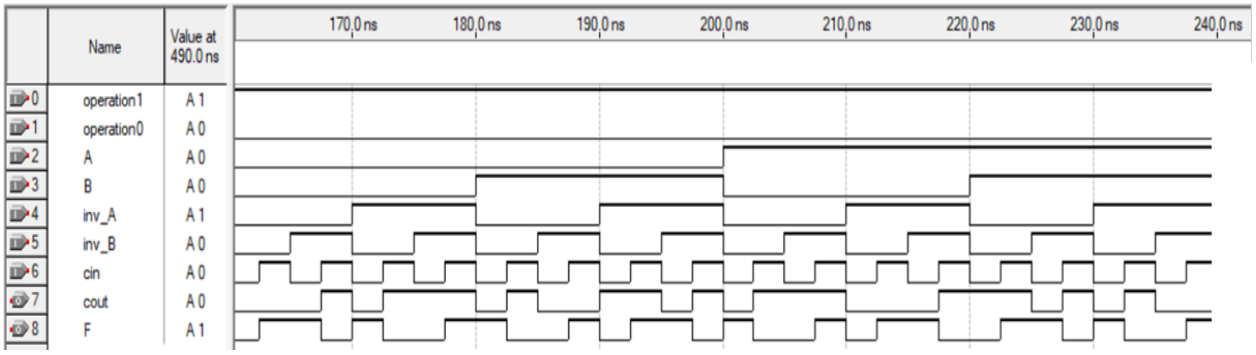
- **When the user enters the operation (01):**

WaveForm



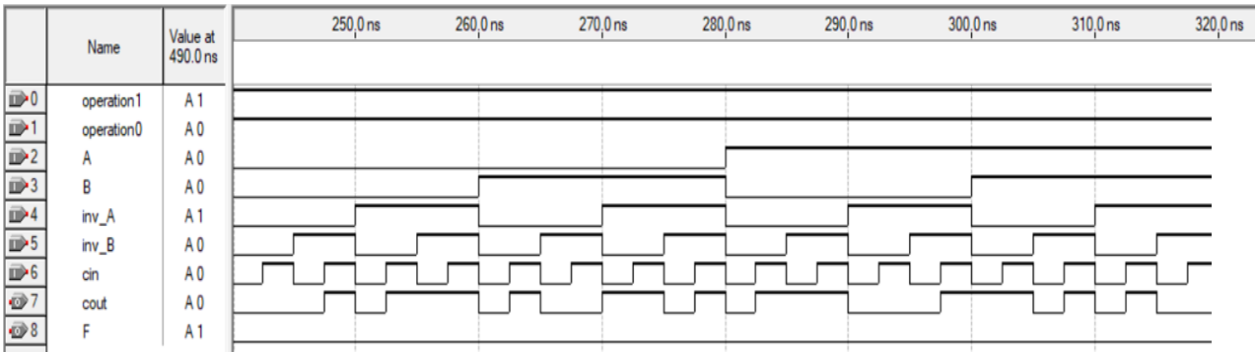
- **When the user enters the operation (10):**

  WaveForm



- **When the user enters the operation (11):**

  WaveForm



Truth table for each operation:

This circuit has 7 inputs then we have 128 combination be input for the circuit , 2 inputs represent the operation that mean the operation will be (00 , 01 , 10 , 11) but (11) operation is invalid (don't care about F) , Hence when the operation be (11) -> F = x and we just find cout .

Attached on the next four pages the Truth table for the four operations :

Operation (00) :

By WaveForm

By experience

	Operation1	Operation0	A	B	inv_A	inv_B	cin	cout	F	cout	F	
1	0	0	0	0	0	0	0	0	0	0	0	✓
2	0	0	0	0	0	0	1	0	0	0	0	✓
3	0	0	0	0	0	1	0	0	0	0	0	✓
4	0	0	0	0	0	1	1	1	0	1	0	✓
5	0	0	0	0	1	0	0	0	0	0	0	✓
6	0	0	0	0	1	0	1	1	0	1	0	✓
7	0	0	0	0	1	1	0	1	1	1	1	✓
8	0	0	0	0	1	1	1	1	1	1	1	✓
9	0	0	0	1	0	0	0	0	0	0	0	✓
10	0	0	0	1	0	0	1	1	0	1	0	✓
11	0	0	0	1	0	1	0	0	0	0	0	✓
12	0	0	0	1	0	1	1	0	0	0	0	✓
13	0	0	0	1	1	0	0	1	1	1	1	✓
14	0	0	0	1	1	0	1	1	1	1	1	✓
15	0	0	0	1	1	1	0	0	0	0	0	✓
16	0	0	0	1	1	1	1	1	0	1	0	✓
17	0	0	1	0	0	0	0	0	0	0	0	✓
18	0	0	1	0	0	0	1	1	0	1	0	✓
19	0	0	1	0	0	1	0	1	1	1	1	✓
20	0	0	1	0	0	1	1	1	1	1	1	✓
21	0	0	1	0	1	0	0	0	0	0	0	✓
22	0	0	1	0	1	0	1	0	0	0	0	✓
23	0	0	1	0	1	1	0	0	0	0	0	✓
24	0	0	1	0	1	1	1	1	0	1	0	✓
25	0	0	1	1	0	0	0	1	1	1	1	✓
26	0	0	1	1	0	0	1	1	1	1	1	✓
27	0	0	1	1	0	1	0	0	0	0	0	✓
28	0	0	1	1	0	1	1	1	0	1	0	✓
29	0	0	1	1	1	0	0	0	0	0	0	✓
30	0	0	1	1	1	0	1	1	0	1	0	✓
31	0	0	1	1	1	1	0	0	0	0	0	✓
32	0	0	1	1	1	1	1	0	0	0	0	✓

Operation (01) :

	Operation1	Operation0	A	B	inv_A	inv_B	cin	cout (By WaveForm)	F (By WaveForm)	cout (By experience)	F (By experience)	
33	0	1	0	0	0	0	0	0	0	0	0	✓
34	0	1	0	0	0	0	1	0	0	0	0	✓
35	0	1	0	0	0	1	0	0	1	0	1	✓
36	0	1	0	0	0	1	1	1	1	1	1	✓
37	0	1	0	0	1	0	0	0	1	0	1	✓
38	0	1	0	0	1	0	1	1	1	1	1	✓
39	0	1	0	0	1	1	0	1	1	1	1	✓
40	0	1	0	0	1	1	1	1	1	1	1	✓
41	0	1	0	1	0	0	0	0	1	0	1	✓
42	0	1	0	1	0	0	1	1	1	1	1	✓
43	0	1	0	1	0	1	0	0	0	0	0	✓
44	0	1	0	1	0	1	1	0	0	0	0	✓
45	0	1	0	1	1	0	0	1	1	1	1	✓
46	0	1	0	1	1	0	1	1	1	1	1	✓
47	0	1	0	1	1	1	0	0	1	0	1	✓
48	0	1	0	1	1	1	1	1	1	1	1	✓
49	0	1	1	0	0	0	0	0	1	0	1	✓
50	0	1	1	0	0	0	1	1	1	1	1	✓
51	0	1	1	0	0	1	0	1	1	1	1	✓
52	0	1	1	0	0	1	1	1	1	1	1	✓
53	0	1	1	0	1	0	0	0	0	0	0	✓
54	0	1	1	0	1	0	1	0	0	0	0	✓
55	0	1	1	0	1	1	0	0	1	0	1	✓
56	0	1	1	0	1	1	1	1	1	1	1	✓
57	0	1	1	1	0	0	0	1	1	1	1	✓
58	0	1	1	1	0	0	1	1	1	1	1	✓
59	0	1	1	1	0	1	0	0	1	0	1	✓
60	0	1	1	1	0	1	1	1	1	1	1	✓
61	0	1	1	1	1	0	0	0	1	0	1	✓
62	0	1	1	1	1	0	1	1	1	1	1	✓
63	0	1	1	1	1	1	0	0	0	0	0	✓
64	0	1	1	1	1	1	1	0	0	0	0	✓

Operation (10) :

	Operation1	Operation0	A	B	inv_A	inv_B	cin	cout (By WaveForm)	F (By WaveForm)	cout (By experience)	F (By experience)	
65	1	0	0	0	0	0	0	0	0	0	0	✓
66	1	0	0	0	0	0	1	0	1	0	1	✓
67	1	0	0	0	0	1	0	0	1	0	1	✓
68	1	0	0	0	0	1	1	1	0	1	0	✓
69	1	0	0	0	1	0	0	0	1	0	1	✓
70	1	0	0	0	1	0	1	1	0	1	0	✓
71	1	0	0	0	1	1	0	1	0	1	0	✓
72	1	0	0	0	1	1	1	1	1	1	1	✓
73	1	0	0	1	0	0	0	0	1	0	1	✓
74	1	0	0	1	0	0	1	1	0	1	0	✓
75	1	0	0	1	0	1	0	0	0	0	0	✓
76	1	0	0	1	0	1	1	0	1	0	1	✓
77	1	0	0	1	1	0	0	1	0	1	0	✓
78	1	0	0	1	1	0	1	1	1	1	1	✓
79	1	0	0	1	1	1	0	0	1	0	1	✓
80	1	0	0	1	1	1	1	1	0	1	0	✓
81	1	0	1	0	0	0	0	0	1	0	1	✓
82	1	0	1	0	0	0	1	1	0	1	0	✓
83	1	0	1	0	0	1	0	1	0	1	0	✓
84	1	0	1	0	0	1	1	1	1	1	1	✓
85	1	0	1	0	1	0	0	0	0	0	0	✓
86	1	0	1	0	1	0	1	0	1	0	1	✓
87	1	0	1	0	1	1	0	0	1	0	1	✓
88	1	0	1	0	1	1	1	1	0	1	0	✓
89	1	0	1	1	0	0	0	1	0	1	0	✓
90	1	0	1	1	0	0	1	1	1	1	1	✓
91	1	0	1	1	0	1	0	0	1	0	1	✓
92	1	0	1	1	0	1	1	1	0	1	0	✓
93	1	0	1	1	1	0	0	0	1	0	1	✓
94	1	0	1	1	1	0	1	1	0	1	0	✓
95	1	0	1	1	1	1	0	0	0	0	0	✓
96	1	0	1	1	1	1	1	0	1	0	1	✓

Operation (11) invalid operation for F (Truth table for Cout) :

By WaveForm

By experience

	Operation1	Operation0	A	B	inv_A	inv_B	cin	cout	F	cout	F	
97	1	1	0	0	0	0	0	0	x	0	x	✓
98	1	1	0	0	0	0	1	0	x	0	x	✓
99	1	1	0	0	0	1	0	0	x	0	x	✓
100	1	1	0	0	0	1	1	1	x	1	x	✓
101	1	1	0	0	1	0	0	0	x	0	x	✓
102	1	1	0	0	1	0	1	1	x	1	x	✓
103	1	1	0	0	1	1	0	1	x	1	x	✓
104	1	1	0	0	1	1	1	1	x	1	x	✓
105	1	1	0	1	0	0	0	0	x	0	x	✓
106	1	1	0	1	0	0	1	1	x	1	x	✓
107	1	1	0	1	0	1	0	0	x	0	x	✓
108	1	1	0	1	0	1	1	0	x	0	x	✓
109	1	1	0	1	1	0	0	1	x	1	x	✓
110	1	1	0	1	1	0	1	1	x	1	x	✓
111	1	1	0	1	1	1	0	0	x	0	x	✓
112	1	1	0	1	1	1	1	1	x	1	x	✓
113	1	1	1	0	0	0	0	0	x	0	x	✓
114	1	1	1	0	0	0	1	1	x	1	x	✓
115	1	1	1	0	0	1	0	1	x	1	x	✓
116	1	1	1	0	0	1	1	1	x	1	x	✓
117	1	1	1	0	1	0	0	0	x	0	x	✓
118	1	1	1	0	1	0	1	0	x	0	x	✓
119	1	1	1	0	1	1	0	0	x	0	x	✓
120	1	1	1	0	1	1	1	1	x	1	x	✓
121	1	1	1	1	0	0	0	1	x	1	x	✓
122	1	1	1	1	0	0	1	1	x	1	x	✓
123	1	1	1	1	0	1	0	0	x	0	x	✓
124	1	1	1	1	0	1	1	1	x	1	x	✓
125	1	1	1	1	1	0	0	0	x	0	x	✓
126	1	1	1	1	1	0	1	1	x	1	x	✓
127	1	1	1	1	1	1	0	0	x	0	x	✓
128	1	1	1	1	1	1	1	0	x	0	x	✓