MUltipleXer (MUX)

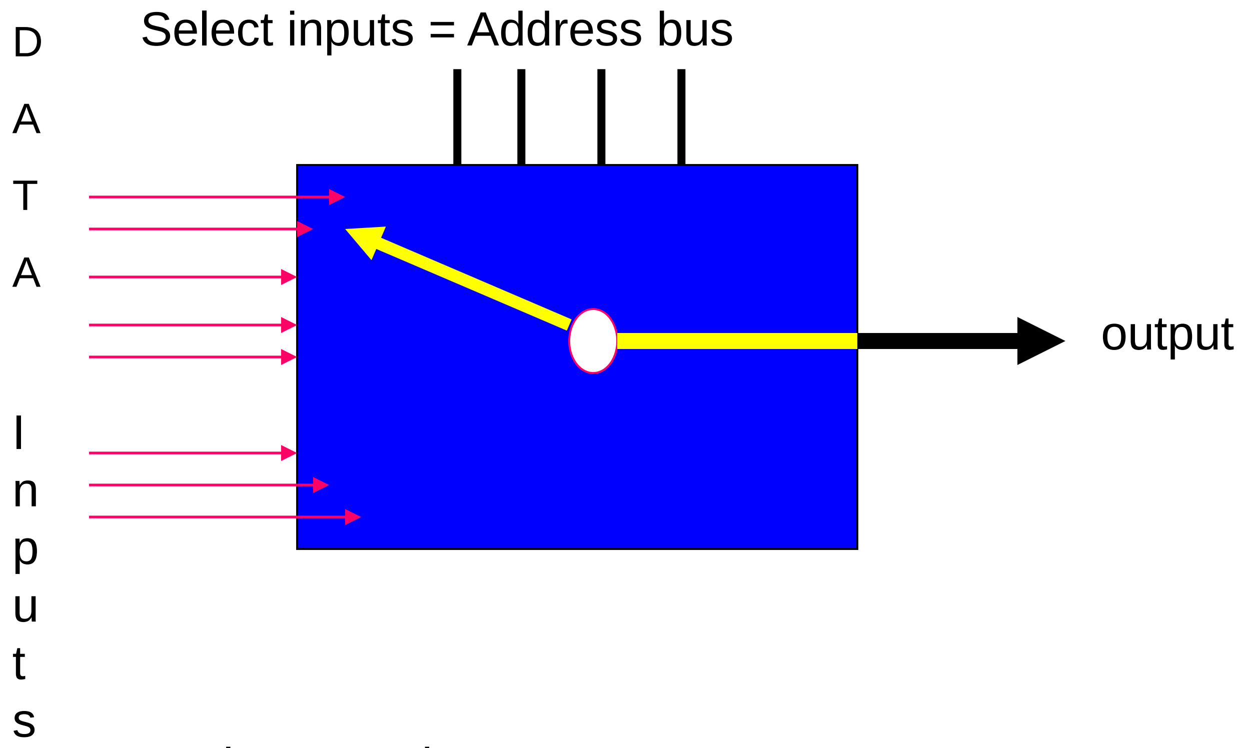
1-bit ALU design

Multiplexer

- A logic circuit that accepts data from more than one inputs and routes the data to a SINGLE output.

16-Channel Analog Multiplexer/Demultiplexer

Multiplexer: Block Diagram

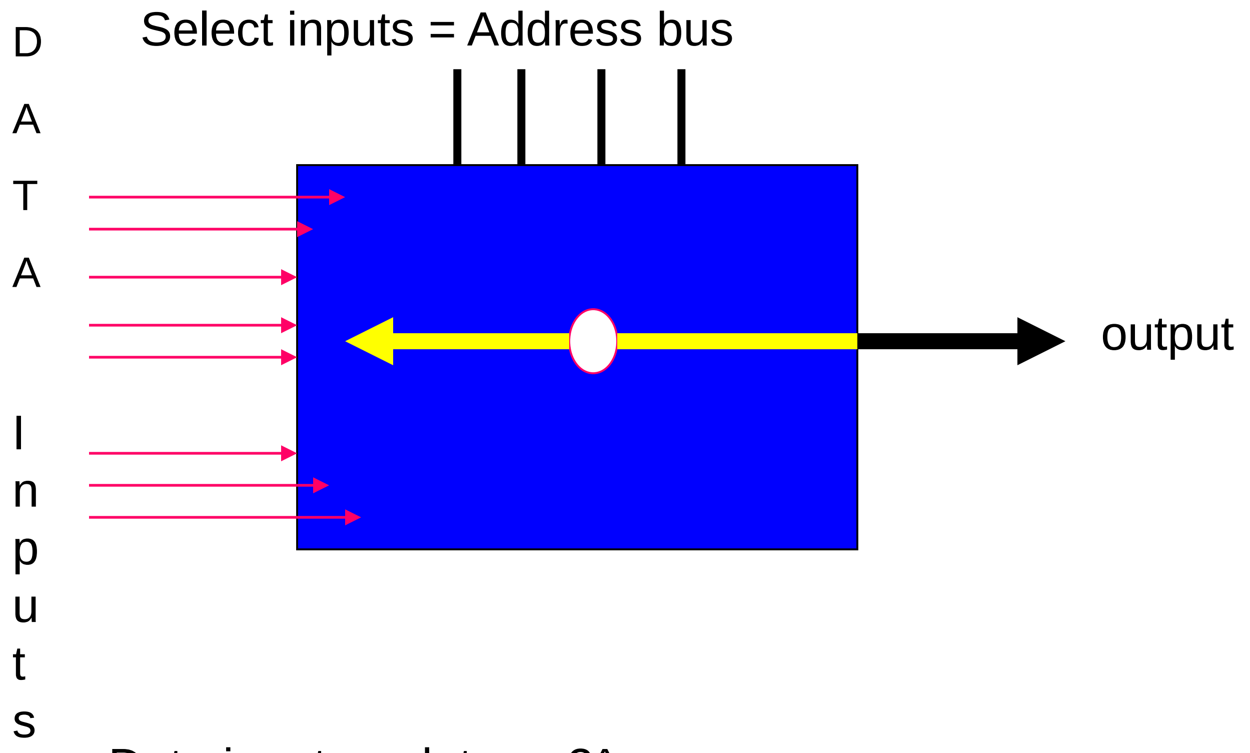


Data inputs = data = 2^n

Select lines = address lines = n

Output = 1

Multiplexer: Block Diagram

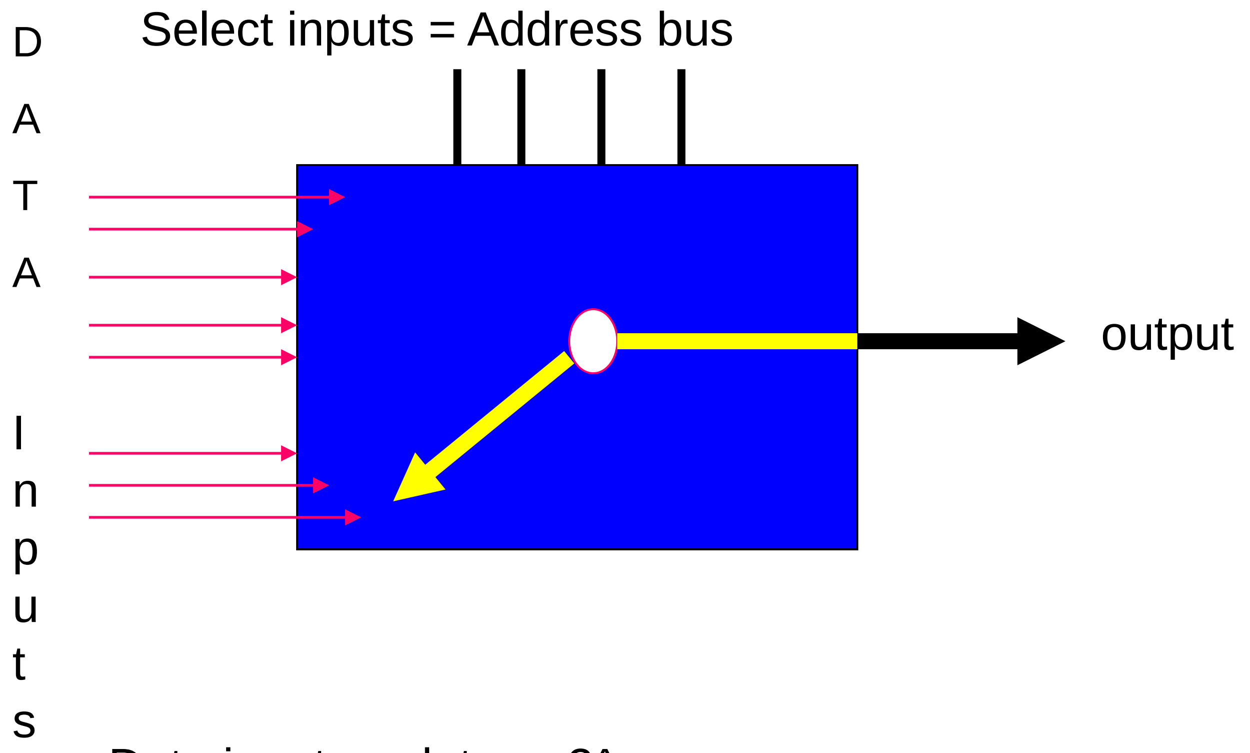


Data inputs = data = 2^n

Select lines = address lines = n

Output = 1

Multiplexer: Block Diagram



Data inputs = data = 2^n

Select lines = address lines = n

Output = 1

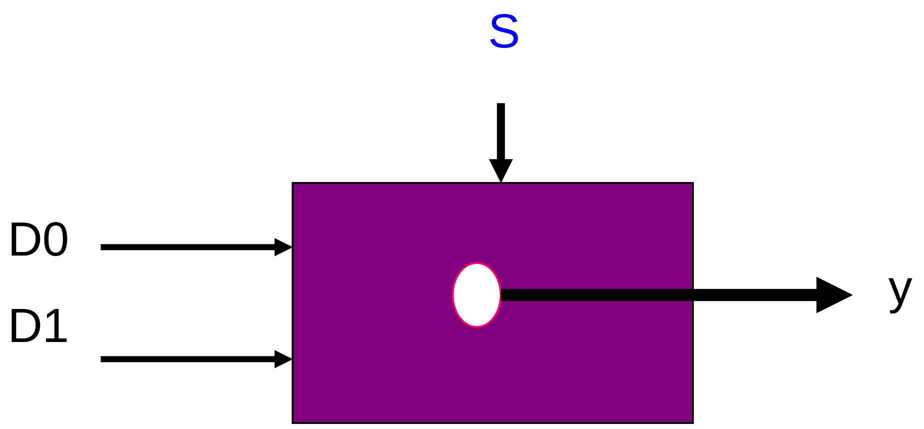
Multiplexer: Characteristic

The multiplexer is programmed through the select inputs to establish a connection between the output and the data inputs.

Design a 2-1 Multiplexer

Inputs = 2

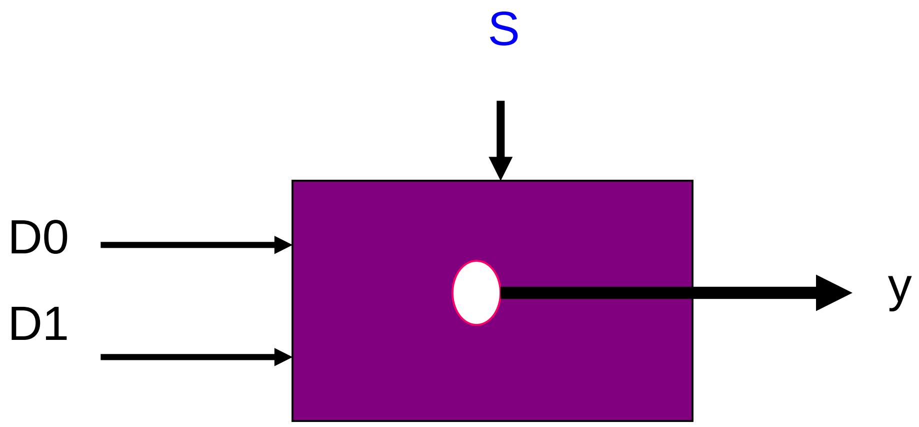
Output = 1



Design a 2-1 Multiplexer

Inputs = 2

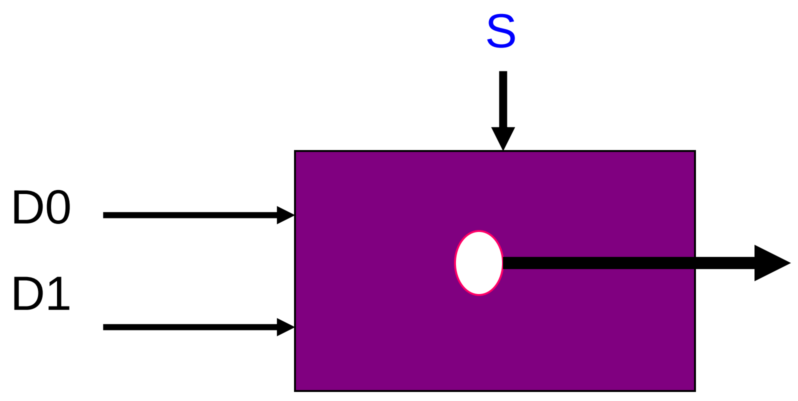
Output = 1



Therefore we need n=1 select line.

Logic: Truth Table

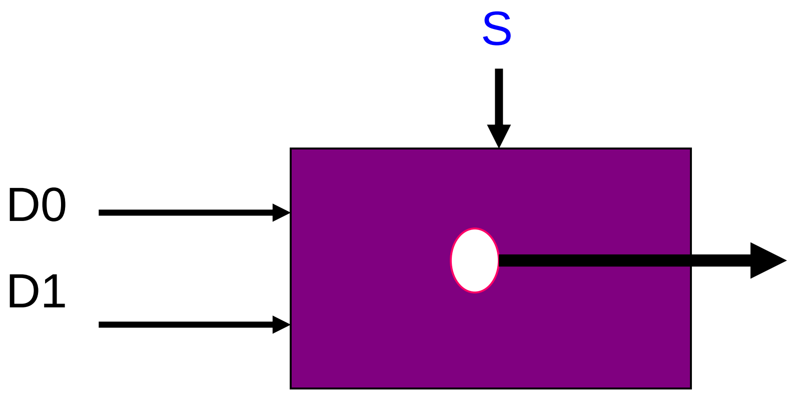
S	D1	D0	Y
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	



If S = 0 => Y = D0
If S = 1 => Y = D1

Logic: Truth Table

S	D1	D0	Y
0	0	0 →	0
0	0	1 →	1
0	1	0 →	0
0	1	1 →	1
1	0	0	
1	0	1	
1	1	0	
1	1	1	

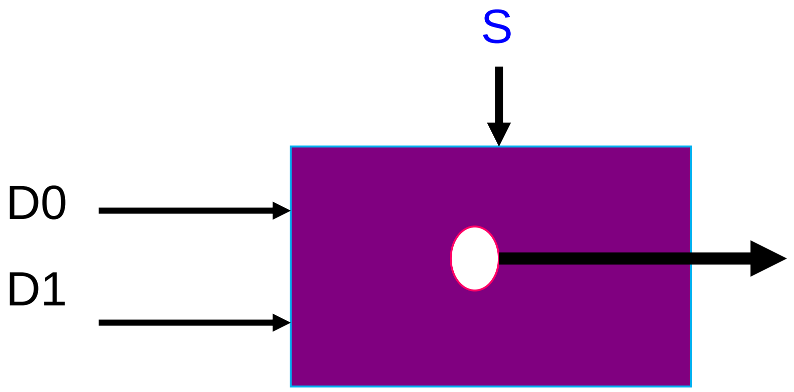


If S = 0 => Y = D0

If S = 1

Logic: Truth Table

S	D1	D0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



If S = 0 => Y = D0

If S = 1 => Y = D1

Logic: Truth Table; K-Map

S	D1	D0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

D1D0	S=0	S=1
00		
01		
11		
10		

Y

Logic: Truth Table; K-Map

S	D1	D0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

D1D0	S=0	S=1
00	0	0
01	1	0
11	1	1
10	0	1

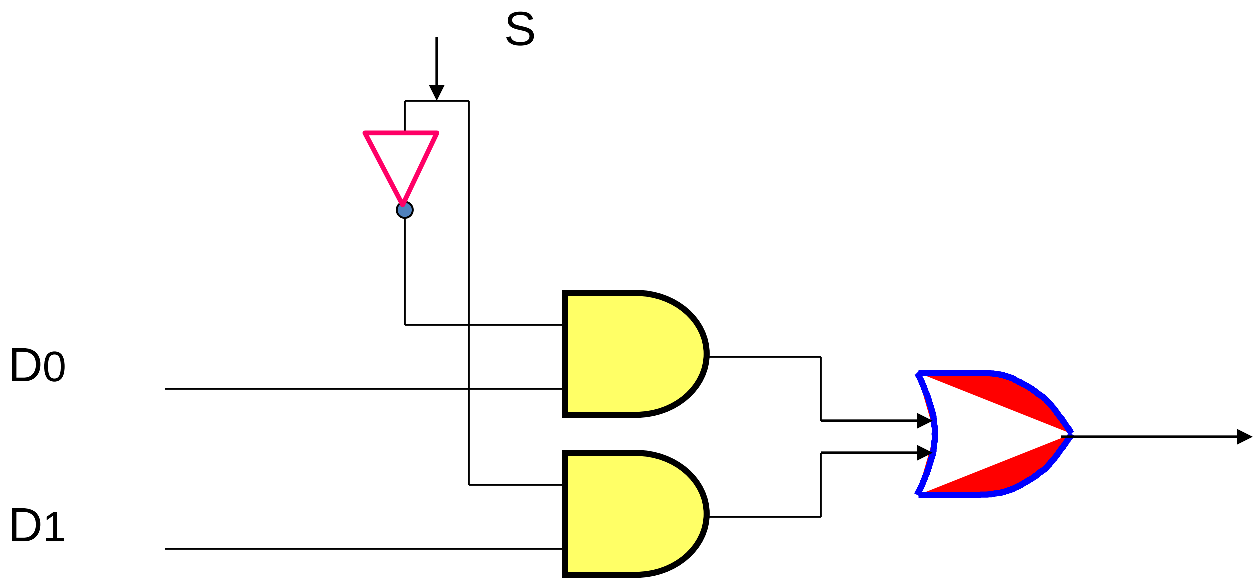
Y

Simplified logic equation

	Y	
D1D0	**S=0**	**S=1**
00	0	0
01	1	0
11	1	1
10	0	1

$$Y = D0\,\overline{S} + D1\,S$$

Logic Diagram



$$Y = D0\,\bar{S} + D1\,S$$

Applications

- Computer Switcher
  - Many PC’s to use a single printer
- Security Monitoring System
  - Use many cameras to monitor an area; 1 computer monitor for the security person
- Data Communications
  - Many users transmit to a single line.

A logic circuit can be implemented with:

- Basic logic gates (AND, OR, NOT)
- NAND gates (universal gate)
- NOR gates (universal gate)
- Decoders
- and … Multiplexers

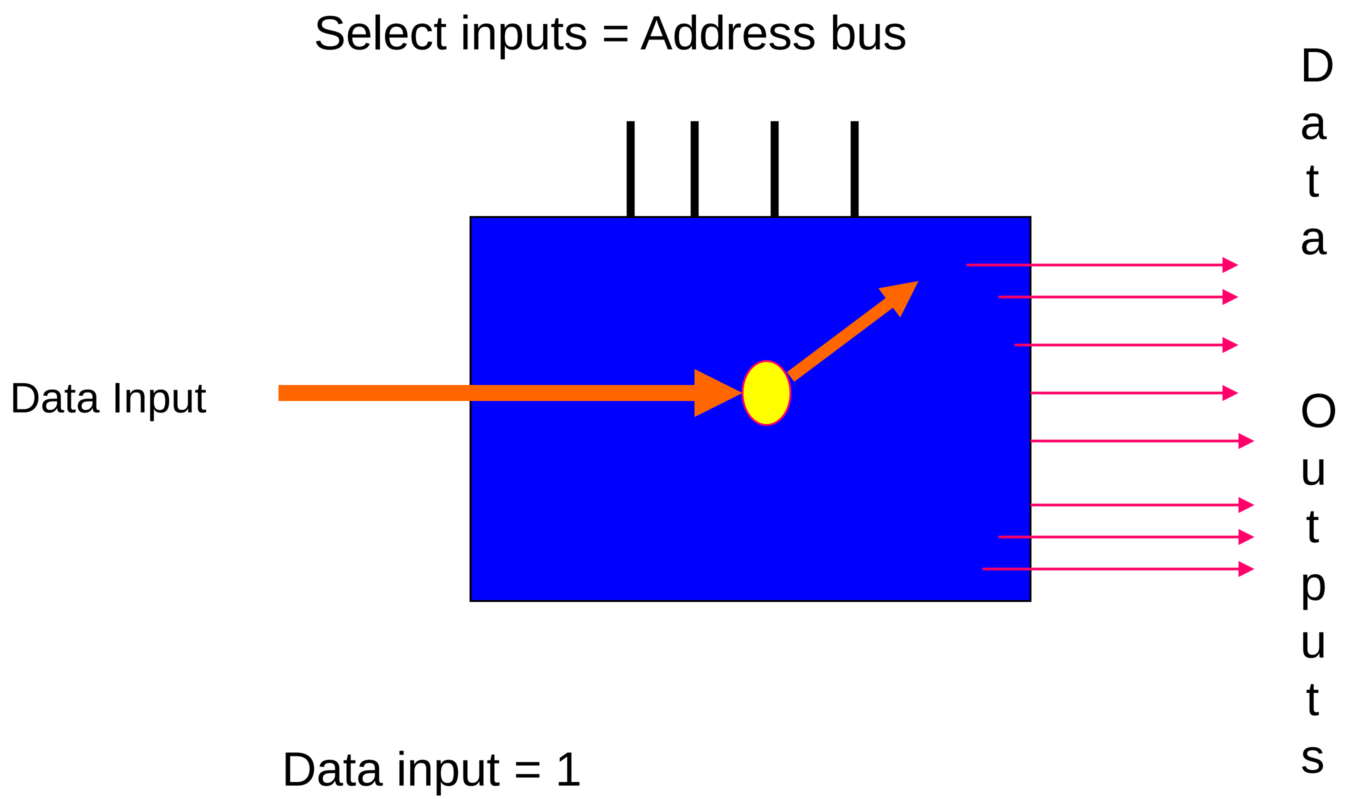
A MULTIPLEXER CAN IMPLEMENT LOGIC FUNCTIONS

Demultiplexer (DMUX)

Demultiplexer

- Is an electronic switch that routes incoming data signals to one of several lines.

Demultiplexer: Block Diagram

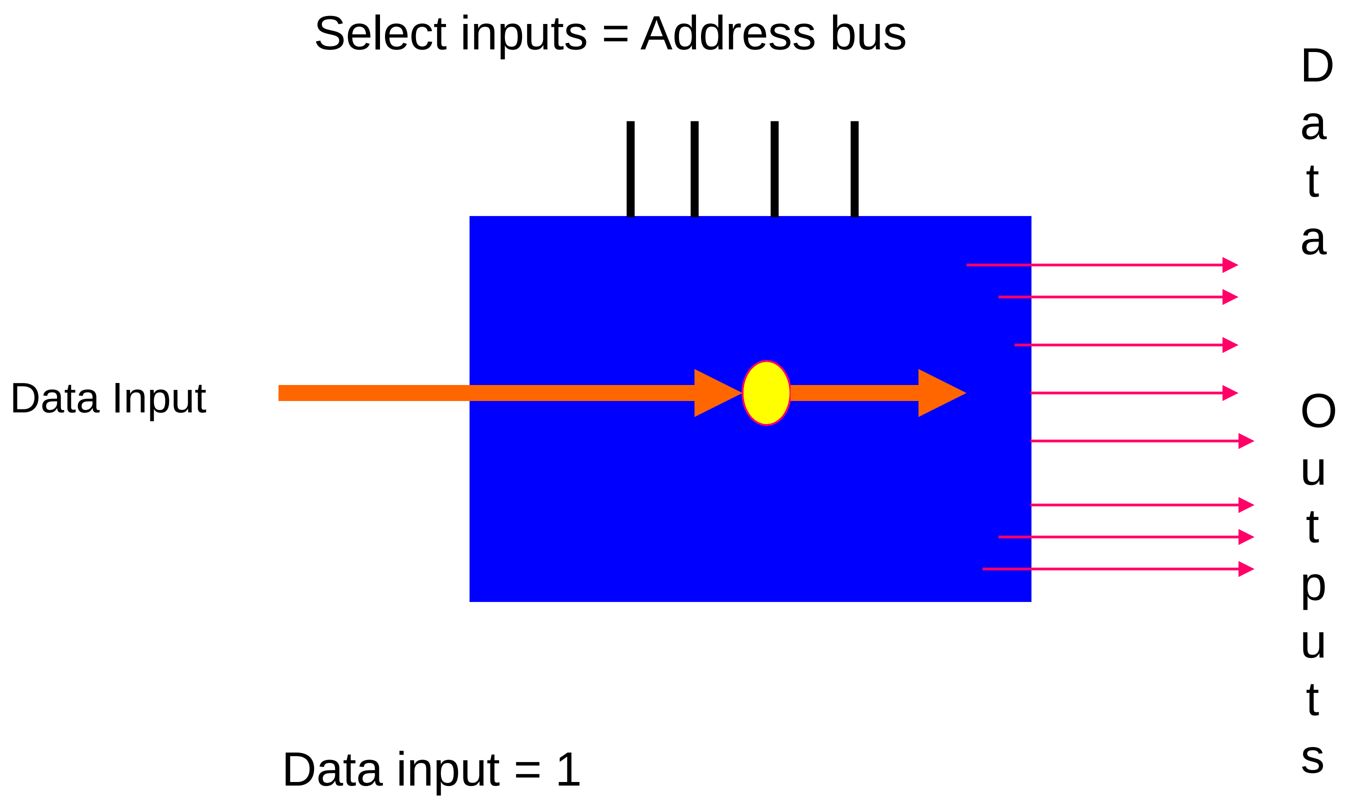


Data input = 1

Select lines = address lines = n

Output lines = 2^n

Demultiplexer: Block Diagram

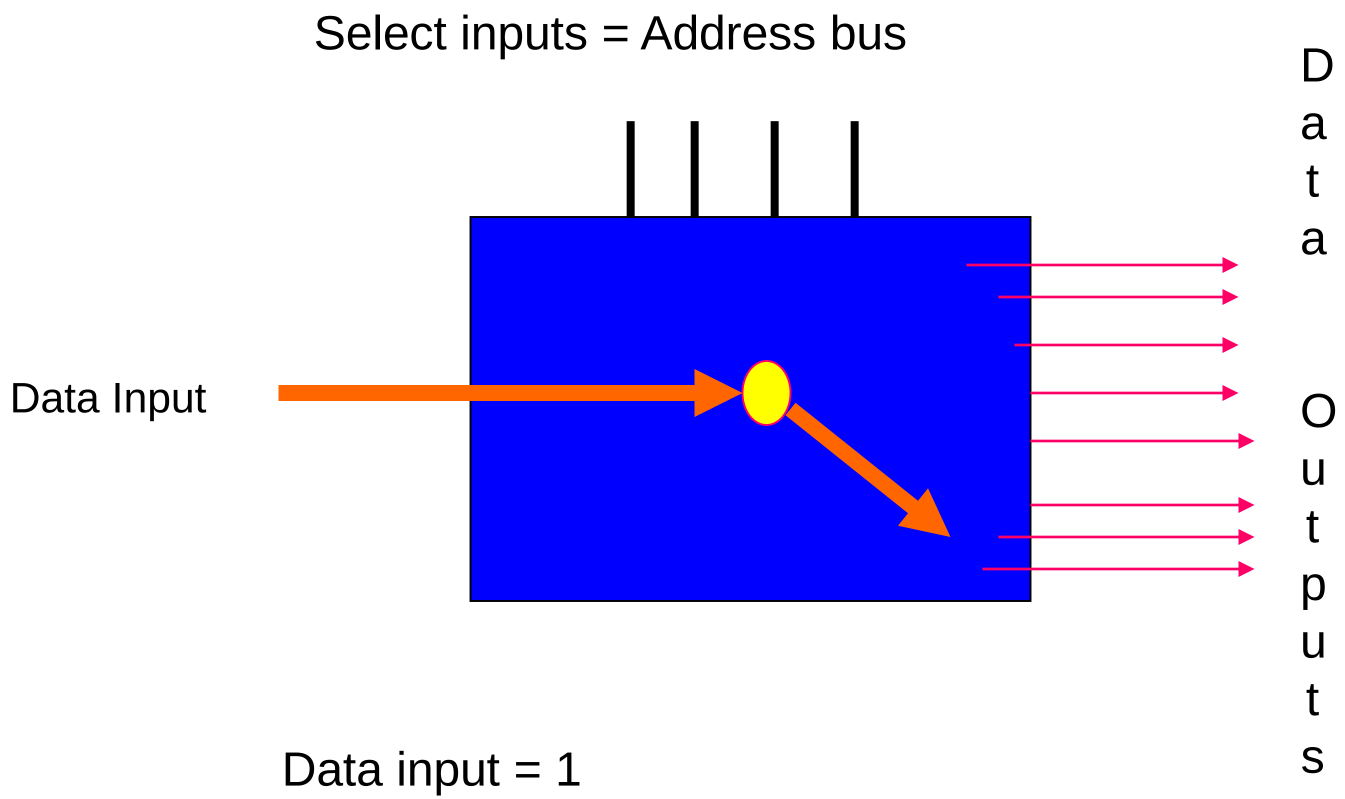


Data input = 1

Select lines = address lines = n

Output lines = 2^n

Demultiplexer: Block Diagram



Data input = 1

Select lines = address lines = n

Output lines = 2^n

Design a 1-2 demultiplexer

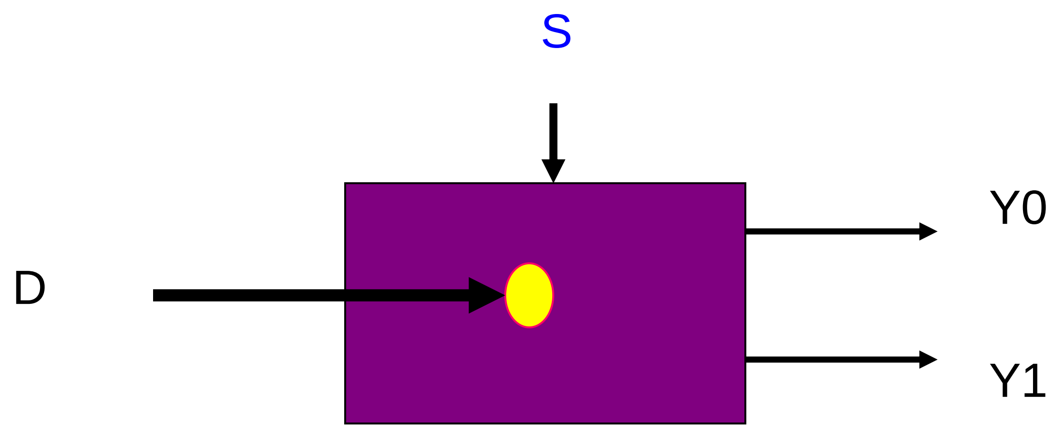
Input = 1

Outputs = 2

Design a 1-2 demultiplexer

Input = 1

Outputs = 2



Therefore we need n = 1 select line.

Truth Table

D	S	Y0	Y1
D	0	D	O
D	1	0	D

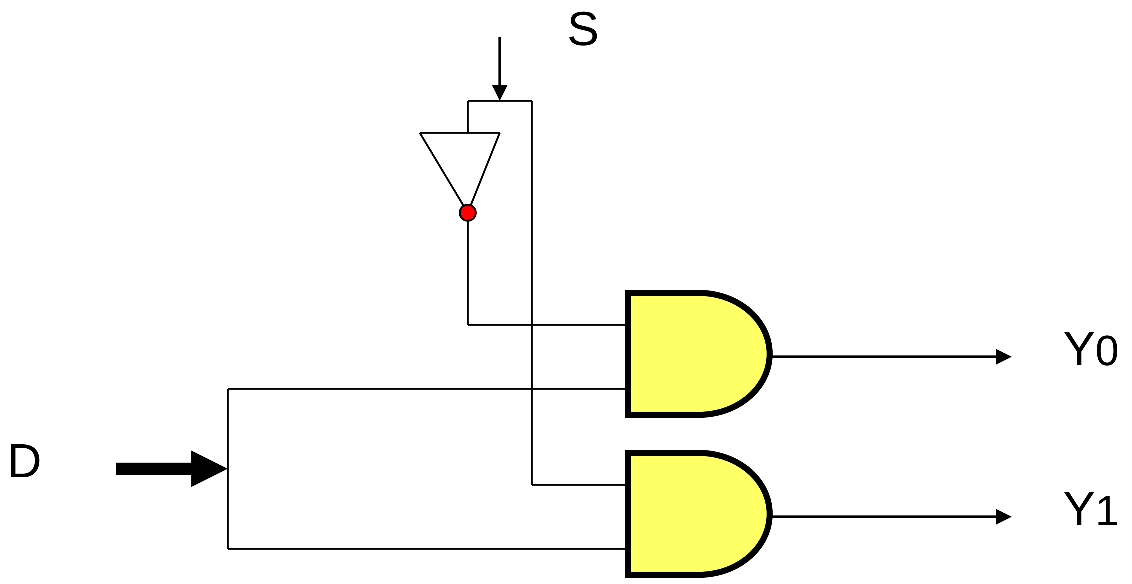
Logic Equations

D	S	Y0	Y1
D	0	D	O
D	1	0	D

$Y0 = \overline{S}\ D$

$Y1 = S\ D$

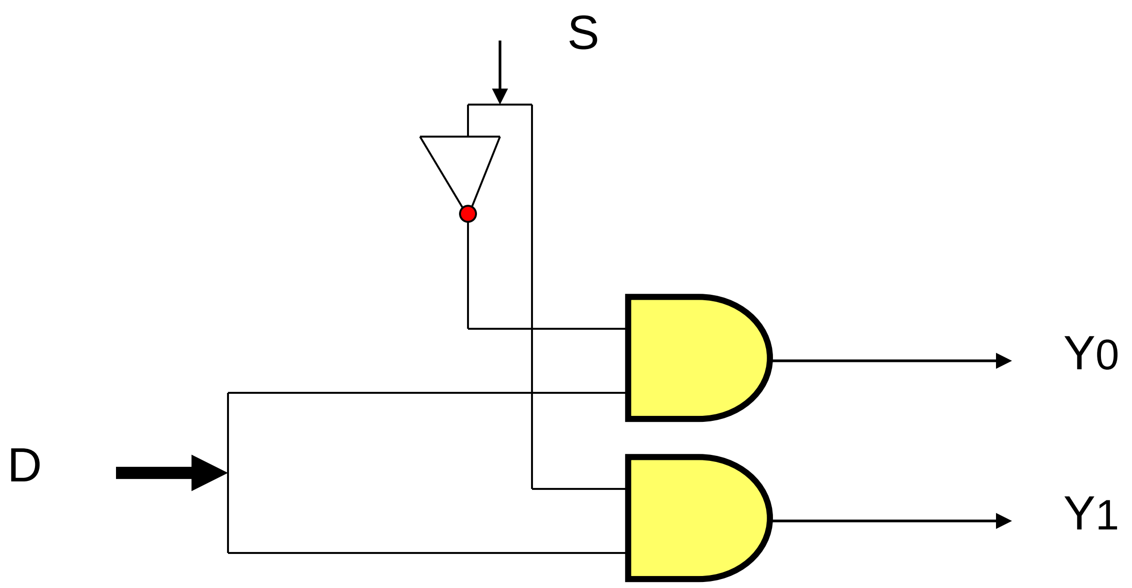
Logic Diagram



$Y0 = \overline{S}\ D$

$Y1 = S\ D$

Logic Diagram

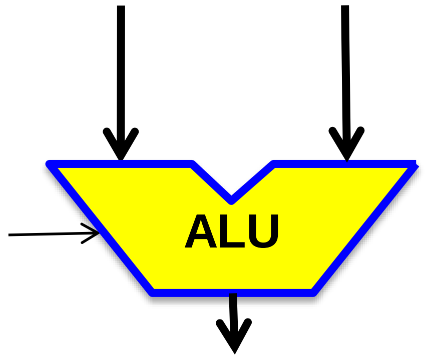


Application; Data Routing

From a digital computer route data to all electronic devices:

- Plotter
- Color Laser Printer
- 3D Printer
- Monitor
- Sensors

(next ALU design)



DESIGN: 1-BIT ALU

1-bit ALU

Design: 1-bit ALU, to perform the following logical and arithmetic operations:

- a AND b
- NOT b
- a + b
- a - b

Just an academic ALU

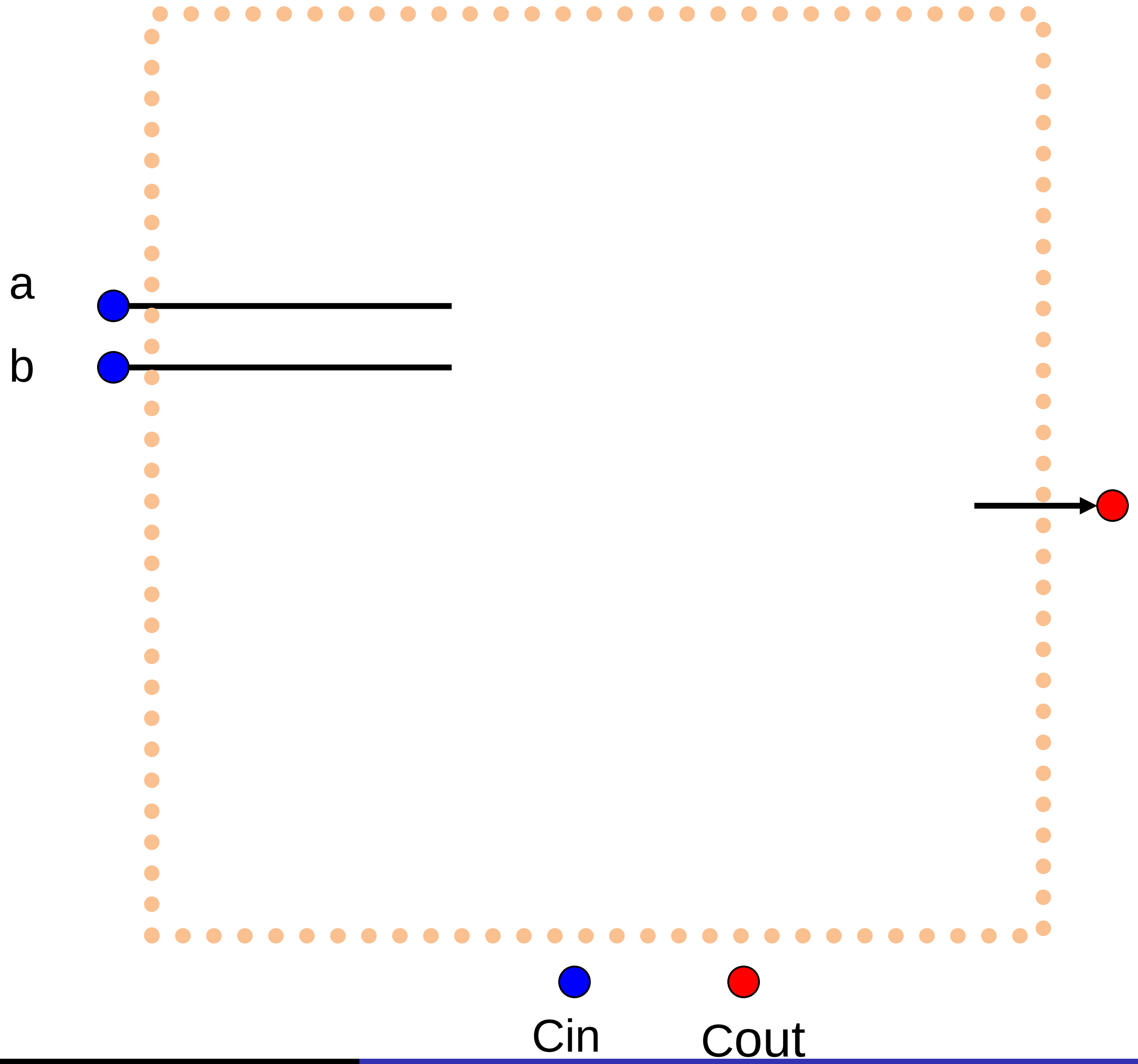
1-bit ALU

Design: 1-bit ALU, to perform the following logical and arithmetic operations:

- a AND b
- NOT b
- a + b
- a - b

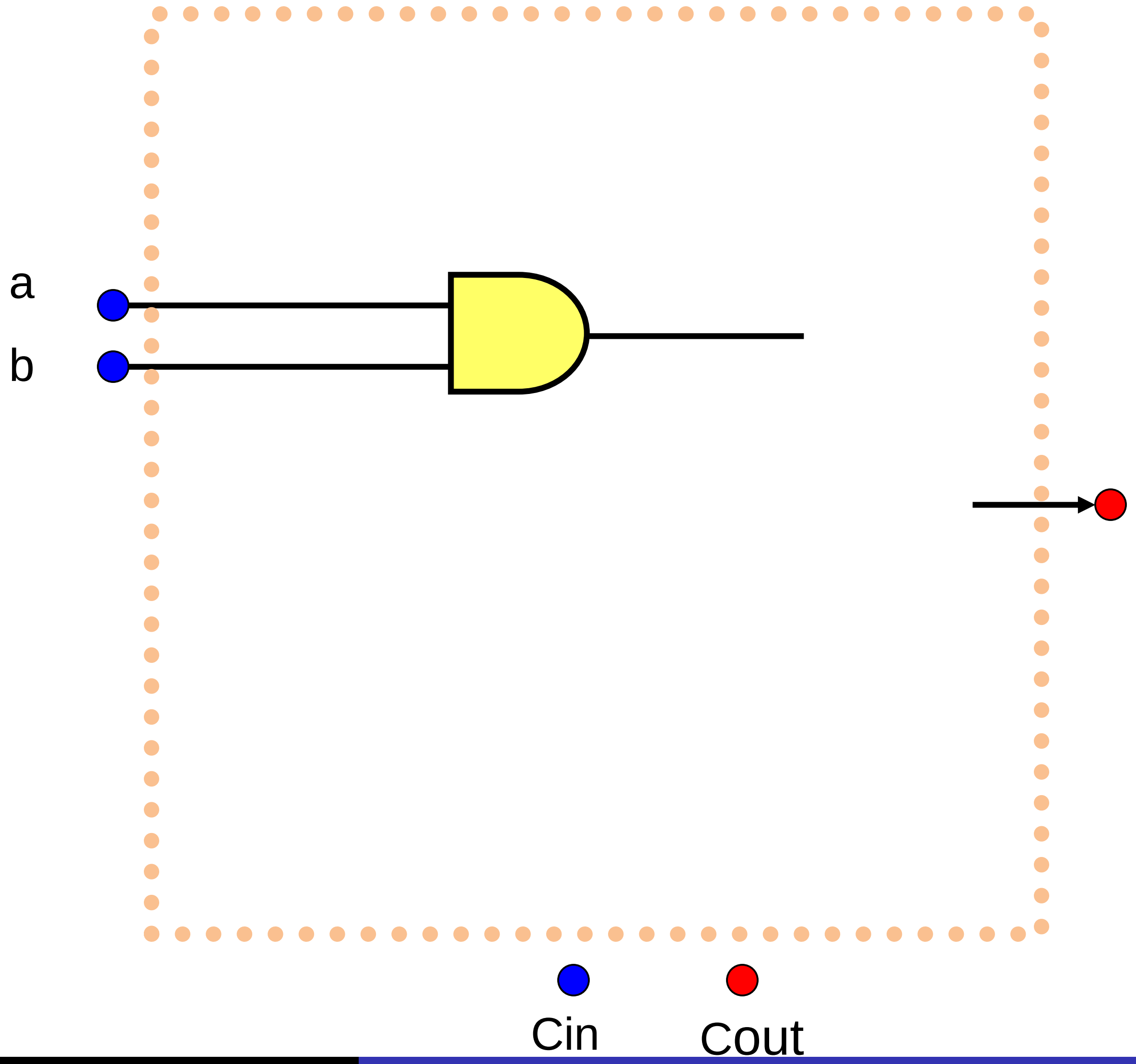
How many inputs … how many outputs ?

(?)-inputs; (?)-output



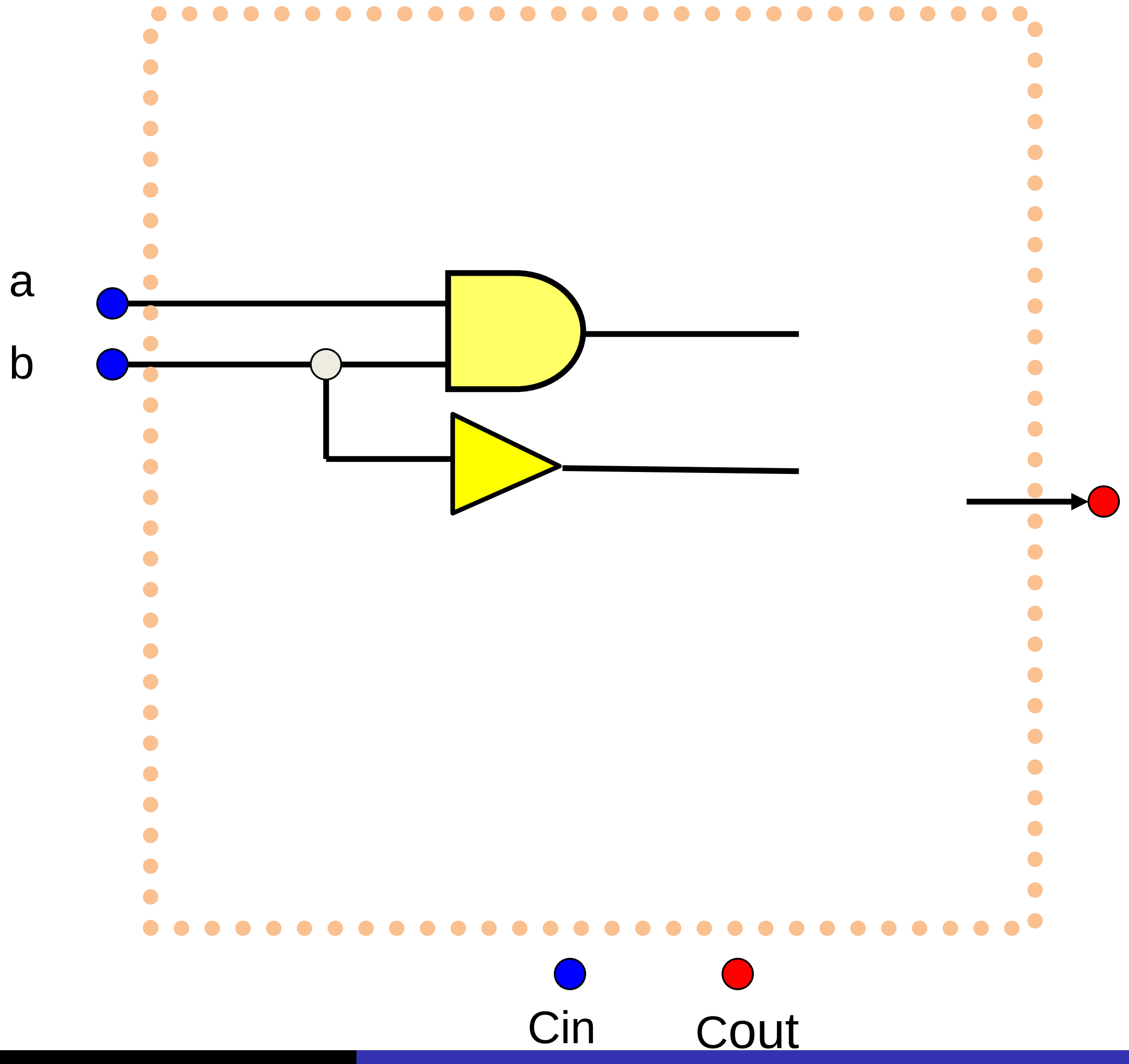
- a AND b
- NOT b
- a + b
- a - b

a AND b



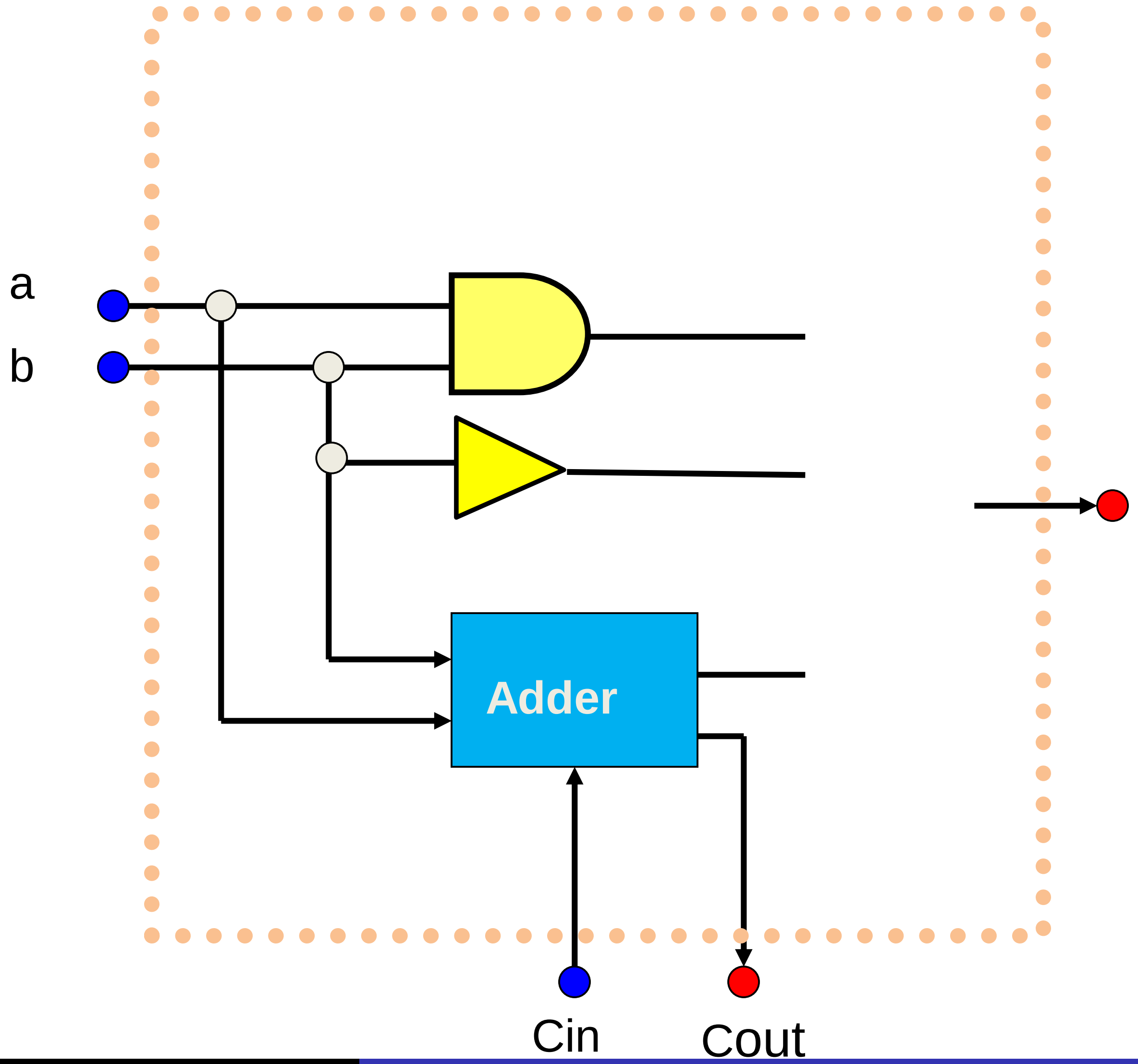
- a AND b
- NOT b
- a + b
- a - b

NOT b



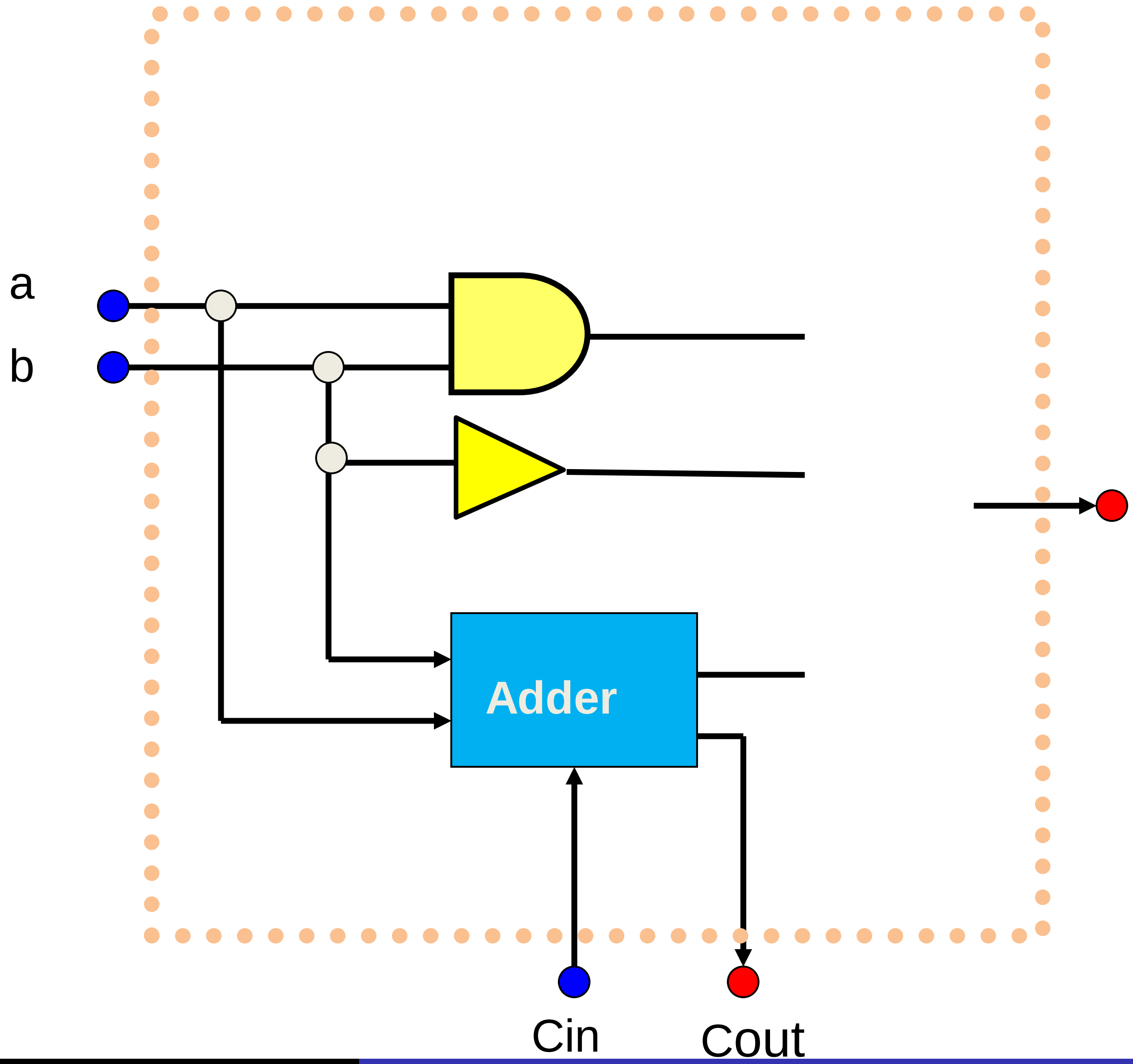
- a AND b
- NOT b
- a + b
- a - b

a+b (C_{in} = 0)



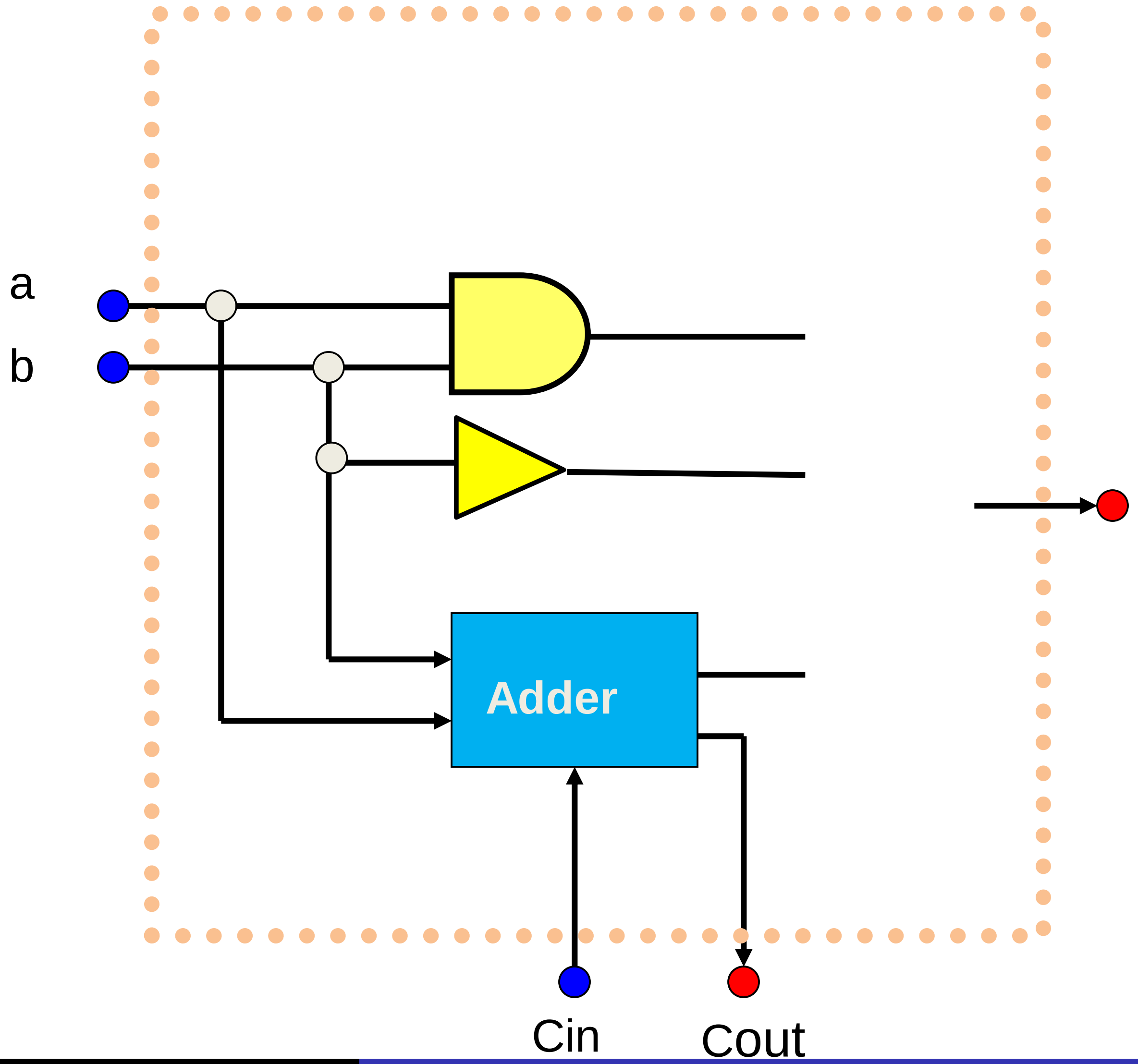
- a AND b
- NOT b
- a + b
- a - b

a-b (C_{in} = 1)



- a AND b
- NOT b
- a + b
- a - b

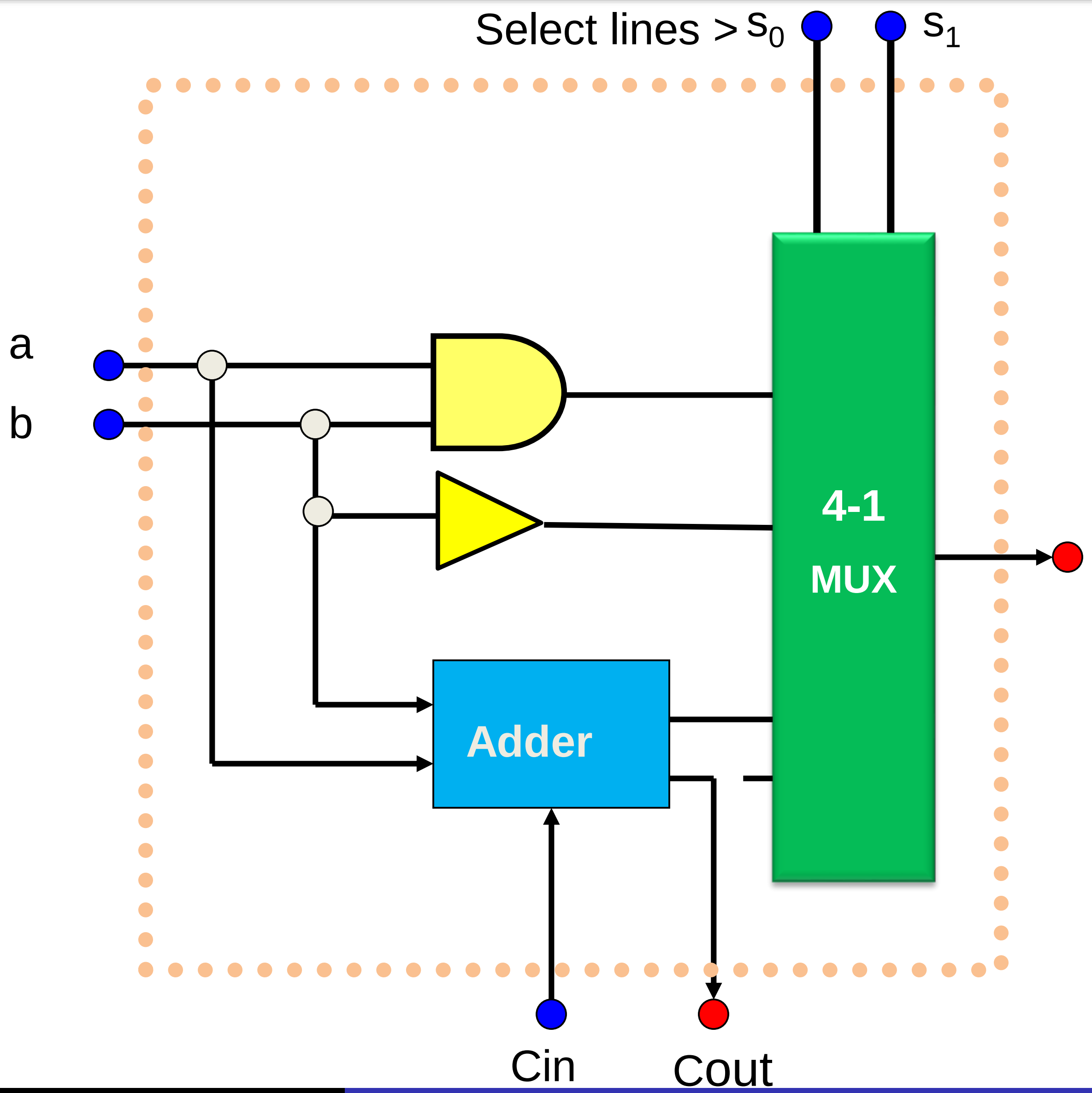
... and a



- a AND b
- NOT b
- a + b
- a - b

… and a 4-1 MUX

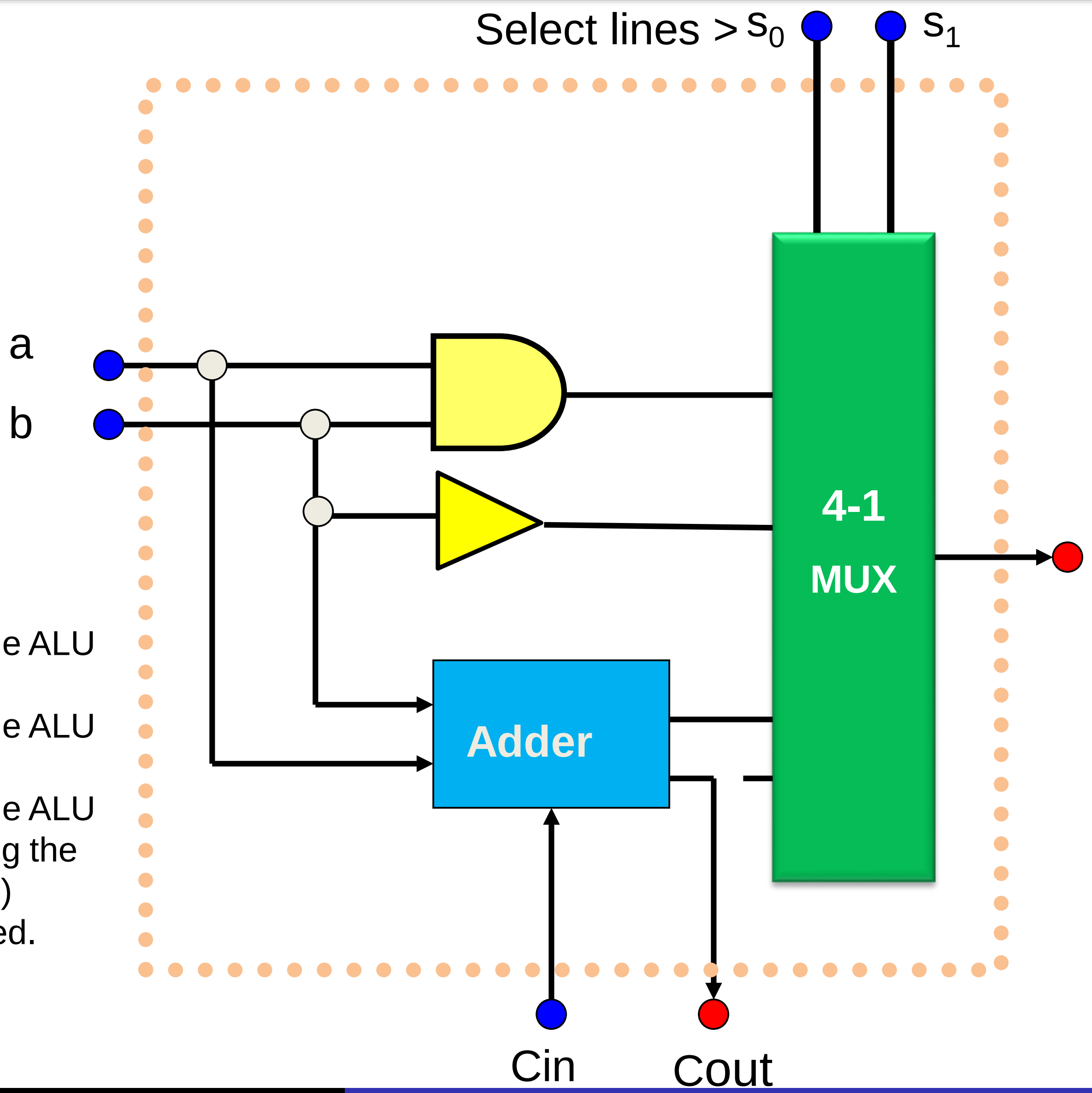
s_0	s_1	Out
0	0	AND
0	1	NOT
1	0	Add & Sub
1	1	x



a AND b
NOT b
a + b
a - b

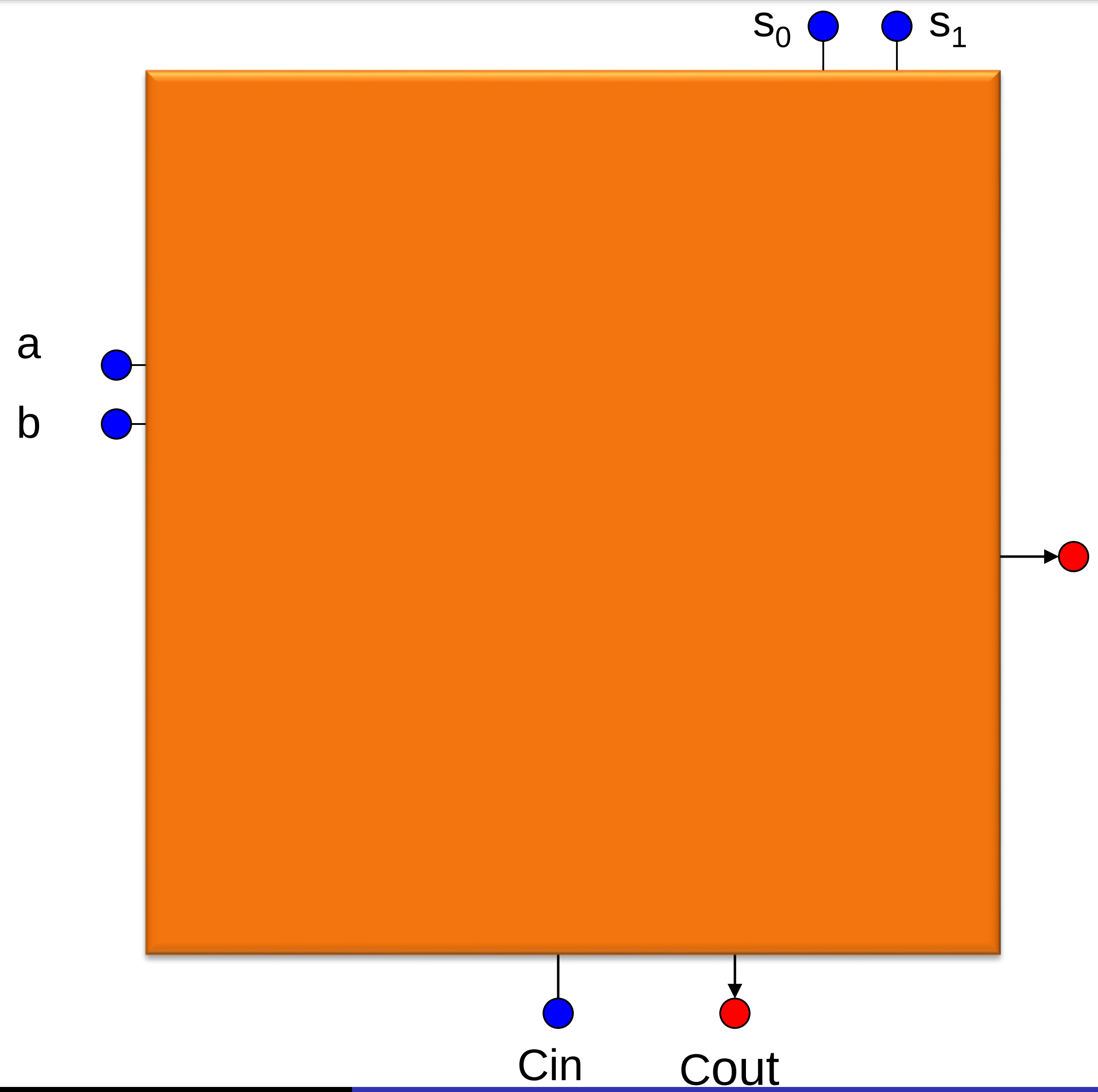
Therefore

S_0	S_1	Out
0	0	AND
0	1	NOT
1	0	Add & Sub
1	1	x

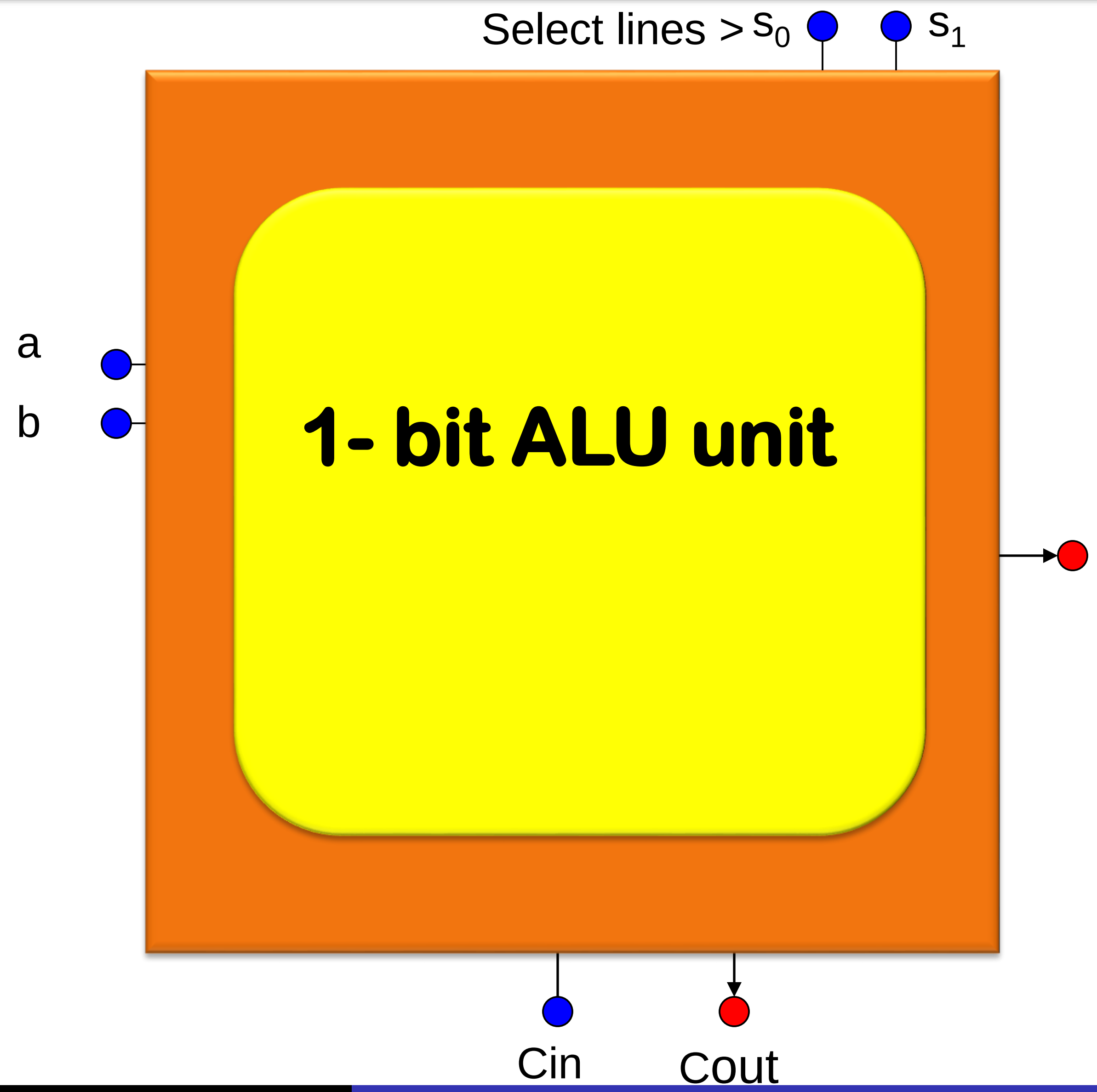


- If the select lines are: (00), the ALU will perform: ADD
- If the select lines are: (01), the ALU will perform: NOT
- If the select lines are: (10), the ALU will perform: Add & Sub (using the Cin, 0 and 1 for Add and Sub)
- The select state (11) is unused.

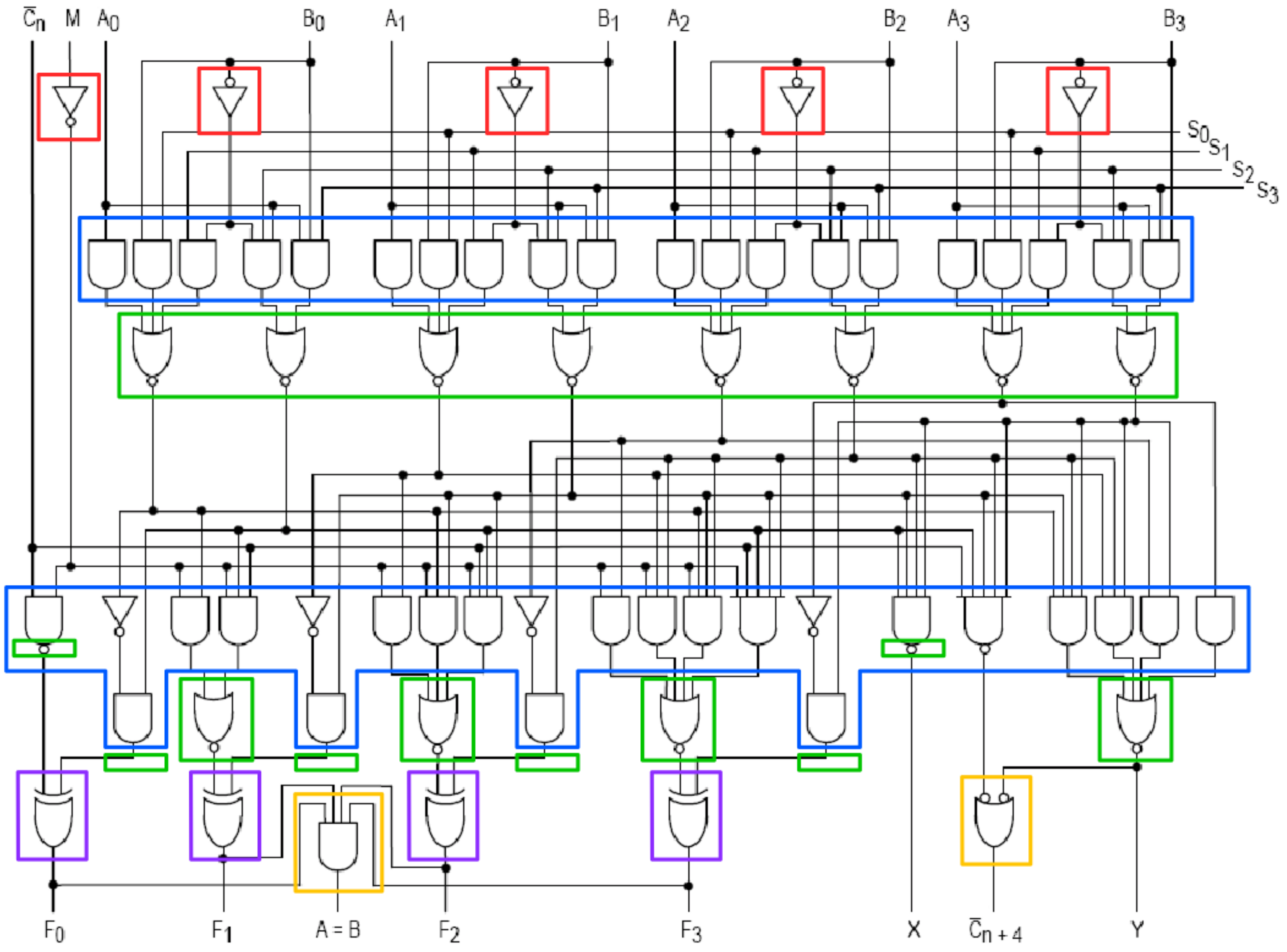
1-bit ALU



1-bit ALU

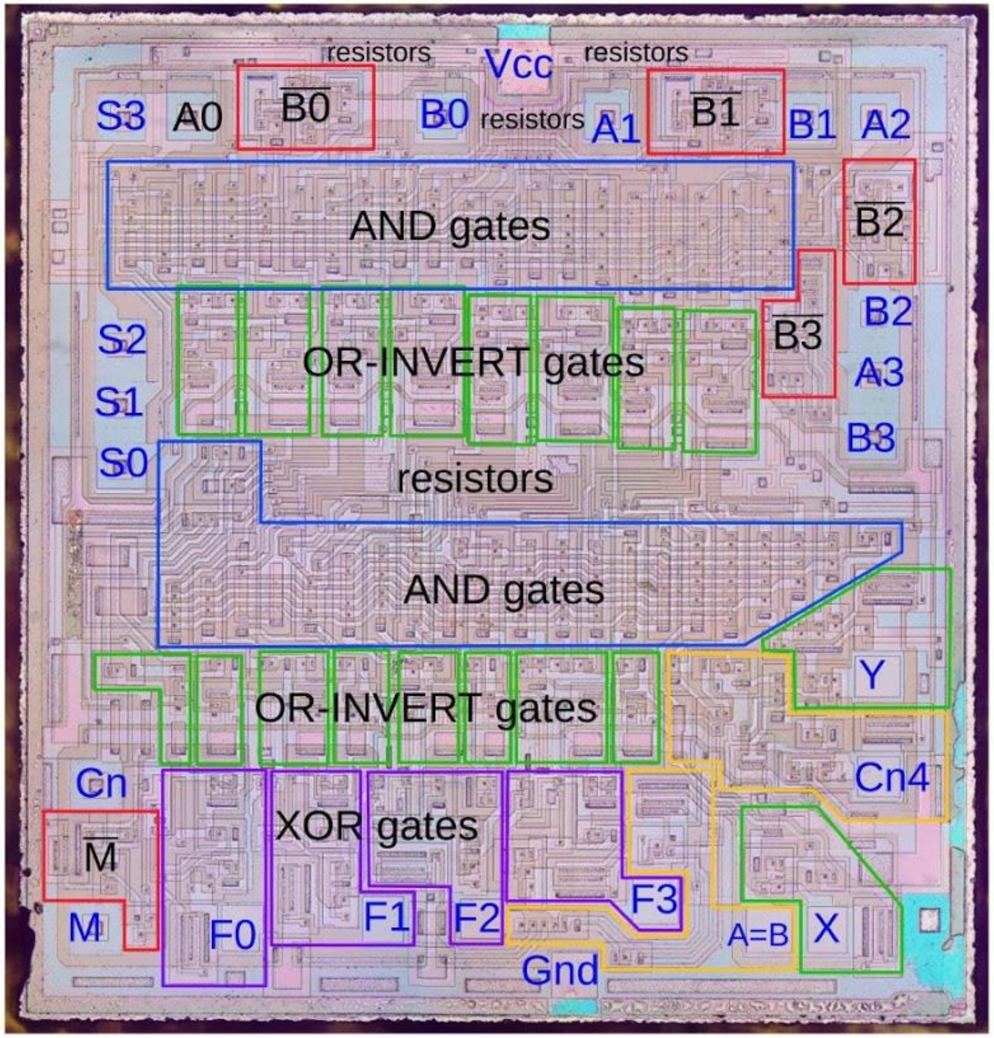


74181 ALU

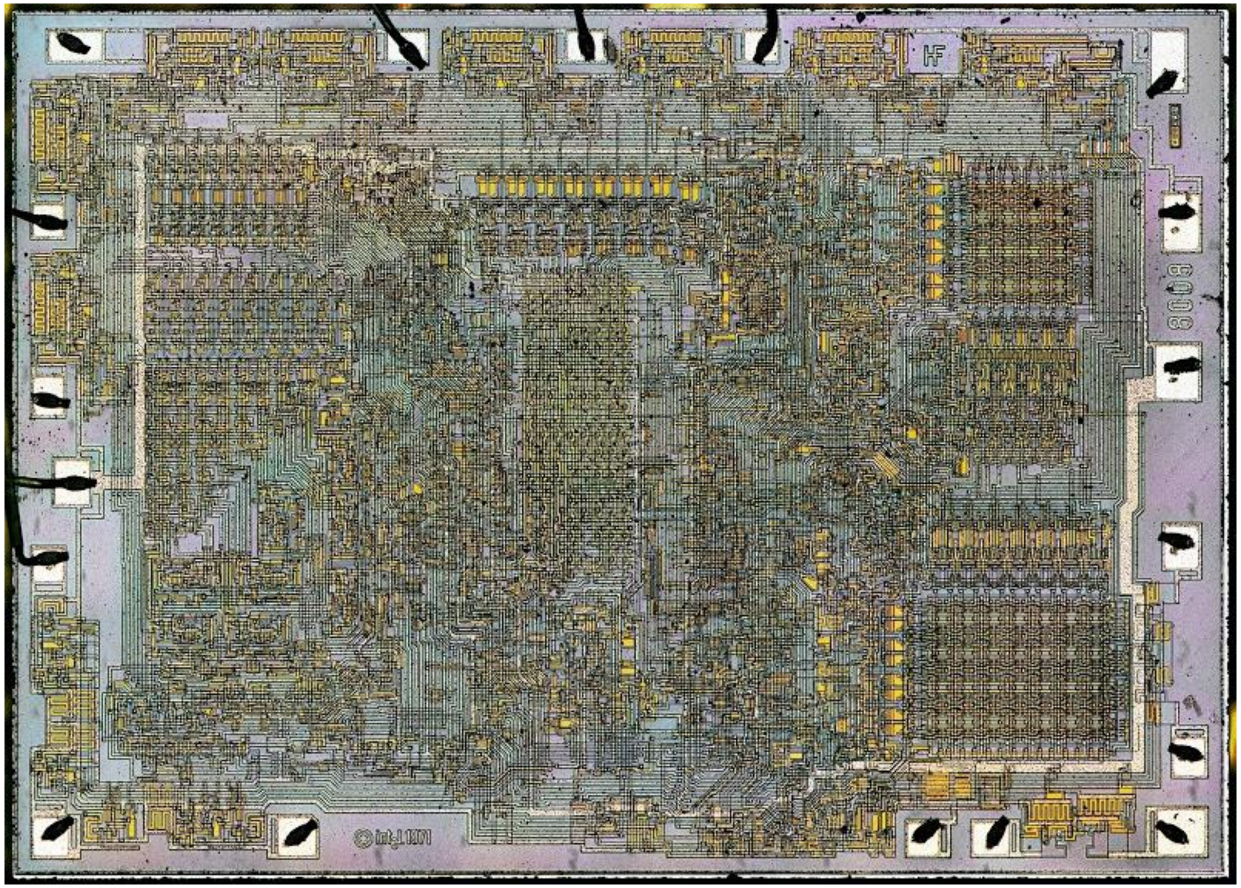


http://www.righto.com/2017/01/die-photos-and-reverse-engineering.html

74181 ALU die

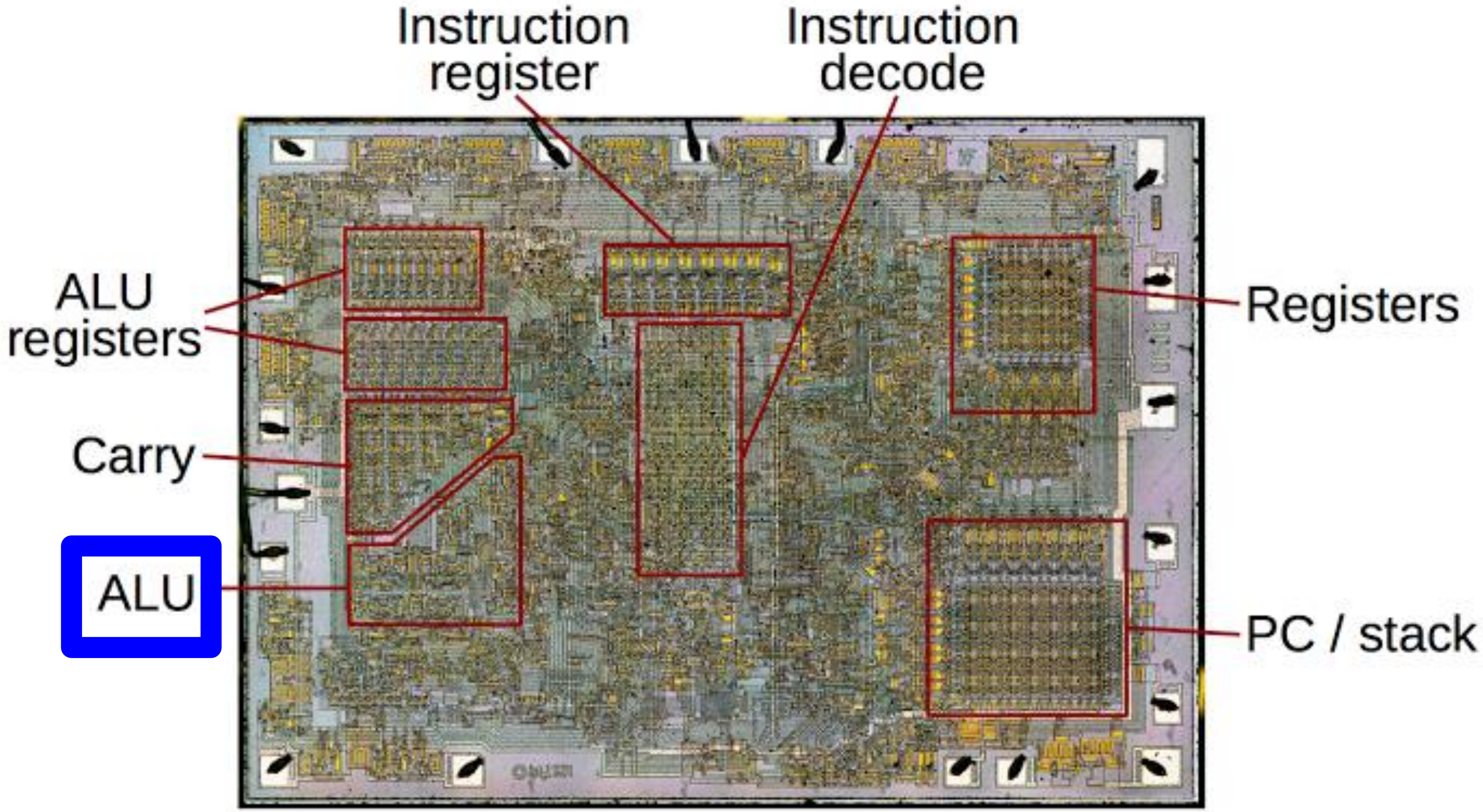


INTEL 8008 8-bit CPU

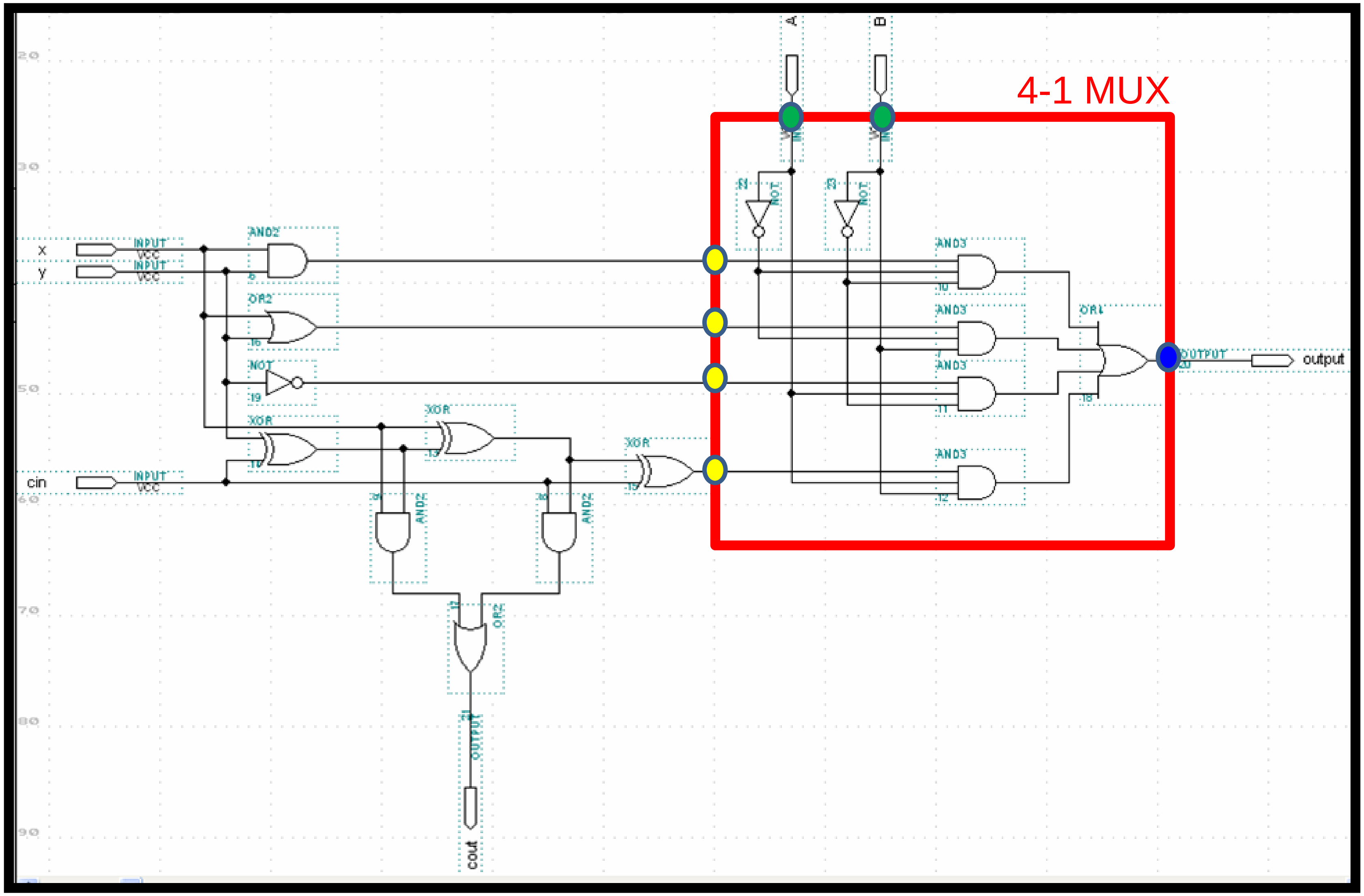


http://www.righto.com/2016/12/die-photos-and-analysis-of_24.html

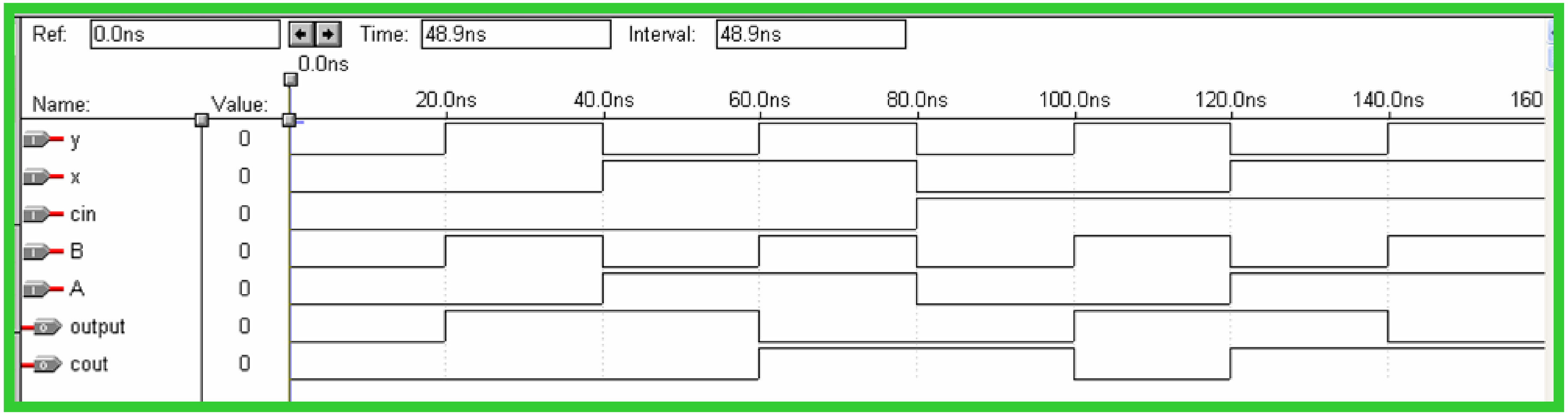
ALU


Instruction register
Instruction decode
ALU registers
Registers
Carry
ALU
PC / stack

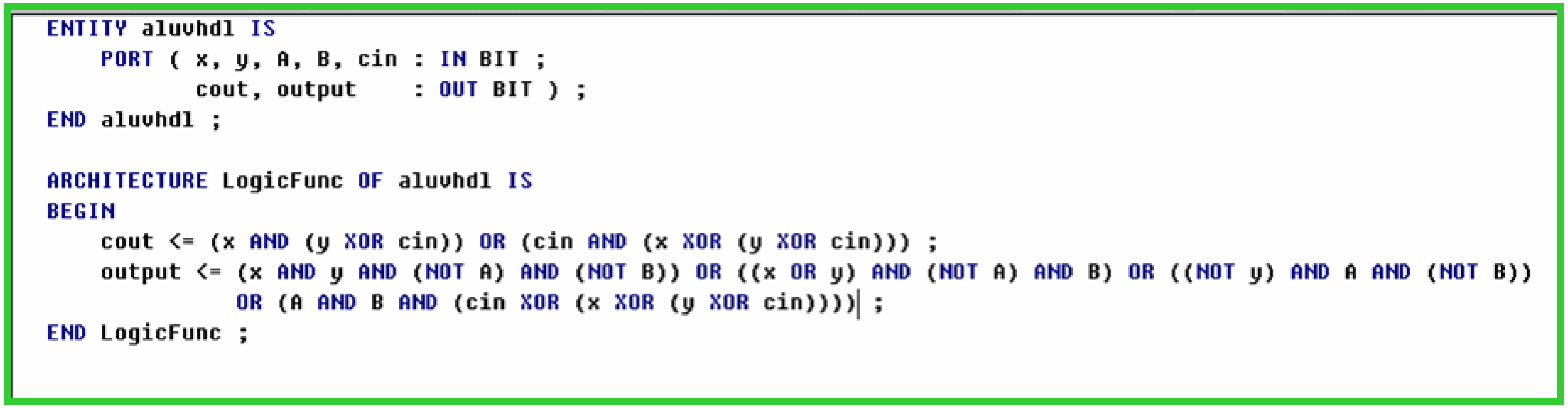
VHDL … 1-bit ALU;



Timing

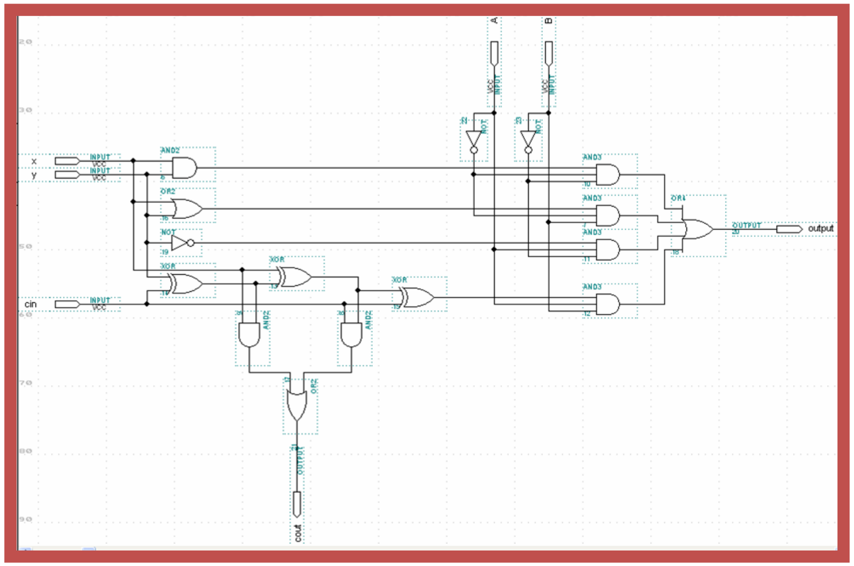

Ref: 0.0ns
Time: 48.9ns
Interval: 48.9ns
0.0ns
Name:
Value:
20.0ns
40.0ns
60.0ns
80.0ns
100.0ns
120.0ns
140.0ns
160
y 0
x 0
cin 0
B 0
A 0
output 0
cout 0

VHDL code

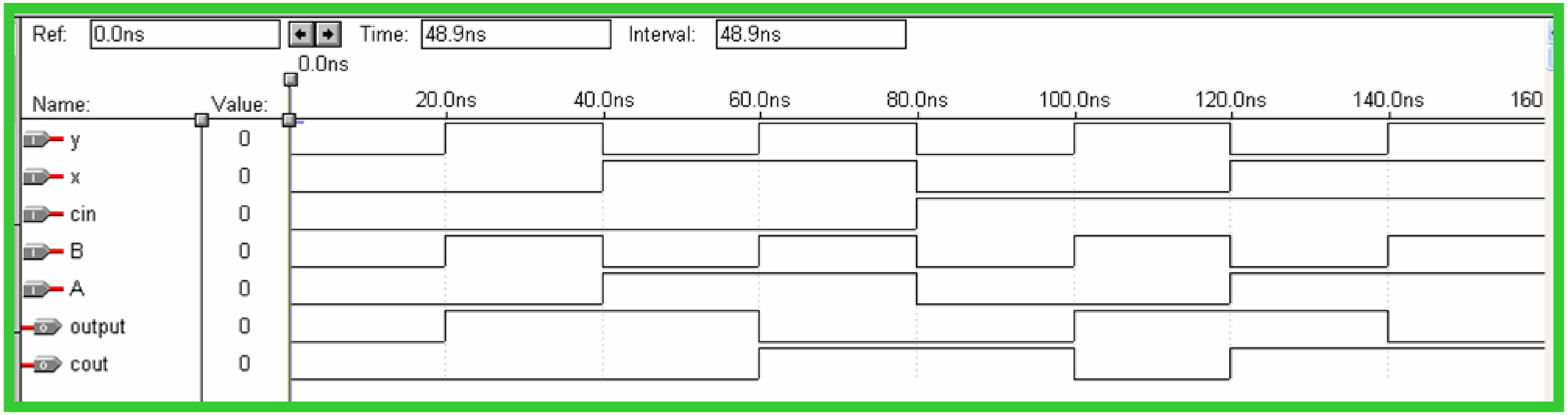


```
ENTITY aluvhdl IS
    PORT ( x, y, A, B, cin : IN BIT ;
           cout, output    : OUT BIT ) ;
END aluvhdl ;

ARCHITECTURE LogicFunc OF aluvhdl IS
BEGIN
    cout <= (x AND (y XOR cin)) OR (cin AND (x XOR (y XOR cin))) ;
    output <= (x AND y AND (NOT A) AND (NOT B)) OR ((x OR y) AND (NOT A) AND B) OR ((NOT y) AND A AND (NOT B))
              OR (A AND B AND (cin XOR (x XOR (y XOR cin)))) ;
END LogicFunc ;
```



… are the same


Ref: 0.0ns
Time: 48.9ns
Interval: 48.9ns
0.0ns
Name:
Value:
20.0ns
40.0ns
60.0ns
80.0ns
100.0ns
120.0ns
140.0ns
160
y 0
x 0
cin 0
B 0
A 0
output 0
cout 0