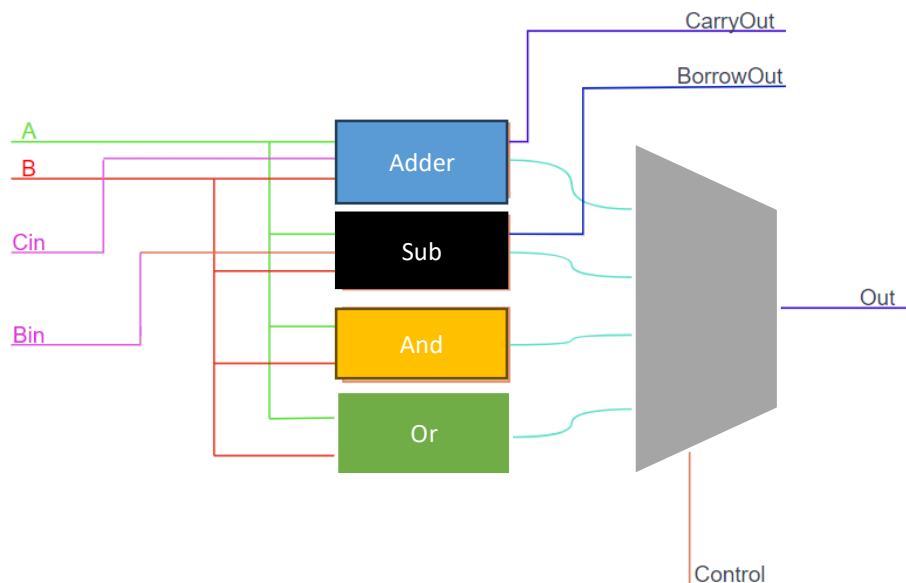


Họ và tên : Dương Minh Vương
MSV : 22022156

Báo cáo tuần 5-6

I. ALU 2 bit 1. Đặc tả chức năng

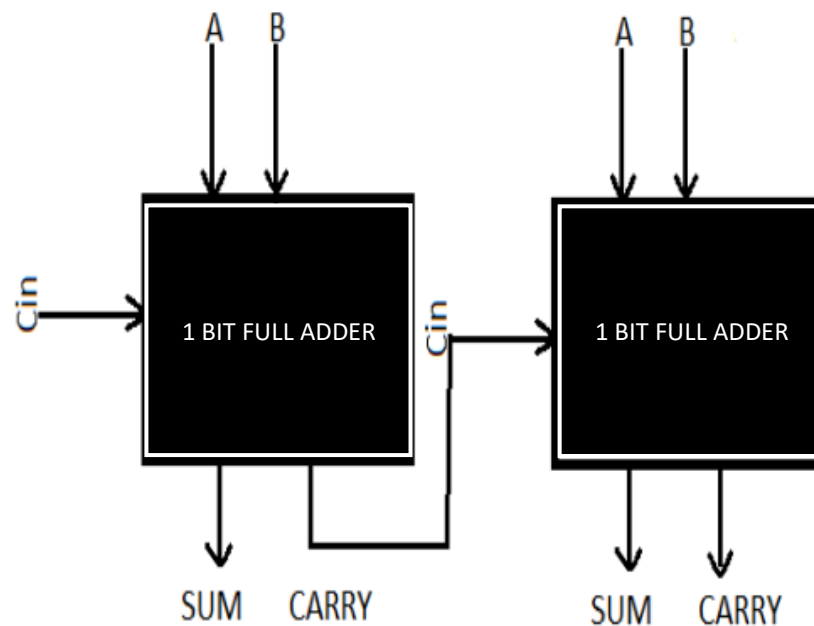


-ALU (Arithmetic Logic Unit) là một phần quan trọng của vi xử lý (CPU), chịu trách nhiệm thực hiện các phép toán số học (cộng, trừ, nhân, chia) và các phép toán logic (AND, OR, NOT, XOR).

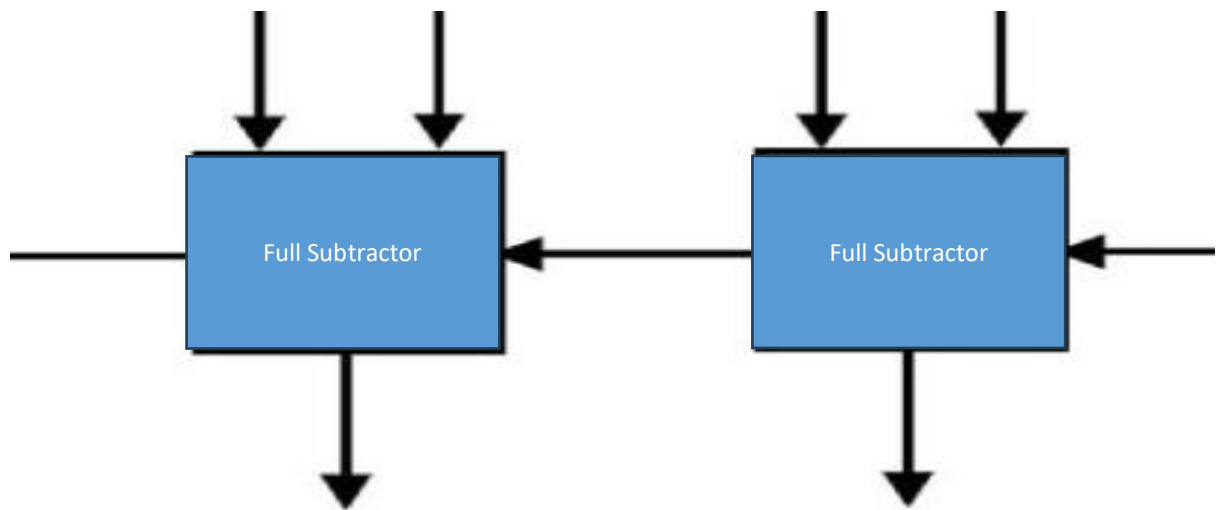
-Trong mạch này ALU thực hiện các phép toán Cộng Trừ, AND , OR

Đầu vào	Đầu ra
<ul style="list-style-type: none">+A: Đầu vào số thứ nhất (2 bit).+ B: Đầu vào số thứ hai (2 bit).+ Cin: Bit mang vào (sử dụng trong phép cộng).+ Bin: Bit vay vào (sử dụng trong phép trừ).+ Control : Tín hiệu điều khiển phép toán nào được thực hiện .Cụ thể<ul style="list-style-type: none">Control =00 : Đầu ra là kết quả phép cộngControl =01 : Đầu ra là kết quả phép trừControl =10 : Đầu ra là kết quả phép AndControl =11 : Đầu ra là kết quả phép Or	<ul style="list-style-type: none">+ CarryOut : Bit nhớ của phép cộng+ BorrowOut : Bit nhớ của phép trừ+Out (Result) : Lỗi ra của ALU

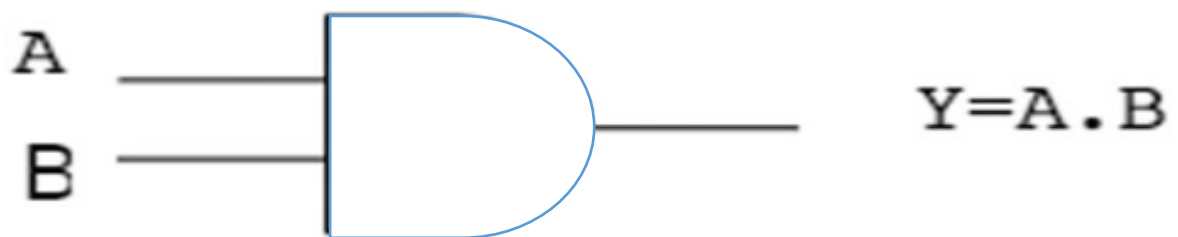
FullAdder2Bit: Đây là module thực hiện phép cộng giữa hai số nhị phân 2-bit. Khi có thêm tín hiệu carry-in (giá trị nhớ từ phép tính trước đó), FullAdder sẽ cộng thêm giá trị này vào, tạo ra kết quả chính xác. Sau khi tính toán, kết quả cộng được lưu vào Sum, và tín hiệu CarryOut sẽ cho biết liệu phép toán có tạo ra một giá trị nhớ (carry) từ bit cao nhất hay không.



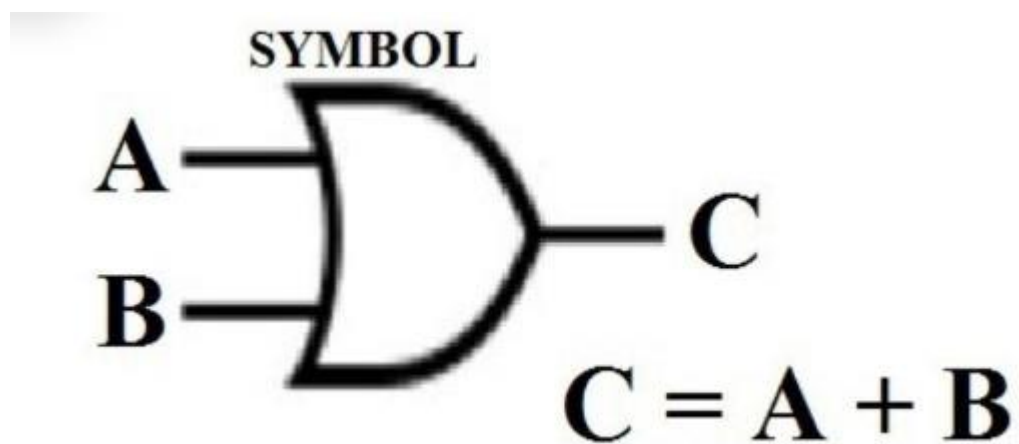
Subtractor2Bit: Đây là module thực hiện phép trừ giữa hai số 2-bit. Subtractor sẽ tính toán hiệu số của hai đầu vào A và B, và nếu cần, sẽ mượn giá trị từ phép toán trước đó (borrow-in). Kết quả của phép trừ được lưu vào Diff, và tín hiệu BorrowOut cho biết nếu có cần phải mượn khi thực hiện phép trừ.



AND_Gate thực hiện phép nhân từng bit nhị phân tương ứng của A và B



OR_Gate thực hiện phép cộng từng bit nhị phân tương ứng của A và B



BẢNG CHÂN LÍ

Control (2bit)	Phép toán	A (2bit)	B (2bit)	Cin/Bin	Đầu ra		
					Result	CarryOut	BorrowOut
00	ADDER	01	10	0	11	0	-
		11	01	1	00	1	-
01	SUB	11	10	0	01	-	0
		10	10	1	00	-	1
10	AND	10	01	-	00	-	-
		11	01	-	01	-	-
11	OR	00	11	-	11	-	-
		10	01	-	11	-	-

2. Thiết kế trên ModelSim

FullAdder2Bit.sv

```

module FullAdder1Bit(
    input logic A,
    input logic B,
    input logic Cin,
    output logic Sum,
    output logic Carry
);
    assign Sum = A ^ B ^ Cin;
    assign Carry = (A & B) | (Cin & (A ^ B));
endmodule

module FullAdder2Bit(
    input logic [1:0] A,
    input logic [1:0] B,
    input logic Cin,
    output logic [1:0] Sum,
    output logic Carry
);
    logic Carry0;

    FullAdder1Bit fa0(
        .A(A[0]),
        .B(B[0]),
        .Cin(Cin),
        .Sum(Sum[0]),
        .Carry(Carry0)
    );

    FullAdder1Bit fa1(
        .A(A[1]),
        .B(B[1]),
        .Cin(Carry0),
        .Sum(Sum[1]),
        .Carry(Carry)
    );
endmodule

```

Subtractor2Bit.sv	<pre> module Subtractor1Bit(input logic A, input logic B, input logic Bin, output logic Diff, output logic Borrow); assign Diff = A ^ B ^ Bin; assign Borrow = (~A & (B ^ Bin)) (B & Bin); endmodule module Subtractor2Bit(input logic [1:0] A, input logic [1:0] B, input logic Bin, output logic [1:0] Diff, output logic Borrow); logic Borrow0; Subtractor1Bit sub0(.A(A[0]), .B(B[0]), .Bin(Bin), .Diff(Diff[0]), .Borrow(Borrow0)); Subtractor1Bit sub1(.A(A[1]), .B(B[1]), .Bin(Borrow0), .Diff(Diff[1]), .Borrow(Borrow)); endmodule </pre>
AND_Gate.sv	<pre> module AND_Gate(input logic [1:0] A, input logic [1:0] B, output logic [1:0] Out); assign Out = A & B; endmodule </pre>
OR_Gate.sv	<pre> module OR_Gate(input logic [1:0] A, input logic [1:0] B, output logic [1:0] Out); assign Out = A B; endmodule </pre>

ALU.sv

```
module ALU(  
    input logic [1:0] A,  
    input logic [1:0] B,  
    input logic Cin,  
    input logic Bin,  
    input logic [1:0] Control,  
    output logic [1:0] Result,  
    output logic Carry,  
    output logic Borrow  
);  
  
    logic [1:0] Sum;  
    logic [1:0] Diff;  
    logic [1:0] OrResult;  
    logic [1:0] AndResult;  
    logic CarryOut;  
    logic BorrowOut;  
  
    FullAdder2Bit adder(  
        .A(A),  
        .B(B),  
        .Cin(Cin),  
        .Sum(Sum),  
        .Carry(CarryOut)  
    );  
  
    Subtractor2Bit subtractor(  
        .A(A),  
        .B(B),  
        .Bin(Bin),  
        .Diff(Diff),  
        .Borrow(BorrowOut)  
    );  
  
    OR_Gate or_gate_module(  
        .A(A),  
        .B(B),  
        .Out(OrResult)
```

```

OR_Gate or_gate_module(
    .A(A),
    .B(B),
    .Out(OrResult)
);

AND_Gate and_gate_module(
    .A(A),
    .B(B),
    .Out(AndResult)
);

always_comb begin
    case (Control)
        2'b00: begin
            Result = Sum;
            Carry = CarryOut;
            Borrow = 1'b0;
        end
        2'b01: begin
            Result = Diff;
            Borrow = BorrowOut;
            Carry = 1'b0;
        end
        2'b10: begin
            Result = AndResult;
            Carry = 1'b0;
            Borrow = 1'b0;
        end
        2'b11: begin
            Result = OrResult;
            Carry = 1'b0;
            Borrow = 1'b0;
        end
        default: begin
            Result = 2'b00;
            Carry = 1'b0;
            Borrow = 1'b0;
        end
    endcase
end
endmodule

```

ALU_Testbench.sv

```

module ALU_Testbench;
    logic [1:0] A;
    logic [1:0] B;
    logic Cin;
    logic Bin;
    logic [1:0] Control;
    logic [1:0] Result;
    logic Carry;
    logic Borrow;

    ALU alu(
        .A(A),
        .B(B),
        .Cin(Cin),
        .Bin(Bin),
        .Control(Control),
        .Result(Result),
        .Carry(Carry),
        .Borrow(Borrow)
    );

    initial begin
        A = 2'b01; B = 2'b10; Cin = 1'b0; Bin = 1'b0; Control = 2'b00;
        #10;

        A = 2'b11; B = 2'b01; Cin = 1'b1; Bin = 1'b0; Control = 2'b00;
        #10;

        A = 2'b11; B = 2'b10; Cin = 1'b0; Bin = 1'b0; Control = 2'b01;
        #10;

        A = 2'b10; B = 2'b10; Cin = 1'b1; Bin = 1'b0; Control = 2'b01;
        #10;

        A = 2'b10; B = 2'b01; Cin = 1'b0; Bin = 1'b0; Control = 2'b10;
        #10;

        A = 2'b00; B = 2'b11; Cin = 1'b0; Bin = 1'b0; Control = 2'b11;
        #10;

        A = 2'b11; B = 2'b11; Cin = 1'b1; Bin = 1'b1; Control = 2'b00;
        #10;

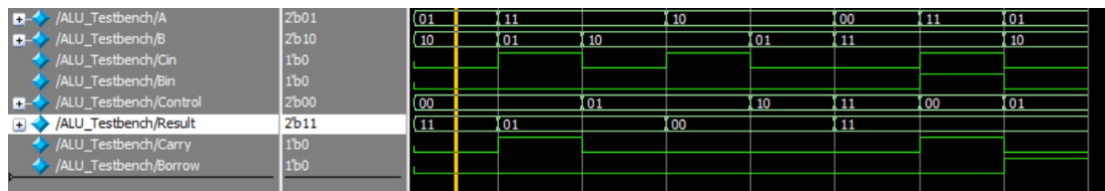
        A = 2'b01; B = 2'b10; Cin = 1'b0; Bin = 1'b0; Control = 2'b01;
        #10;

        $finish;
    end
endmodule

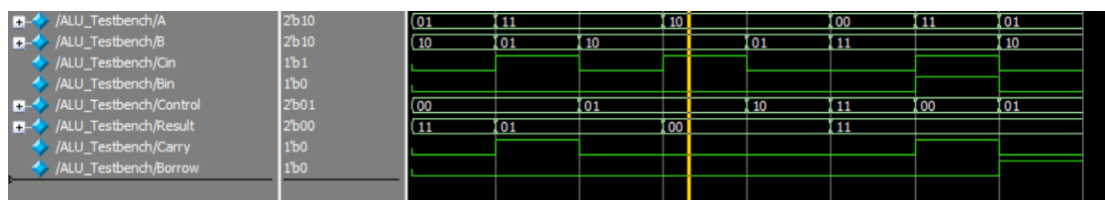
```

3. Mô phỏng

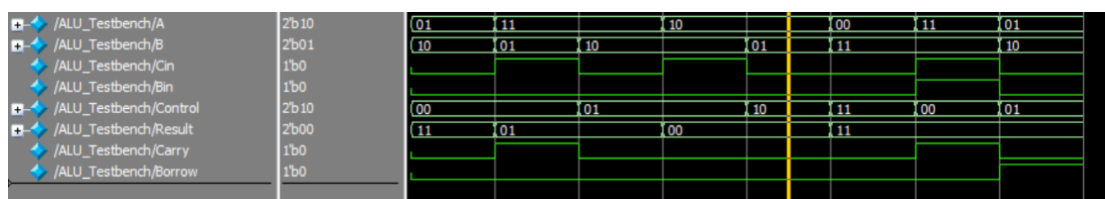
Trường hợp cộng



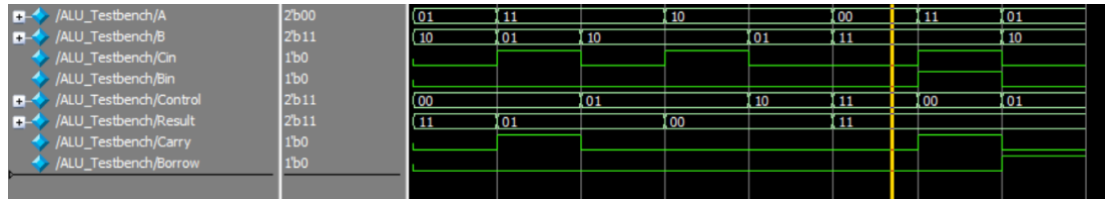
Trường hợp trừ



Trường hợp AND

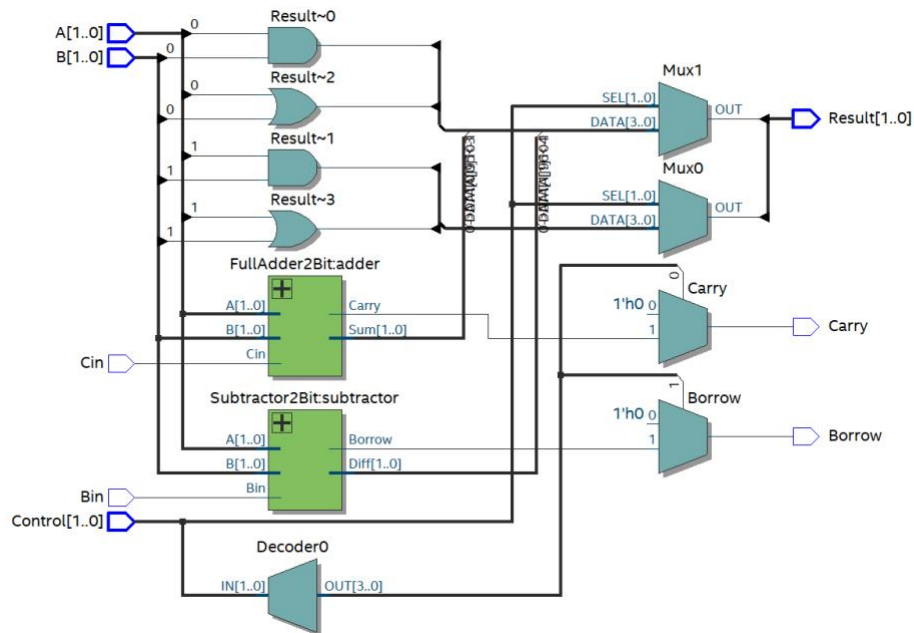


Trường hợp OR

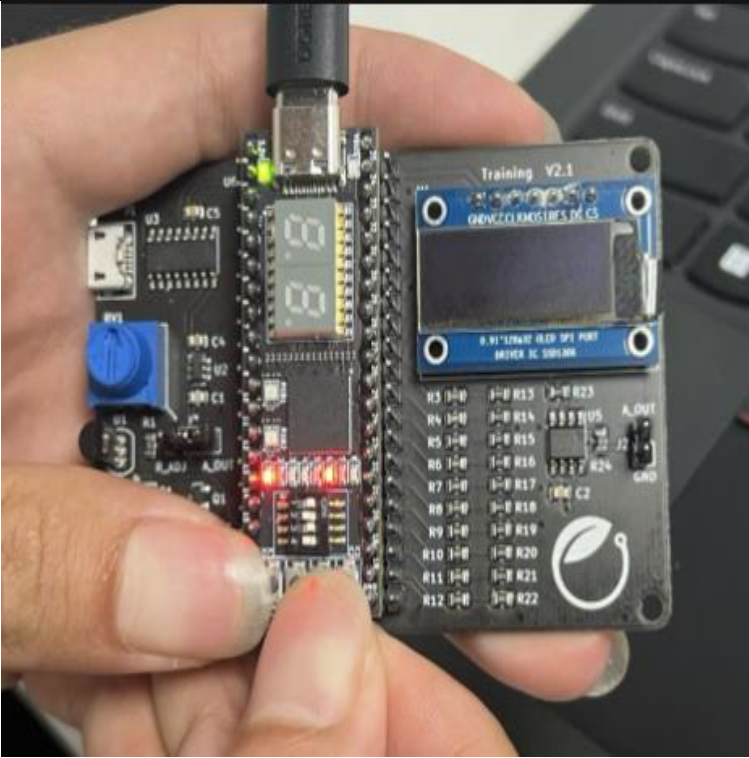
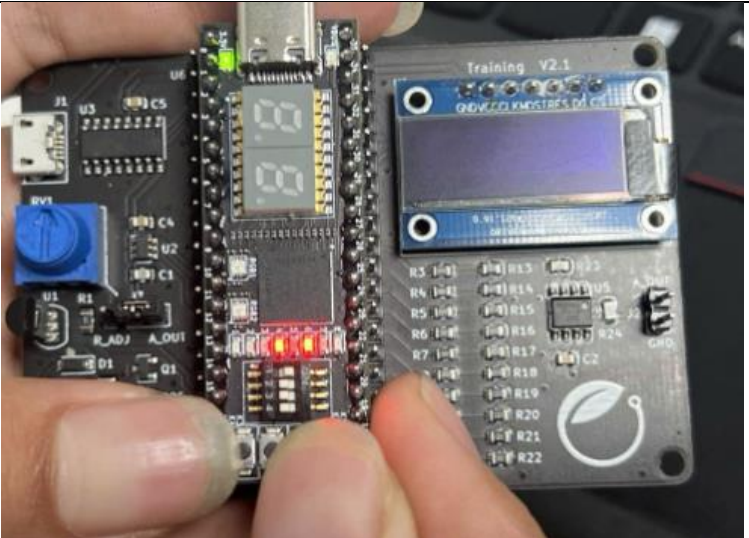


4. Tổng hợp trên Quartus

RTL Viewer



5. Kiểm thử trên bo mạch

Đầu vào	Ảnh
<p>Control = 00 (Cộng) A = 10 B = 10 Cin = 1 => Result = 01 , CarryOut = 1</p>	
<p>Control = 00 (Cộng) A = 10 B = 00 Cin = 1 => Result = 11 , CarryOut = 0</p>	

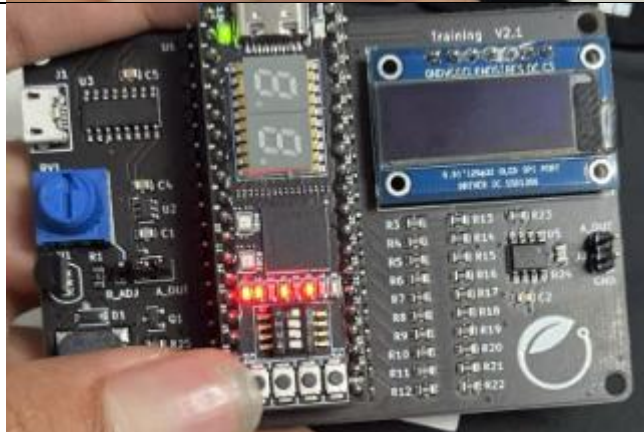
Control = 01 (Trừ)
A = 11
B = 10
Cin = 1
=> Result = 00 ,
BorrowOut = 0



Control = 01 (Trừ)
A = 10
B = 10
Cin = 1
=> Result = 11,
BorrowOut = 1



Control = 10 (AND)
A = 10
B = 00
=> Result = 00

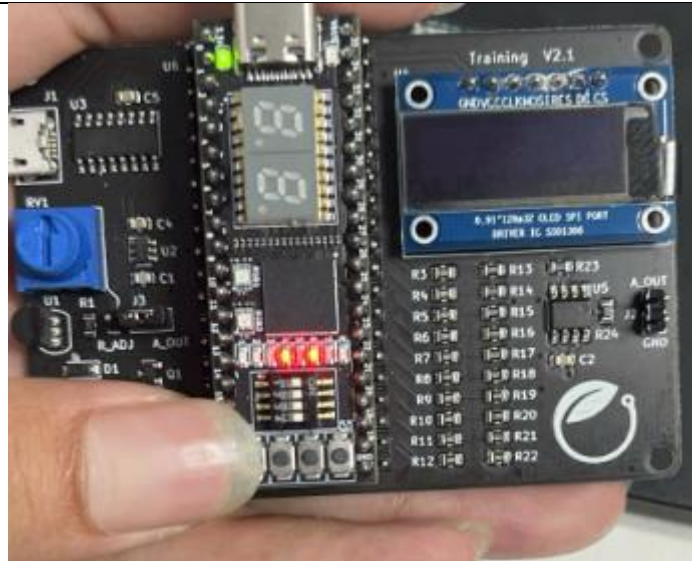


Control = 10 (AND)

A = 11

B = 11

=> Result = 11



Control = 11 (OR)

A = 10

B = 10

=> Result = 10



Control = 11 (OR)

A = 00

B = 00

=> Result = 00

