

**ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH**

**ĐẠI HỌC BÁCH KHOA HỒ CHÍ MINH**

**KHOA ĐIỆN-ĐIÊN TỬ**

**BỘ MÔN ĐIỆN TỬ**

-----o0o-----



**THIẾT KẾ VI MẠCH**  
**BÁO CÁO TKVM SỐ LAB 1:**  
**THIẾT KẾ BỘ ALU**

GVHD: PGS.TS Hoàng Trang

TGHD: Đỗ Quang Thịnh

NTH : Lớp L05 – Nhóm 06

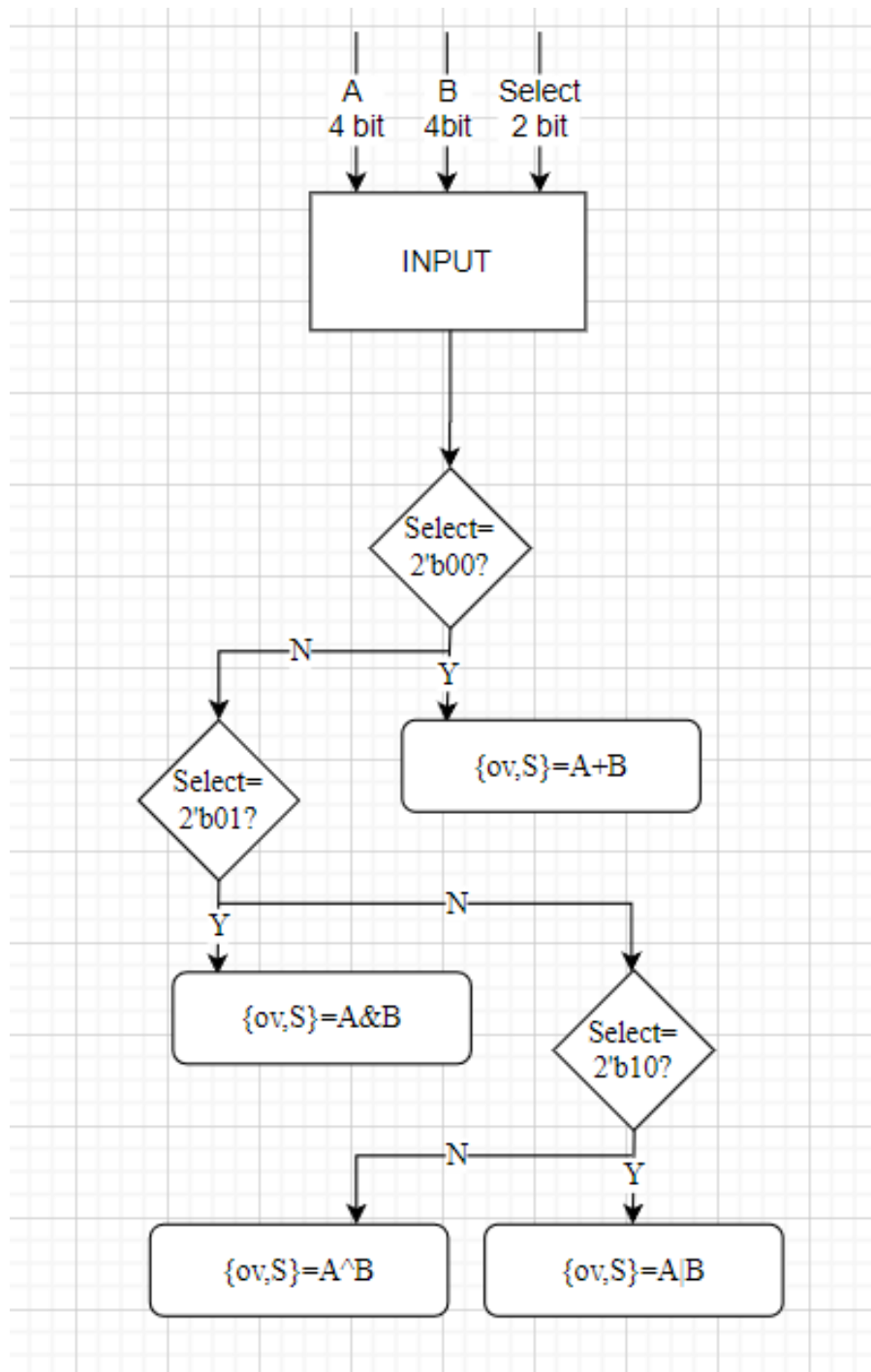
STT	Họ và tên	MSSV	Email
1	Nguyễn Phan Vĩnh Khang (0915 282 427)	1910242	khang.nguyennpvk@hcmut.edu.vn
2	Nguyễn Quang Minh	1911612	minh.nguyen711@hcmut.edu.vn
3	Phạm Hồng Thái	1915119	thai.phamhongthai10@hcmut.edu.vn
4	Nguyễn Văn Thăng	1910543	thang.nguyen28@hcmut.edu.vn
5	Huỳnh Phú Cường	1912821	cuong.huynh922035@hcmut.edu.vn

Thành phố Hồ Chí Minh, ngày 28 tháng 5 năm 2022

## 1. Tổng quan lý thuyết

### 1.1. Thiết kế cấp độ Specification

#### A. Sơ đồ khối bộ ALU



Sơ đồ khối bộ ALU

## B.Cách thức hoạt động của bộ ALU

### -Khối input:

- +A, B: 2 dữ liệu đầu vào 4 bit.
- +Select(cmd): Tín hiệu control 2 bit,

### -Khối output:

- + Select = 2'b00: Ngõ ra là phép cộng  $A + B$ .
- + Select = 2'b01: Ngõ ra là phép A AND B ( $A \& B$ )
- + Select = 2'b10: Ngõ ra là phép A OR B ( $A | B$ )
- + Select = 2'b11: Ngõ ra là phép A XOR B ( $A \wedge B$ )

## C.Ý tưởng thực thi thiết kế

- Việc đầu tiên là hình thành sơ đồ khối của thiết kế
- Hiểu được cách thực hiện các phép toán adder, and, or, xor và phép mở rộng bit.
- Viết khối ALU dựa vào input và chọn output từ tín hiệu điều khiển cmd.

## 2.Thực hiện thiết kế

### 2.1.Mạch thiết kế cấp cổng dùng Verilog

#### A.Mã code

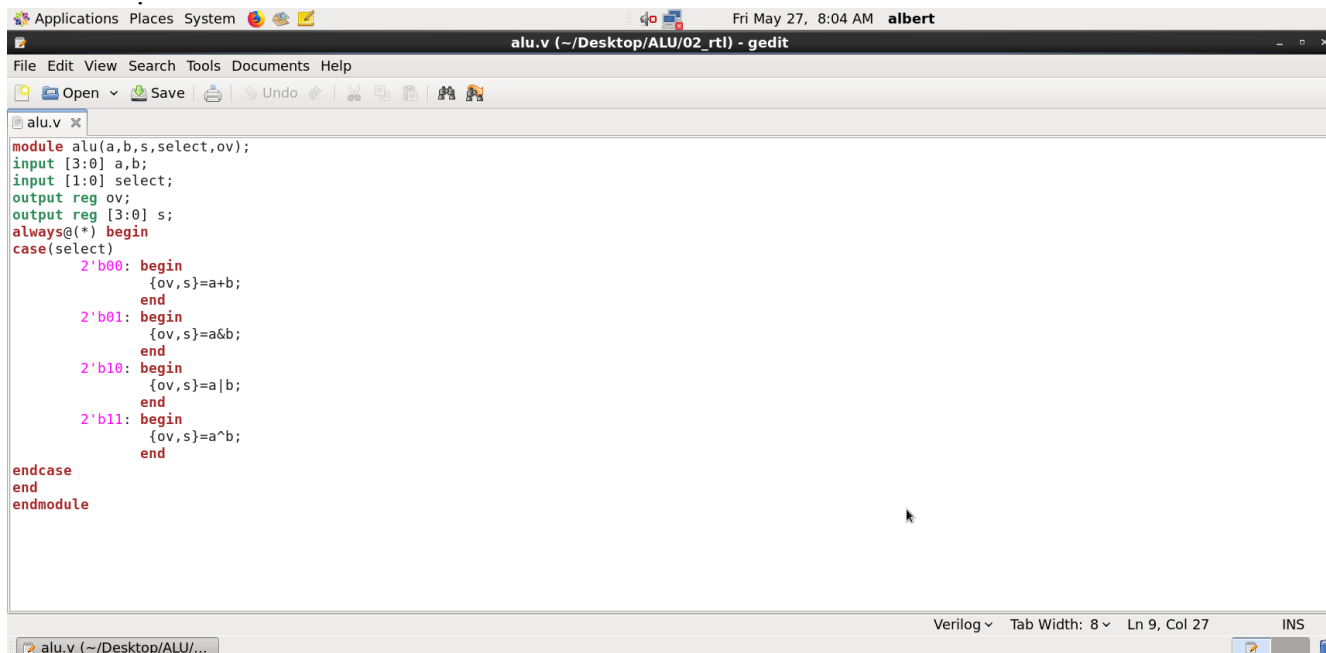
##### \*Khối alu:

##### -Input:

- + a,b: 4 bit ngõ vào
- +select:2 bit ngõ vào

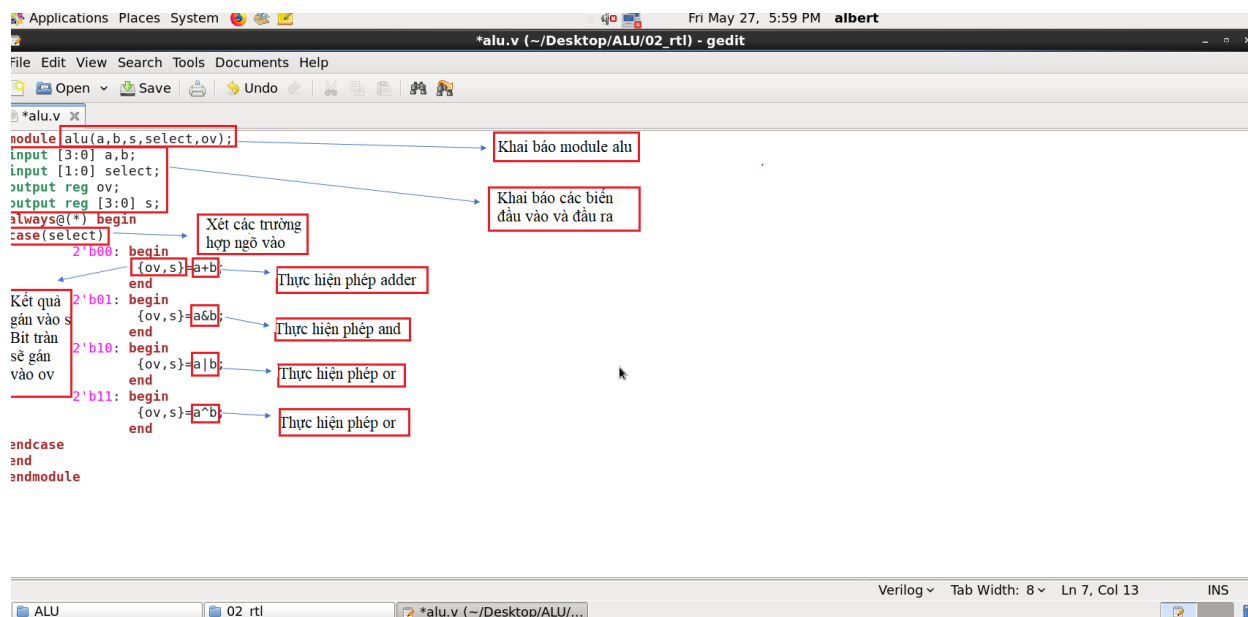
##### -Output

- +s: 4 bit ngõ ra
- +ov:1 bit ngõ ra



```
module alu(a,b,s,select,ov);
input [3:0] a,b;
input [1:0] select;
output reg ov;
output reg [3:0] s;
always@(*) begin
case(select)
2'b00: begin
{ov,s}=a+b;
end
2'b01: begin
{ov,s}=a&b;
end
2'b10: begin
{ov,s}=a|b;
end
2'b11: begin
{ov,s}=a^b;
end
endcase
end
endmodule
```

## B.Giải thích code



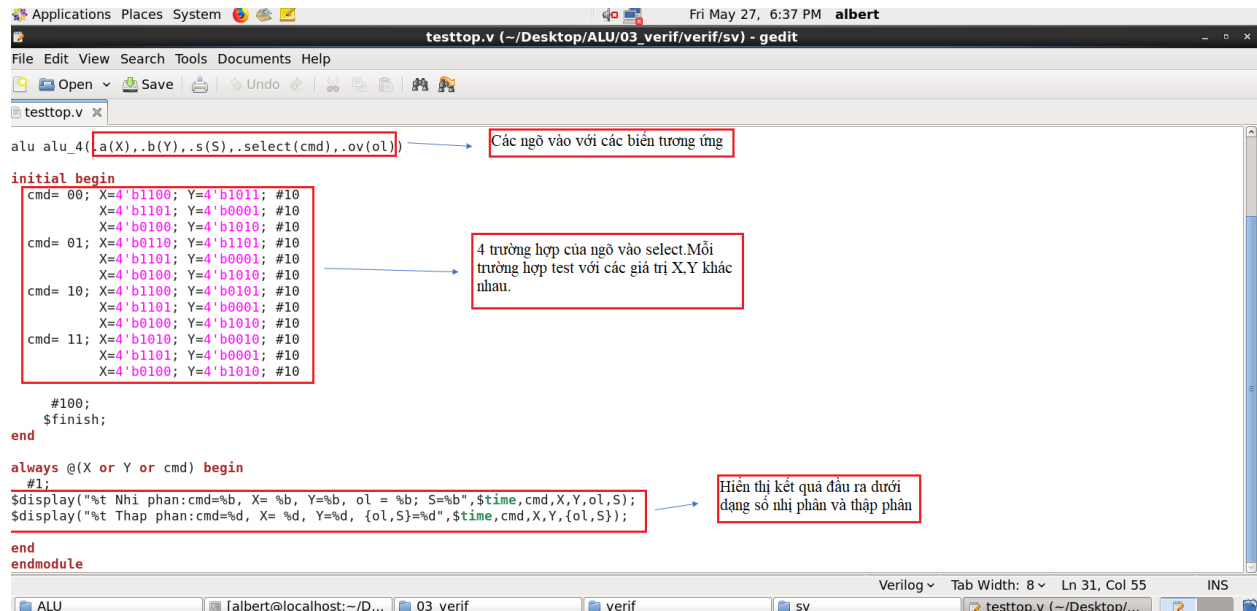
```
module alu(a,b,s,select,ov);
input [3:0] a,b;
input [1:0] select;
output reg ov;
output reg [3:0] s;
always@(*) begin
case(select)
2'b00: begin
{ov,s}=a+b;
end
2'b01: begin
{ov,s}=a&b;
end
2'b10: begin
{ov,s}=a|b;
end
2'b11: begin
{ov,s}=a^b;
end
endcase
end
endmodule
```

Annotations:

- Khái báo module alu
- Khái báo các biến đầu vào và đầu ra
- Xét các trường hợp ngõ vào
- Thực hiện phép adder
- Thực hiện phép and
- Thực hiện phép or
- Thực hiện phép or
- Kết quả gán vào s. Bit tràn sẽ gán vào ov

## 2.2.Kiểm định RTL

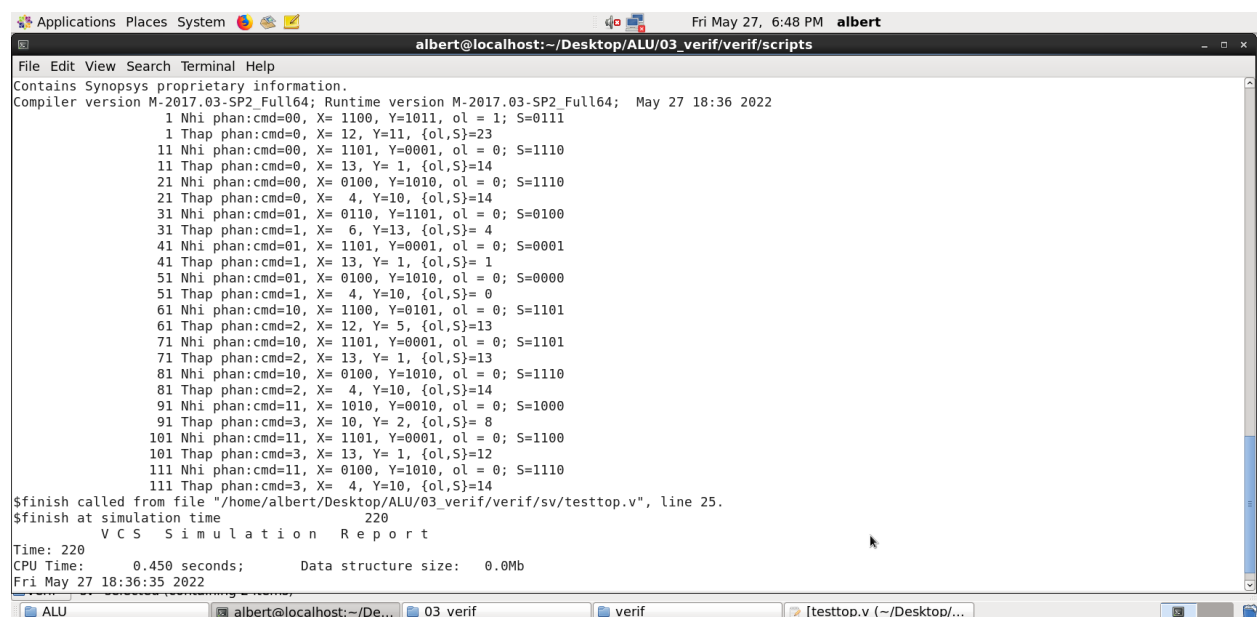
### A.File testtop.sv



The screenshot shows a Verilog code editor window titled 'testtop.v (~/Desktop/ALU/03\_verif/verif/sv) - gedit'. The code defines an ALU module with inputs X, Y, and S, and output ol. It includes an initial block for setting initial values and an always block for performing operations based on the command (cmd). Annotations in red boxes point to specific parts of the code:

- Annotation 1: Points to the input selection logic: `alu alu_4(a(X),.b(Y),.s(S),.select(cmd),.ov(ol))`. Text: "Các ngõ vào với các biến tương ứng".
- Annotation 2: Points to the initial block. Text: "4 trường hợp của ngõ vào select. Mỗi trường hợp test với các giá trị X,Y khác nhau."
- Annotation 3: Points to the display statements: `$display("%t Nhi phan:cmd=%b, X= %b, Y=%b, ol = %b; S=%b", $time, cmd, X, Y, ol, S);` and `$display("%t Thap phan:cmd=%d, X= %d, Y=%d, {ol,S}=%d", $time, cmd, X, Y, {ol,S});`. Text: "Hiện thị kết quả đầu ra dưới dạng số nhị phân và thập phân".

## B. Màn hình terminal sau khi chạy makefile

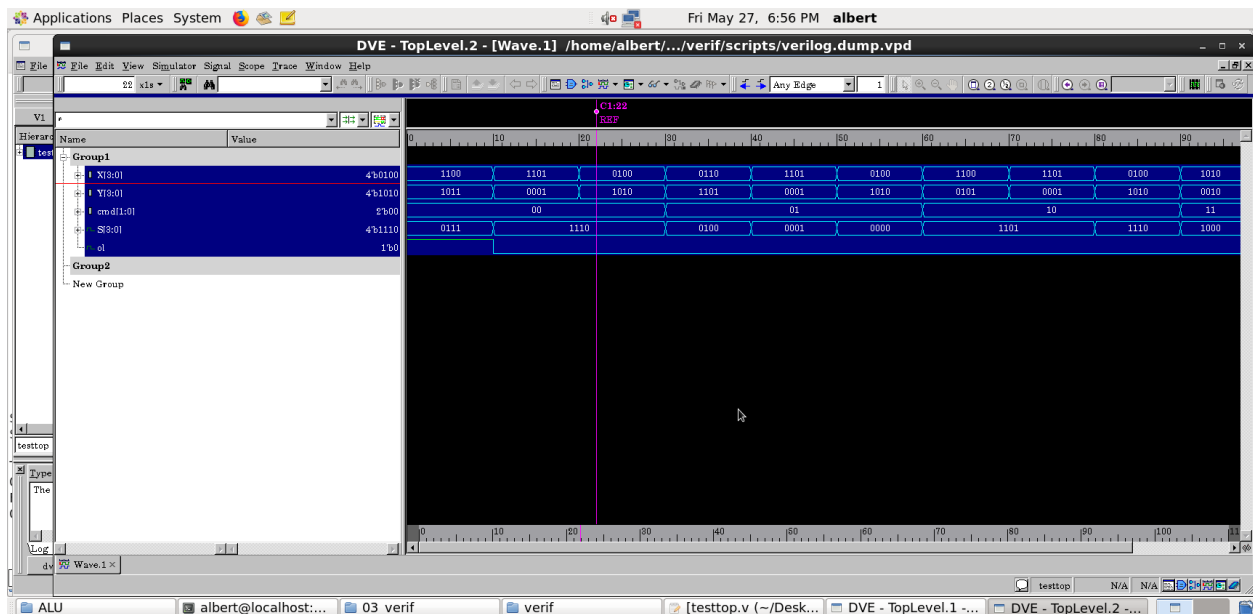


The screenshot shows a terminal window titled 'albert@localhost:~/Desktop/ALU/03\_verif/scripts'. It displays the output of a Verilog simulation. The output includes the compiler version, the simulation time, and the results of the operations performed by the ALU. The results are shown in a table format, with columns for the command (cmd), the inputs (X, Y), and the output (ol, S). The simulation results are as follows:

Command	X	Y	ol	S
00	1100	1011	1	0111
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101	0	1110
01	1101	0001	0	1110
10	1100	0101	0	1110
11	1101	0010	0	1110
00	1100	0101		

- + Cmd: 10 thực hiện phép OR.
- + Cmd: 11 thực hiện phép XOR.
- Cột thứ 3: giá trị input X.
- Cột thứ 4: giá trị input Y.
- Cột thứ 5: giá trị output s

### C.Dùng DVE mở dạng sóng

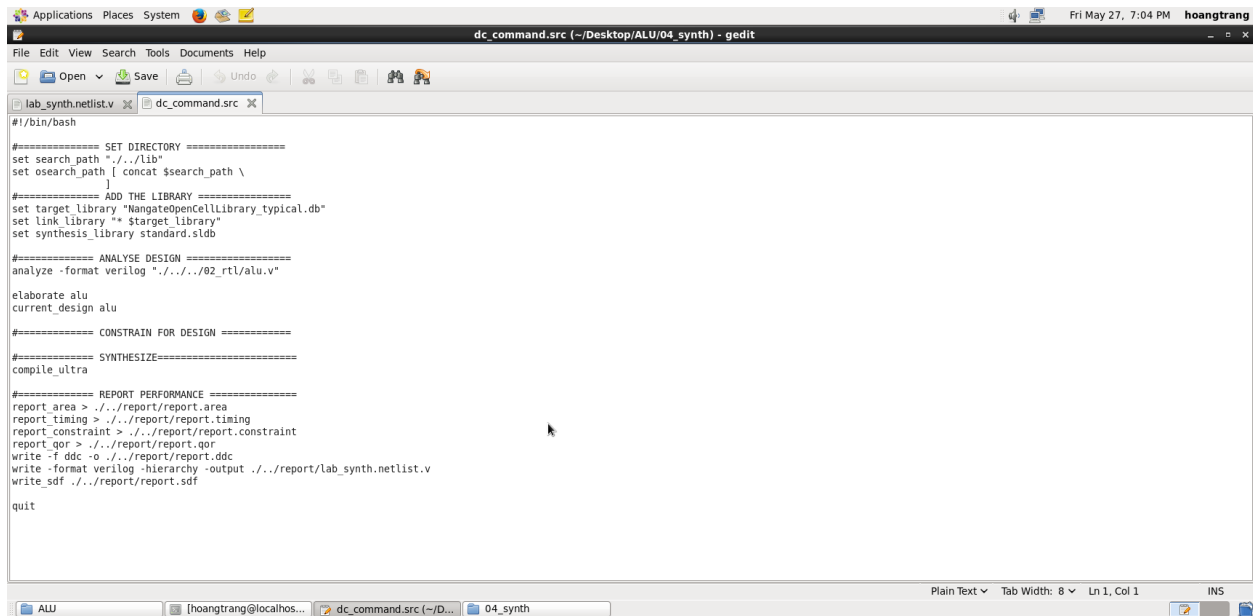


-Kết quả mô phỏng gồm 5 hàng:

- +Hàng 1:Giá trị của ngõ vào X
- +Hàng 2:Giá trị của ngõ vào Y
- +Hàng 3:Giá trị của select(cmd)
- +Hàng 4:Giá trị của ngõ ra S
- +Hàng 5:Giá trị của ngõ ra ov(overflow)

### 2.3.Synthesis tổng hợp

-File dc.command.src



```
#!/bin/bash

#===== SET DIRECTORY =====
set search_path "-./../lib"
set osearch_path [ concat $search_path \
]
#===== ADD THE LIBRARY =====
set target_library "NangateOpenCellLibrary_typical.db"
set link_library "+ target_library"
set synthesis_library standard.sldb

#===== ANALYSE DESIGN =====
analyze -format verilog "-./../02_rtl/alu.v"

elaborate alu
current_design alu

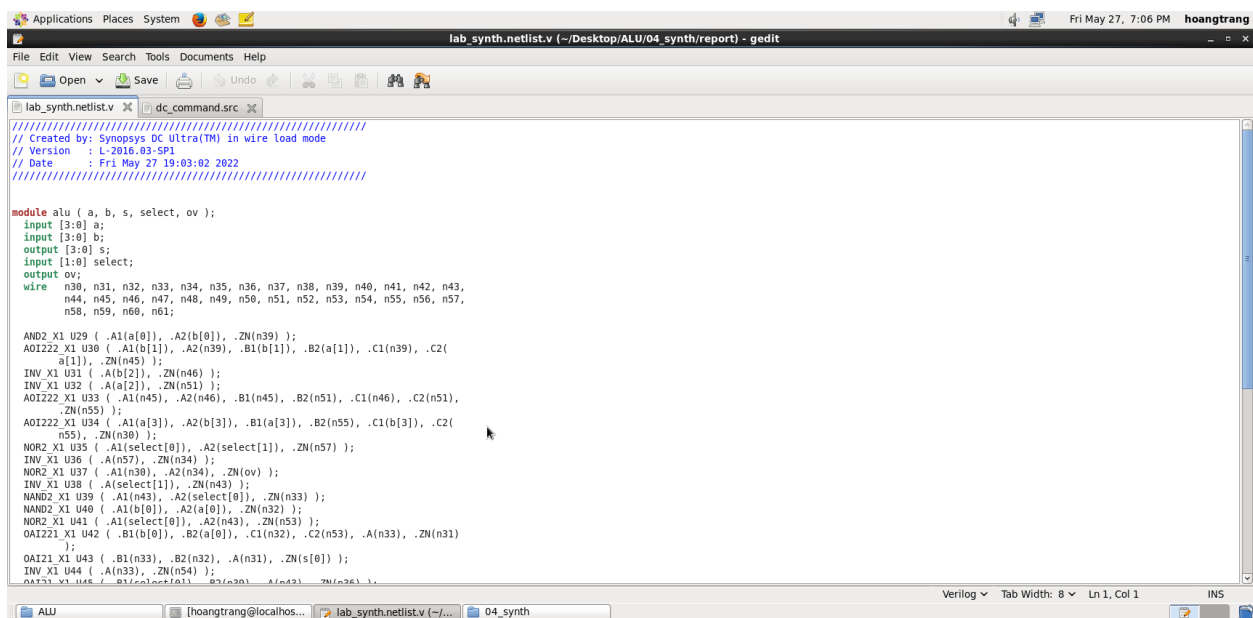
#===== CONSTRAIN FOR DESIGN =====

#===== SYNTHESIZE=====
compile_ultra

#===== REPORT PERFORMANCE =====
report_area > ../report/report.area
report_timing > ../report/report.timing
report_constraint > ../report/report.constraint
report_qor > ../report/report.qor
write -f ddc -o ../report/report.ddc
write -format verilog -hierarchy -output ../report/lab_synth.netlist.v
write_sdf ../report/report.sdf

quit
```

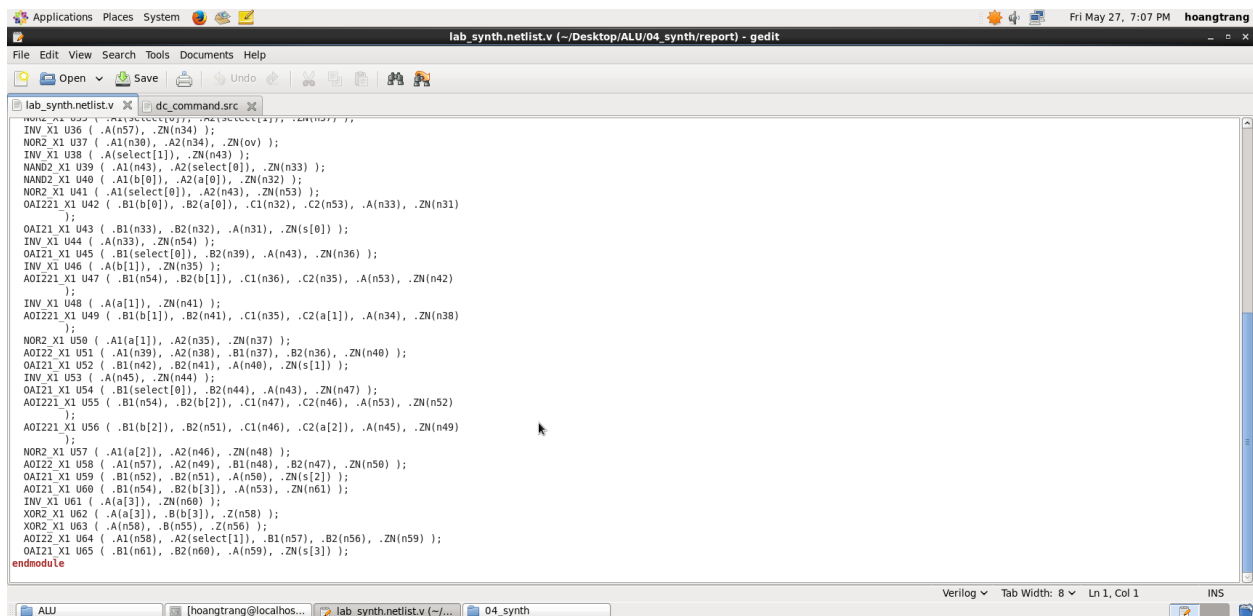
-File Verilog của bộ ALU sau khi đã Synthesis



```
// Created by: Synopsys DC Ultra(TM) in wire load mode
// Version : L-2016.03-SP1
// Date : Fri May 27 19:03:02 2022

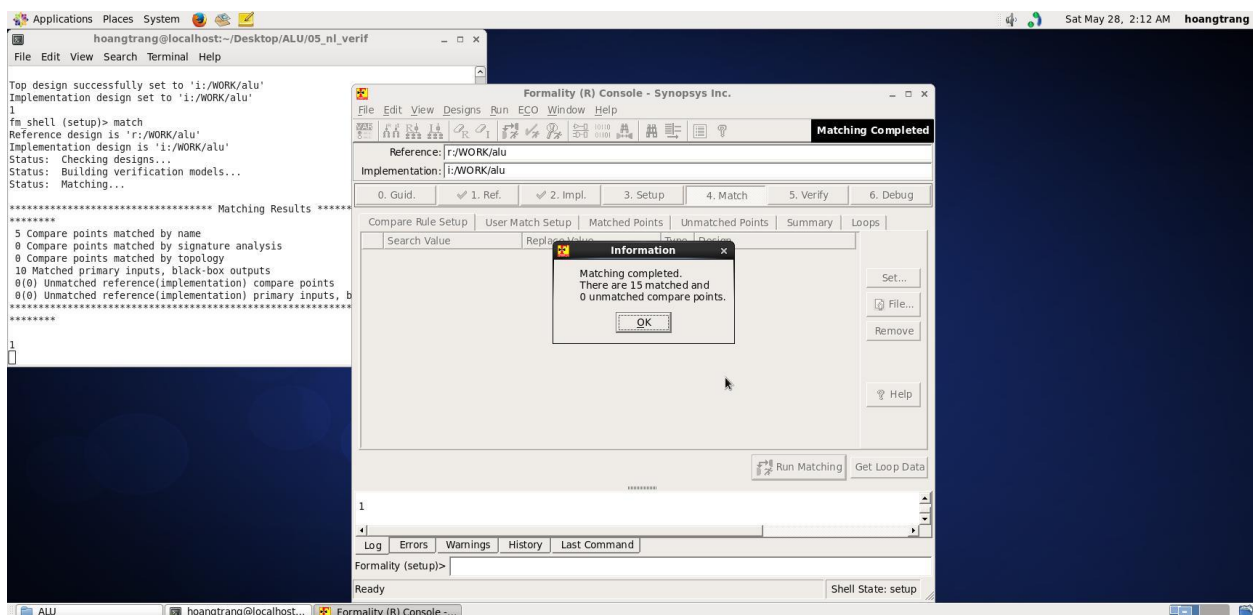
module alu ( a, b, s, select, ov );
input [3:0] a;
input [3:0] b;
output [3:0] s;
input [1:0] select;
output ov;
wire n30, n31, n32, n33, n34, n35, n36, n37, n38, n39, n40, n41, n42, n43,
n44, n45, n46, n47, n48, n49, n50, n51, n52, n53, n54, n55, n56, n57,
n58, n59, n60, n61;

AND2_X1 U29 ( .A1(a[0]), .A2(b[0]), .ZN(n39) );
AOI222_X1 U30 ( .A1(a[1]), .A2(n39), .B1(b[1]), .B2(a[1]), .C1(n39), .C2(
a[1]), .ZN(n45) );
INV_X1 U31 ( .A(b[2]), .ZN(n46) );
INV_X1 U32 ( .A(a[2]), .ZN(n51) );
AOI222_X1 U33 ( .A1(n45), .A2(n46), .B1(n45), .B2(n51), .C1(n46), .C2(n51),
.ZN(n55) );
AOI222_X1 U34 ( .A1(a[3]), .A2(b[3]), .B1(a[3]), .B2(n55), .C1(b[3]), .C2(
n55), .ZN(n30) );
NOR2_X1 U35 ( .A1(select[0]), .A2(select[1]), .ZN(n57) );
INV_X1 U36 ( .A(n57), .ZN(n34) );
NOR2_X1 U37 ( .A1(n30), .A2(n34), .ZN(ov) );
INV_X1 U38 ( .A(select[1]), .ZN(n43) );
NAND2_X1 U39 ( .A1(n43), .A2(select[0]), .ZN(n33) );
NAND2_X1 U40 ( .A1(b[0]), .A2(a[0]), .ZN(n32) );
NOR2_X1 U41 ( .A1(select[0]), .A2(n43), .ZN(n53) );
OAI221_X1 U42 ( .B1(b[0]), .B2(a[0]), .C1(n32), .C2(n53), .A(n33), .ZN(n31)
);
OAI221_X1 U43 ( .B1(n33), .B2(n32), .A(n31), .ZN(s[0]) );
INV_X1 U44 ( .A(n33), .ZN(n54) );
OAI221_X1 U45 ( .B1(select[0]), .B2(n30), .A(n43), .ZN(n36) );
```



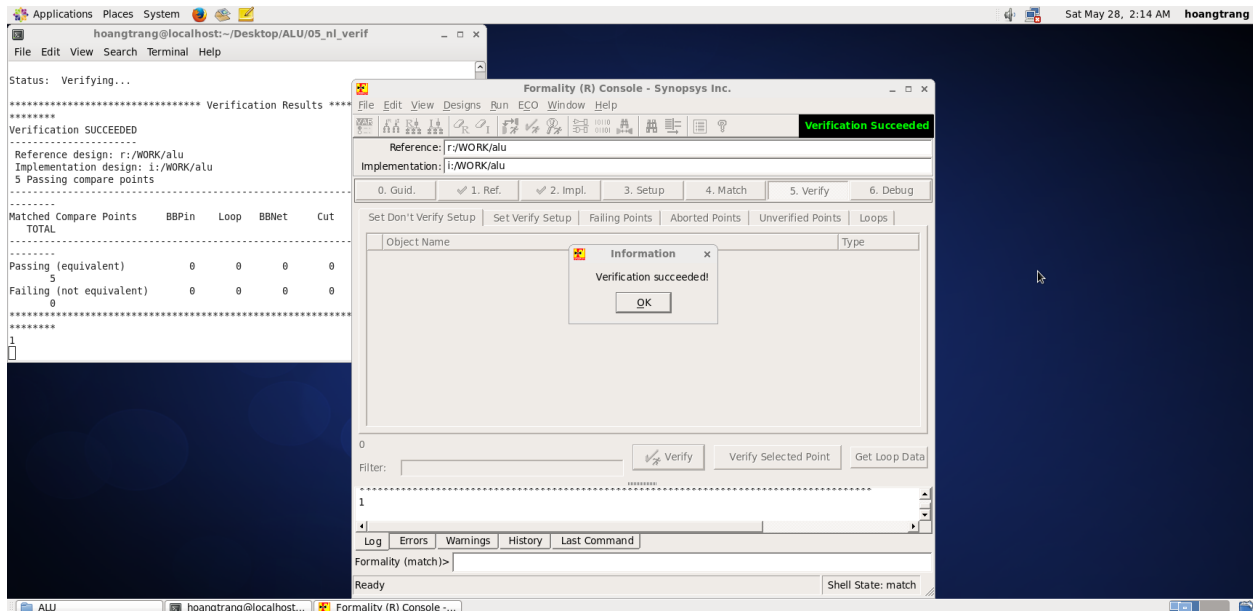
## 2.4. Kiểm định netlist

### A. Thông báo của Formality sau khi chạy matching





-Thông báo của Formality sau khi chạy Verify



### 3. Kết luận

#### 3.1. Nhận xét

- Mạch thiết kế ALU này là dạng cơ bản của một khối ALU, các chức năng ở dạng đơn giản, không có nhiều chế độ tính toán, tuy nhiên với một số cá nhân không nhớ syntax của verilog là 1 trở ngại.
- Việc xây dựng khối ALU này được dựa trên các phép toán cơ bản khác nhau.
- Sinh viên chưa làm quen với Linux và đặc biệt là flow của thiết kế sẽ gặp nhiều khó khăn.
- Sinh viên lần đầu gặp 1 số lỗi vẫn chưa biết khắc phục và cần có sự giúp đỡ của các thành viên và giảng viên hướng dẫn.

#### 3.2. Kết luận

- Sau khi thực hiện khối ALU 4 chức năng giúp sinh viên ôn lại kiến thức về kỹ thuật số,biết cách code verilog và biết được môi trường làm việc sau này sẽ như thế nào.
- Bài lab giúp sinh viên hiểu được những bước đầu trong quá trình thiết kế vi mạch.

- Giúp sinh viên nắm được cơ bản các tool trong các bước thiết kế vi mạch.
- Nắm được cách sử dụng Linux và một số cài đặt cần thiết.

#### **4.Phụ lục**

#### **5.Tài liệu tham khảo**

- Synthesis Tool Commands (DC\_Tool\_Commands) [1]
- VCS® MX/VCS MXi™ User Guide [2]
- Tài liệu hướng dẫn thí nghiệm Quy trình thiết kế vi mạch số.

Link google drive file làm việc lab1:

[https://drive.google.com/drive/folders/1dTSaEsADl6Z1lNAcp8SqHOqthmxWd1Q0?usp=s\\_haring](https://drive.google.com/drive/folders/1dTSaEsADl6Z1lNAcp8SqHOqthmxWd1Q0?usp=s_haring)

