

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**  
**TRƯỜNG ĐẠI HỌC BÁCH KHOA**  
**KHOA ĐIỆN – ĐIỆN TỬ**  
**BỘ MÔN KỸ THUẬT ĐIỆN TỬ**  
--oOo--



# **TÀI LIỆU THÍ NGHIỆM KỸ THUẬT SỐ**

---

## MỤC LỤC

<b>LỜI NÓI ĐẦU.....</b>	<b>1</b>
<b>LAB 1: CÁC CÔNG LOGIC VÀ IC CHỨC NĂNG CƠ BẢN.....</b>	<b>2</b>
A. PRELAB .....	2
I. HƯỚNG DẪN CƠ BẢN VỀ BREADBOARD .....	2
II. CÂU HỎI CHUẨN BỊ: .....	11
B. HƯỚNG DẪN THÍ NGHIỆM: .....	22
I. MỤC TIÊU .....	22
II. CHUẨN BỊ.....	22
III. HƯỚNG DẪN THÍ NGHIỆM .....	22
<b>LAB 2: THỰC HIỆN CÁC IC CHỨC NĂNG CƠ BẢN TRÊN FPGA.....</b>	<b>35</b>
A. PRELAB .....	35
I. MỤC TIÊU .....	35
II. CHUẨN BỊ.....	35
B. HƯỚNG DẪN THÍ NGHIỆM: .....	42
I. MỤC TIÊU .....	42
II. CHUẨN BỊ.....	42
III. HƯỚNG DẪN THÍ NGHIỆM .....	42
<b>LAB 3: THIẾT KẾ HỆ TỔ HỢP      THIẾT KẾ HỆ TUẦN TỤ.....</b>	<b>64</b>
A. PRELAB .....	64
B. HƯỚNG DẪN THÍ NGHIỆM: .....	70
I. MỤC TIÊU .....	70
II. CHUẨN BỊ.....	70
III. HƯỚNG DẪN THÍ NGHIỆM: .....	70
<b>LAB 4: THỰC HIỆN HỆ TỔ HỢP VÀ HỆ TUẦN TỤ CƠ BẢN TRÊN FPGA</b>	<b>79</b>
A. PRELAB .....	79
B. HƯỚNG DẪN THÍ NGHIỆM .....	87
I. MỤC TIÊU .....	87
II. CHUẨN BỊ: .....	87



---

III.	HƯỚNG DẪN THÍ NGHIỆM .....	87
<b>LAB 5:</b>	<b>BÀI TỔNG HỢP THỰC HIỆN MẠCH SỐ TRÊN BREADBOARD – GIAO TIẾP VỚI FPGA.....</b>	<b>105</b>
A.	PRELAB .....	105
B.	HƯỚNG DẪN THÍ NGHIỆM .....	119
I.	MỤC TIÊU .....	119
II.	CHUẨN BỊ: .....	119
III.	HƯỚNG DẪN THÍ NGHIỆM .....	119
<b>PHỤ LỤC 1:</b>	<b>HƯỚNG DẪN CÀI ĐẶT QUARTUS VÀ UBUNTU TRONG WINDOWS</b>	<b>134</b>
A.	HƯỚNG DẪN CÀI ĐẶT UBUNTU TRONG WINDOWS: .....	134
I.	Download và cài đặt công cụ: .....	134
II.	Cài đặt Ubuntu trong Windows: .....	134
III.	Một số lệnh cơ bản trong Linux.....	136
B.	HƯỚNG DẪN CÀI ĐẶT QUARTUS 13.0SP1.....	138
<b>PHỤ LỤC 2:</b>	<b>QUY TRÌNH THIẾT KẾ MẠCH SỐ SỬ DỤNG NGÔN NGỮ SYSTEMVERILOG.....</b>	<b>139</b>
A.	QUY TRÌNH THIẾT KẾ .....	139
B.	VÍ DỤ THIẾT KẾ HỆ TỔ HỢP.....	140
I.	Yêu cầu: .....	140
II.	Thiết kế: .....	140
C.	VÍ DỤ THIẾT KẾ HỆ TUẦN TỤ.....	154
I.	Yêu cầu: .....	154
II.	Phân tích: .....	154
III.	Thiết kế: .....	155

---

## LỜI NÓI ĐẦU

Tài liệu này được sử dụng trong môn Thí nghiệm Kỹ thuật số, giành cho sinh viên khoa Điện – Điện tử, trường Đại học Bách Khoa TPHCM. Các bài thí nghiệm được xây dựng theo 2 hướng:

- Sử dụng các vi mạch (IC) số để thiết kế, thực hiện các mạch tổ hợp và mạch tuần tự trên breadboard.
- Sử dụng ngôn ngữ mô tả phần cứng (HDL), thiết kế mạch số trên FPGA.

Tài liệu gồm 5 bài thí nghiệm

Lab 1 – Các cổng logic và các IC chức năng cơ bản.

Lab 2 – Thực hiện các IC chức năng cơ bản trên FPGA.

Lab 3 – Thiết kế mạch tổ hợp - Mạch cộng – Thiết kế mạch tuần tự.

Lab 4 – Thực hiện mạch tổ hợp và mạch tuần tự trên FPGA.

Lab 5 – Bài thiết kế tổng hợp.

Để làm tốt bài thí nghiệm, sinh viên **PHẢI** thực hiện bài chuẩn bị thí nghiệm (Prelab) trước mỗi buổi thí nghiệm.



Electronics Department

Ho Chi Minh City University of Technology, Vietnam

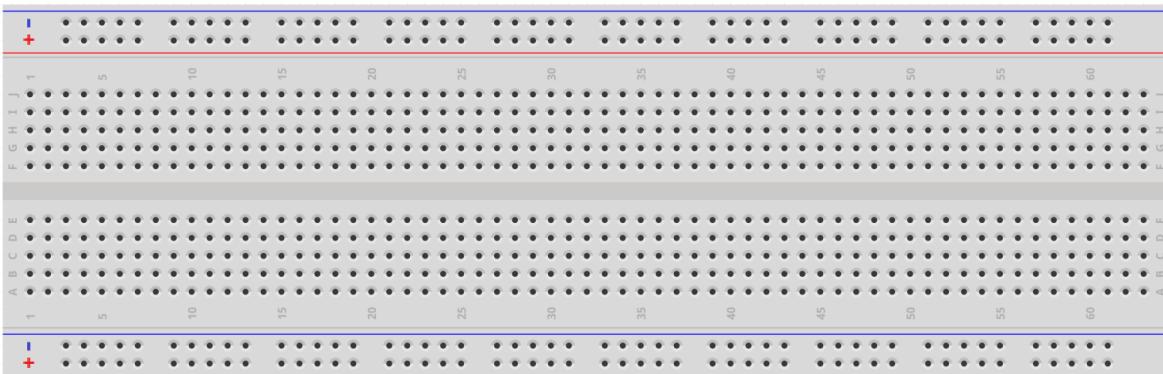
# LAB 1: CÁC CÔNG LOGIC VÀ IC CHỨC NĂNG CƠ BẢN

Họ và tên:	Lớp TN:
MSSV:	Ngày:

## A. PRELAB

### I. HƯỚNG DẪN CƠ BẢN VỀ BREADBOARD

Breadboard là sản phẩm không thể thiếu trong quá trình học tập điện tử, được sử dụng để lắp và kiểm tra mạch điện tử.



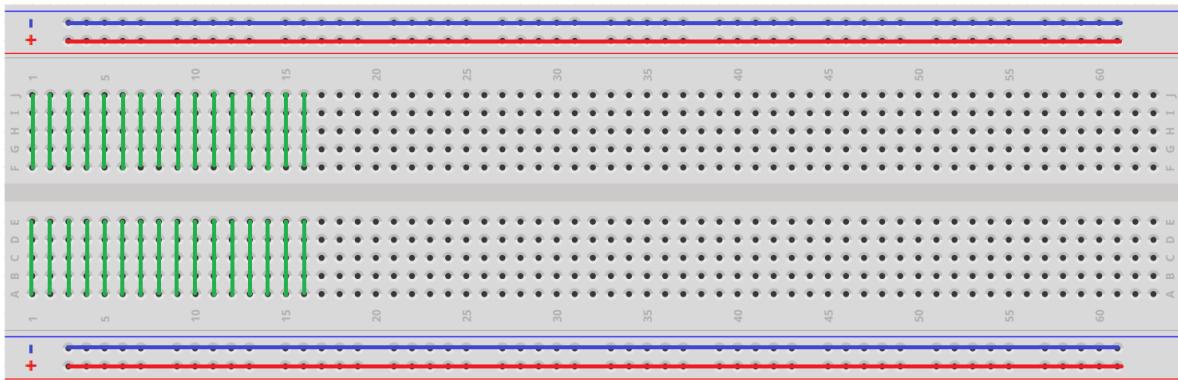
**Hình 1.1: Hình minh họa breadboard**

Như các bạn thấy, breadboard có rất nhiều lỗ, giữa các lỗ cách nhau một khoảng tiêu chuẩn để các thành phần như IC, tụ điện, điện trở, LED... khi lắp vào sẽ khớp hoàn toàn. Dưới các lỗ là các dải kim loại có thể tiếp xúc với các chân của linh kiện hoặc jumper (dây nối), giúp các chân linh kiện, jumper có thể nối với nhau hình thành một mạch điện kín mà không cần đến thao tác hàn dây, đồng thời các phần tử trong mạch điện có thể được gỡ ra, chỉnh sửa theo mong muốn một cách dễ dàng. Các dải kim loại này kết nối với nhiều chân, cụ thể như sau:

- Các dải nguồn: hai rìa của breadboard có các hàng kí hiệu +/- được kết nối theo hàng ngang. Hai dải này được sử dụng để phân phối nguồn cho toàn mạch.



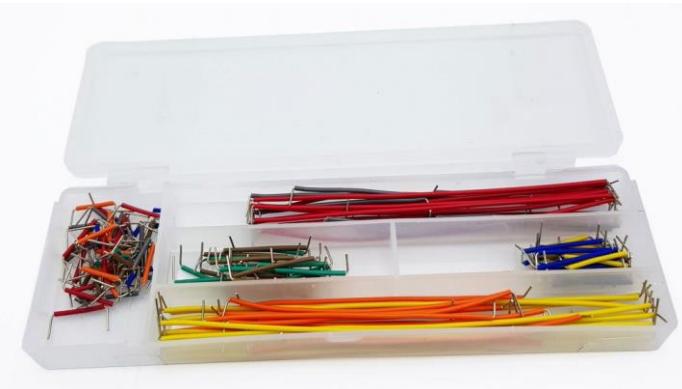
- Các lưới thành phần: phần giữa của breadboard được kết nối theo hàng dọc theo hình minh họa sau.



**Hình 1.2:** Hình minh họa vị trí các dải kim loại trên breadboard

- Một số lưu ý khi cắm breadboard.

Để kết nối các linh kiện trên breadboard, ta sử dụng các dây dẫn. Trong môn học Thí nghiệm Kỹ Thuật Số cung cấp cho sinh viên bộ dây cắm breadboard chuyên dụng. Trong mỗi bộ dây gồm có 14 loại dây có kích thước vừa đủ để kết nối các lỗ trên breadboard với nhau (độ dài 2 lỗ, 3 lỗ, 4 lỗ...). Mỗi loại dây trong bộ gồm 10 sợi.

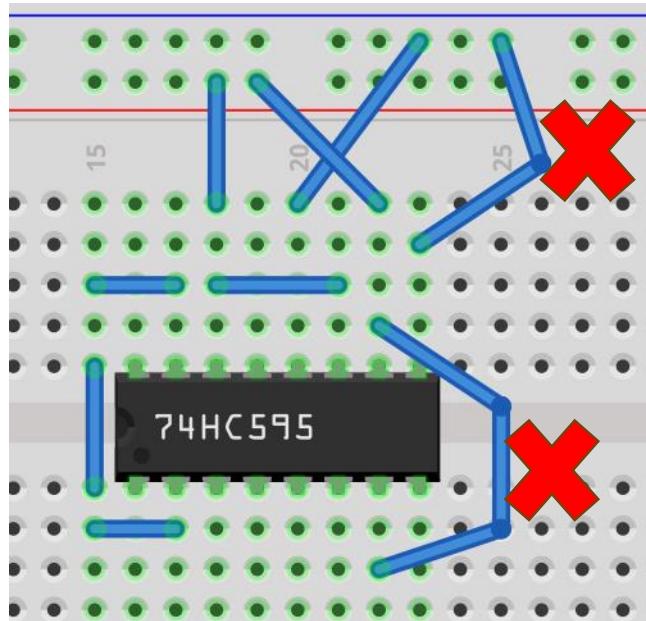


**Hình 1.3:** Bộ dây dẫn chuyên dụng cắm breadboard

Sinh viên khi dùng phải chú ý thao tác, đảm bảo cho dây không bị méo dạng sau khi sử dụng. Trước khi cắm dây, cần đảm bảo đầu dây phải thẳng. Sử dụng kìm kẹp để cắm dây sẽ dễ thao tác hơn.

Lưu ý: Các dây jumper được cung cấp cho thí nghiệm có kích thước và hình dạng cố định, khi thí nghiệm không được tự ý thay đổi kích thước và hình dạng dây. Trong trường hợp dây

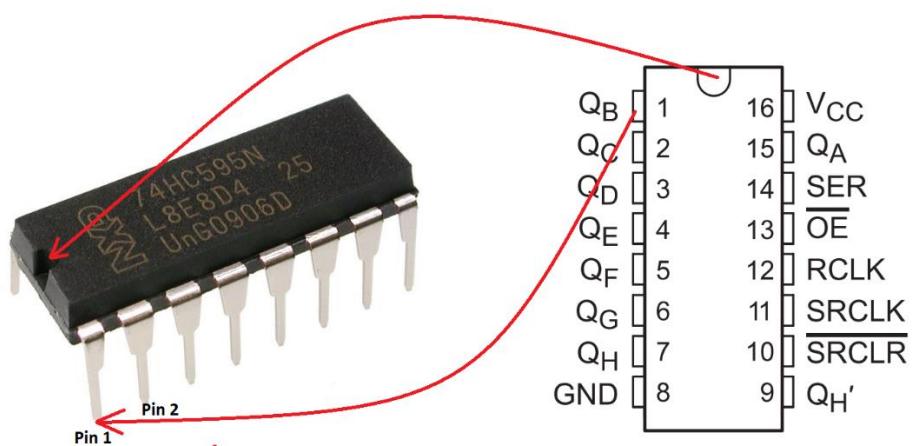
jumper bị méo dạng trong quá trình tháo, lắp, cần trả về hình dạng gốc của dây. Có thể để hai dây jumper chéo nhau, nhưng hạn chế.



**Hình 1.4:** Hình minh họa việc cắm dây (dấu X: không cho phép bẻ dây).

Khi cắm các điện trở, tụ điện, LED, BJT... có thể cắt ngắn chân các linh kiện này vừa đủ để cắm trên breadboard. Trong trường hợp không cắt ngắn, cần chú ý không để các chân này chạm nhau gây chập mạch.

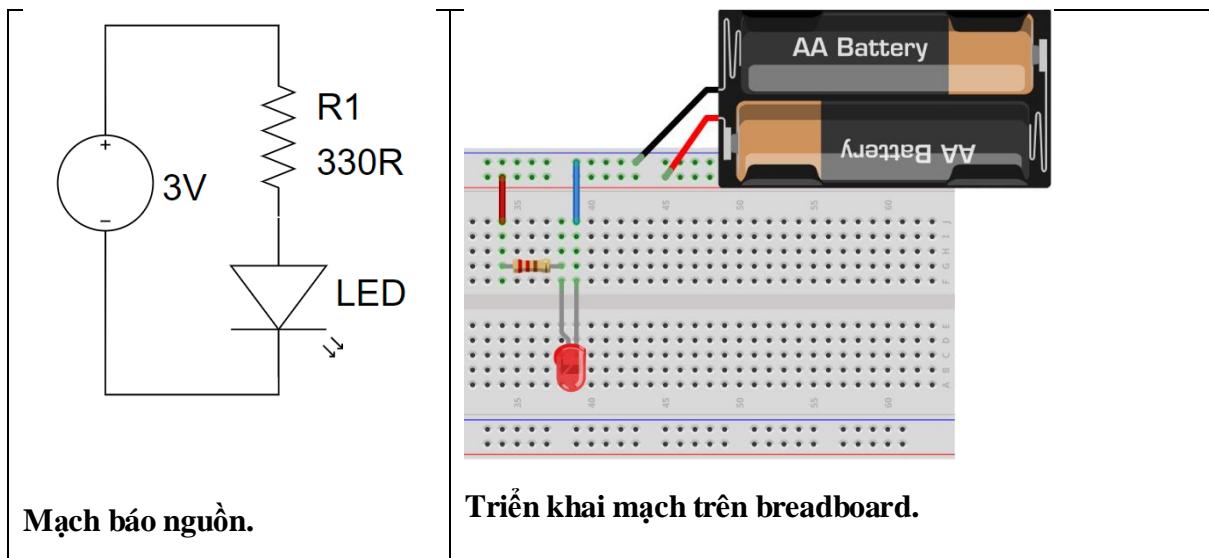
Khi làm việc với IC, cần đọc datasheet tương ứng của IC, nắm được vị trí các chân của IC trên datasheet tương ứng với trên thực tế. Quan sát trên IC, sẽ có kí hiệu chỉ ra chân số 1, từ vị trí chân số 1 nhìn IC từ trên xuống, số thứ tự chân của IC được đánh số theo chiều ngược chiều kim đồng hồ. Kí hiệu chỉ ra chân số 1 của IC có thể là một vết khoét hình vòng cung, hoặc 1 dấu chấm đối với footprint dạng DIP.



Hình 1.5: Xác định chân của IC.

- Ví dụ mạch cắm bread board.

Cho mạch điện LED báo nguồn như sau.



Hình 1.6: Mạch LED báo nguồn

- Hướng dẫn cắm input và output.

Theo lý thuyết, trong một hệ nhị phân, 1 và 0 được gọi là bit (bit là dạng rút gọn của binary digit). Nhưng trong mạch số thực tế, hai bit nói trên được biểu diễn bằng hai mức điện áp khác nhau. Có hai cách biểu diễn là mức logic dương và mức logic âm. Mức logic dương: 1 được biểu diễn bởi mức điện áp cao mà chúng ta thường gọi là MỨC CAO (HIGH), và 0 được biểu diễn bởi mức điện áp thấp hơn mà chúng ta sẽ gọi là MỨC THẤP (LOW). Mức

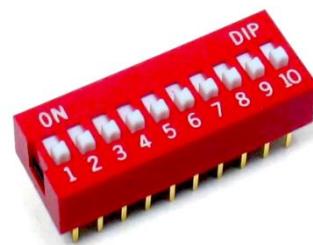
logic âm: ngược lại so với mức logic dương. Trong Thí nghiệm Kỹ Thuật Số ta sẽ dùng mức logic dương để biểu diễn hệ nhị phân trong suốt quá trình học tập.

Theo lý thuyết, sẽ có một mức điện áp biểu diễn MỨC CAO (1) và một điện áp biểu diễn MỨC THẤP (0). Tuy nhiên trong thực tế, MỨC CAO sẽ được hiểu là điện áp nằm giữ giá trị cực đại và cực tiểu. Điều này tương tự với MỨC THẤP. Lấy chuẩn TTL làm ví dụ, MỨC CAO (1) sẽ rơi trong khoảng điện áp 2 đến 5V đối với tín hiệu là ngõ vào (Input) và 2.7 đến 5V đối với tín hiệu là ngõ ra (Output), MỨC THẤP (0) sẽ rơi trong khoảng điện áp 0 đến 0.8V đối với tín hiệu là ngõ vào (Input) và 0 đến 0.5V đối với tín hiệu là ngõ ra (Output). Khi mức điện áp ngõ vào nằm giữ MỨC THẤP và MỨC CAO (từ 0.8V đến 2.7V) đây là dải điện áp không xác định và dẫn đến trạng thái không hợp lệ gọi là thả nổi (floating). Trong môn học Thí nghiệm Kỹ Thuật Số, ta sử dụng mức điện áp 5V cho mức cao (1) và 0V cho mức thấp (0).



Hình 1.7: Sơ đồ khối mạch số

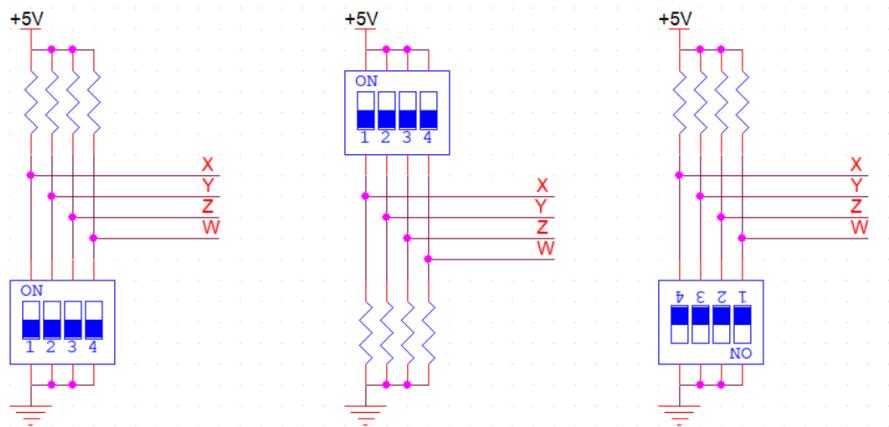
Trong một bài Thí nghiệm Kỹ Thuật Số sẽ yêu cầu sinh viên thiết kế một mạch có các ngõ vào, ngõ ra, mối liên hệ giữa ngõ vào và ngõ ra để thực hiện một chức năng nào đó. Vì thế khi cắm mạch trên breadboard, sinh viên cũng phải cắm đủ 3 phần: ngõ vào (input, một dải các tín hiệu đầu vào), mạch được thiết kế, ngõ ra (output, dùng để hiển thị kết quả một cách trực quan).



Phần mạch input, cần thiết kế sao cho dễ dàng thay đổi giữa mức 0 và 1. Ta sẽ sử dụng công tắc (switch) để làm điều này, switch được sử dụng là loại dip switch: gồm một dãy các switch độc lập, như hình minh họa đây là loại dip switch 10 positions, gồm 10 switch độc lập với nhau được đánh số từ 1 đến 10, công tắc tương ứng nằm ở phía trên, gạt công tắc ở mức ON thì hai chân tương ứng của công tắc sẽ nối với nhau.

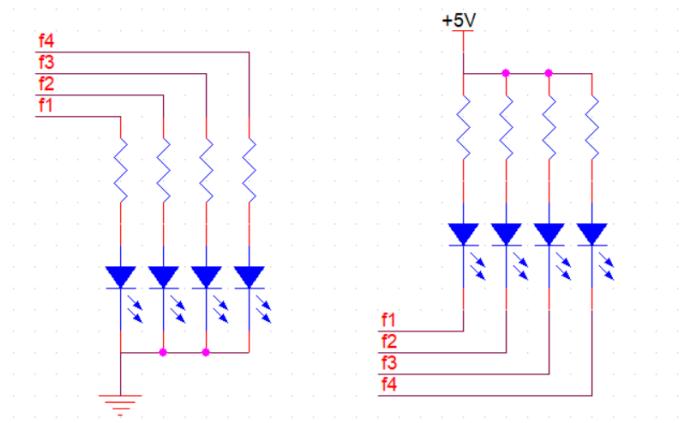
**Hình 1.8: DIP Switch**

Có hai cách thiết kế mạch input: công tắt ON thì ngõ vào mức 0 (hình trái) hoặc công tắt ON thì ngõ vào mức 1 (hình giữa) như sơ đồ nguyên lý sau. Trong đó các điện trở có giá trị là **10Kohm**. Theo kinh nghiệm cá nhân, nên thiết kế mạch input như hình bên phải sẽ ổn định, dễ thao tác hơn.



**Hình 1.9: Kết nối DIP Switch**

Về phần mạch output, cần thiết kế để quan sát các ngõ ra, thông thường sẽ hiển thị output trên các LED đơn, bar LED, LED 7 đoạn (đối với mạch đếm). Một số mạch hiển thị output trên LED đơn. Hình bên trái: ngõ ra mức 1 LED sẽ sáng, ngõ ra mức 0 LED sẽ tắt. Hình bên phải: ngược lại. Các điện trở hạn dòng sử dụng có giá trị **1Kohm**.



**Hình 1.10:** Kết nối LED

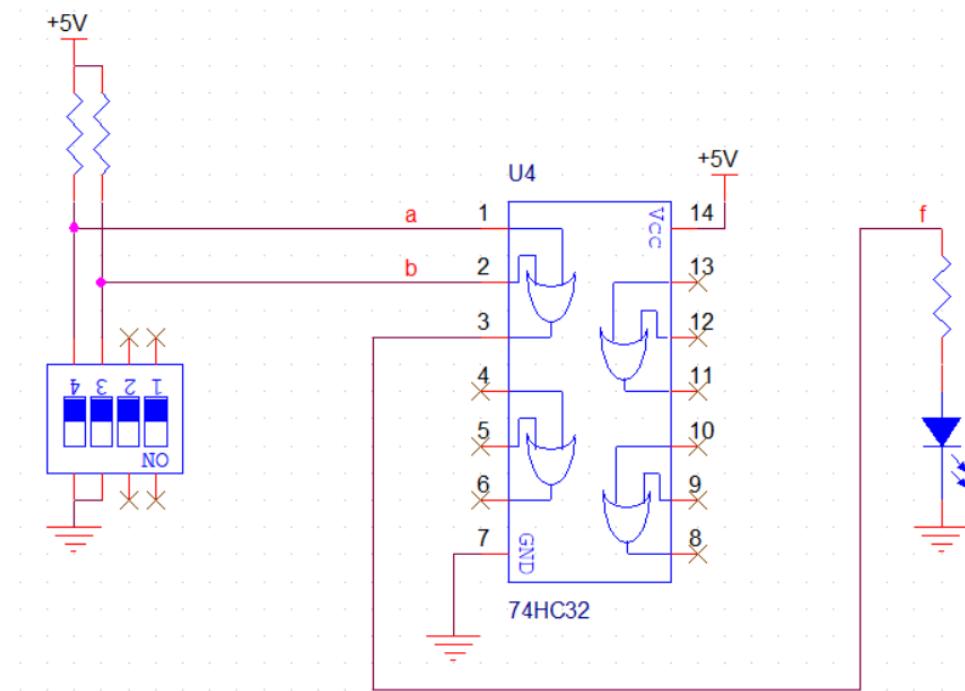
Ví dụ: thiết kế hàm  $f(a, b) = a + b$ .

Đầu tiên cần phải phân tích đề bài:

- Ngõ vào: có 2 ngõ vào được đặt tên là a và b.
- Ngõ ra: có 1 ngõ ra là f.
- Mối liên hệ giữa ngõ vào và ngõ ra:  $f = a + b$ . Vì thế ta sử dụng IC thực hiện phép toán OR là 74HC32. Xem datasheet của IC 74HC32 để biết chi tiết các chân của IC. Nhớ cấp nguồn đúng cho IC.

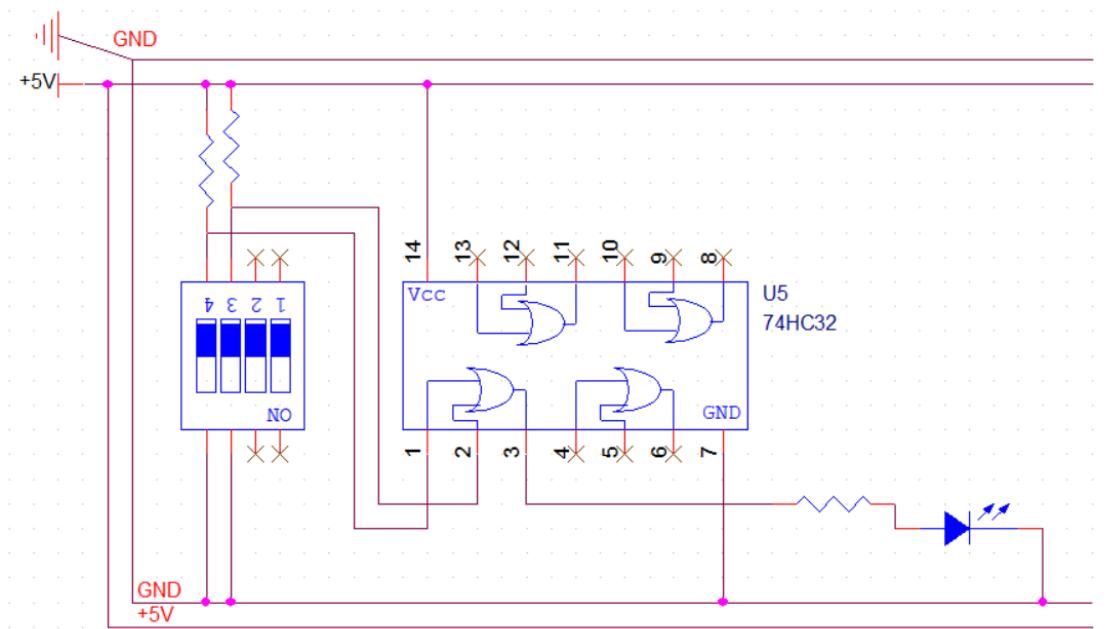
Sơ đồ nguyên lý cần thiết kế: gồm 3 phần: input, phần mạch, và output.





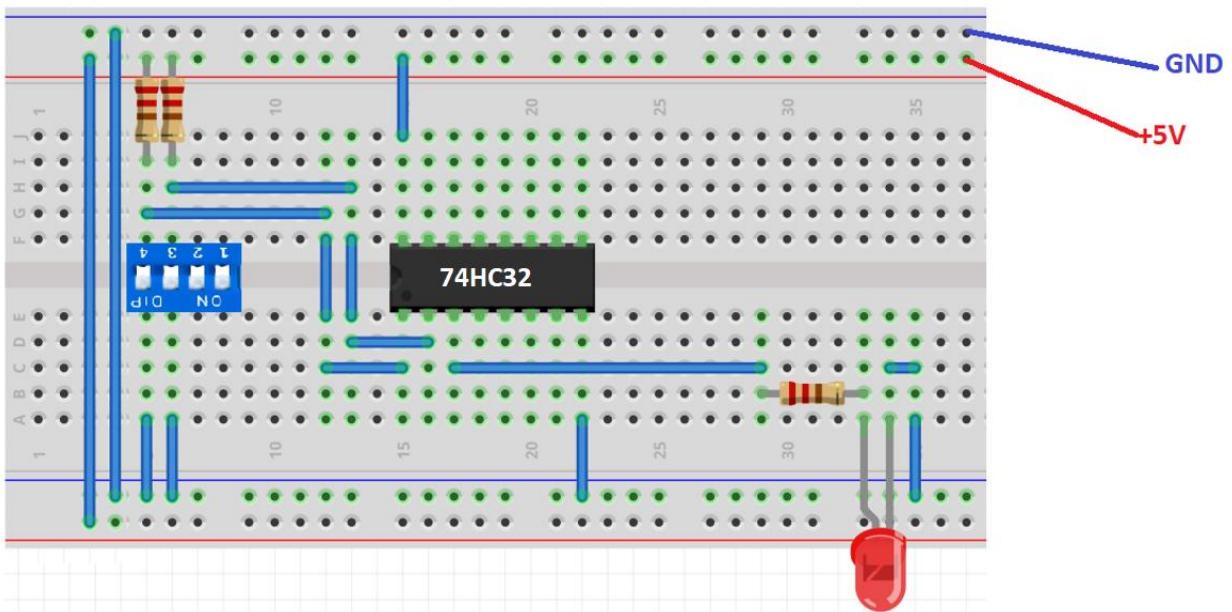
**Hình 1.11:** Sơ đồ nguyên lý mạch thực hiện hàm  $F(a,b) = a + b$

Dưới đây là sơ đồ nguyên lý của mạch cần thiết kế nhưng được sắp xếp lại để dễ dàng triển khai trên breadboard hơn.



**Hình 1.12:** Sơ đồ nguyên lý mạch trên breadboard thực hiện hàm  $F(a,b) = a + b$

Triển khai sơ đồ nguyên lý lên trên breadboard.



**Hình 1.13:** Triển khai sơ đồ nguyên lý  $F(a,b) = a + b$  lên breadboard



## II. CÂU HỎI CHUẨN BI:

1. Khi cắm jumper cần lưu ý điều gì.

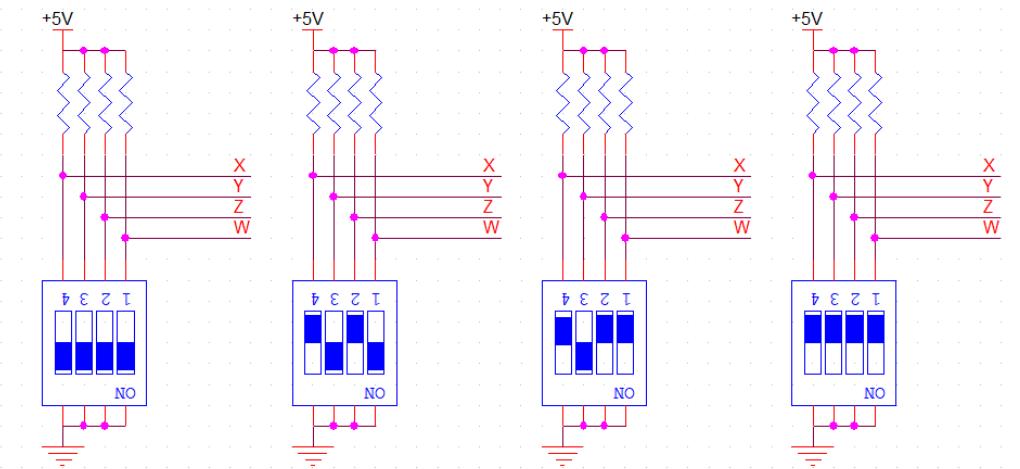
2. Kích thước giữa 2 lỗ gần nhau nhất trên breadboard là bao nhiêu?

3. Cách xác định chân số 1 của IC.

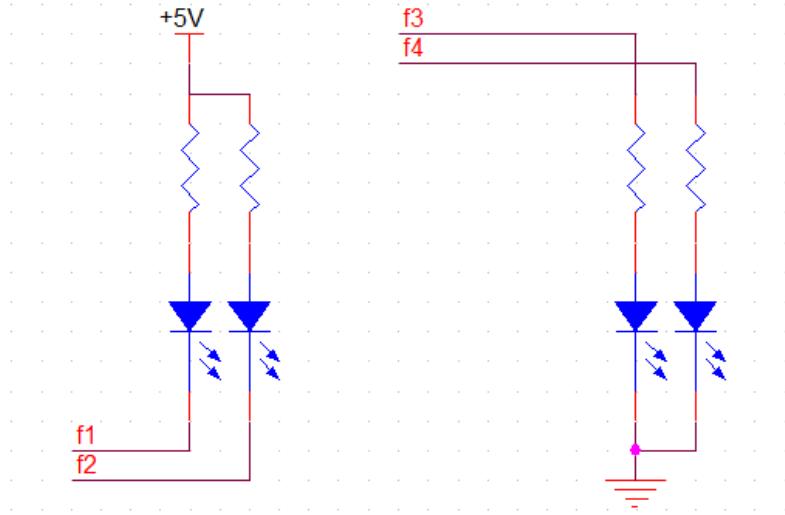
4. Cách đánh số thứ tự chân của IC.

5. Có mấy cách biểu diễn mức logic trong thực tế, kể tên và trong TN KTS sử dụng cách nào? Mức điện áp sử dụng để biểu diễn?

6. Một bạn sinh viên lắp mạch cho 4 tín hiệu input X, Y, Z, W. Khi DIP switch ở trạng thái như các hình thì các tín hiệu ngõ vào X, Y, Z, W là mức cao hay thấp (1 hay 0), ghi trực tiếp vào hình vẽ.



7. Một bạn sinh viên lắp mạch sau cho tín hiệu output. Khi các tín hiệu f1, f2, f3, f4 lần lượt là 0,1,1,0 thì trạng thái các đèn LED (sáng/tắt) như thế nào, chỉ rõ trên hình.



8. Hoàn thành bảng sau (xem datasheet của chúng)

Tên IC	Chức năng	Tóm tắt chức năng các chân của IC
74LS00	4 cổng NAND	14-VCC; 7-GND; 3 = 1 nand 2; 6 = 4 nand 5; .....
74LS02		
74LS04		
74LS08		
74LS32		
74LS86		
74LS125		
74LS126		
74LS138		
74LS151		

9. Tham khảo datasheet của IC 74LS125 và 74LS126, trả lời các câu hỏi sau.

Hai IC đã cho là hai loại công logic cơ bản, sinh viên vẽ công logic cơ bản của hai IC này ra và giải thích hoạt động của nó.

Sự khác nhau giữa 74LS125 và 74LS126 là gì?

10. Cho hàm boolean  $f(x, y, z) = \bar{x} \cdot y + \bar{y}z$ , trả lời các câu hỏi sau:

Để thiết kế hàm trên bằng các IC các công logic cơ bản, ta cần những IC nào, số lượng bao nhiêu.

Phân tích mạch cần thiết kế:

Ngõ vào:

Ngõ ra:

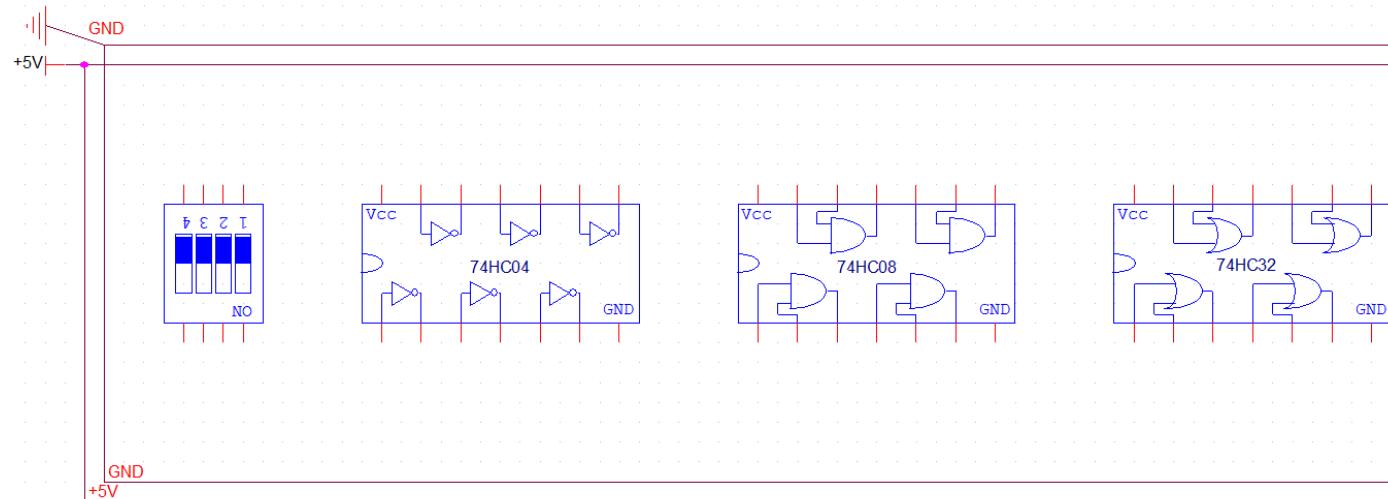
Mối liên hệ ngõ vào và ngõ ra:



Electronics Department

Ho Chi Minh City University of Technology, Vietnam

Vẽ sơ đồ nguyên lý của mạch cần thiết kế. Ghi rõ chân của IC.

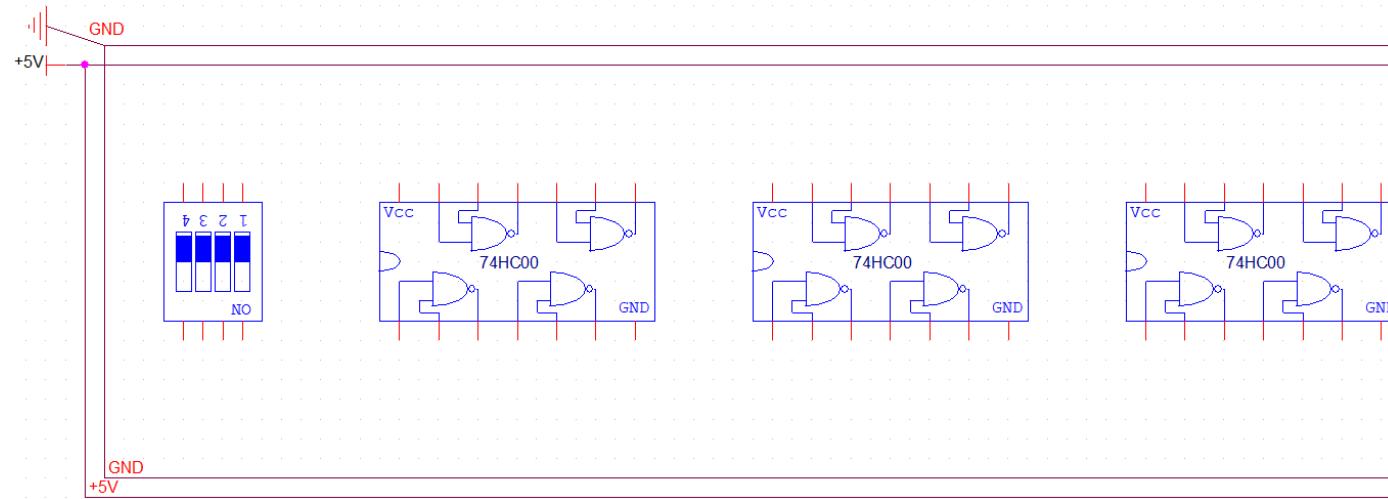


11. Biến đổi hàm boolean  $f(x, y, z) = \bar{x} \cdot y + \bar{y} \cdot z$  để thực hiện hàm f bằng các công NAND2 (tức công NAND 2 ngõ vào). Trả lời các câu hỏi sau:  
Biến đổi hàm:

Cần sử dụng bao nhiêu công NAND2.

Nếu thực hiện hàm f trên khi chỉ được dùng IC 74'00 thì cần bao nhiêu IC 74'00?

Vẽ sơ đồ nguyên lý của mạch cần thiết kế. Ghi rõ chân của IC.



Electronics Department

Ho Chi Minh City University of Technology, Vietnam

12. Cho hàm boolean  $f(x, y, z) = m_1 + m_3 + m_6$ , cho biết z là LSB, trả lời các câu hỏi sau:

Viết biểu thức ngõ ra f theo các ngõ vào x, y, z.

Để thiết kế hàm trên bằng các IC các công logic cơ bản, ta cần những IC nào, số lượng bao nhiêu. (Không dùng AND 3 ngõ vào)

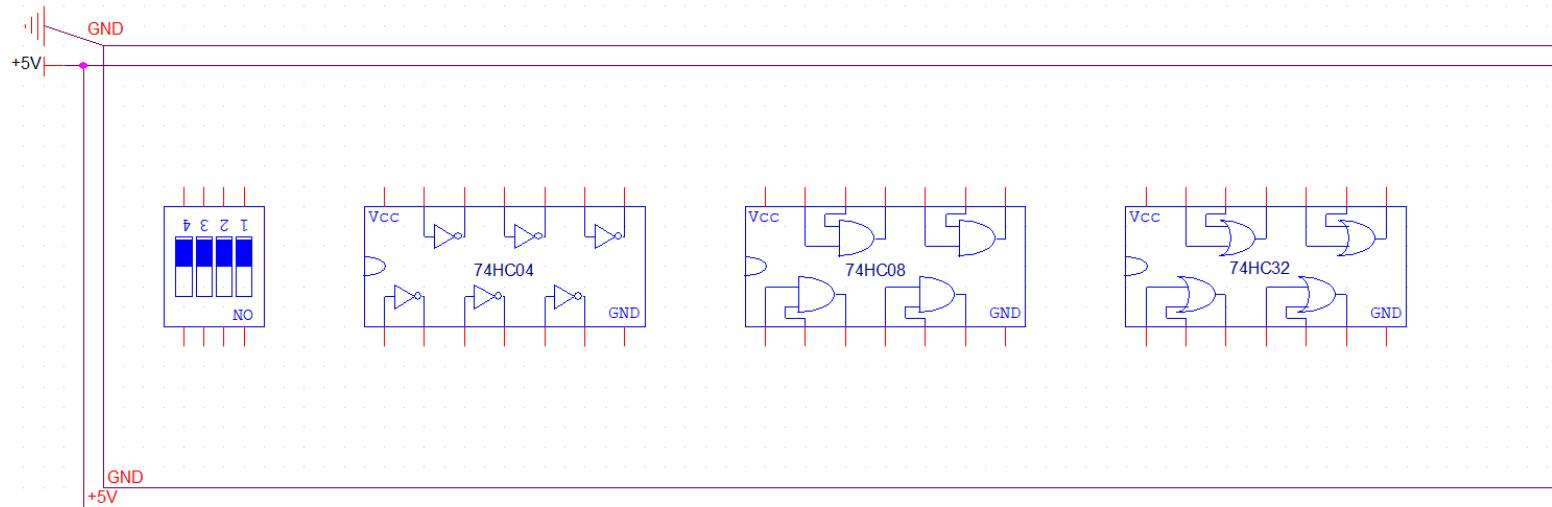
Phân tích mạch cần thiết kế:

Ngõ vào:

Ngõ ra:

Mỗi liên hệ ngõ vào và ngõ ra:

Vẽ sơ đồ nguyên lý của mạch cần thiết kế. Ghi rõ chân của IC.

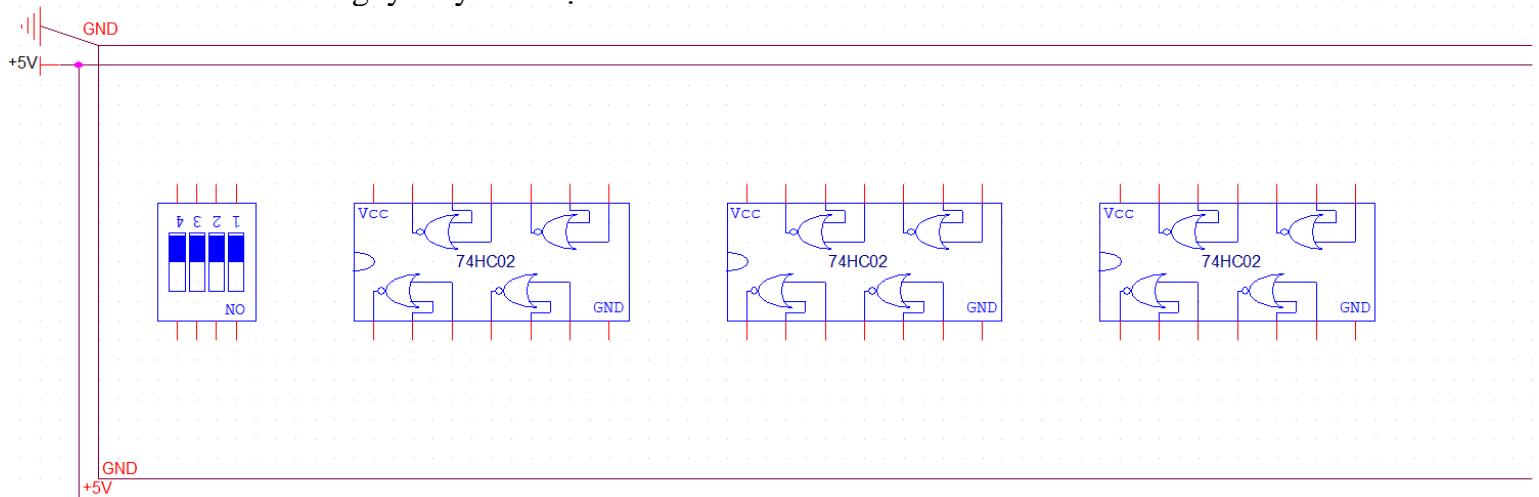


13. Cho hàm boolean  $f(x, y, z) = m_1 + m_3 + m_6$ , hãy biểu diễn hàm boolean dưới dạng NOR2.

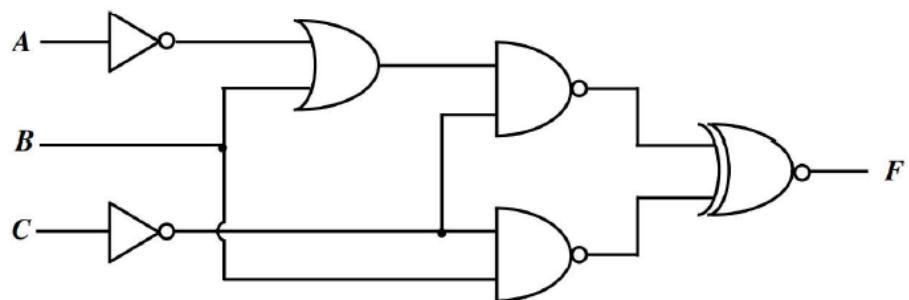
Biểu diễn dưới dạng NOR2.

Nếu thực hiện hàm  $f$  trên khi chỉ được dùng các cổng NOR thì cần IC gì và số lượng là bao nhiêu?

Vẽ sơ đồ nguyên lý của mạch cần thiết kế. Ghi rõ chân của IC.



14. Cho sơ đồ logic sau:



Viết biểu diễn hàm  $f(x, y, z)$

Giả sử không có công XNOR, chỉ có XOR, để thiết kế hàm trên bằng các IC các công logic cơ bản, ta cần những IC nào, số lượng bao nhiêu.

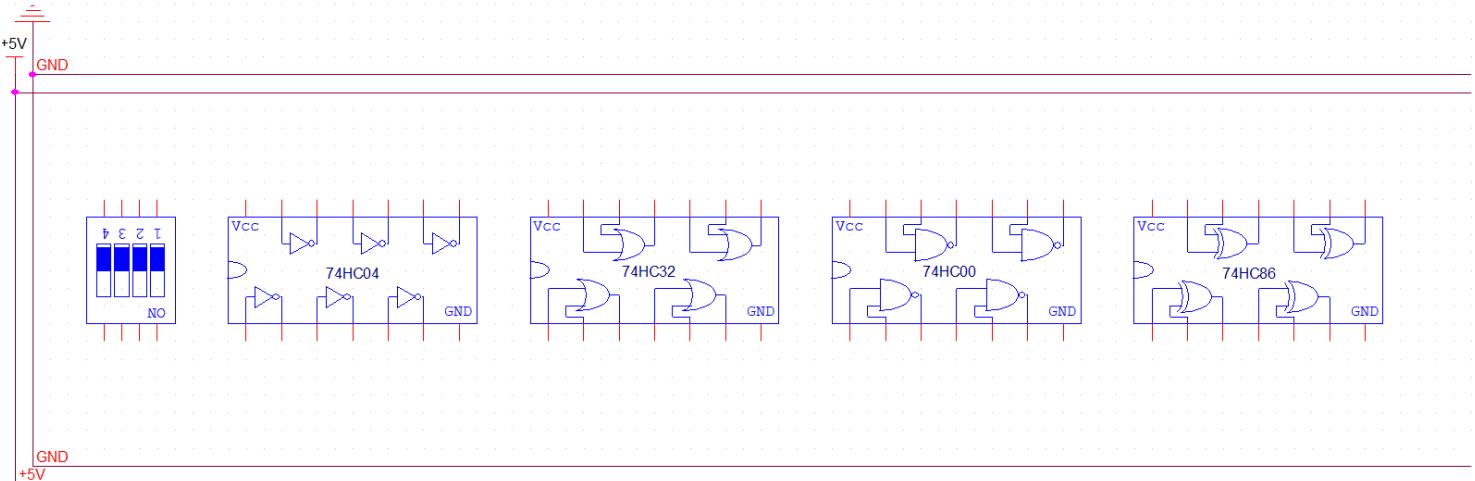
Phân tích mạch cần thiết kế:

Ngõ vào:

Ngõ ra:

Mỗi liên hệ ngõ vào và ngõ ra:

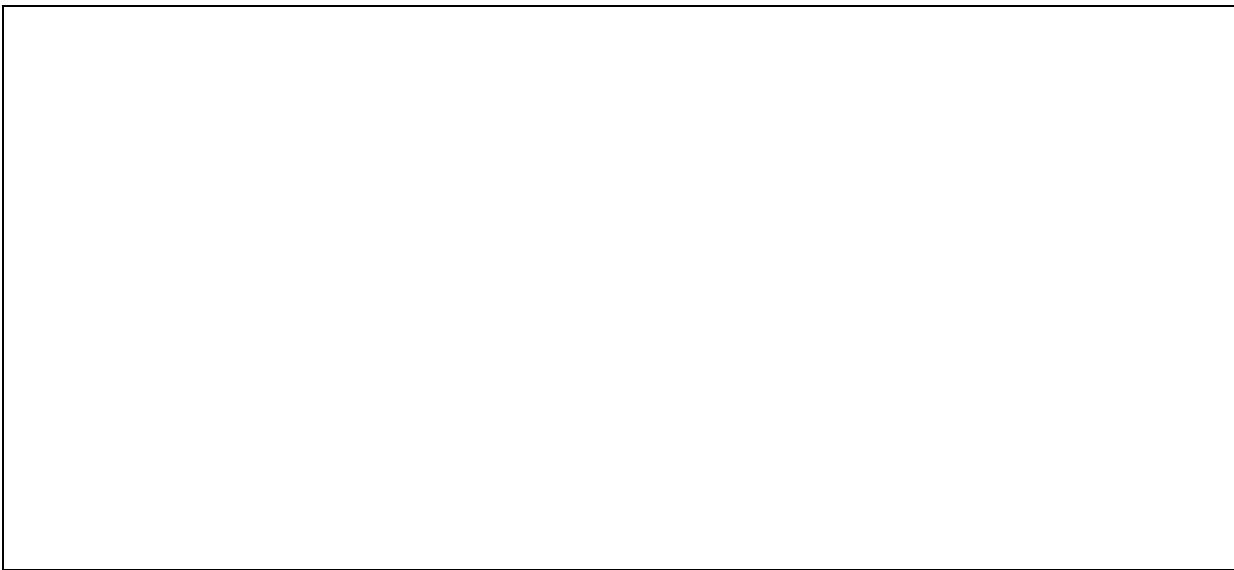
Vẽ sơ đồ nguyên lý của mạch cần thiết kế. Ghi rõ chân của IC.



15. Cho hàm boolean  $f(x, y, z) = \Sigma(2,3,5,7)$ , trả lời các câu hỏi sau:

Sử dụng IC chức năng 74LS151 và các công logic cần thiết, thiết kế mạch để thực hiện hàm boolean trên.

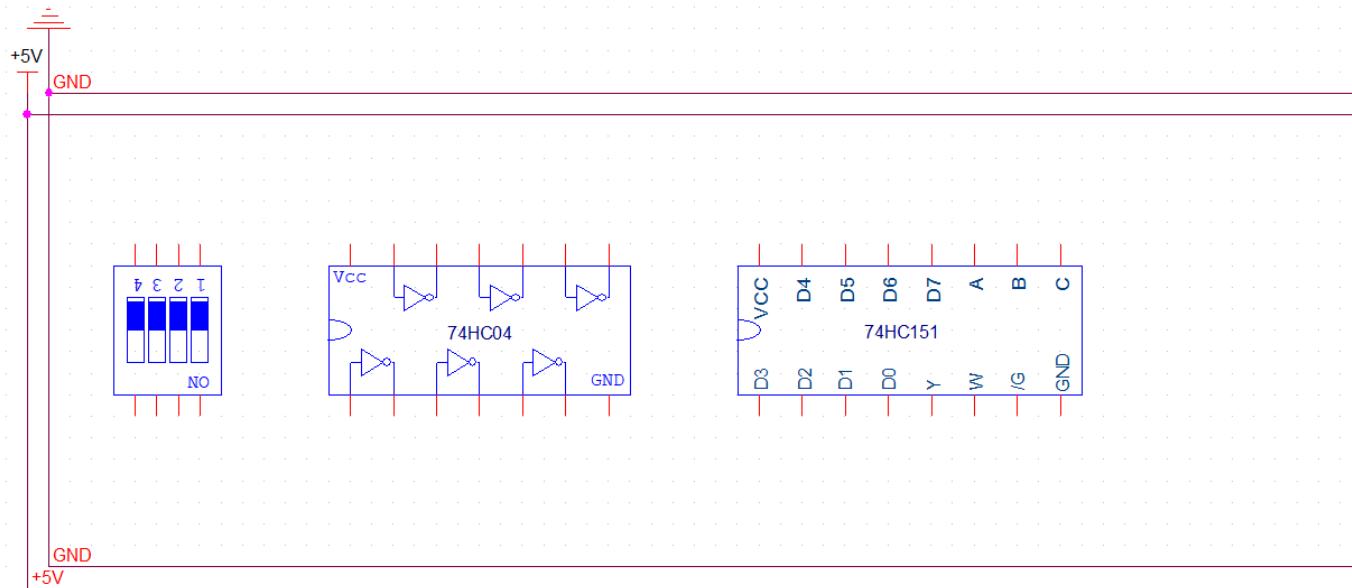




Để triển khai mạch trên thực tế, ta cần sử dụng những IC nào, kể tên.



Vẽ sơ đồ nguyên lý của mạch cân thiết kế. Ghi rõ chân của IC, tự vẽ thêm chân IC nếu thiếu.



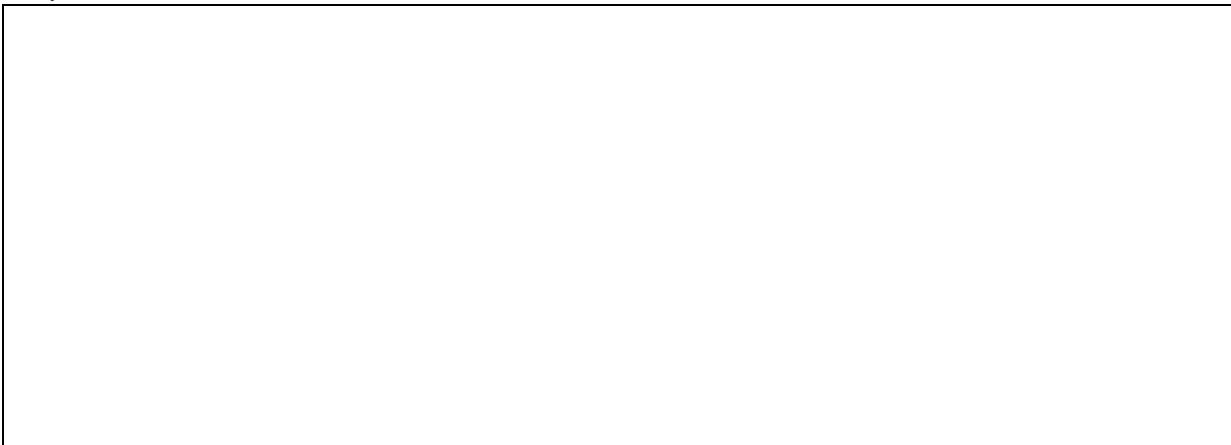
16. Cho hàm boolean  $f(x, y, z, w) = \sum(2, 6, 8, 9, 11, 13)$ , trả lời các câu hỏi sau:



Electronics Department

Ho Chi Minh City University of Technology, Vietnam

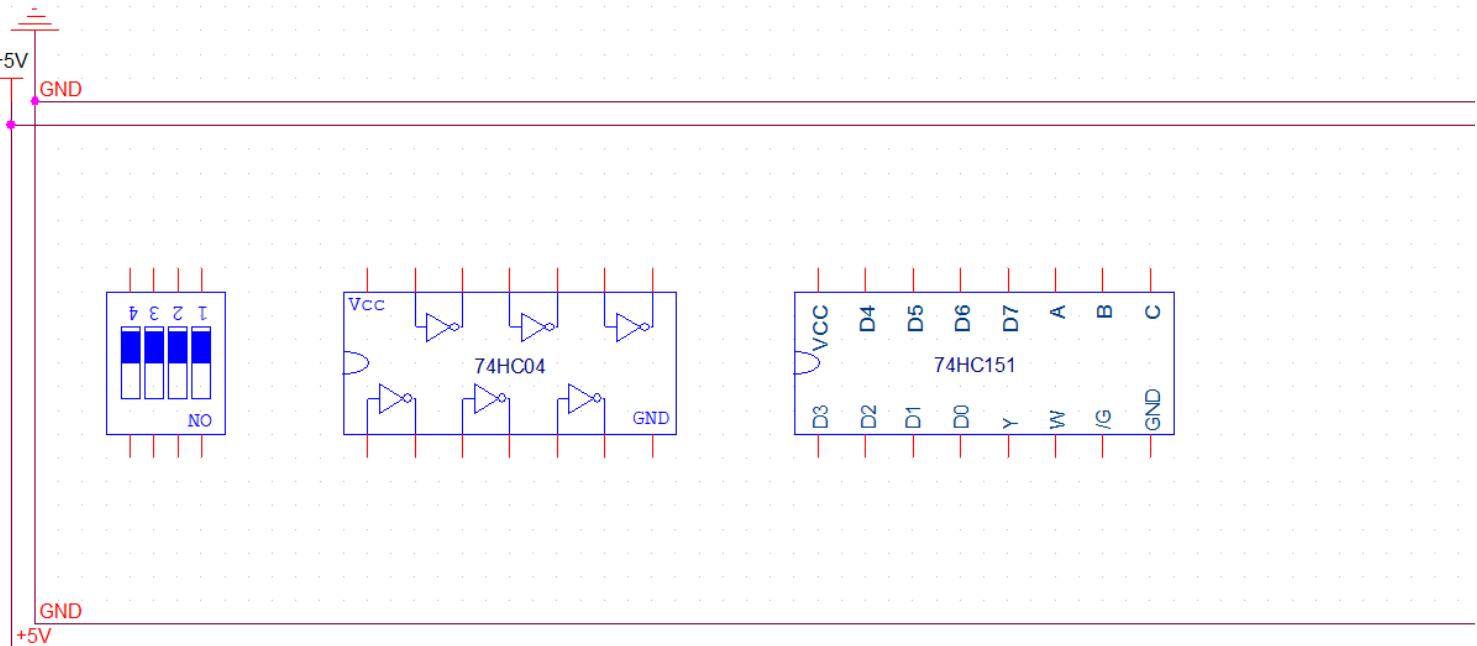
Sử dụng IC chức năng 74LS151 và các công logic cần thiết, thiết kế mạch để thực hiện hàm boolean trên.



Để triển khai mạch trên trong thực tế, ta cần sử dụng những IC nào, kể tên.



Vẽ sơ đồ nguyên lý của mạch cần thiết kế. Ghi rõ chân của IC.

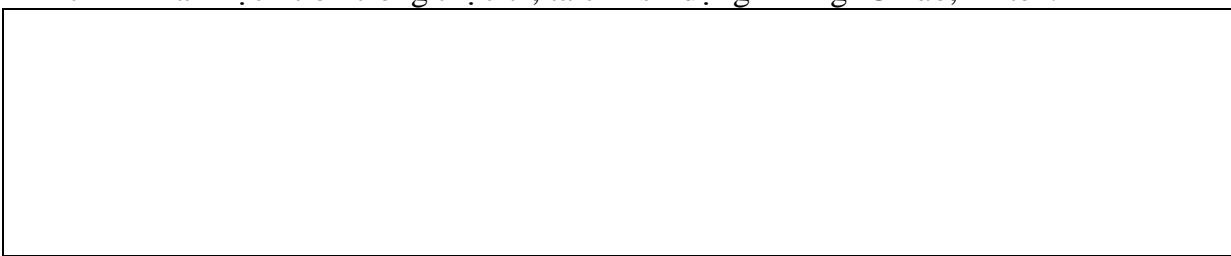


17. Cho hàm boolean  $f(x, y, z) = \Sigma(0, 3, 4, 7)$ , trả lời các câu hỏi sau:

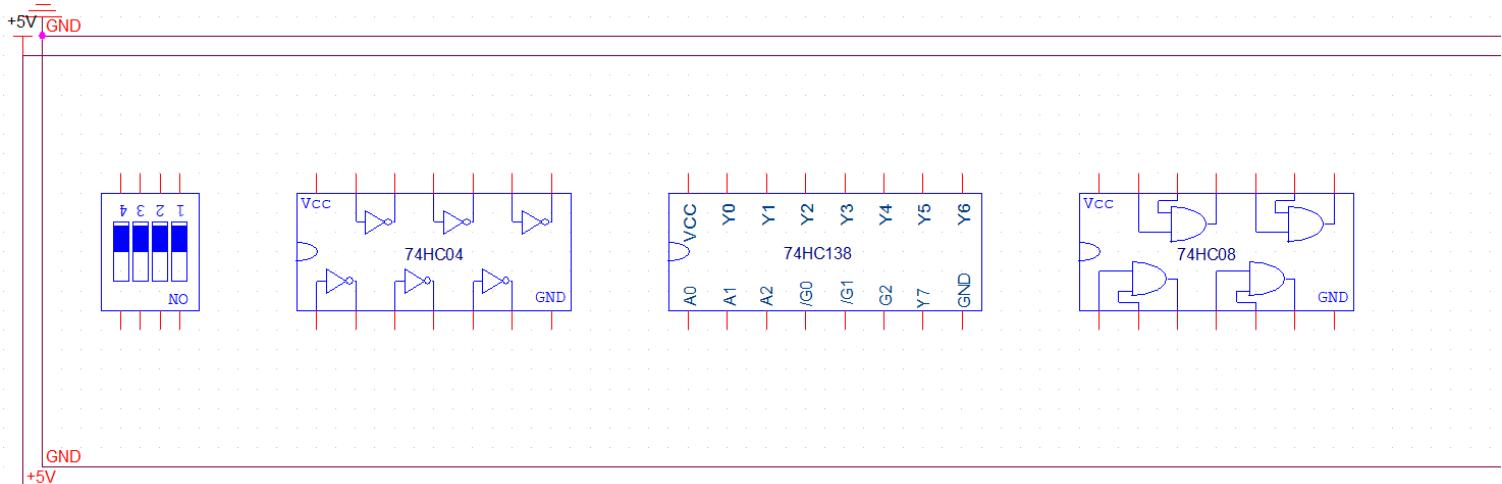
Sử dụng IC chức năng 74LS138 và các công logic cần thiết, thiết kế mạch để thực hiện hàm boolean trên.



Để triển khai mạch trên thực tế, ta cần sử dụng những IC nào, kê tên.



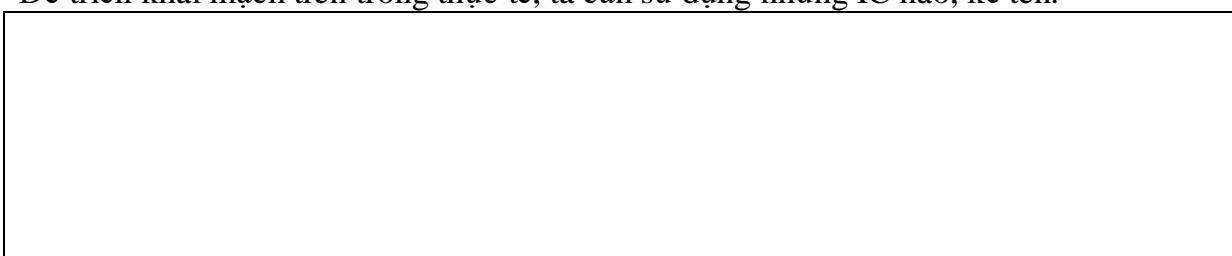
Vẽ sơ đồ nguyên lý của mạch cần thiết kế. Ghi rõ chân của IC.



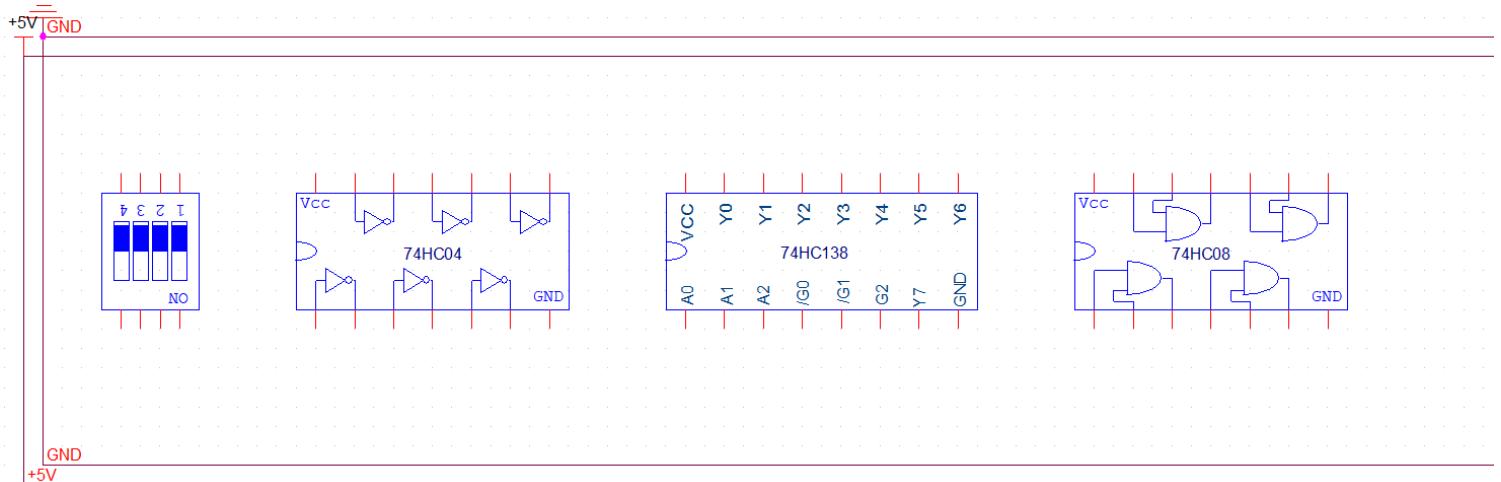
18. Cho hàm boolean  $f(x, y, z, w) = \sum(1, 5, 6, 7, 10, 12, 15)$ , trả lời các câu hỏi sau:  
 Sử dụng IC chức năng 74LS138 và các công logic cần thiết, thiết kế mạch để thực hiện hàm boolean trên.



Để triển khai mạch trên thực tế, ta cần sử dụng những IC nào, kể tên.



Vẽ sơ đồ nguyên lý của mạch cần thiết kế. Ghi rõ chân của IC.



## B. HƯỚNG DẪN THÍ NGHIỆM:

### I. MỤC TIÊU

- Nắm được cách sử dụng kit thí nghiệm, phần mềm lập trình.
- Nắm được cách khảo sát và thiết kế hàm boolean sử dụng các công logic.
- Nắm được quy trình mô tả phần cứng trên FPGA.

### II. CHUẨN BI

Tự hoàn thành bài prelab1 trước khi tham gia buổi học thí nghiệm đầu tiên. Không hoàn thành bài prelab1 sẽ không được tham gia buổi thí nghiệm, mọi hình thức sao chép đều sẽ bị xử lý nặng.

### III. HƯỚNG DẪN THÍ NGHIỆM

#### THÍ NGHIỆM 1

Mục tiêu: Nắm được cách khảo sát hàm boolean sử dụng các công logic.

Yêu cầu: Sinh viên thực hiện khảo sát hoạt động của hàm  $f(x, y, z) = \bar{x} \cdot y + \bar{y}z$  và điền các kết quả khảo sát vào **Bảng 1.1** theo hướng dẫn ở mục **Kiểm tra**.

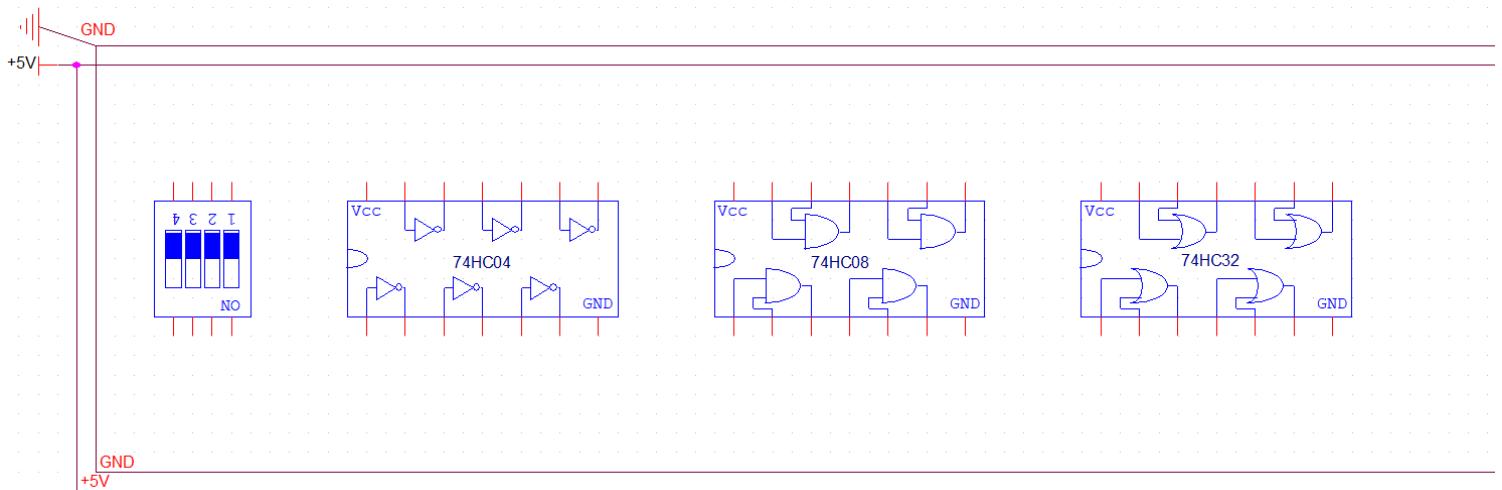
x	y	z	f	f <sub>Test</sub>	f <sub>nand</sub>
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

*Bảng 1.1 Kết quả khảo sát hoạt động của hàm boolean*



**Kiểm tra:**

- Sinh viên tiến hành thay các giá trị của x, y, z vào hàm  $f(x, y, z) = \bar{x} \cdot y + \bar{y}z$  đã cho, điền kết quả vào cột f của **Bảng 1.1**.
- Sinh viên tiến hành vẽ sơ đồ nguyên lý của mạch cần thiết kế. Ghi rõ chân của IC.

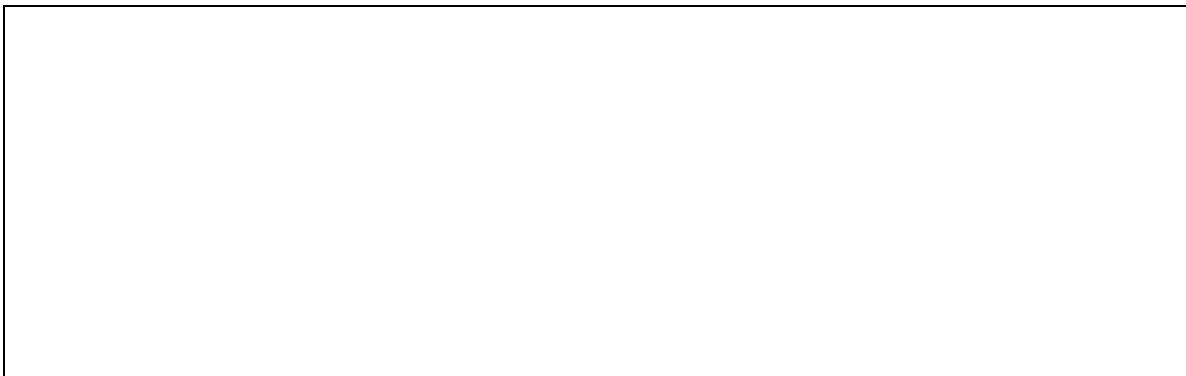


- Sinh viên tiến hành lắp mạch và khảo sát hoạt động của hàm, ghi kết quả vào cột  $f_{Test}$  của **Bảng 1.1**.

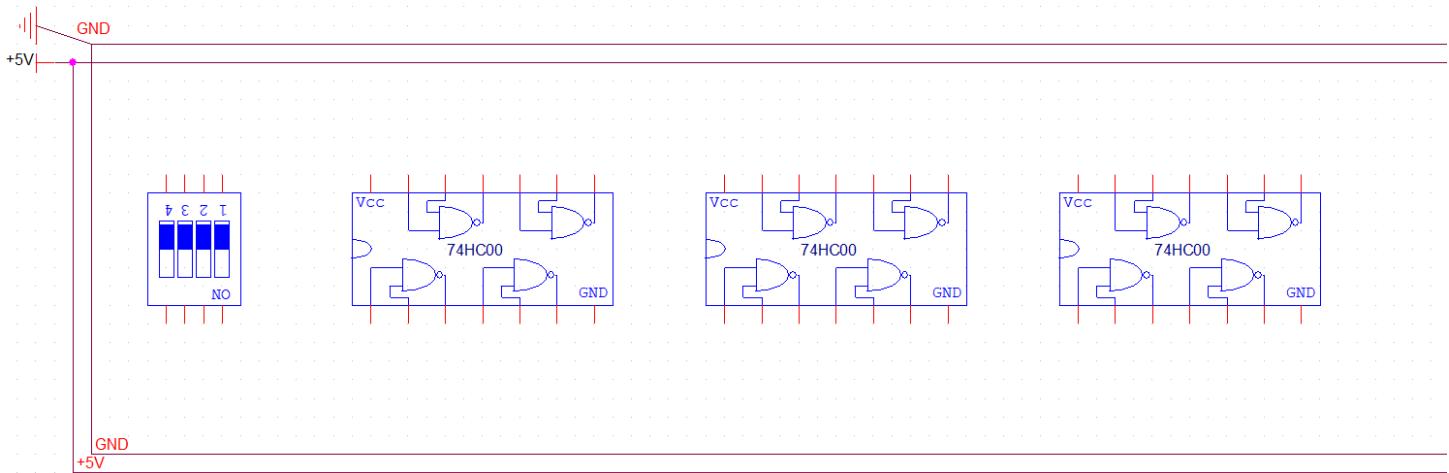
(Chèn hình chụp minh chứng sinh viên đã lắp xong mạch hoặc xác nhận của GVHD)



- Thực hiện hàm boolean trên chỉ sử dụng công NAND 2 ngõ vào, cần sử dụng bao nhiêu công NAND? Tại sao?



- Sinh viên tiến hành vẽ sơ đồ nguyên lý của mạch cần thiết kế (toute NAND).



- Sinh viên tiến hành lắp lại mạch và thực hiện với hàm chỉ sử dụng công NAND. Kết quả kiểm tra điền vào cột  $f_{\text{nand}}$  của **Bảng 1.1**.

(Chèn hình chụp minh chứng sinh viên đã lắp xong mạch hoặc xác nhận của GVHD)



## THÍ NGHIỆM 2

**Mục tiêu:** Nắm được cách khảo sát hàm boolean sử dụng các công logic.

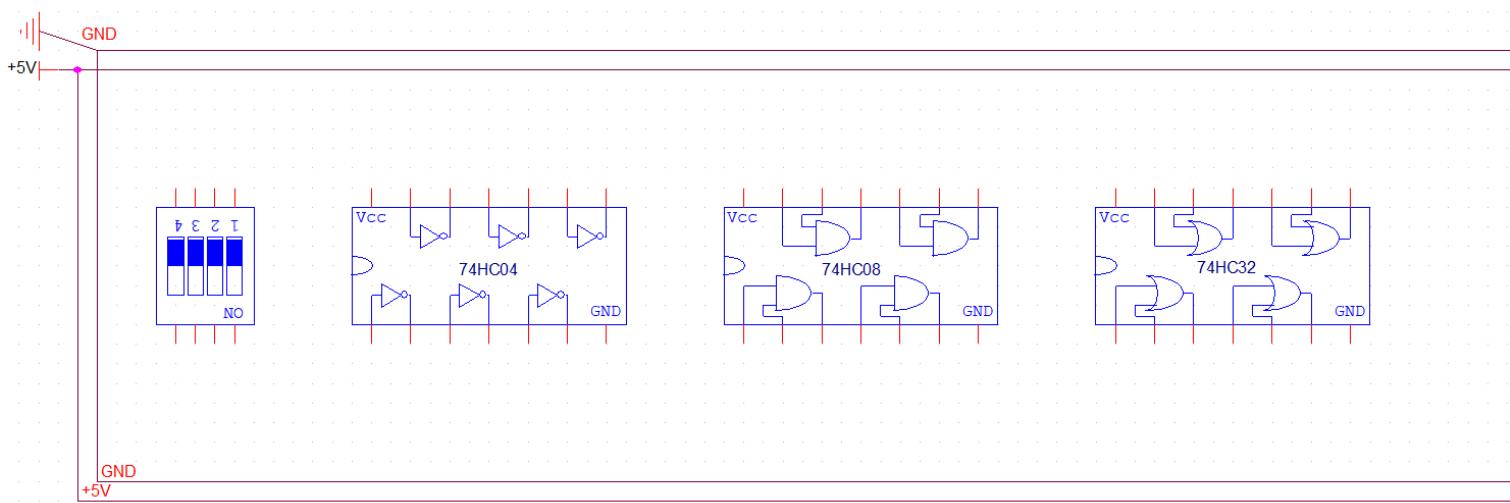
**Yêu cầu:** Sinh viên thực hiện khảo sát hoạt động của hàm được cho bởi **Bảng 1.2** và điền các kết quả khảo sát vào **Bảng 1.2** theo hướng dẫn ở mục **Kiểm tra**.

x	y	z	f	$f_{Test}$	$f_{nor}$
<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>		
<b>0</b>	<b>0</b>	<b>1</b>	<b>1</b>		
<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>		
<b>0</b>	<b>1</b>	<b>1</b>	<b>1</b>		
<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>		
<b>1</b>	<b>0</b>	<b>1</b>	<b>0</b>		
<b>1</b>	<b>1</b>	<b>0</b>	<b>1</b>		
<b>1</b>	<b>1</b>	<b>1</b>	<b>0</b>		

Bảng 1.2: Kết quả khảo sát hoạt động của hàm boolean

### Kiểm tra:

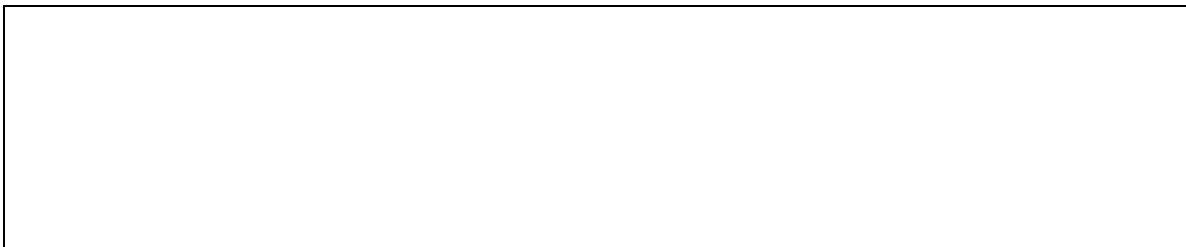
- Viết biểu thức ngõ f theo các ngõ vào x, y, z:
- Sinh viên tiến hành vẽ sơ đồ nguyên lý của mạch cần thiết kế. Ghi rõ chân của IC.



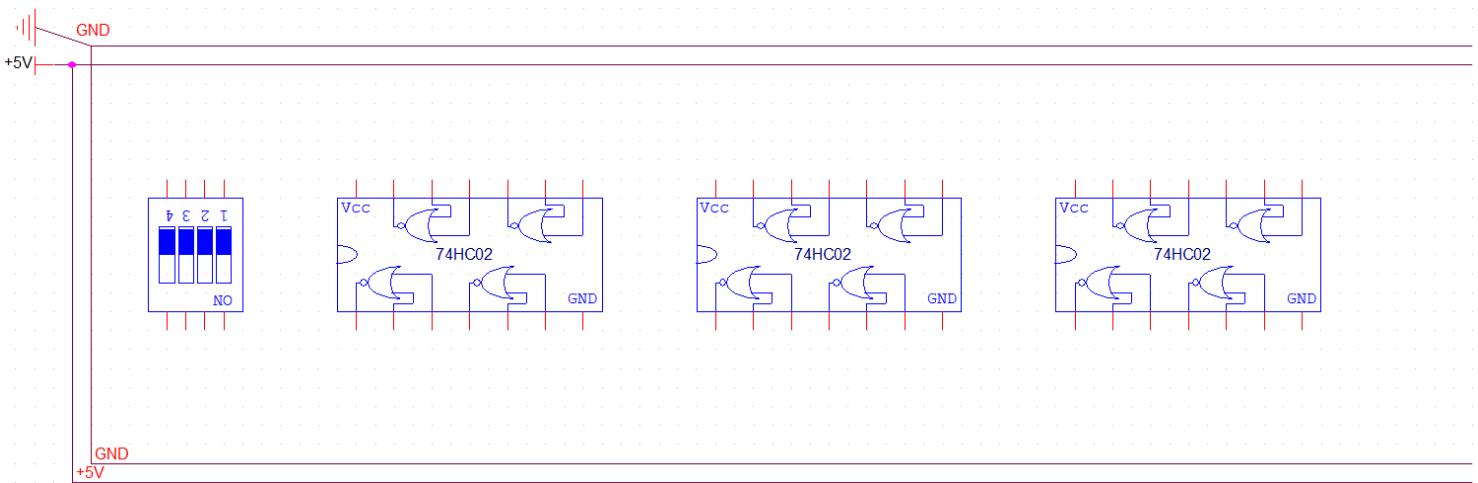
- Sinh viên tiến hành lắp mạch và khảo sát hoạt động của hàm, ghi kết quả vào cột  $f_{Test}$  của **Bảng 1.2**. (Chèn hình chụp minh chứng sinh viên đã lắp xong mạch hoặc xác nhận của GVHD)



- Thực hiện hàm boolean trên chỉ sử dụng công NOR 2 ngõ vào, cần sử dụng bao nhiêu công NOR? Tại sao?



- Sinh viên tiến hành vẽ sơ đồ nguyên lý của mạch cần thiết kế (tồn NOR).



- Sinh viên tiến hành lắp lại mạch và thực hiện với hàm chỉ sử dụng công NOR. Kết quả kiểm tra điền vào cột  $f_{nor}$  của **Bảng 1.2**.

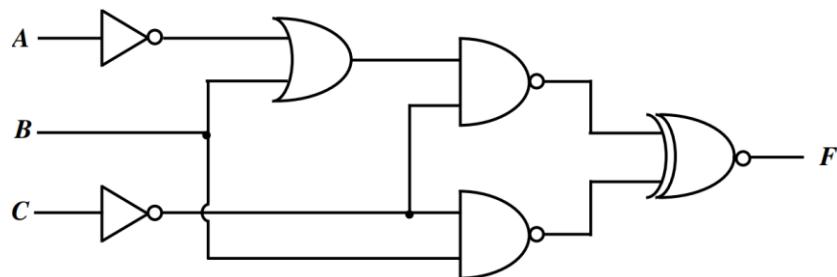
(Chèn hình chụp minh chứng sinh viên đã lắp xong mạch hoặc xác nhận của GVHD)



### THÍ NGHIỆM 3

Mục tiêu: Nắm được cách khảo sát hàm boolean sử dụng các công logic.

Yêu cầu: Sinh viên thực hiện khảo sát hoạt động của hàm được cho bởi **Hình 3** và điền các kết quả khảo sát vào **Bảng 3** theo hướng dẫn ở mục **Kiểm tra**.



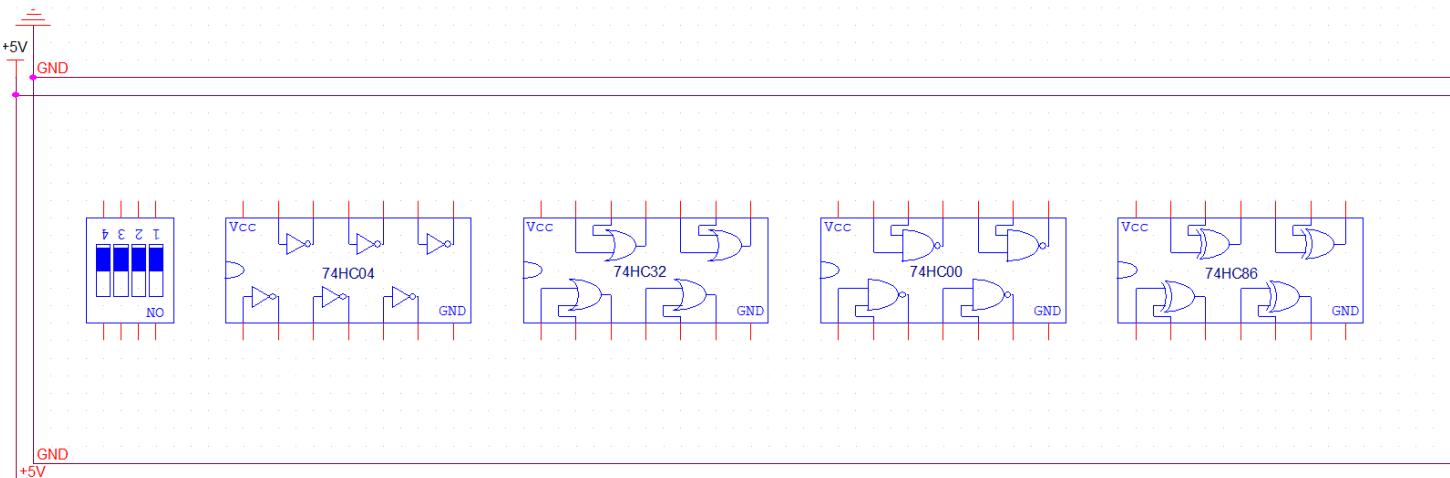
*Hình 1.14: Hàm Boolean của thí nghiệm 3*

x	y	z	F <sub>1</sub>	F <sub>2</sub>
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

*Bảng 1.3: Kết quả khảo sát*

**Kiểm tra:**

- Sinh viên tiến hành lắp mạch và khảo sát hoạt động của hàm, ghi kết quả vào cột F1 của **Bảng 1.3**.
- Sinh viên tiến hành vẽ sơ đồ nguyên lý của mạch cần thiết kế. Ghi rõ chân của IC.



- Chèn hình chụp minh chứng sinh viên đã lắp xong mạch hoặc xác nhận của GVHD:

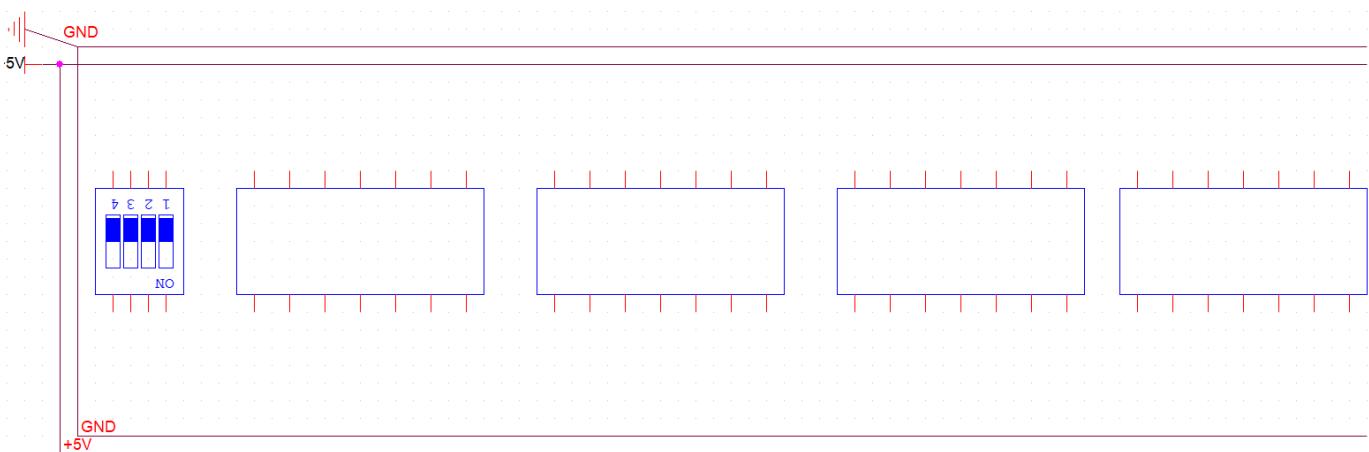


- Sinh viên tiến hành rút gọn hàm đã cho ở Hình 1.14 và vẽ lại mạch logic mô tả hàm boolean đã cho. Sinh viên tiến hành lắp mạch và khảo sát hoạt động của hàm, ghi kết quả vào cột F2 ở Bảng 1.3.

Rút gọn hàm:



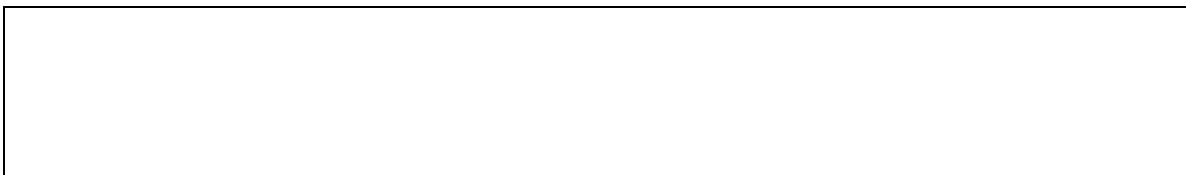
Sinh viên tiến hành vẽ sơ đồ nguyên lý của mạch cần thiết kế. Ghi rõ các IC cần sử dụng.



Chèn hình chụp minh chứng sinh viên đã lắp xong mạch hoặc xác nhận của GVHD:



- Từ kết quả thí nghiệm, sinh viên rút ra nhận xét.



## THÍ NGHIỆM 4

**Mục tiêu:** Nắm được cách thức sử dụng các IC chức năng để thực hiện hàm boolean.

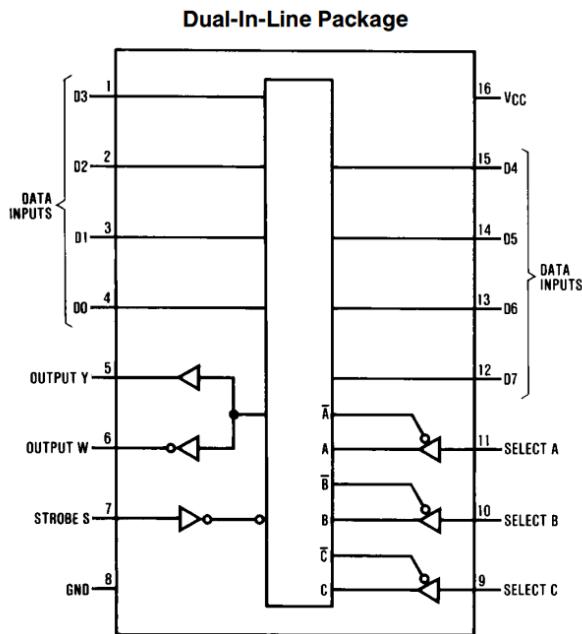
**Yêu cầu:** Sinh viên thực hiện thiết kế hàm boolean  $f(x, y, z) = \sum(2, 3, 5, 7)$  sử dụng IC chức năng 74LS151 và các công logic cần thiết. Kết quả khảo sát điền vào **Bảng 1.4** theo hướng dẫn ở mục **Kiểm tra**.

x	y	z	f	f <sub>Test</sub>
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

**Bảng 1.4:** Kết quả khảo sát hoạt động của hàm boolean

Sơ đồ kết nối và bảng hoạt động của 74LS151:

### Connection Diagram



### Function Table

Inputs			Outputs	
Select			Strobe S	Y      W
C	B	A		
X	X	X	H	L      H
L	L	L	L	D0 $\overline{D0}$
L	L	H	L	D1 $\overline{D1}$
L	H	L	L	D2 $\overline{D2}$
L	H	H	L	D3 $\overline{D3}$
H	L	L	L	D4 $\overline{D4}$
H	L	H	L	D5 $\overline{D5}$
H	H	L	L	D6 $\overline{D6}$
H	H	H	L	D7 $\overline{D7}$

H = High Level, L = Low Level, X = Don't Care  
D0 thru D7 = the level of the respective D input

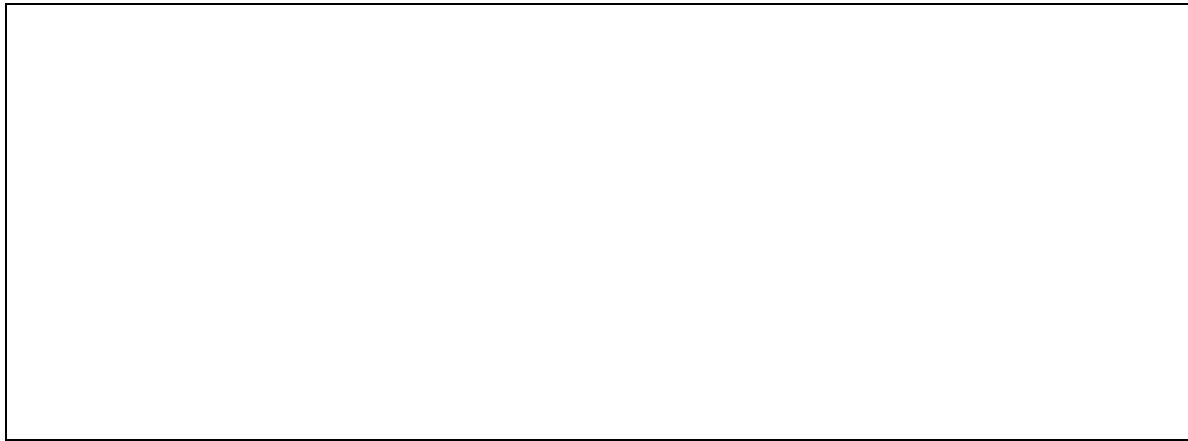
Kiểm tra:



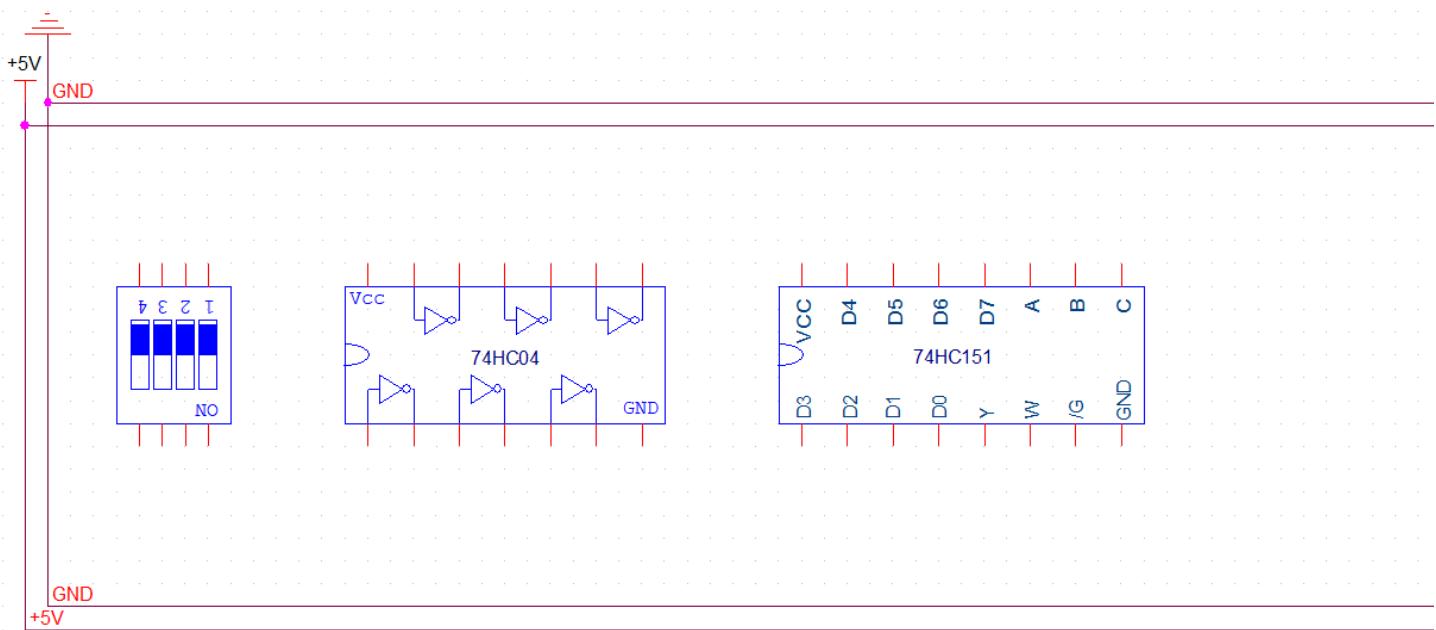
Electronics Department

Ho Chi Minh City University of Technology, Vietnam

- Sinh viên sử dụng IC 74LS151 và các công logic cần thiết để thiết kế mạch thực hiện hàm boolean đã cho.



- Vẽ sơ đồ nguyên lý của mạch đã thiết kế.



- Lắp mạch thực tế và khảo sát mạch, điền kết quả vào cột  $f_{Test}$  ở bảng 1.4  
➤ Chèn hình chụp minh chứng sinh viên đã lắp xong mạch hoặc xác nhận của GVHD:

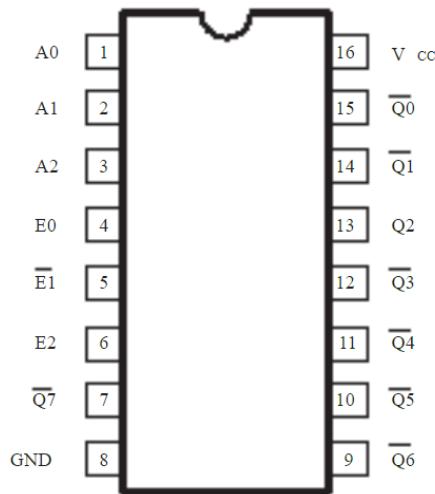


## THÍ NGHIỆM 5

**Mục tiêu:** Nắm được cách thức sử dụng các IC chức năng để thực hiện hàm boolean.

**Yêu cầu:** Sinh viên thực hiện thiết kế hàm boolean  $f(x, y, z) = \sum(2, 3, 5, 7)$  sử dụng IC chức năng 74LS138 và các công logic cần thiết. Kết quả khảo sát điền vào **Bảng 1.5** theo hướng dẫn ở mục **Kiểm tra**.

Sơ đồ chân của IC 74LS138:



x	y	z	f	f <sub>Test</sub>
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

**Bảng 1.5:** Kết quả khảo sát hoạt động của hàm boolean

**Bảng hoạt động của 74LS138:**

INPUTS						OUTPUTS							
$\overline{E_0}$	$\overline{E_1}$	E2	A0	A1	A2	$\overline{Q_0}$	$\overline{Q_1}$	$\overline{Q_2}$	$\overline{Q_3}$	$\overline{Q_4}$	$\overline{Q_5}$	$\overline{Q_6}$	$\overline{Q_7}$
H	X	X	X	X	X	H	H	H	H	H	H	H	H
X	H	X	X	X	X	H	H	H	H	H	H	H	H
X	X	L	X	X	X	H	H	H	H	H	H	H	H
L	L	H	L	L	L	L	H	H	H	H	H	H	H
L	L	H	H	L	L	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	L	H	H	H	H	H
L	L	H	H	H	L	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	L	H	H	H	H
L	L	H	H	L	H	H	H	H	H	L	H	H	H
L	L	H	L	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	L	H

H = High voltage level

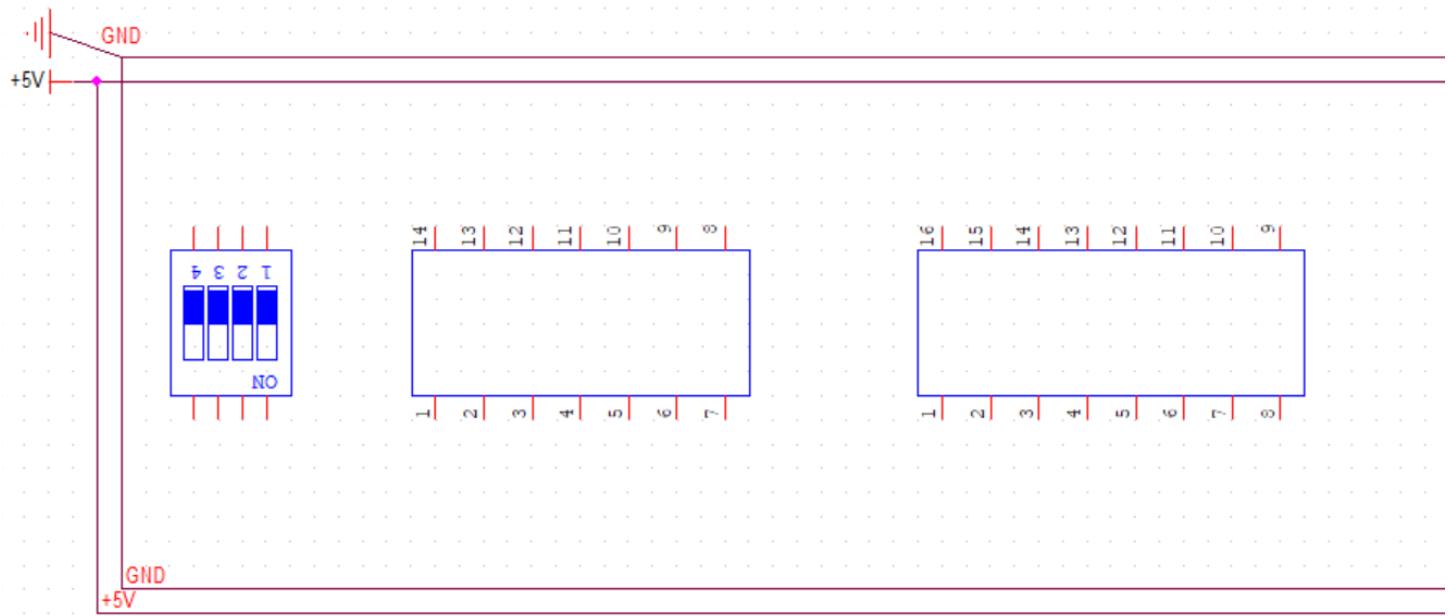
L = Low voltage level

X = Don't care

**Kiểm tra:**

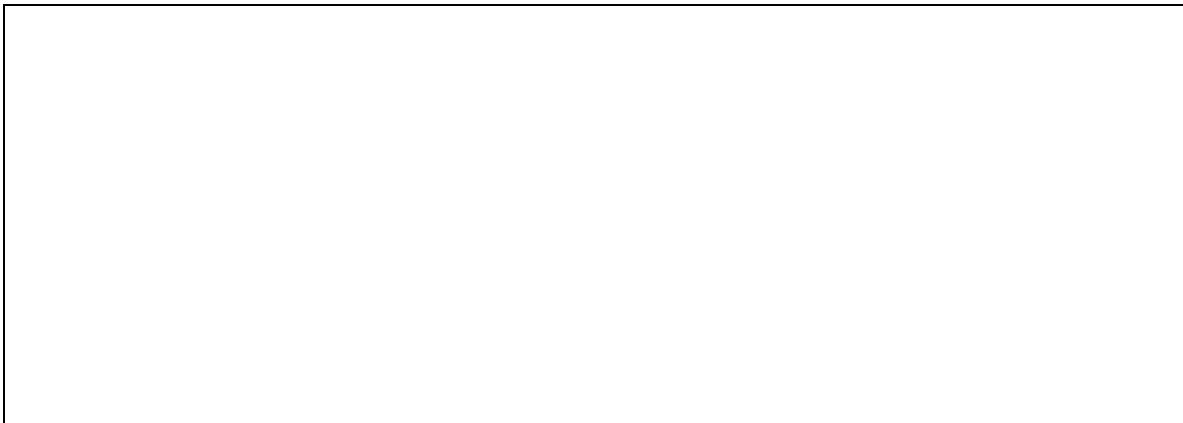
- Sinh viên sử dụng IC 74LS138 và các công logic cần thiết để thiết kế mạch thực hiện hàm boolean đã cho.

- Vẽ sơ đồ nguyên lý của mạch đã thiết kế.



- Lắp mạch thực tế và khảo sát mạch, điền kết quả vào cột  $f_{Test}$  ở bảng

Chèn hình chụp minh chứng sinh viên đã lắp xong mạch hoặc xác nhận của GVHD:



# LAB 2: THỰC HIỆN CÁC IC CHỨC NĂNG CƠ BẢN TRÊN FPGA

Họ và tên:	Lớp TN:
MSSV:	Ngày:

## A. PRELAB

### I. MỤC TIÊU

- Nắm được cách sử dụng kit thí nghiệm, phần mềm lập trình.
- Nắm được cách khảo sát và thiết kế hệ mạch để sử dụng các IC chức năng cơ bản.
- Nắm được quy trình mô tả phần cứng trên FPGA.

### II. CHUẨN BI

- Để chuẩn bị tốt cho bài thí nghiệm, sinh viên **PHẢI** đọc trước phần Phụ lục 1 và hoàn thành các bước của Sample lab trong Phụ lục 2, nộp kèm với Prelab 2 trước khi vào lớp.
- Sinh viên **phải** hoàn thành và nộp Prelab 2 trước khi vào lớp.

Sinh viên xem phần phụ lục và Kit DE2 Manual để hiểu cách sử dụng Kit DE2, nối dây, các thiết bị ngoại vi, cách sử dụng phần mềm Quartus để mô phỏng, tổng hợp mạch. Sinh viên tham khảo các tài liệu để hiểu cách viết thiết kế phần cứng bằng ngôn ngữ SystemVerilog.

Với Lab 2 sinh viên sẽ sử dụng các ngoại vi LEDR, LEDG, và SW. Dưới đây là header nối dây cẩn bản của các ngoại vi này khi sinh viên tổng hợp project trên Intel Quartus. (Lưu ý import file DE2\_pin\_list/assignment).



Lưu ý, top-level của project phải đặt tên là tên của file header (nối pin), ví dụ là “lab2tn1\_wrapper”, với module “lab2tn1” là module chứa đoạn mã mô tả thiết kế của thí nghiệm 1.

Giả sử, với thí nghiệm viết thiết kế phần cứng bằng ngôn ngữ SystemVerilog cho hàm  $f(x, y, z) = \bar{x}yz$  là thí nghiệm 1 của lab 2, ta có đoạn mã SystemVerilog như sau:

```

≡ lab2tn1.sv
1  module lab2tn1 (
2      input x,y,z,
3      output f
4  );
5
6  assign f = !x & y & z
7
8  endmodule

```

*Hình 2.1: module “lab2tn1” thực hiện hàm  $f(x, y, z) = \bar{x}yz$*

Vậy đoạn mã của top level “lab2tn1\_wrapper” như sau:

```

≡ lab2tn1_wrapper.sv
1  module lab2tn1_wrapper (
2      input [2:0] SW,
3      output [2:0] LEDR,
4      output [0:0] LEDG
5  );
6
7  lab2tn1 ilab2tn1 (SW[2],SW[1],SW[0],LEDG);
8
9  assign LEDR = SW;
10
11 endmodule

```

*Hình 2.2: Top level “lab2tn1\_wrapper” thực hiện hàm  $f(x, y, z) = \bar{x}yz$*



Như vậy ta đã hoàn thành phần viết đoạn mã của chương trình và project Quartus, sinh viên thực hiện mô phỏng, nạp Kit như đã được hướng dẫn tại Lab 0.

Với các bài thí nghiệm yêu cầu sử dụng IC 74LS151/74LS138, sinh viên viết đoạn mã mô tả IC này và tiến hành gọi module như trên.

## BÀI CHUẨN BỊ 1

**Mục tiêu:** Thực hiện khảo sát hoạt động của hàm Boolean trên Kit DE2

**Yêu cầu:**

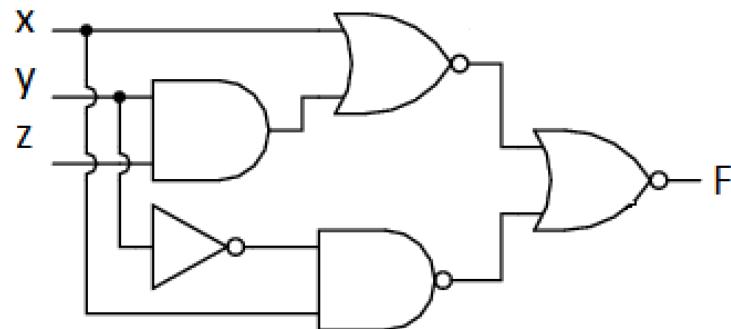
- Sinh viên thực hiện khảo sát hoạt động của hàm  $f(x, y, z) = \bar{x}yz + \bar{x}\bar{y}z + xy$
- Sinh viên viết đoạn mã SystemVerilog thực hiện hoạt động của hàm  $f(x, y, z) = \bar{x}yz + \bar{x}\bar{y}z + xy$  như **hướng dẫn** từ lab 0 với sơ đồ gán chân như sau:
  - Chân x, y, z gán tới SW2, SW1, SW0 và LEDR2, LEDR1, LEDR0 tương ứng
  - Chân f gán tới LEDG0
- Sinh viên tiến hành biên dịch đoạn chương trình trên. Sau đó, sinh viên tiến mô phỏng dạng sóng ngõ ra trong tất cả các trường hợp của ngõ vào. Chụp hình dạng sóng ngõ ra.

## BÀI CHUẨN BỊ 2

**Mục tiêu:** Nắm được cách khảo sát hàm boolean sử dụng kit DE2

**Yêu cầu:** Sinh viên xem mạch được cho bởi **Hình 2.3**





**Hình 2.3**

- Sinh viên viết đoạn mã SystemVerilog thực hiện hoạt động của hàm  $F$  trên **Hình 2.3** như hướng dẫn từ lab 0 với sơ đồ gán chân như sau:
  - Chân x, y, z gán tới SW2, SW1, SW0 và LEDR2, LEDR1, LEDR0 tương ứng
  - Chân f gán tới LEDG0
- Sinh viên tiến hành biên dịch đoạn chương trình trên. Sau đó, sinh viên tiến mô phỏng dạng sóng ngõ ra trong tất cả các trường hợp của ngõ vào. Chụp hình dạng sóng ngõ ra.

### BÀI CHUẨN BỊ 3

**Mục tiêu:** Nắm được cách thức thiết kế các IC chức năng bằng ngôn ngữ SystemVerilog và ứng dụng trên kit DE2

**Yêu cầu:**



Electronics Department

Ho Chi Minh City University of Technology, Vietnam

- Cho hàm boolean  $f(x, y, z) = \Sigma(1,2,4,7)$
- Sinh viên sử dụng IC 74LS151 và các công logic cần thiết để thiết kế mạch thực hiện hàm boolean đã cho.
- Sinh viên viết đoạn mã SystemVerilog thực hiện hoạt động của hàm  $f(x, y, z) = \Sigma(1,2,4,7)$  như **hướng dẫn** từ Lab 0 với sơ đồ gán chân như sau:
  - Chân x, y, z gán tới SW2, SW1, SW0 và LEDR2, LEDR1, LEDR0 tương ứng
  - Chân f gán tới LEDG0
  - Trong đó, gọi IC 74LS151 như một module con
- Sinh viên tiến hành biên dịch đoạn chương trình trên. Sau đó, sinh viên tiến mô phỏng dạng sóng ngõ ra trong tất cả các trường hợp của ngõ vào. Chụp hình dạng sóng ngõ ra.

## BÀI CHUẨN BỊ 4

**Mục tiêu:** Nắm được cách thức thiết kế các IC chức năng bằng ngôn ngữ SystemVerilog và ứng dụng trên kit DE2

### Yêu cầu:

- Cho hàm boolean  $f(x, y, z, w) = \Sigma(2,5,7,9,12,13)$
- Sinh viên sử dụng IC 74LS151 và các công logic cần thiết để thiết kế mạch thực hiện hàm boolean đã cho.
- Sinh viên viết đoạn mã SystemVerilog thực hiện hoạt động của hàm  $f(x, y, z, w)$  như **hướng dẫn** từ Prelab với sơ đồ gán chân như sau:
  - Chân x, y, z, w gán tới SW3, SW2, SW1, SW0 và LEDR3, LEDR2, LEDR1, LEDR0 tương ứng
  - Chân f gán tới LEDG0



- Trong đó, gọi IC 74LS151 như một module con
- Sinh viên tiến hành biên dịch đoạn chương trình trên. Sau đó, sinh viên tiến mô phỏng dạng sóng ngõ ra trong tất cả các trường hợp của ngõ vào. Chụp hình dạng sóng ngõ ra.

## BÀI CHUẨN BỊ 5

**Mục tiêu:** Nắm được cách thức thiết kế các IC chức năng bằng ngôn ngữ SystemVerilog và ứng dụng trên kit DE2

### Yêu cầu:

- Cho hàm boolean  $f(x, y, z) = \Sigma(0, 2, 5, 7)$
- Sinh viên sử dụng IC 74LS138 và các công logic cần thiết để thiết kế mạch thực hiện hàm boolean đã cho.
- Sinh viên viết đoạn mã SystemVerilog thực hiện hoạt động của hàm  $f(x, y, z)$  như hướng dẫn từ Prelab với sơ đồ gán chân như sau:
  - Chân x, y, z gán tới SW2, SW1, SW0 và LEDR2, LEDR1, LEDR0 tương ứng
  - Chân f gán tới LEDG0
  - Trong đó, gọi IC 74LS138 như một module con
- Sinh viên tiến hành biên dịch đoạn chương trình trên. Sau đó, sinh viên tiến mô phỏng dạng sóng ngõ ra trong tất cả các trường hợp của ngõ vào. Chụp hình dạng sóng ngõ ra.



## BÀI CHUẨN BỊ 6

**Mục tiêu:** Nắm được cách thức thiết kế các IC chức năng bằng ngôn ngữ SystemVerilog và ứng dụng trên kit DE2

### Yêu cầu:

- Cho hàm boolean  $f(x, y, z, w) = \sum(1, 3, 5, 7, 10, 12, 14)$
- Sinh viên sử dụng IC 74LS138 và các cổng logic cần thiết để thiết kế mạch thực hiện hàm boolean đã cho.
- Sinh viên viết đoạn mã SystemVerilog thực hiện hoạt động của hàm  $f(x, y, z, w)$  như **hướng dẫn** từ Prelab với sơ đồ gán chân như sau:
  - Chân x, y, z, w gán tới SW3, SW2, SW1, SW0 và LEDR3, LEDR2, LEDR1, LEDR0 tương ứng
  - Chân f gán tới LEDG0
  - Trong đó, gọi IC 74LS138 như một module con
- Sinh viên tiến hành biên dịch đoạn chương trình trên. Sau đó, sinh viên tiến mô phỏng dạng sóng ngõ ra trong tất cả các trường hợp của ngõ vào. Chụp hình dạng sóng ngõ ra.



## B. HƯỚNG DẪN THÍ NGHIỆM:

### I. MỤC TIÊU

- Nắm được cách sử dụng kit DE-2, FPGA Cyclone II
- Nắm được các kiến thức từ bài thí nghiệm 1 – các IC chức năng cơ bản
- Nắm được quy trình mô tả phần cứng trên FPGA.

### II. CHUẨN BI

- Đã chuẩn bị tốt cho bài thí nghiệm, sinh viên PHẢI đọc trước phần Phụ lục 1 và hoàn thành các bước của Sample lab trong Phụ lục 2, nộp kèm với Prelab 2 trước khi vào lớp.
- Tự hoàn thành bài prelab2 trước khi tham gia buổi học thí nghiệm. Không hoàn thành bài prelab2 sẽ không được tham gia buổi thí nghiệm
- Mọi hình thức sao chép đều sẽ bị xử lý nặng.
- Tuân thủ sơ đồ gán chân để dễ kiểm tra kết quả

### III. HƯỚNG DẪN THÍ NGHIỆM

#### THÍ NGHIỆM 1

Mục tiêu: Thực hiện khảo sát hoạt động của hàm Boolean trên Kit DE 2

Yêu cầu: Sinh viên thực hiện khảo sát hoạt động của hàm  $f(x, y, z) = \bar{x}yz + \bar{x}\bar{y}z + xy$  và điền các kết quả khảo sát vào **Bảng 2.1** theo hướng dẫn ở mục **Kiểm tra**.

x	y	z	f	f <sub>Sim</sub>	f <sub>Kit</sub>
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

*Bảng 2.1: Kết quả khảo sát hoạt động của hàm boolean.*



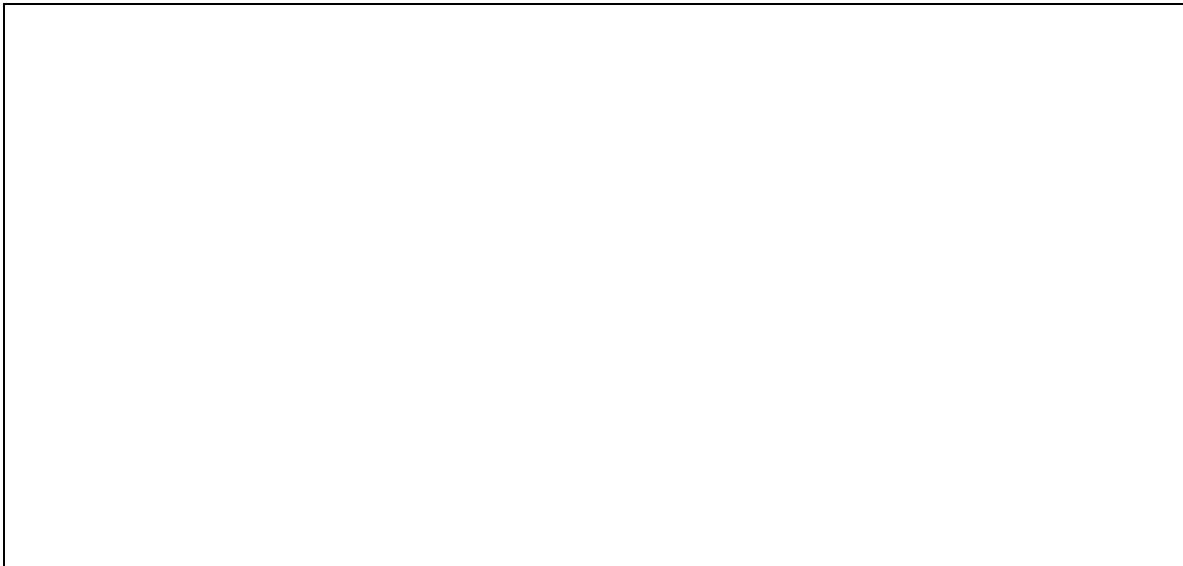
**Kiểm tra:**

- Sinh viên tiến hành thay các giá trị của x, y, z vào hàm  $f(x, y, z) = \bar{x}yz + \bar{x}\bar{y}z + xy$  đã cho, điền kết quả vào cột **f** của **Bảng 2.1**.
- Sinh viên tiến hành vẽ sơ đồ cổng logic của mạch cần thiết kế

- Sinh viên viết đoạn mã SystemVerilog thực hiện hoạt động của hàm  $f(x, y, z) = \bar{x}yz + \bar{x}\bar{y}z + xy$  như **hướng dẫn** từ Prelab với sơ đồ gán chân như sau:
  - Chân x, y, z gán tới SW2, SW1, SW0 và LEDR2, LEDR1, LEDR0 tương ứng
  - Chân f gán tới LEDG0

- Sinh viên tiến hành tổng hợp thiết kế System Verilog và khảo sát hoạt động của hàm trên mô phỏng, ghi kết quả vào cột  $f_{sim}$  của **Bảng 2.1**.

(Chèn hình chụp minh chứng sinh viên đã mô phỏng mạch)



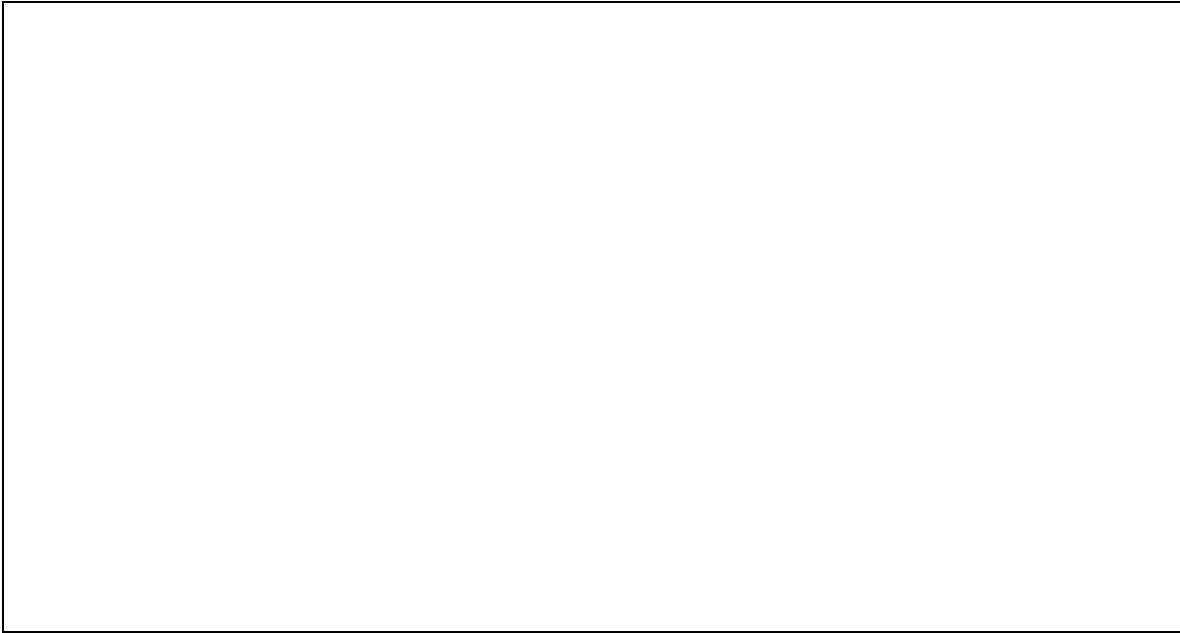
- Sinh viên xem kết quả Netlist > RTL Viewer của mạch. Kết quả này có giống với sơ đồ công logic sinh viên đã vẽ không, tại sao?

(Chèn hình chụp Netlist > RTL Viewer)



- Sinh viên thực hiện nạp đoạn mã lên Kit DE2, thiết kế có thực hiện chính xác như yêu cầu hay không? Ghi kết quả vào cột  $f_{Kit}$  của **Bảng 2.1**.

(Chèn hình chụp minh chứng sinh viên đã nạp lên Kit hoạt động)



## THÍ NGHIỆM 2

Mục tiêu: Nắm được cách khảo sát hàm boolean sử dụng kit DE2

Yêu cầu: Sinh viên thực hiện khảo sát hoạt động của hàm được cho bởi **Bảng 2.2** và điền các kết quả khảo sát vào **Bảng 2.2** theo hướng dẫn ở mục **Kiểm tra**.

x	y	z	f	$f_{Sim}$	$f_{Kit}$
0	0	0	1		
0	0	1	1		
0	1	0	0		
0	1	1	1		
1	0	0	1		
1	0	1	0		
1	1	0	0		
1	1	1	1		

*Bảng 2.2: Khảo sát hoạt động hàm Boolean*

Kiểm tra:



Electronics Department

Ho Chi Minh City University of Technology, Vietnam

- Viết biểu thức ngõ f theo các ngõ vào x, y, z.

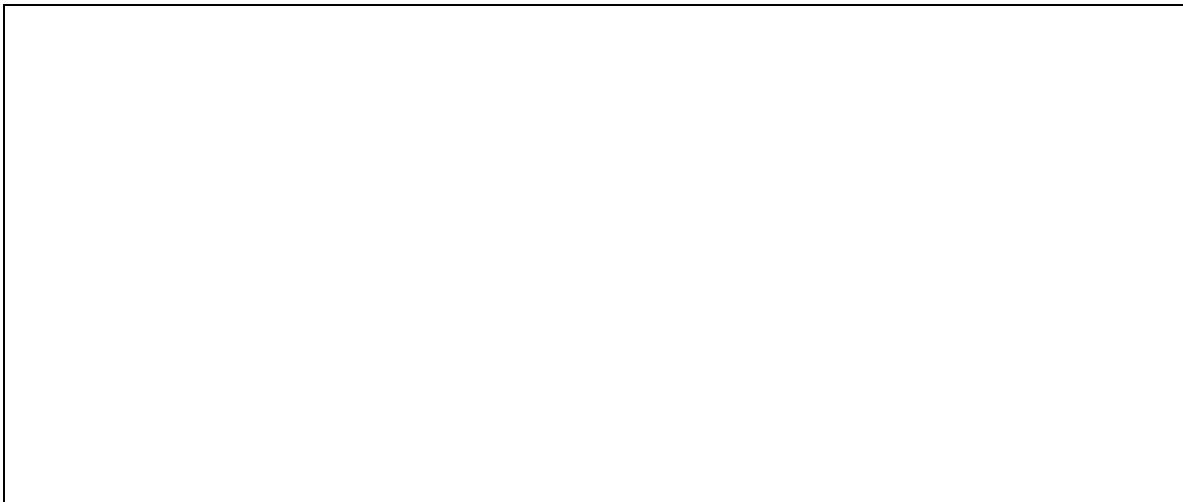
- Sinh viên tiến hành vẽ sơ đồ công logic của mạch cần thiết kế.

- Sinh viên viết đoạn mã SystemVerilog thực hiện hoạt động của hàm  $f(x, y, z)$  như hướng dẫn từ Prelab với sơ đồ gán chân như sau:

- Chân x, y, z gán tới SW2, SW1, SW0 và LEDR2, LEDR1, LEDR0 tương ứng
- Chân f gán tới LEDG0

- 
- Sinh viên tiến hành tổng hợp thiết kế System Verilog và khảo sát hoạt động của hàm trên mô phỏng, ghi kết quả vào cột  $f_{sim}$  của **Bảng 2.2**.

(Chèn hình chụp minh chứng sinh viên đã mô phỏng mạch)



- Sinh viên xem kết quả Netlist > RTL Viewer của mạch. Kết quả này có giống với sơ đồ công logic sinh viên đã vẽ không, tại sao?

(Chèn hình chụp Netlist > RTL Viewer)



- Sinh viên thực hiện nạp đoạn mã lên Kit DE2, thiết kế có thực hiện chính xác như yêu cầu hay không? Ghi kết quả vào cột  $f_{Kit}$  của **Bảng 2.2**.

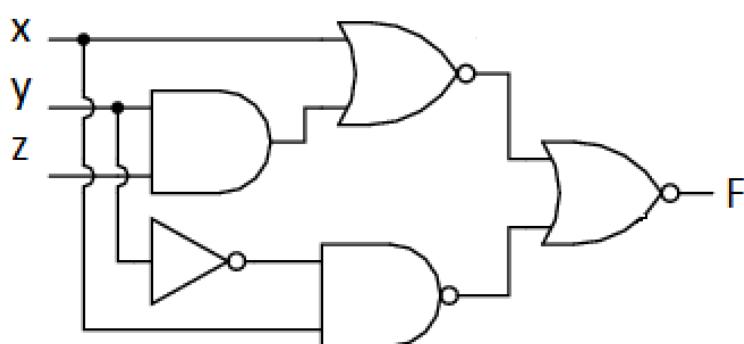
(Chèn hình chụp minh chứng sinh viên đã nạp lên Kit hoạt động)



### THÍ NGHIỆM 3

Mục tiêu: Näm được cách khảo sát hàm boolean sử dụng kit DE2.

Yêu cầu: Sinh viên xem mạch được cho bởi **Hình 2.4** và điền các kết quả khảo sát vào **Bảng 2.3** theo hướng dẫn ở mục **Kiểm tra**.



**Hình 2.4**

x	y	z	$f_{sim}$	$f_{Kit}$
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

**Bảng 2.3** Kết quả khảo sát hoạt động của hàm Boolean - thí nghiệm 3

**Kiểm tra:**

- Sinh viên viết đoạn mã SystemVerilog thực hiện hoạt động của hàm  $F$  trên **Hình 2.4** như **hướng dẫn** từ Prelab với sơ đồ gán chân như sau:
- Chân x, y, z gán tới SW2, SW1, SW0 và LEDR2, LEDR1, LEDR0 tương ứng
  - Chân f gán tới LEDG0

- Sinh viên xem kết quả Netlist > RTL Viewer của mạch. Kết quả này có giống với sơ đồ công logic trên **Hình 2.4 không**, tại sao?

(Chèn hình chụp Netlist > RTL Viewer)

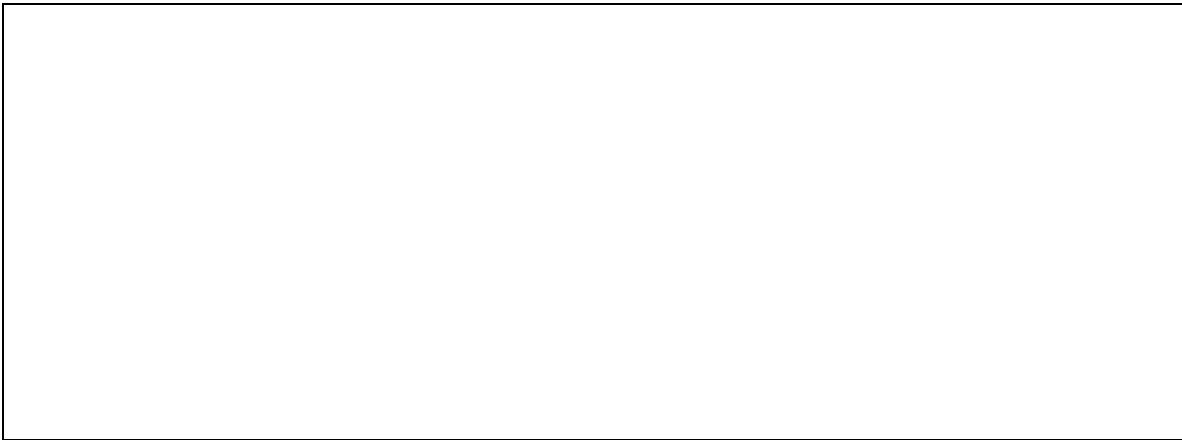
- Sinh viên tiến hành tổng hợp thiết kế System Verilog và khảo sát hoạt động của hàm trên mô phỏng, ghi kết quả vào cột  $f_{sim}$  của **Bảng 2.3**.

(Chèn hình chụp minh chứng sinh viên đã mô phỏng mạch)



- Sinh viên thực hiện nạp đoạn mã lên Kit DE2, thiết kế có thực hiện chính xác như yêu cầu hay không? Ghi kết quả vào cột  $f_{Kit}$  của **Bảng 2.3**.

(Chèn hình chụp minh chứng sinh viên đã nạp lên Kit hoạt động)



## THÍ NGHIỆM 4

**Mục tiêu:** Nắm được cách thức thiết kế các IC chức năng bằng ngôn ngữ SystemVerilog và ứng dụng trên kit DE2

**Yêu cầu:** Sinh viên thực hiện thiết kế hàm boolean  $f(x, y, z) = \Sigma(1, 2, 4, 7)$  sử dụng IC chức năng 74LS151 và các công logic cần thiết trên ngôn ngữ SystemVerilog. Kết quả khảo sát điền vào **Bảng 2.4** theo hướng dẫn ở mục **Kiểm tra**.

x	y	z	f	$f_{sim}$	$f_{Kit}$
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

**Bảng 2.4** Kết quả khảo sát hoạt động của hàm boolean – thí nghiệm 4

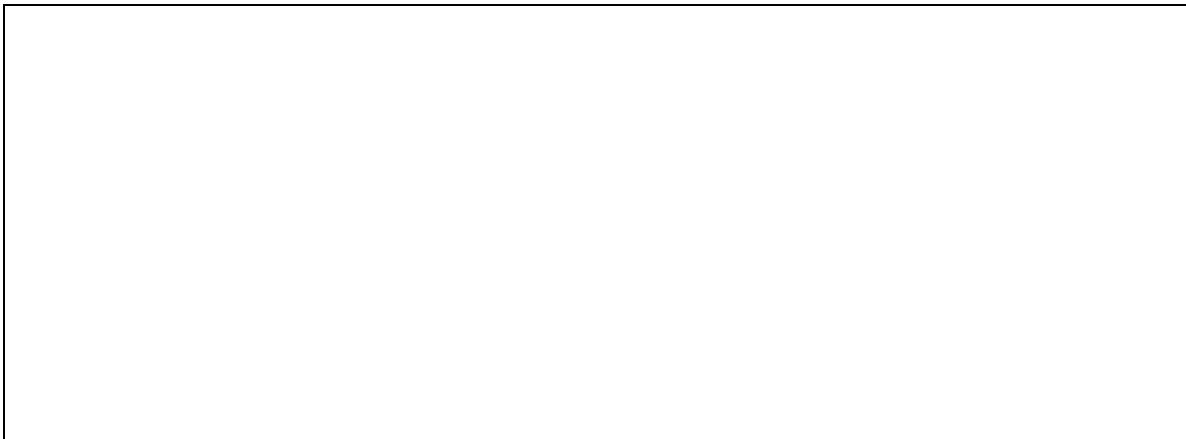
### Kiểm tra:

- Sinh viên sử dụng IC 74LS151 và các công logic cần thiết để thiết kế mạch thực hiện hàm boolean đã cho.

- Sinh viên viết đoạn mã SystemVerilog thực hiện hoạt động của hàm  $f(x, y, z) = \Sigma(1, 2, 4, 7)$  như hướng dẫn từ Prelab với sơ đồ gán chân như sau:

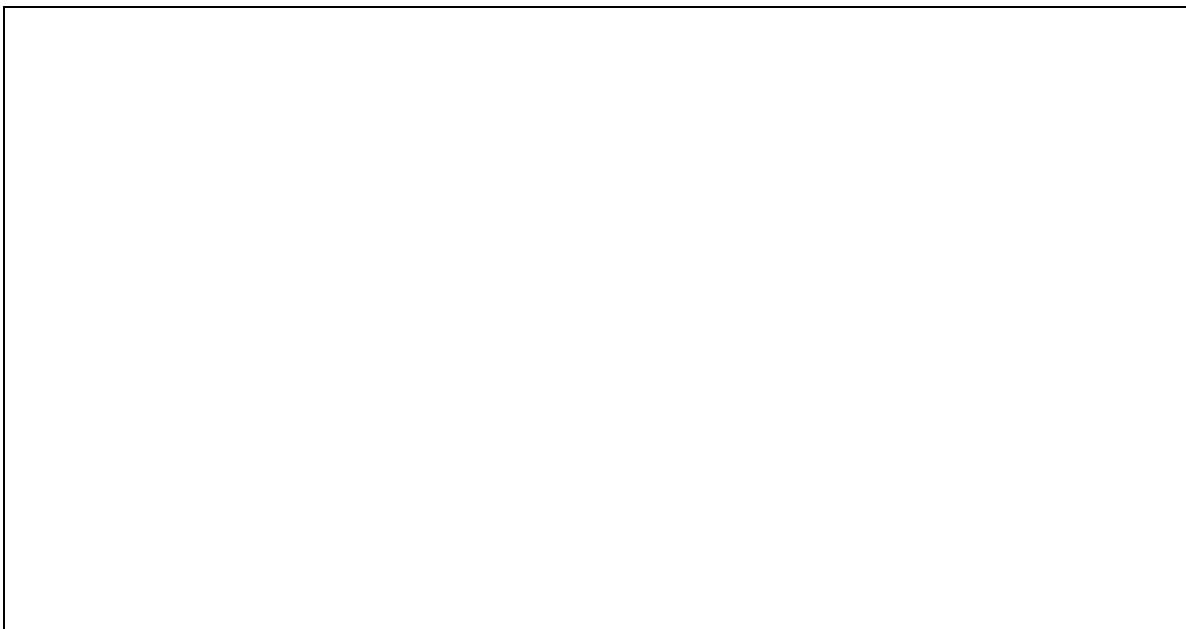


- Chân x, y, z gán tới SW2, SW1, SW0 và LEDR2, LEDR1, LEDR0 tương ứng
- Chân f gán tới LEDG0
- Trong đó, gọi IC 74LS151 như một module con



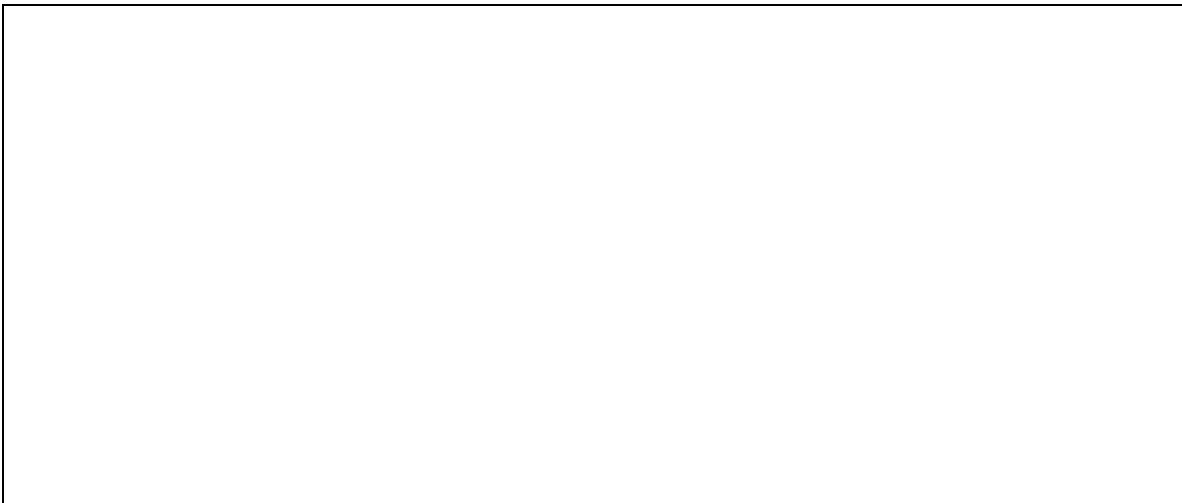
- Sinh viên xem kết quả Netlist > RTL Viewer của mạch. Kết quả này có giống với sơ đồ đã thiết kế hay không, tại sao?

(Chèn hình chụp Netlist > RTL Viewer)



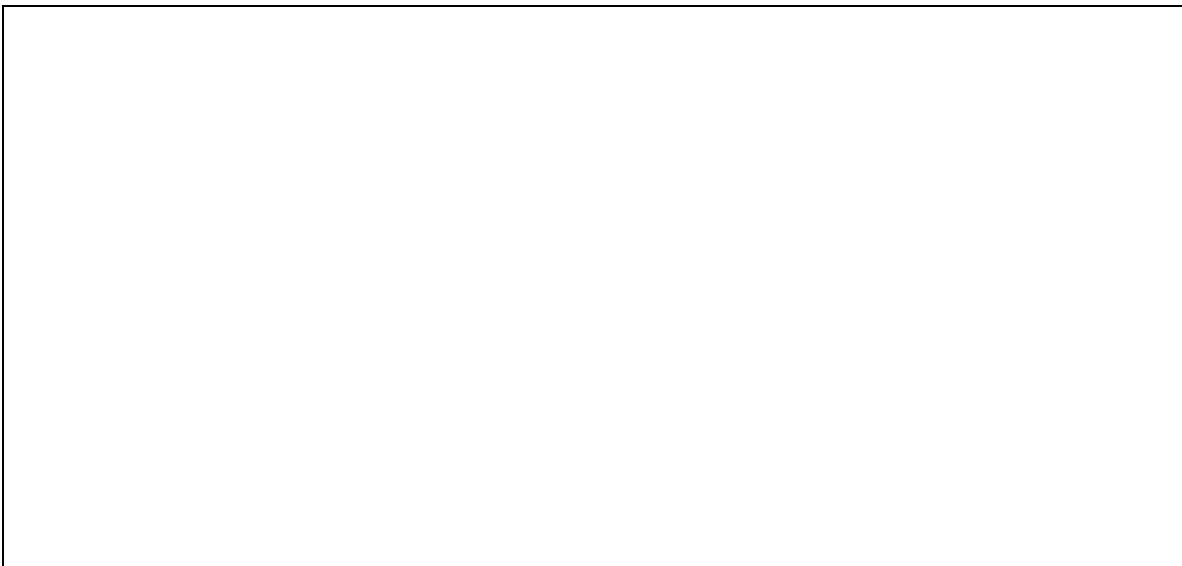
- Sinh viên tiến hành tổng hợp thiết kế System Verilog và khảo sát hoạt động của hàm trên mô phỏng, ghi kết quả vào cột  $f_{sim}$  của **Bảng 2.4**.

(Chèn hình chụp minh chứng sinh viên đã mô phỏng mạch)



- Sinh viên thực hiện nạp đoạn mã lên Kit DE2, thiết kế có thực hiện chính xác như yêu cầu hay không? Ghi kết quả vào cột  $f_{Kit}$  của **Bảng 2.4**.

(Chèn hình chụp minh chứng sinh viên đã nạp lên Kit hoạt động)



## THÍ NGHIỆM 5

**Mục tiêu:** Nắm được cách thức thiết kế các IC chức năng bằng ngôn ngữ SystemVerilog và ứng dụng trên kit DE2

**Yêu cầu:** Sinh viên thực hiện thiết kế hàm boolean  $f(x, y, z, w) = \Sigma(2, 5, 7, 9, 12, 13)$  sử dụng IC chức năng 74LS151 và các công logic cần thiết trên ngôn ngữ SystemVerilog. Kết quả khảo sát điền vào **bảng 2.5** theo hướng dẫn ở mục **Kiểm tra**.

x	y	z	w	f	f <sub>Sim</sub>	f <sub>Kit</sub>
0	0	0	0			
0	0	0	1			
0	0	1	0			
0	0	1	1			
0	1	0	0			
0	1	0	1			
0	1	1	0			
0	1	1	1			
1	0	0	0			
1	0	0	1			
1	0	1	0			
1	0	1	1			
1	1	0	0			
1	1	0	1			
1	1	1	0			
1	1	1	1			

Bảng 2.5: Kết quả khảo sát hoạt động của hàm boolean – thí nghiệm 5

### Kiểm tra:

- Sinh viên sử dụng IC 74LS151 và các công logic cần thiết để thiết kế mạch thực hiện hàm boolean đã cho.

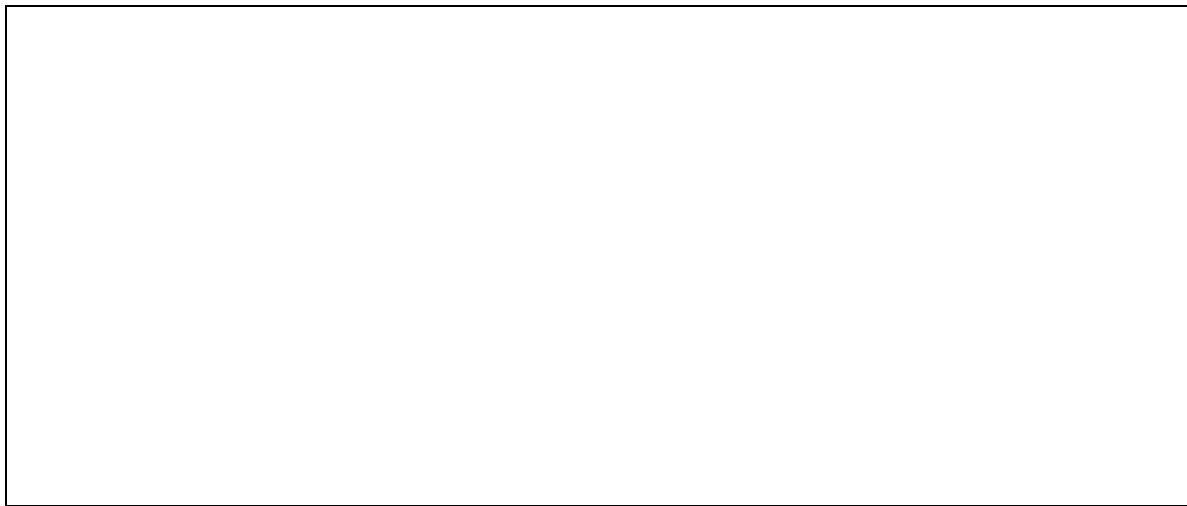




- Sinh viên viết đoạn mã SystemVerilog thực hiện hoạt động của hàm  $f(x, y, z, w)$  như **hướng dẫn** từ Prelab với sơ đồ gán chân như sau:
- Chân x, y, z, w gán tới SW3, SW2, SW1, SW0 và LEDR3, LEDR2, LEDR1, LEDR0 tương ứng
  - Chân f gán tới LEDG0
  - Trong đó, gọi IC 74LS151 như một module con



- Sinh viên xem kết quả Netlist > RTL Viewer của mạch. Kết quả này có giống với sơ đồ đã thiết kế hay không, tại sao?  
(Chèn hình chụp Netlist > RTL Viewer)



- Sinh viên tiến hành tổng hợp thiết kế System Verilog và khảo sát hoạt động của hàm trên mô phỏng, ghi kết quả vào cột  $f_{sim}$  của **Bảng 2.5**.

(Chèn hình chụp minh chứng sinh viên đã mô phỏng mạch)



- Sinh viên thực hiện nạp đoạn mã lên Kit DE2, thiết kế có thực hiện chính xác như yêu cầu hay không? Ghi kết quả vào cột  $f_{kit}$  của **Bảng 2.5**.

(Chèn hình chụp minh chứng sinh viên đã nạp lên Kit hoạt động)



## THÍ NGHIỆM 6

**Mục tiêu:** Nắm được cách thức thiết kế các IC chức năng bằng ngôn ngữ SystemVerilog và ứng dụng trên kit DE2

**Yêu cầu:** Sinh viên thực hiện thiết kế hàm boolean  $f(x, y, z) = \Sigma(0, 2, 5, 7)$  sử dụng IC chức năng 74LS138 và các công logic cần thiết trên ngôn ngữ SystemVerilog.. Kết quả khảo sát điền vào **Bảng 2.6** theo hướng dẫn ở mục **Kiểm tra**.

x	y	z	f	$f_{Sim}$	$f_{Kit}$
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Bảng 2.6: Kết quả khảo sát hoạt động hàm Boolean – thí nghiệm 6

### Kiểm tra:

- Sinh viên sử dụng IC 74LS138 và các công logic cần thiết để thiết kế mạch thực hiện hàm boolean đã cho.





- Sinh viên viết đoạn mã SystemVerilog thực hiện hoạt động của hàm  $f(x, y, z)$  như hướng dẫn từ Prelab với sơ đồ gán chân như sau:

- Chân x, y, z gán tới SW2, SW1, SW0 và LEDR2, LEDR1, LEDR0 tương ứng
- Chân f gán tới LEDG0
- Trong đó, gọi IC 74LS138 như một module con



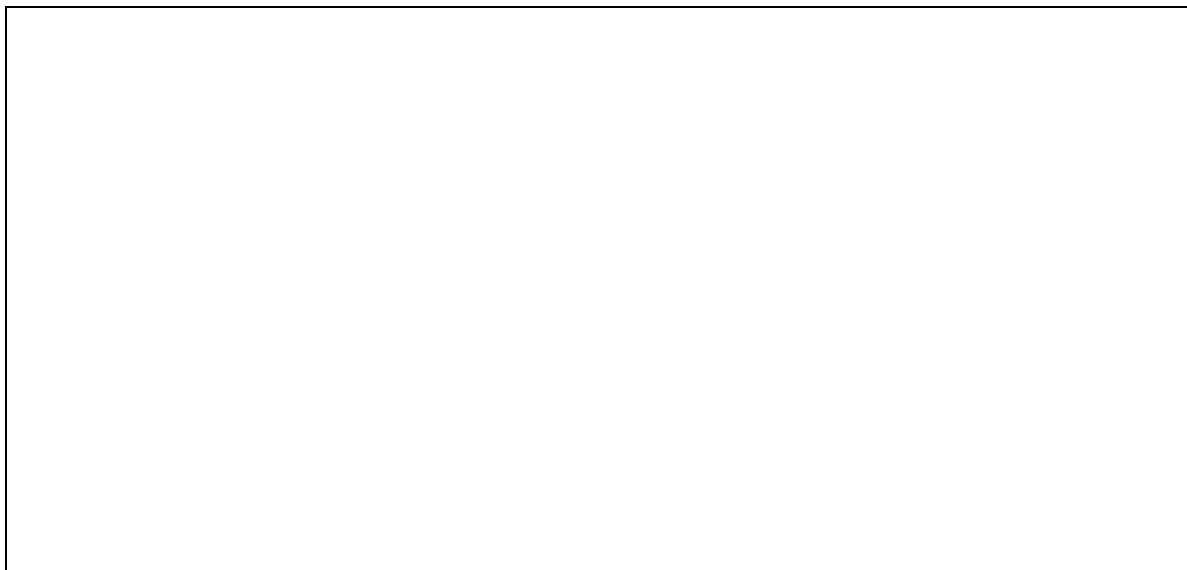
- Sinh viên xem kết quả Netlist > RTL Viewer của mạch. Kết quả này có giống với sơ đồ đã thiết kế hay không, tại sao?

(Chèn hình chụp Netlist > RTL Viewer)



- Sinh viên tiến hành tổng hợp thiết kế System Verilog và khảo sát hoạt động của hàm trên mô phỏng, ghi kết quả vào cột  $f_{sim}$  của **Bảng 2.6**.

(Chèn hình chụp minh chứng sinh viên đã mô phỏng mạch)



- Sinh viên thực hiện nạp đoạn mã lên Kit DE2, thiết kế có thực hiện chính xác như yêu cầu hay không? Ghi kết quả vào cột  $f_{Kit}$  của **Bảng 2.6**.

(Chèn hình chụp minh chứng sinh viên đã nạp lên Kit hoạt động)



## THÍ NGHIỆM 7

**Mục tiêu:** Nắm được cách thức thiết kế các IC chức năng bằng ngôn ngữ SystemVerilog và ứng dụng trên kit DE2

**Yêu cầu:** Sinh viên thực hiện thiết kế hàm boolean  $f(x, y, z, w) = \sum(1, 3, 5, 7, 10, 12, 14)$  sử dụng IC chức năng 74LS138 và các công logic cần thiết trên ngôn ngữ SystemVerilog. Kết quả khảo sát điền vào **bảng 2.7** theo hướng dẫn ở mục **Kiểm tra**.

x	y	z	w	f	$f_{Sim}$	$f_{Kit}$
0	0	0	0			
0	0	0	1			
0	0	1	0			
0	0	1	1			
0	1	0	0			
0	1	0	1			
0	1	1	0			
0	1	1	1			
1	0	0	0			
1	0	0	1			
1	0	1	0			
1	0	1	1			
1	1	0	0			
1	1	0	1			
1	1	1	0			
1	1	1	1			

Bảng 2.7: Kết quả khảo sát hoạt động của hàm Boolean – thí nghiệm 7



**Kiểm tra:**

- Sinh viên sử dụng IC 74LS138 và các công logic cần thiết để thiết kế mạch thực hiện hàm boolean đã cho.

- Sinh viên viết đoạn mã SystemVerilog thực hiện hoạt động của hàm  $f(x, y, z, w)$  như **hướng dẫn** từ Prelab với sơ đồ gán chân như sau:

- Chân x, y, z, w gán tới SW3, SW2, SW1, SW0 và LEDR3, LEDR2, LEDR1, LEDR0 tương ứng
- Chân f gán tới LEDG0
- Trong đó, gọi IC 74LS138 như một module con

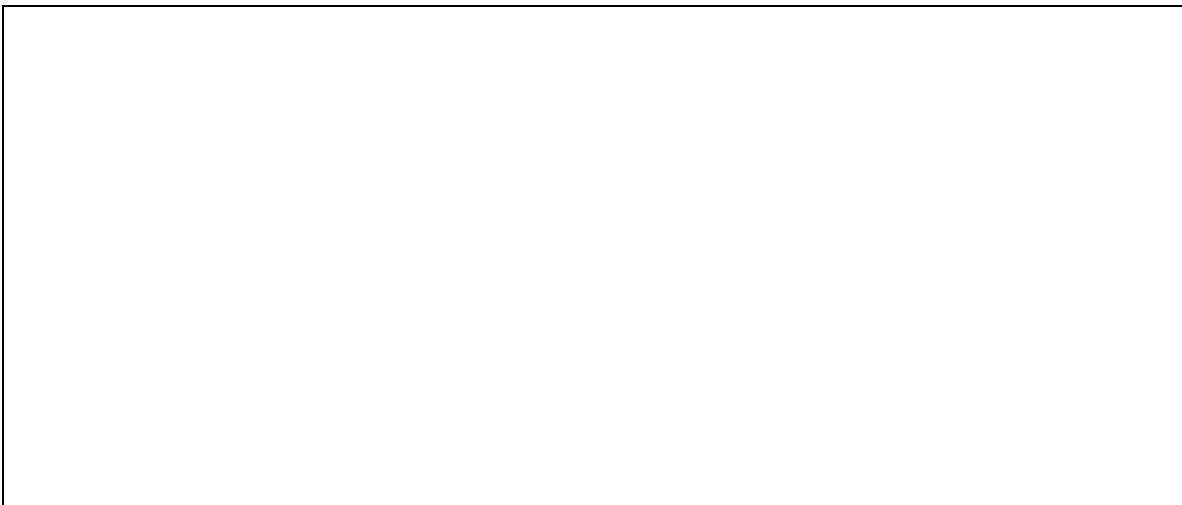
- Sinh viên xem kết quả Netlist > RTL Viewer của mạch. Kết quả này có giống với sơ đồ đã thiết kế hay không, tại sao?

(Chèn hình chụp Netlist > RTL Viewer)



- Sinh viên tiến hành tổng hợp thiết kế System Verilog và khảo sát hoạt động của hàm trên mô phỏng, ghi kết quả vào cột  $f_{sim}$  của **Bảng 2.7**.

(Chèn hình chụp minh chứng sinh viên đã mô phỏng mạch)



- 
- Sinh viên thực hiện nạp đoạn mã lên Kit DE2, thiết kế có thực hiện chính xác như yêu cầu hay không? Ghi kết quả vào cột  $f_{Kit}$  của **Bảng 2.7**.

(Chèn hình chụp minh chứng sinh viên đã nạp lên Kit hoạt động)



# LAB 3: THIẾT KẾ HỆ TỔ HỢP THIẾT KẾ HỆ TUẦN TỰ

Họ và tên:	Lớp TN:
MSSV:	Ngày:

## A. PRELAB

**Câu 1:** Thiết kế mạch cộng toàn phần Full Adder:

Mạch bao gồm 3 ngõ vào và 2 ngõ ra, thực hiện phép cộng 3 bit nhị phân ngõ vào. Hai ngõ vào A và B là 2 bit nhị phân cùng trọng số của 2 số cần thực hiện phép cộng, ngõ vào thứ 3, Ci, là số nhớ của phép cộng 2 bit có trọng số nhỏ hơn trước đó. Hai ngõ ra của mạch cộng toàn phần là tổng S và bit nhớ Co.

**Bảng chân trị:**

A	B	Ci	S	Co

*Bảng 3.1 Bảng chân trị bộ cộng toàn phần*

Sử dụng các IC cổng logic, lắp mạch cộng toàn phần trên Breadboard.

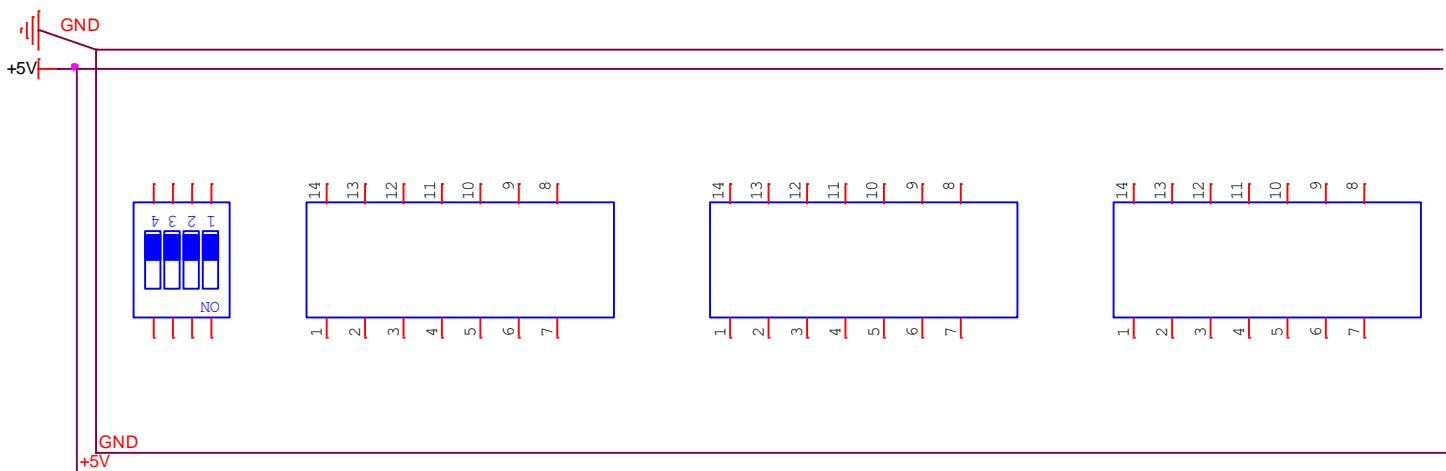
**Xác định các IC cần thiết:**

**Sơ đồ kết nối các IC:**



Electronics Department

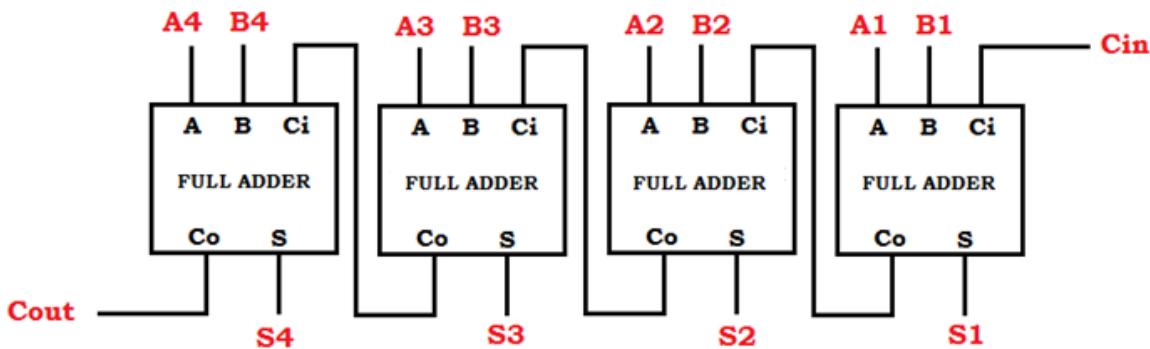
Ho Chi Minh City University of Technology, Vietnam



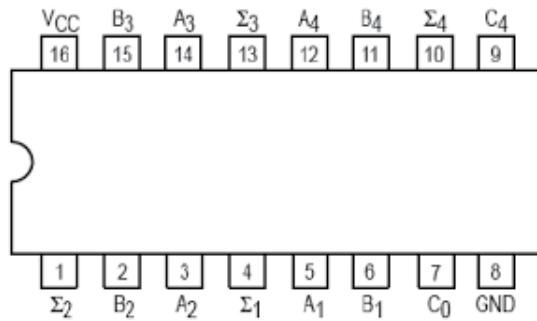
**Câu 2:** Sử dụng IC cộng 74LS283 để thiết kế mạch cộng/trừ hai số nhị phân. Mạch có các ngõ vào S (1bit), A (4bit) và B (4bit) thực hiện chức năng:

- Khi  $S = 0$ , mạch thực hiện  $A + B$
- Khi  $S = 1$ , mạch thực hiện  $A - B$

74LS283 là IC thực hiện phép cộng song song 4 bit, nhận ngõ vào là 2 số A ( $A_3A_2A_1A_0$ ) và B ( $B_3B_2B_1B_0$ ) cùng số nhớ Cin. Mạch cộng 4 bit song song bao gồm 4 mạch cộng toàn phần. Mỗi bộ cộng toàn phần thực hiện phép cộng 2 bit tương ứng của số A và số B, tạo ra tổng S và số nhớ Co; số nhớ Co này được đưa vào ngõ Ci của bộ toàn phần tiếp theo. Kết quả của mạch cộng song song gồm tổng S ( $S_4S_3S_2S_1$ ) và 1 bit nhớ Cout.



Hình 3.1: Mạch cộng song song 4 bit



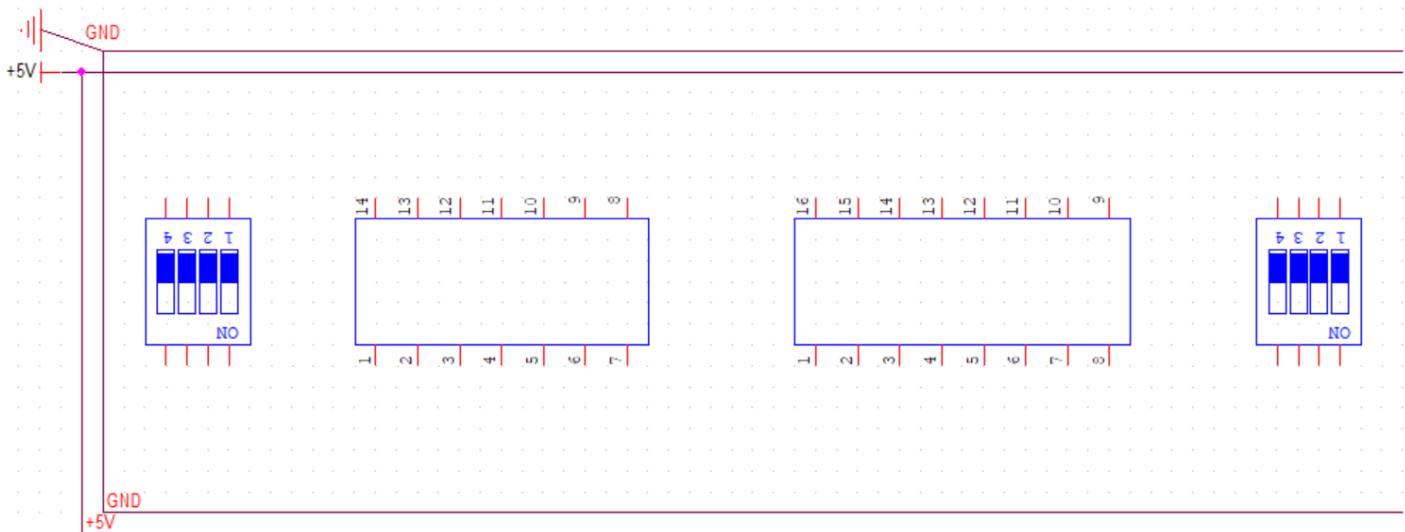
Hình 3.2: Sơ đồ chân của IC 74LS283

### Sơ đồ logic của mạch cộng song song 4 bit – Giải thích cách thiết kế?

S	Phép toán	A <sub>i</sub>	B <sub>i</sub>	C <sub>i</sub>

Lắp mạch cộng toàn phần trên Breadboard, xác định các IC cần thiết:

### Sơ đồ kết nối IC:



### Câu 3: Thiết kế mạch đếm lên từ 0 → 7 sử dụng IC D-FF 74LS74

Bộ đếm nối tiếp thực hiện việc đếm lên hoặc đếm xuống, các trạng thái của bộ đếm hơn/kém nhau 1 đơn vị. Bộ đếm nối tiếp được xây dựng dựa trên các flip-flop, ngõ ra của flip-flop trước là tín hiệu clock của flip-flop tiếp theo.

Schematic Symbol

**Function Table**

Inputs				Outputs	
PR	CLR	CLK	D	Q	$\bar{Q}$
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H (Note 1)	H (Note 1)
H	H	↑	H	H	L
H	H	↑	L	L	H
H	H	L	X	$Q_0$	$\bar{Q}_0$

H = HIGH Logic Level  
 X = Either LOW or HIGH Logic Level  
 L = LOW Logic Level  
 ↑ = Positive-going Transition  
 $Q_0$  = The output logic level of Q before the indicated input conditions were established.  
 Note 1: This configuration is nonstable; that is, it will not persist when either the preset and/or clear inputs return to their inactive (HIGH) level.

*Hình 3.3: Sơ đồ chân và bảng hoạt động của IC 74LS74*

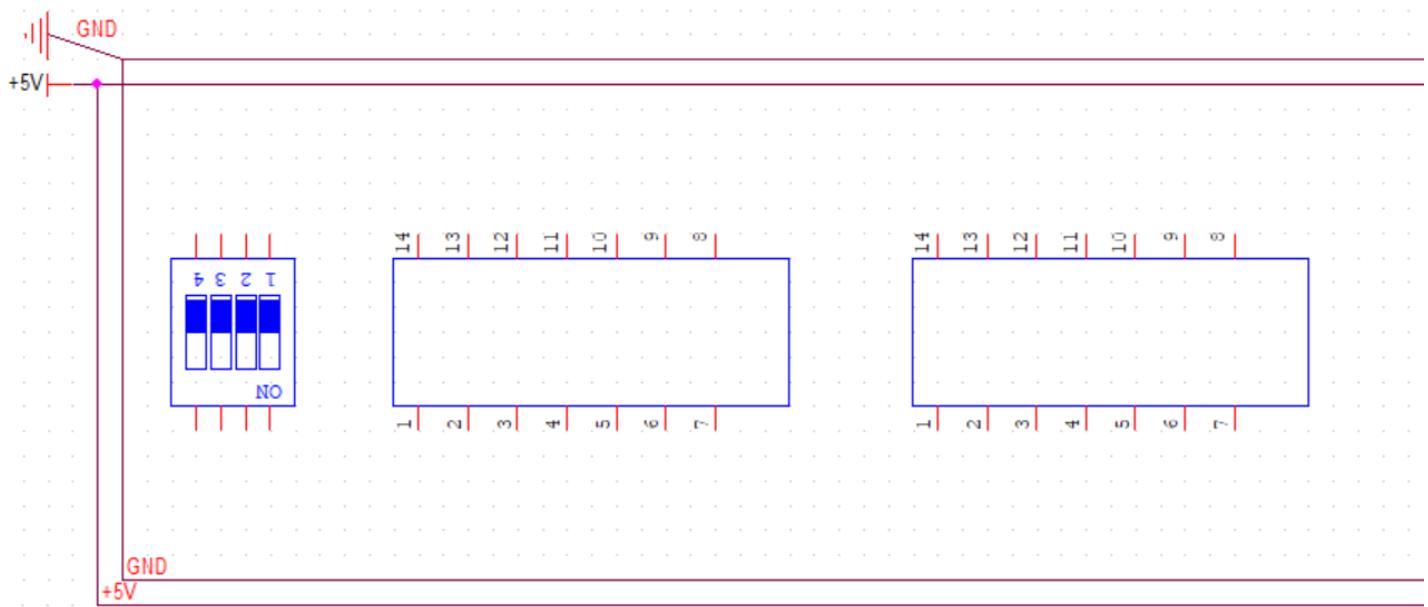
### Sơ đồ mạch đếm nối tiếp từ 0 – 7:

### Sơ đồ kết nối IC:



Electronics Department

Ho Chi Minh City University of Technology, Vietnam



**Câu 4:** Thiết kế hệ tuần tự có giản đồ trạng thái như hình 3.4:

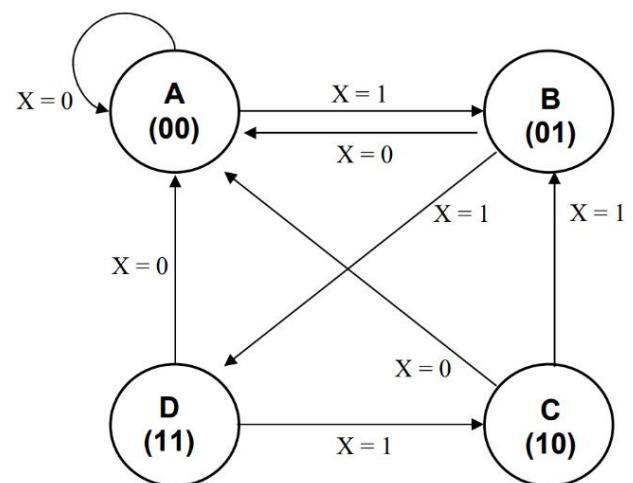
**Bảng chuyển trạng thái:**

TT hiện tại		Ngõ vào	TT kề tiếp		
	S1	S0	X	S1+	S0+
A	0	0	0		
	0	0	1		
	0	1	0		
	0	1	1		
	1	0	0		
	1	0	1		
	1	1	1		

*Bảng 3.2: Bảng chuyển trạng thái*

**Sơ đồ mạch của máy trạng thái:**

$$S1+ = \dots ; S0+ = \dots$$



*Hình 3.4: Giản đồ trạng thái hệ tuần tự*

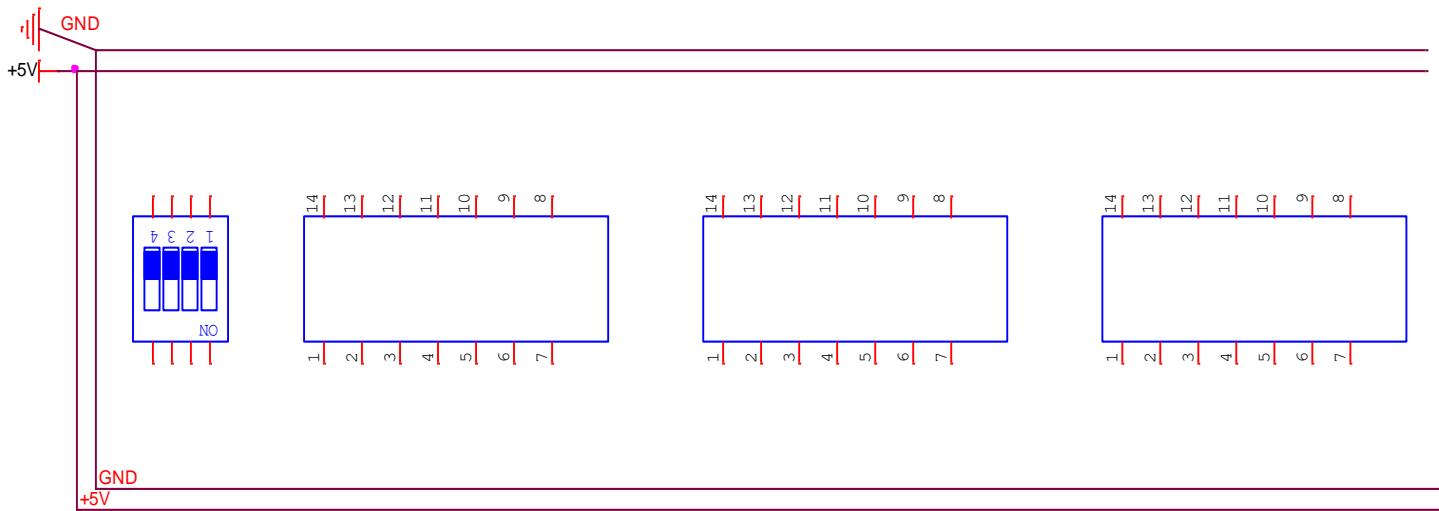


Electronics Department

Ho Chi Minh City University of Technology, Vietnam

Các IC cần sử dụng:

Sơ đồ kết nối IC:



## B. HƯỚNG DẪN THÍ NGHIỆM:

### I. MỤC TIÊU

- Sử dụng vi mạch cộng để thực hiện phép toán cộng/trừ 2 số nhị phân.
- Thiết kế hệ tuần tự tổng quát.

### II. CHUẨN BỊ

- Sinh viên thực hiện PreLab3 tại nhà và nộp kết quả PreLab3 trước khi vào lớp.

*Nếu không thực hiện bài PreLab, sinh viên sẽ không được tham gia thí nghiệm và được xem như vắng buổi học hôm đó.*

### III. HƯỚNG DẪN THÍ NGHIỆM:

#### THÍ NGHIỆM 1

Mục tiêu: Sử dụng cổng logic để thiết kế mạch tổ hợp.

Yêu cầu: Thiết kế mạch cộng toàn phần Full Adder.

Thiết bị:

- IC 74LS08, 74LS32, 74LS86.
- LEDs, điện trở, DIP switch, bộ dây nối.
- Breadboard, nguồn 5V DC.

Sơ đồ thiết kế:

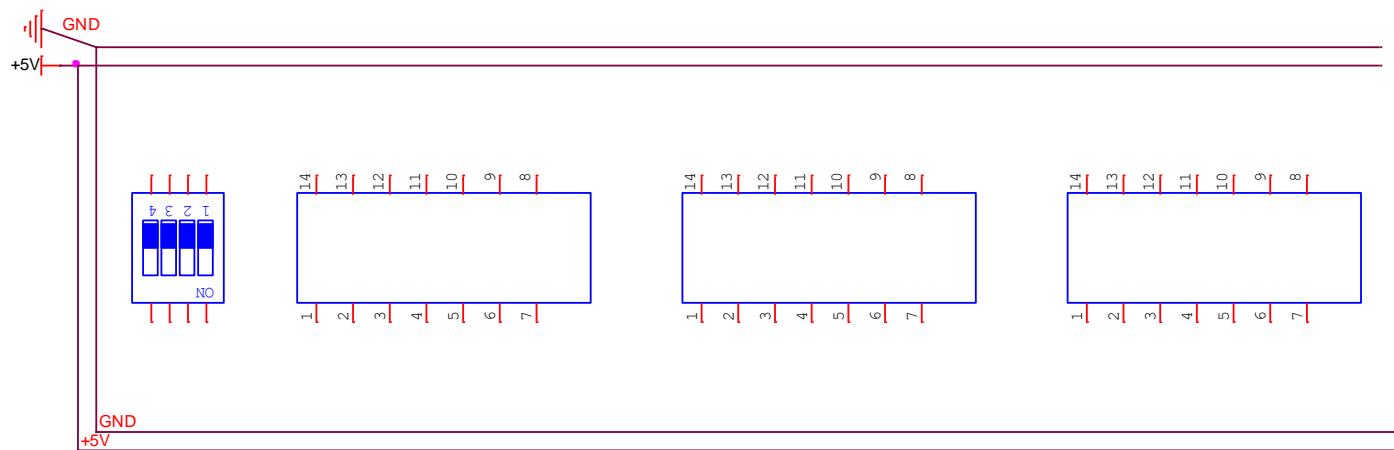
- Sơ đồ mạch:

- Sơ đồ kết nối IC:



Electronics Department

Ho Chi Minh City University of Technology, Vietnam



### Kết quả thí nghiệm:

- Thay đổi các tín hiệu ngõ vào và ghi nhận giá trị ngõ ra vào bảng 3.3:

INPUTS			OUTPUTS	
A	B	Ci	S	Co

Bảng 3.3

## THÍ NGHIỆM 2

Mục tiêu: Sử dụng IC công 74LS283 để thiết kế mạch cộng/trừ hai số nhị phân.

Yêu cầu: Thiết kế mạch có các ngõ vào S (1bit), A (4bit) và B (4bit) thực hiện chức năng:

- Khi S = 0, mạch thực hiện A + B
- Khi S = 1, mạch thực hiện A - B

### Thiết bị:

- IC 74LS283, 74LS86.

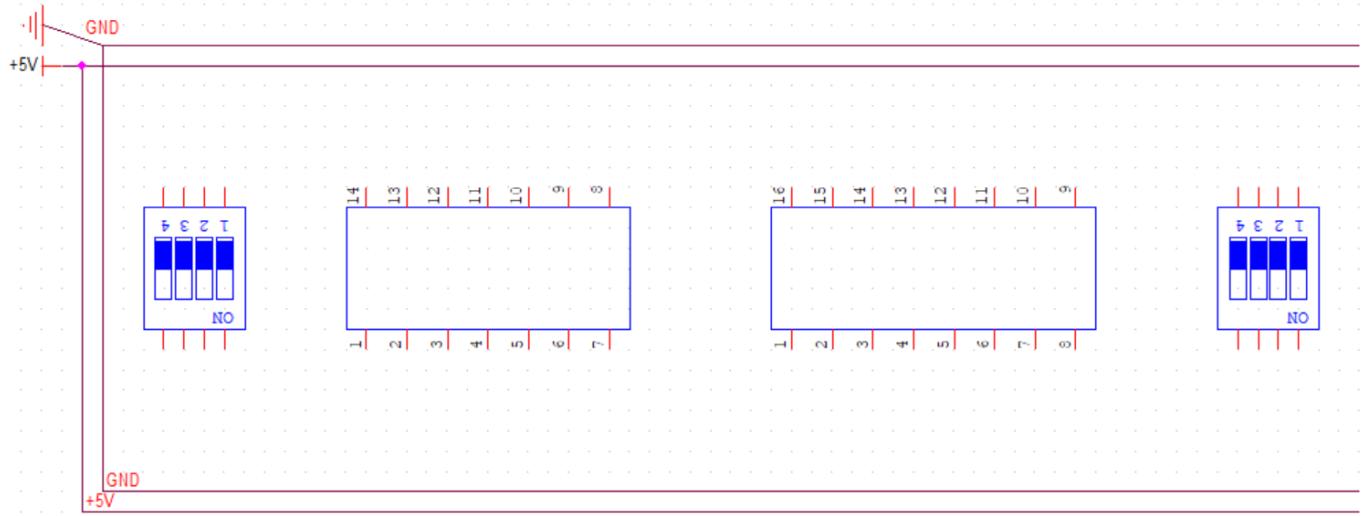


- 
- LEDs, điện trở, DIP switch, bộ dây nối.
  - Breadboard, nguồn 5V DC.

### Sơ đồ thiết kế:

- Sơ đồ mạch:

- Sơ đồ kết nối IC:



### Kết quả thí nghiệm:



Electronics Department

Ho Chi Minh City University of Technology, Vietnam

➤ Thay đổi các tín hiệu ngõ vào và ghi nhận giá trị ngõ ra vào bảng 3.4:

S	A (base 10)	Số A				Số B				B (base 10)	Ngõ ra					(base 10)
		A4	A3	A2	A1	B4	B3	B2	B1		Cout	S4	S3	S2	S1	
0	10	1	0	1	0	1	0	1	1	11						
0	14	1	1	1	0	1	0	1	0	10						
0	12	1	1	0	0	0	1	0	0	4						
0	5	0	1	0	1	0	0	1	1	3						
0	4	0	1	0	0	0	1	0	1	5						
0	13	1	1	0	1	1	1	0	1	13						
0	6	0	1	1	0	1	0	0	1	9						
0	2	0	0	1	0	1	0	0	0	8						
0	1	0	0	0	1	0	0	1	0	2						
0	7	0	1	1	1	0	0	0	1	1						
0	15	1	1	1	1	0	1	1	0	6						
0	9	1	0	0	1	1	1	1	1	15						
1	10	1	0	1	0	1	0	1	1	11						
1	14	1	1	1	0	1	0	1	0	10						
1	12	1	1	0	0	0	1	0	0	4						
1	5	0	1	0	1	0	0	1	1	3						
1	4	0	1	0	0	0	1	0	1	5						
1	13	1	1	0	1	1	1	0	1	13						
1	6	0	1	1	0	1	0	0	1	9						
1	2	0	0	1	0	1	0	0	0	8						
1	1	0	0	0	1	0	0	1	0	2						
1	7	0	1	1	1	0	0	0	1	1						
1	15	1	1	1	1	0	1	1	0	6						
1	9	1	0	0	1	1	1	1	1	15						

Bảng 3.4



### THÍ NGHIỆM 3

**Mục tiêu:** Kiểm chứng hoạt động của D Flipflop – IC 74LS74

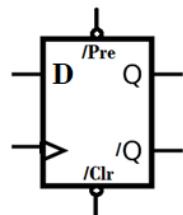
**Yêu cầu:** Khảo sát hoạt động của D Flipflop – IC 74LS74: thay đổi giá trị các ngõ vào D, Preset, Clear, Clock và ghi nhận giá trị ngõ ra của Flipflop.

#### Thiết bị:

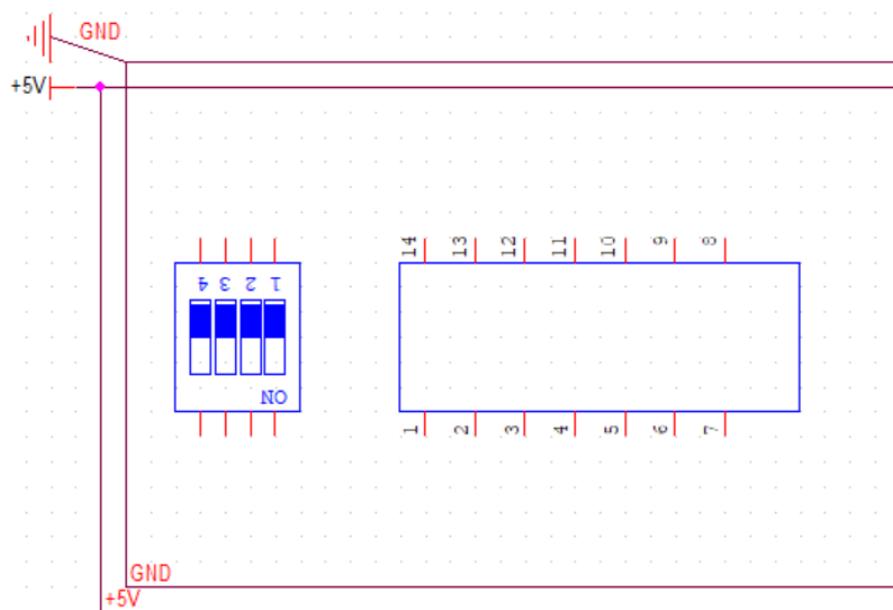
- IC 74LS74.
- LEDs, điện trở, DIP switch, bộ dây nối.
- Breadboard, nguồn 5V DC.

#### Sơ đồ thiết kế:

- Sơ đồ mạch:



- Sơ đồ kết nối IC:



**Kết quả thí nghiệm:**

- Lần lượt thay đổi các giá trị Preset, Clear, D và Clock, ghi nhận giá trị ngõ ra của DFF và điền vào bảng 3.5:

Lưu ý:

- Ngõ ra chỉ thay đổi khi có cạnh lên của xung clock. Cạnh lên được tạo ra khi công tắc chuyển từ mức 0 sang mức 1.
- Ngõ vào D phải được thiết lập trước khi xuất hiện cạnh lên của xung clock.

ASYNCHRONOUS INPUTS		SYNCHRONOUS INPUTS		OUTPUTS		COMMENT
PRESET	CLEAR	D	CLK	Q	Q'	
0	0	X	X			
1	0	X	X			
0	1	X	X			
1	1	0	0			
1	1	1	0			
1	1	0	1			
1	1	1	1			
1	1	0	↓			
1	1	1	↓			
1	1	0	↑			
1	1	1	↑			

Bảng 3.5

## THÍ NGHIỆM 4

**Mục tiêu:** Sử dụng D Flipflop – IC 74LS74 để thiết kế mạch đếm nối tiếp.

**Yêu cầu:** Thiết kế mạch đếm lên từ 0 → 7 sử dụng D-FF, kết quả thể hiện lên LED đơn.

**Thiết bị:**

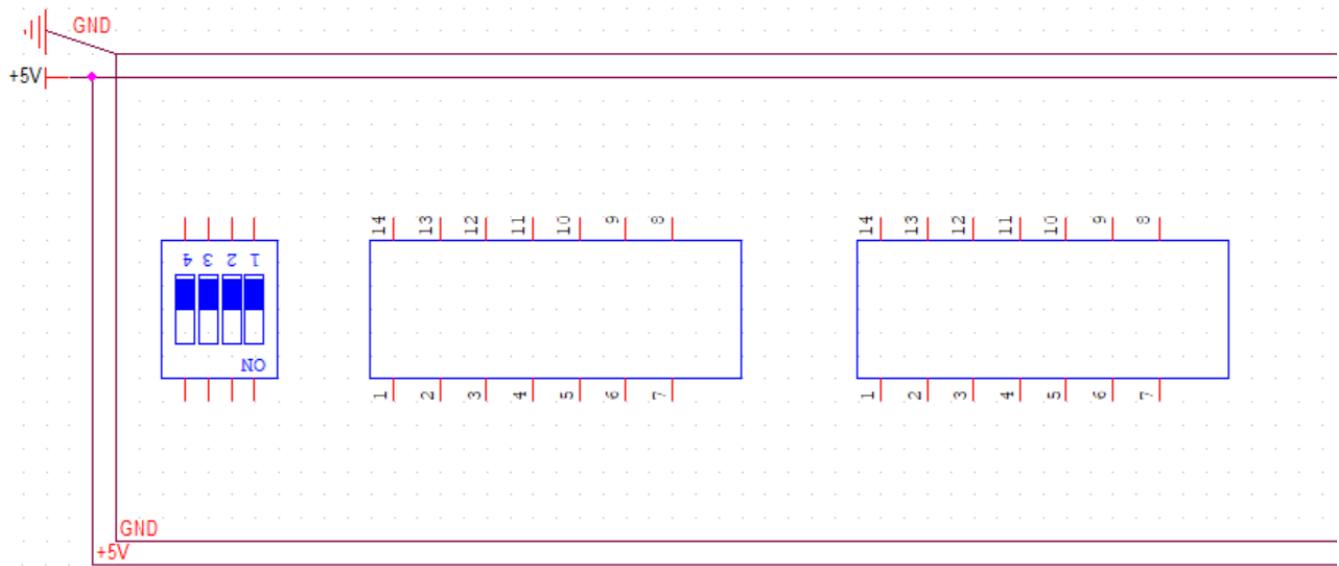
- IC 74LS74.
- LEDs, điện trở, DIP switch, bộ dây nối.
- Breadboard, nguồn 5V DC, máy phát sóng.



### Sơ đồ thiết kế:

- Sơ đồ mạch:

- Sơ đồ kết nối IC:



### Kết quả thí nghiệm:

- Cáp tín hiệu Preset = 1, Clear = 0, ghi nhận kết quả ngõ ra:

- Cáp tín hiệu Preset = 0, Clear = 1, ghi nhận kết quả ngõ ra:

- Cấp tín hiệu Preset = Clear = 1. Sử dụng máy phát sóng, tạo tín hiệu xung vuông tuần hoàn có tần số  $f = 1$  KHz, biên độ điện áp  $V_{pp} = 5V$ ,  $V_{offset} = 2.5V$ ; dùng xung này làm xung clock cho mạch đếm. Quan sát ngõ ra của mạch đếm và nhận xét.

## THÍ NGHIỆM 5

Mục tiêu: Thiết kế hệ tuần tự tổng quát.

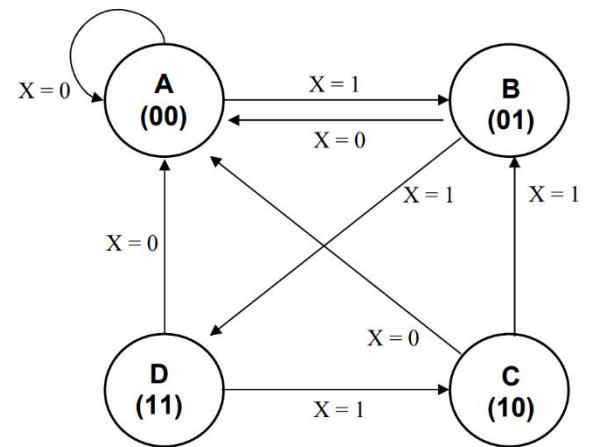
Yêu cầu: Thiết kế hệ tuần tự có giản đồ trạng thái như hình 3.5:

Thiết bị:

- IC 74LS74, 74LS08, 74LS32.
- LEDs, điện trở, DIP switch, bộ dây nối.
- Breadboard, nguồn 5V DC, máy phát sóng.

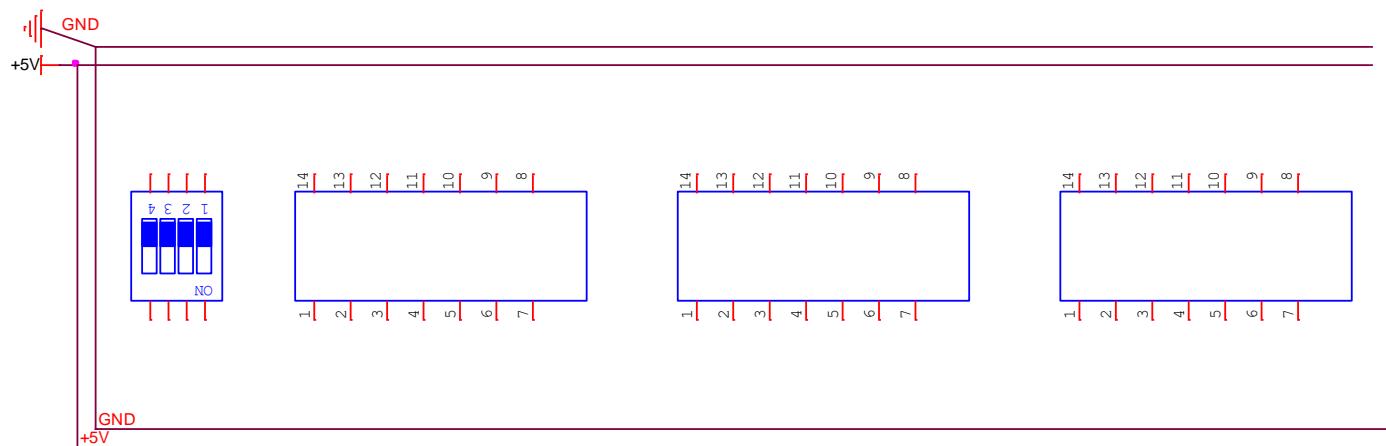
Sơ đồ thiết kế:

- Sơ đồ mạch:



Hình 3.5

- Sơ đồ kết nối IC:



**Kết quả thí nghiệm:**

- Reset máy trạng thái để hệ bắt đầu hoạt động từ trạng thái A. Sử dụng máy phát sóng, tạo tín hiệu xung vuông tuần hoàn có tần số  $f = 1$  KHz, biên độ điện áp  $V_{pp} = 5V$ ,  $V_{offset} = 2.5V$ ; dùng xung này làm xung clock cho mạch.
- Thay đổi giá trị ngõ vào và ghi nhận vào bảng sau.

Input		0	1	1	0	0	0	1
State	A							

So sánh kết quả ghi nhận được với giản đồ xung ở trên.

# LAB 4: THỰC HIỆN HỆ TỔ HỢP VÀ HỆ TUẦN TỰ CƠ BẢN TRÊN FPGA

Họ và tên:	Lớp TN:
MSSV:	Ngày:

## A. PRELAB

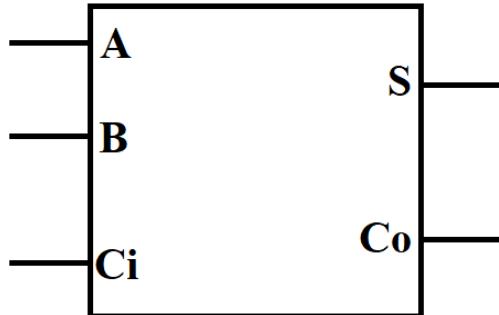
### BÀI CHUẨN BỊ 1

#### Mục tiêu:

Năm được các thức mô tả mạch cộng toàn phần (FA) sử dụng ngôn ngữ SystemVerilog.

Yêu cầu: Sinh viên đọc đoạn chương trình sau đây dùng để mô tả bộ FA.

```
module FA_ex1(
    input A,
    input B,
    input Ci,
    output S,
    output Co);
    assign S = A ^ B ^ Ci;
    assign Co=(A&B)|(A&Ci)|(B&Ci);
endmodule
```



**Hình 4.1:** Bộ cộng toàn phần Full Adder

- Sinh viên tiến hành biên dịch đoạn chương trình trên. Sau đó, sinh viên tiến mô phỏng dạng sóng ngõ ra trong tất cả các trường hợp của ngõ vào. Chụp hình dạng sóng ngõ ra.

## BÀI CHUẨN BỊ 2

**Mục tiêu:** Năm được các thức mô tả mạch cộng trừ 2 số 4 bit sử dụng ngôn ngữ systemverilog.

**Yêu cầu:** Sinh viên thực hiện mô tả mạch cộng trừ 2 số 4 bit (2 số ngõ vào lần lượt là A và B, ngõ ra là S, cờ nhớ Ci và Co) bằng thông qua ngõ vào điều khiển sel:

- sel=0:  $S = A + B$

- sel=1:  $S = A - B$

Sinh viên thực hiện mô tả bằng 2 cách:

- Cách 1: sử dụng các Full Adder ghép nối lại với nhau (sử dụng bài chuẩn bị 1).

- Cách 2: mô tả theo phương pháp luồng dữ liệu (data flow model).

➤ Sinh viên thực hiện mô tả theo yêu cầu của bài toán sử dụng ngôn ngữ systemverilog.

*Cách 1: Gợi ý:* sử dụng dạng cấu trúc, ghép nối các bộ FA.

```
wire wire0, wire1, wire2  
FA_ex1 u0(.Ci(0),.A(A[0]),.B(B[0]),.S(S[0]),.Co(wire0));  
FA_ex1 u1(.Ci(wire1),.A([1]),.B([1]),.S(S[1]),.Co(wire1));  
...
```

*Cách 2: Gợi ý:* sử dụng câu điều kiện if ... else ...

always@(*) if (sel) ... else ... end	assign S=(sel)?A+B:A-B;
---	-------------------------

➤ Sinh viên tiến mô phỏng dạng sóng ngõ ra trong một vài trường hợp của ngõ vào. Chụp hình dạng sóng ngõ ra.

### BÀI CHUẨN BỊ 3

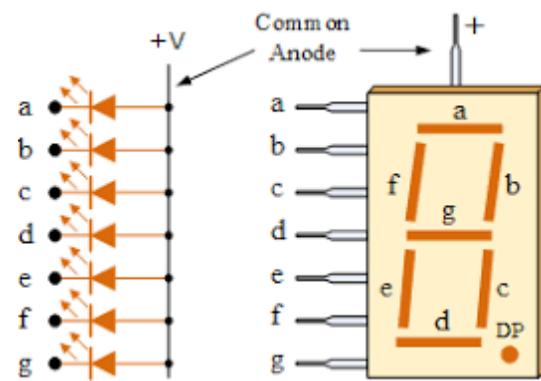
**Mục tiêu:** Nắm được các thức mô tả mạch giải mã số BCD sang mã LED 7 đoạn loại anode chung sử dụng ngôn ngữ systemverilog.

**Yêu cầu:** Sinh viên thực hiện mô tả mạch giải mã số BCD sang mã LED 7 đoạn loại anode chung.

- Sinh viên hoàn thành bảng chân trị của mạch giải mã.

I <sub>3</sub>	I <sub>2</sub>	I <sub>1</sub>	I <sub>0</sub>	g	f	e	d	c	b	a
0	0	0	0	1	0	0	0	0	0	0
0	0	0	1							
0	0	1	0							
0	0	1	1							
0	1	0	0							
0	1	0	1							
0	1	1	0							
0	1	1	1							
1	0	0	0							
1	0	0	1							
1	0	1	0							
1	0	1	1							
1	1	0	0							
1	1	0	1							
1	1	1	0							
1	1	1	1							

Bảng 4.1: Bảng chân trị mạch giải mã LED 7 đoạn



Hình 4.2 LED 7 đoạn loại anode chung

(source: internet)

- Sinh viên thực hiện mô tả theo yêu cầu của bài toán sử dụng ngôn ngữ systemverilog.

Gợi ý: sử dụng câu lệnh case .... .

```

input [3:0]I;
always@(*) begin
    case(I)
        4'b0000: out <= 7'b1000000;
        4'b0001: out <= ...
    endcase
end

```

```
...
endcase;
end
```

- Sinh viên tiến mô phỏng dạng sóng ngõ ra trong một vài trường hợp của ngõ vào. Chụp hình dạng sóng ngõ ra.

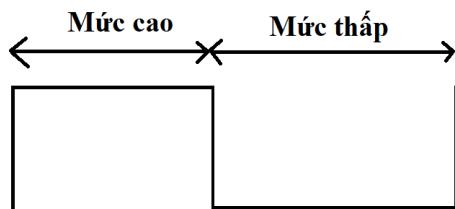
## BÀI CHUẨN BỊ 4

**Mục tiêu:** Nắm được các thức mô tả mạch chuyển đổi 1 xung có tần số 50MHz sang 1 xung có tần số 1Hz sử dụng ngôn ngữ systemverilog.

**Yêu cầu:** Sinh viên thực hiện mô tả mạch chuyển đổi 1 xung có tần số 50MHz sang 1 xung có tần số 1Hz.

Gợi ý:

- Xung có tần số 50MHz sẽ thực hiện 50 000 000 dao động trong 1s. (1 dao động bao gồm 1 lần mức cao và 1 lần mức thấp).
- Xung có tần số 1Hz sẽ thực hiện 1 dao động trong 1s.
- Vận dụng nguyên lý của mạch đếm, mạch nhận ngõ vào là xung 50MHz, khi xung ngõ vào đếm 25 000 000 sẽ tiến hành đảo trạng thái của xung ngõ ra.



Đoạn chương trình minh họa:

```
integer i=0;  
reg temp=0;  
always_ff@(posedge clk)  
begin  
    i = i + 1;  
    if (i == 25 000 000) begin  
        out = ~ out;  
        i = 0;  
    end  
end
```

- Sinh viên sử dụng gợi ý trên và thực hiện mô tả theo yêu cầu của bài toán sử dụng ngôn ngữ systemverilog.

## BÀI CHUẨN BỊ 5

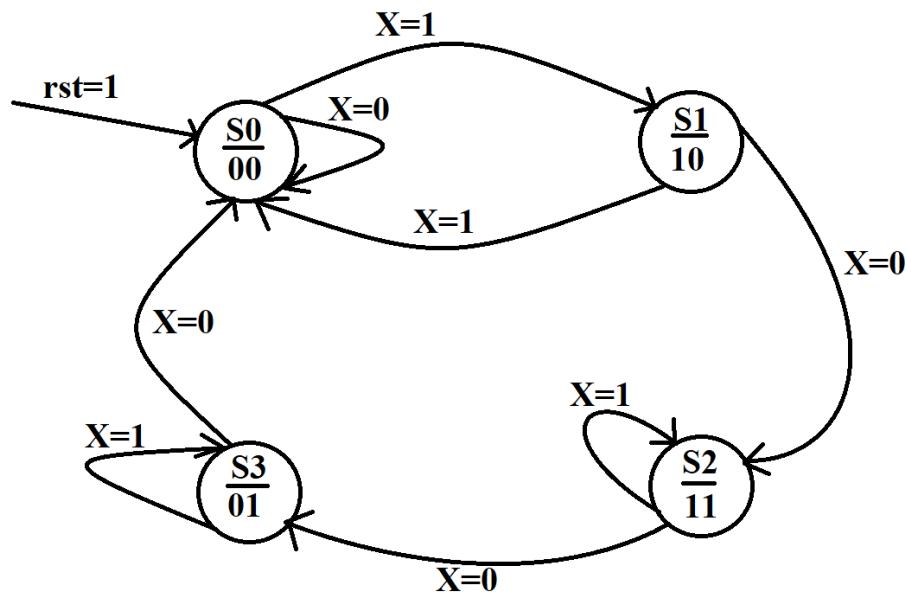
**Mục tiêu:** Năm được các thức mô tả máy trạng thái sử dụng ngôn ngữ systemverilog.

### Yêu cầu:

- Sinh viên vẽ máy trạng thái cho yêu cầu ở thí nghiệm 5 của bài Lab 4. Sinh viên giải thích ý nghĩa của các trạng thái và tiến hành mã hóa trạng thái thành chuỗi bit.

- Sinh viên thực hiện mô tả máy trạng thái ở phần trên sử dụng ngôn ngữ systemverilog. (Tham khảo đoạn chương trình ở phần dưới về cách viết máy trạng thái).

Giả sử có máy trạng thái: Ngõ vào X, ngõ ra Y (2 bit)



Các trạng thái lần lượt là: S0 (00), S1 (01), S2 (10), S3(11)

## Đoạn chương trình cho máy trạng thái trên:

```
input X;          //Khai báo ngõ vào 1 bit
input clk;        //Khai báo clock
output reg [1:0]Y; //Khai báo ngõ ra 2 bit
parameter S0=2'b00, S1=2'b01, S2=2'b10, S3=2'b11; //Định nghĩa các trạng thái
reg [1:0]pre_state, next_state; //Các thanh ghi chưa trạng thái
//Khối chuyển trạng thái
always@(posedge clk) begin
    if (rst) begin
        pre_state <= S0;
    end
    else
        pre_state <= next_state;
end
//Khối chuyển trạng thái
always@(pre_state or X) begin
    case(pre_state)
        S0: if (X) next_state <= S1;
            else next_state <= S0;
        S1: if (!X) next_state <= S2;
            else next_state <= S0;
        S2: if (!X) next_state <= S3;
            else next_state <= S2;
        S3: if (!X) next_state <= S0;
            else next_state <= S3;
    endcase;
end
```

**//Khối tạo ngõ ra**

```
always@(*) begin  
    case (pre_state)  
        S0: Y <= 2'b00;  
        S1: Y <= 2'b10;  
        S2: Y <= 2'b11;  
        S3: Y <= 2'b01;  
    endcase;  
end
```

## B. HƯỚNG DẪN THÍ NGHIỆM

### I. MỤC TIÊU

- Nắm được cách sử dụng kit thí nghiệm, phần mềm lập trình.
- Nắm được cách khảo sát và thiết kế hệ mạch đếm sử dụng các IC chức năng cơ bản.
- Nắm được quy trình mô tả phần cứng trên FPGA.

### II. CHUẨN BI:

- Đã chuẩn bị tốt cho bài thí nghiệm, sinh viên PHẢI đọc trước phần Phụ lục 1 và hoàn thành các bước của Sample lab trong Phụ lục 2.
- Sinh viên phải hoàn thành và nộp PRELAB 4 trước khi vào lớp.

### III. HƯỚNG DẪN THÍ NGHIỆM

#### THÍ NGHIỆM 1

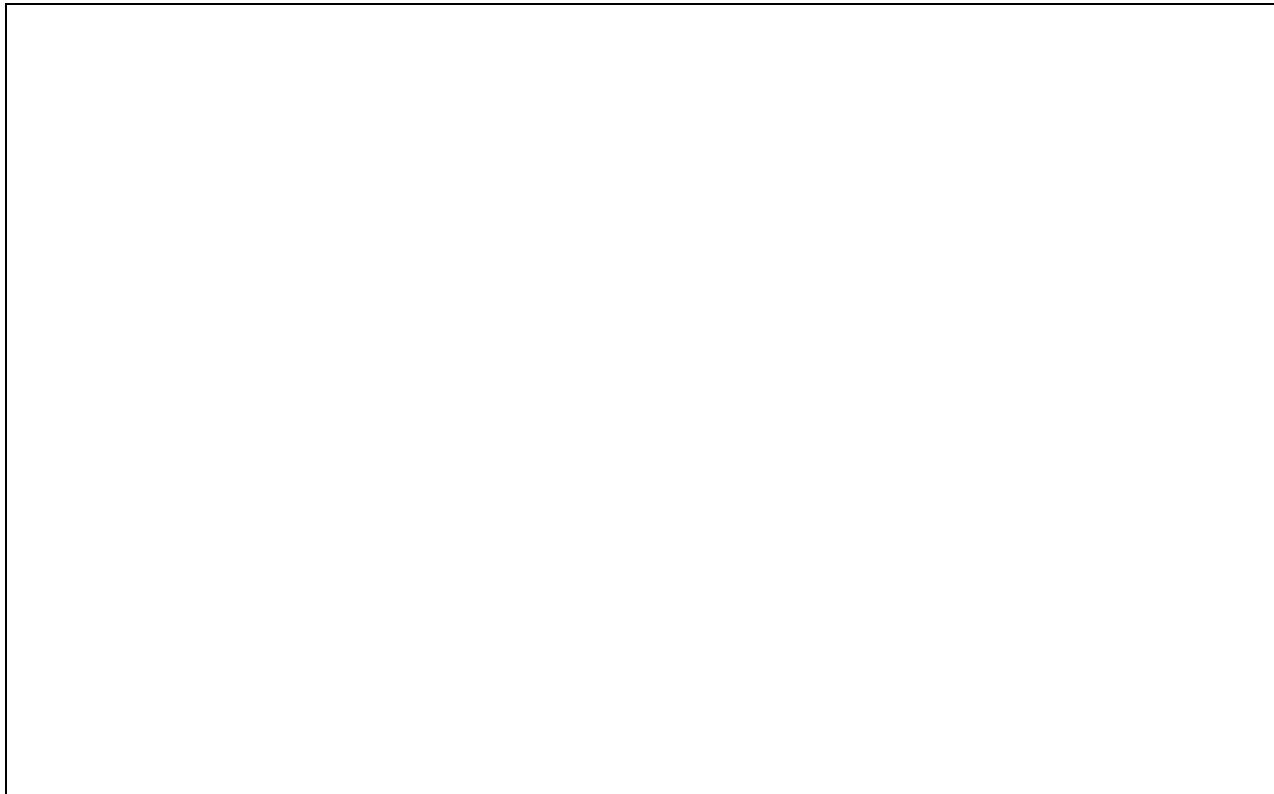
**Mục tiêu:** Nắm được các thức mô tả mạch tính giá trị tuyệt đối của một số 4 bit sử dụng ngôn ngữ systemverilog và thực hiện kiểm tra hoạt động trên kit FPGA.

**Yêu cầu:** Sinh viên thực hiện mô tả mạch tính giá trị tuyệt đối của một số 4 bit (số ngõ vào lần lượt là A, ngõ ra là S).

#### Kiểm tra:

- Sinh viên trình bày ý tưởng của thiết kế. (Sinh viên có thể vẽ sơ đồ khối và/hoặc diễn giải để giáo viên hiểu được ý tưởng của mình)

- Chương trình mô tả hoạt động của thiết kế.



- Kết quả mô phỏng dạng sóng.

- Kết quả RTL viewer.

- Sinh viên thực hiện gán chân theo yêu cầu và sau đó đỗ lên kit FPGA DE2. Sau đó ghi nhận kết quả.

Gán chân theo mẫu sau:

Chân A[3:0] được nối với SW0-SW3

Chân S[3:0] được nối với LEDR[3:0]

## THÍ NGHIỆM 2

**Mục tiêu:** Năm được các thức mô tả khối ALU (bộ tính toán) có chức năng đơn giản sử dụng systemverilog và thực hiện kiểm tra hoạt động trên kit FPGA.

**Yêu cầu:** Sinh viên thực hiện mô tả mạch cho mạch thực hiện bộ ALU tính toán 2 số 4 bit (hai số ngõ vào lần lượt là A và B, ngõ ra là S, cờ nhớ Ci và Co) thông qua ngõ vào điều khiển 2 bit Sel.

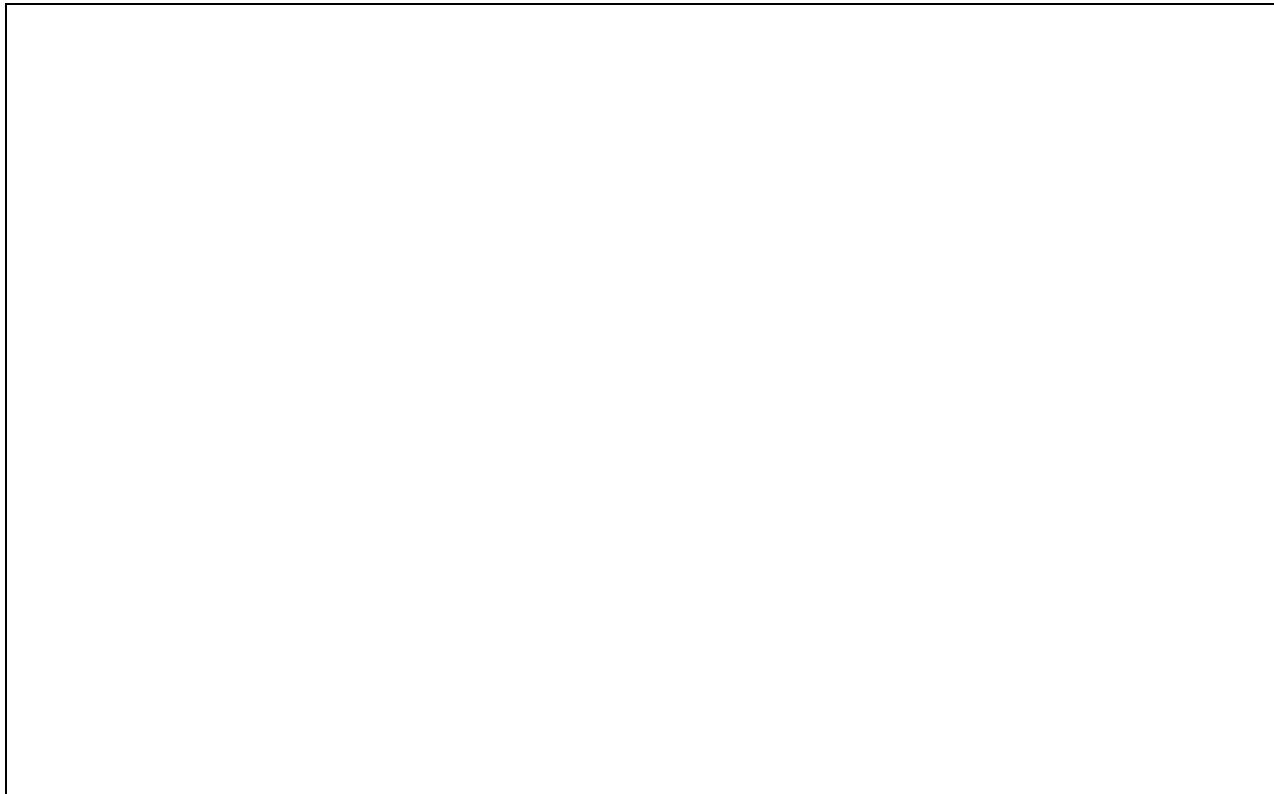
- Nếu Sel=00:  $S=A+B$
- Nếu Sel=01:  $S=A-B$
- Nếu Sel=10:  $S=A \text{ AND } B$
- Nếu Sel=11:  $S=A \text{ OR } B$

**Kiểm tra:**

➤ Sinh viên trình bày ý tưởng của thiết kế. (Sinh viên có thể vẽ sơ đồ khối và/hoặc diễn giải để giáo viên hiểu được ý tưởng của mình)

➤ Chương trình mô tả hoạt động của thiết kế.





➤ Kết quả mô phỏng dạng sóng.

➤ Kết quả RTL viewer.

- Sinh viên thực hiện gán chân theo yêu cầu và sau đó đố lên kit FPGA DE2. Sau đó ghi nhận kết quả.

Gán chân theo mẫu sau:

Chân A[3:0] được nối với SW0-SW3

Chân B[3:0] được nối với SW4-SW7

Chân Ci được nối với SW8

Chân S[3:0] được nối với LEDR[3:0]

Chân Co được nối với LEDR[4]

### THÍ NGHIỆM 3

**Mục tiêu:** Nắm được các thức mô tả mạch đếm đầy đủ sử dụng ngôn ngữ systemverilog và thực hiện kiểm tra hoạt động trên kit FPGA.

**Yêu cầu:** Sinh viên thực hiện thiết kế mô tả mạch đếm xuống 3 bit đầy đủ sử dụng ngôn ngữ system verilog. Giá trị đếm thay đổi sau mỗi 1s. Ngõ ra được kết nối với LED 7 đoạn loại anode chung. Ngoài ra, bộ đếm còn có chân RST (tích cực cao) dùng để reset trạng thái bộ đếm về 0.

*Gợi ý:*

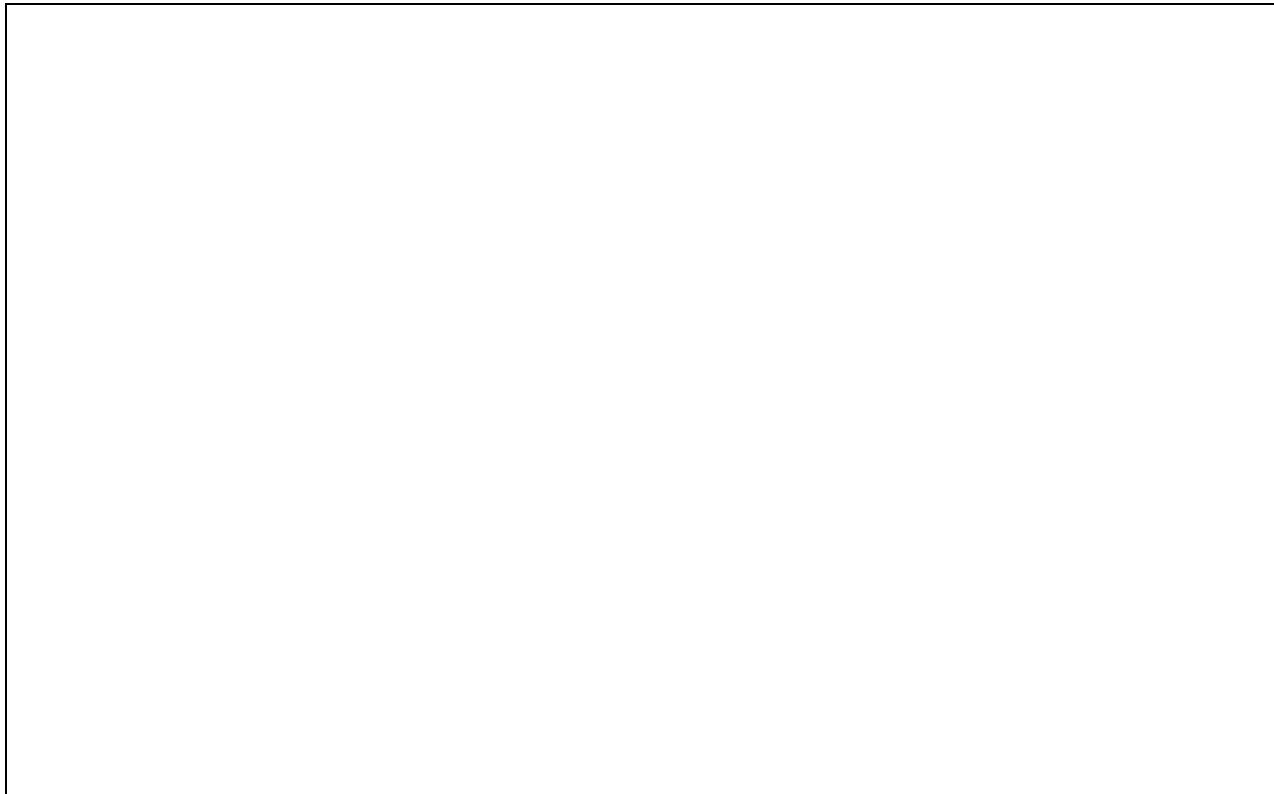
- Tín hiệu clock được tạo từ bộ chia tần số từ 50MHz sang 1s.
- Sinh viên sử dụng chương trình chuyển từ mã BCD sang LED 7 đoạn trong PRELAB, kết nối ngõ ra của bộ đếm với ngõ vào của bộ chuyển đổi.

**Kiểm tra:**

- Sinh viên trình bày ý tưởng của thiết kế. (Sinh viên có thể vẽ sơ đồ khái và/hoặc diễn giải để giáo viên hiểu được ý tưởng của mình)

- Chương trình mô tả hoạt động của thiết kế.





➤ Kết quả mô phỏng dạng sóng.

➤ Kết quả RTL viewer.

- Sinh viên thực hiện gán chân theo yêu cầu và sau đó đỗ lên kit FPGA DE2. Sau đó ghi nhận kết quả.

Gán chân theo mẫu sau:

LED 7 đoạn là HEX0.

Chân RST là SW0.

## THÍ NGHIỆM 4

**Mục tiêu:** Nắm được các thức mô tả mạch đếm đầy đủ sử dụng ngôn ngữ systemverilog và thực hiện kiểm tra hoạt động trên kit FPGA.

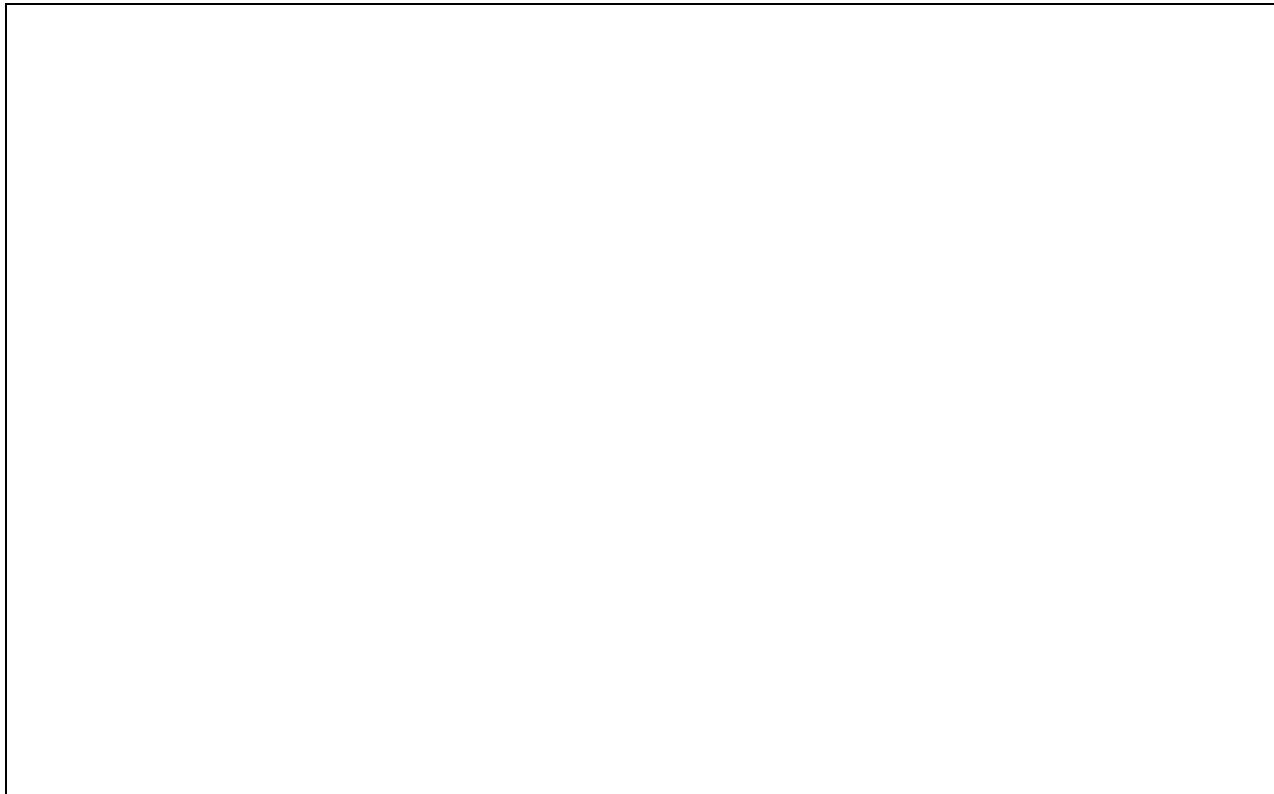
**Yêu cầu:** Sinh viên thực hiện thiết kế mô tả mạch **đếm lên 4 bit từ 5 đến 14** sử dụng ngôn ngữ system verilog. Giá trị đếm thay đổi sau mỗi 1s. Ngõ ra 4 bit được kết nối với 2 LED 7 đoạn loại anode chung. Ngoài ra, bộ đếm còn có chân RST (tích cực cao) dùng để reset trạng thái bộ đếm về 0.

*Gợi ý:*

- Tín hiệu clock được tạo từ bộ chia tần số từ 50MHz sang 1s.
- Sinh viên sử dụng chương trình chuyển từ mã BCD sang LED 7 đoạn trong PRELAB, kết nối ngõ ra của bộ đếm với ngõ vào của bộ chuyển đổi.
- Sinh viên cần viết thêm bộ chuyển đổi từ số 4 bit sang số BCD.

**Kiểm tra:**

- Sinh viên trình bày ý tưởng của thiết kế. (Sinh viên có thể vẽ sơ đồ khối và/hoặc diễn giải để giáo viên hiểu được ý tưởng của mình)
- Chương trình mô tả hoạt động của thiết kế.



➤ Kết quả mô phỏng dạng sóng.

➤ Kết quả RTL viewer.

- Sinh viên thực hiện gán chân theo yêu cầu và sau đó đỗ lên kit FPGA DE2. Sau đó ghi nhận kết quả.

Gán chân theo mẫu sau:

Hai LED 7 đoạn là HEX1 (trọng số cao) và HEX0 (trọng số thấp).  
Chân RST là SW0.

## THÍ NGHIỆM 5

**Mục tiêu:** Nắm được các thức mô tả máy trạng thái sử dụng ngôn ngữ systemverilog và thực hiện kiểm tra hoạt động trên kit FPGA.

**Yêu cầu:** Sinh viên tiến hành mô tả hệ tuần tự sau bằng VHDL sau đó nạp chương trình xuống kit DE2 để kiểm tra hoạt động:

Hệ tuần tự có 1 ngõ vào (X) và 1 ngõ ra (Z). Ngõ ra  $Z = 1$  nếu tổng số bit 1 nhận được chia hết cho 3 (quy ước 0, 3, 6, 9, ... là các số chia hết cho 3) và tổng số bit 0 nhận được là 1 số chẵn (lớn hơn 0).

**Ghi chú:** Sinh viên có thể lựa chọn thiết kế theo máy trạng thái kiểu Mealy hoặc Moore.

Gán chân theo mẫu sau:

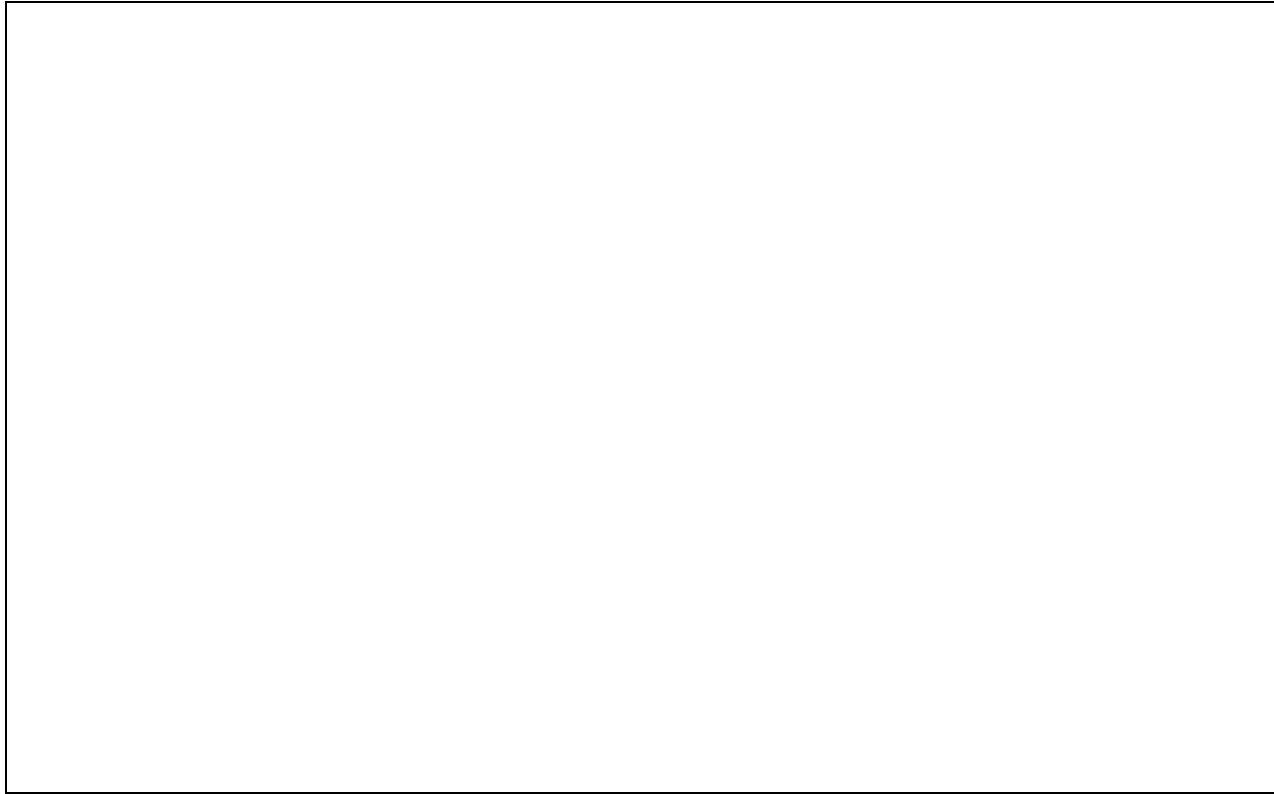
Ngõ vào X được nối với SW0.

Tín hiệu CLK được nối với xung clock 1Hz (Trong bài PRELAB).

Ngõ ra Z được nối với LED0.

Kiểm tra:

- Sinh viên trình bày ý tưởng của thiết kế. (Sinh viên có thể vẽ sơ đồ khái (máy trạng thái) và/hoặc diễn giải để giáo viên hiểu được ý tưởng của mình)
  
- Chương trình mô tả hoạt động của thiết kế.



➤ Kết quả mô phỏng dạng sóng.

➤ Kết quả RTL viewer.

- Sinh viên thực hiện gán chân theo yêu cầu và sau đó đố lên kit FPGA DE2. Sau đó ghi nhận kết quả.

*Lưu ý:* Sinh viên nên nối clock 1Hz ra 1 LED để dễ quan sát tín hiệu clock.

## THÍ NGHIỆM 6

**Mục tiêu:** Nắm được các thức mô tả máy trạng thái sử dụng ngôn ngữ systemverilog và thực hiện kiểm tra hoạt động trên kit FPGA.

**Yêu cầu:** Sinh viên tiến hành mô tả hệ tuần tự sau bằng VHDL sau đó nạp chương trình xuống kit DE2 để kiểm tra hoạt động:

Một hệ thống cung cấp thức ăn và nước uống tự động cho thú cưng gồm 2 ngõ vào là 2 nút nhấn RED, BLUE; và 2 ngõ ra là tín hiệu FOOD, WATER để kích hoạt máy cung cấp thức ăn và nước uống

- Nút RED (tín hiệu R; khi nhấn nút R=1, ngược lại R=0): khi con vật đói muốn ăn thì cần nhấn nút RED 3 lần liên tiếp. Khi đó tín hiệu F (FOOD) = 1 để kích hoạt máy cung cấp thức ăn.
- Nút BLUE (tín hiệu B; khi nhấn nút B=1, ngược lại B=0): khi con vật khát muốn uống thì cần nhấn nút BLUE 2 lần liên tiếp. Khi đó tín hiệu W (WATER) = 1 để kích hoạt máy cung cấp nước uống.

### Chú ý:

- Khi tín hiệu kích hoạt F hay W bằng 1, nếu nhấn 1 nút bất kỳ hệ thống sẽ trở lại trạng thái reset.
- Ở mỗi thời điểm, chỉ có 1 nút nhấn.
- Các nút nhấn cần phải được tác động liên tiếp, nếu có 1 nút sai trình tự, máy trạng thái sẽ quay trở về trạng thái ban đầu. (ví dụ: khi các nút nhấn được tác động theo trình tự (RED, RED, BLUE), máy trạng thái quay về trạng thái đầu reset).

**Ghi chú:** Sinh viên có thể lựa chọn thiết kế theo máy trạng thái kiểu Mealy hoặc Moore.

### Gán chân theo mẫu sau:

Ngõ vào RED và BLUE lần lượt được nối với SW0 và SW1.

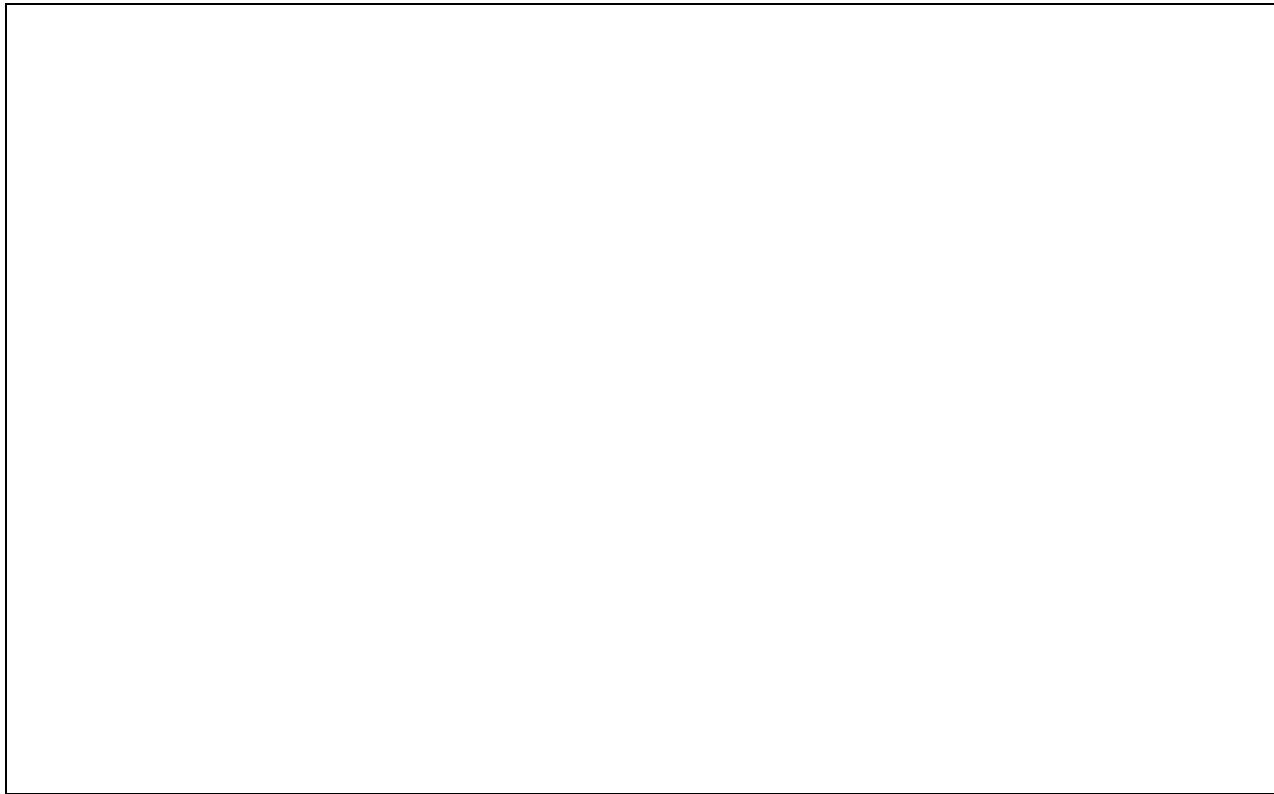
Tín hiệu CLK được nối với xung clok 1Hz (Trong bài PRELAB).

Ngõ ra FOOD và WATER lần lượt được nối với LED0 và LED1.

### Kiểm tra:

- Sinh viên trình bày ý tưởng của thiết kế. (Sinh viên có thể vẽ sơ đồ khối (máy trạng thái) và/hoặc diễn dãi để giáo viên hiểu được ý tưởng của mình)

- Chương trình mô tả hoạt động của thiết kế.



- Kết quả mô phỏng dạng sóng.

- Kết quả RTL viewer.

- Sinh viên thực hiện gán chân theo yêu cầu và sau đó đố lên kit FPGA DE2. Sau đó ghi nhận kết quả.

Lưu ý: Sinh viên nên nối clock 1Hz ra 1 LED để dễ quan sát tín hiệu clock.



# LAB 5: BÀI TỔNG HỢP THỰC HIỆN MẠCH SỐ TRÊN BREADBOARD – GIAO TIẾP VỚI FPGA

Họ và tên:	Lớp TN:
MSSV:	Ngày:

## A. PRELAB

### Phần I: Tạo Clock 1 Hz cho Kit DE 2

Kit De 2 cung cấp chân CLOCK\_50 được nối tới một bộ tạo clock 50 MHz. Nếu nối module tới chân này, ta sẽ nhận được CLOCK 50 MHz (Giao động 50 triệu lần 1 giây). Trong các bài toán thực tế, ta thường thực hiện các thay đổi ở mức 1 giây (Clock 1 Hz).

Để tạo Clock 1 Hz trên Kit DE 2, hoạt động thực tế, ta có thể dùng một bộ nạp thanh ghi giá trị 50M, sau đó trừ đi 1 cho mỗi clock. Sau khi hết 50 M Clock, một bộ so sánh giá trị thanh ghi với 0 sẽ cho ra kết quả 1 và cứ thế lặp lại.

Kết quả của bộ so sánh đó chính là một tín hiệu flag cho kết quả tương tự Clock 1 Hz. Nếu ta đặt các module có enable hoạt động với tín hiệu flag này, chúng sẽ hoạt động (nhảy số,...) mỗi 1 giây.

**Yêu cầu:** Sinh viên viết mã phần cứng SystemVerilog cho module Clock 1 Hz thực hiện nháy nháy đèn LEDR0 trên Kit De2 mỗi 1 giây.

Đoạn mã System Verilog tham khảo cho module này:

```
//////////  
// Ho Chi Minh University of Technology  
// Faculty of Electrical & Electronics Engineering  
// Department of Electronics  
// EE1016 - EE1010 : Ky Thuat So  
//  
// Lab 5 : Thiet ke he thong den giao thong  
// Thuc hien giao tiep DE2 - GPIO - Breadboard - ICs/LEDs/LCDs  
// Module change 50 MHz to 1 sec  
//  
//////////  
  
module secgen (
```

```
input clk,  
      rst,  
  
      output sec  
);  
  
/////////////////////////////  
parameter CLKCONST = 50000000; //clock 50 MHz  
/////////////////////////////  
  
reg [25:0] temp; // 26-bit 67M  
  
wire tmpzr;  
  
assign tmpzr = temp == 0;  
  
always_ff @(posedge clk) begin  
    if (rst)  
        temp <= CLKCONST; //50M  
    else if (tmpzr)  
        temp <= CLKCONST;  
    else  
        temp <= temp - 1;  
end  
  
assign sec = tmpzr;  
  
endmodule
```





## Phần II: Viết máy trạng thái đèn giao thông điều khiển

Hệ thống đèn giao thông điều khiển được yêu cầu của Thí nghiệm 1 Lab 5 yêu cầu hoạt động theo một máy trạng thái.

**Yêu cầu:** Sinh viên hãy tham khảo đề bài và vẽ máy trạng thái tương ứng cho bộ điều khiển chính của hệ thống này:

## Phần III: Viết máy trạng thái đèn giao thông tự động

Hệ thống đèn giao thông tự động theo yêu cầu của Thí nghiệm 2 Lab 5 yêu cầu hoạt động theo một máy trạng thái.

**Yêu cầu:** Sinh viên hãy tham khảo đề bài và vẽ máy trạng thái tương ứng cho bộ điều khiển chính của hệ thống này:





#### Phần IV: Thành phần GPIO của Kit DE2

Sinh viên tham khảo datasheet kit DE2 Terrasic và tìm hiểu thứ tự của GPIO\_0 và GPIO\_1, cách thức lắp đèn LED tới GPIO\_0 của kit DE2.

**Yêu cầu:** Sinh viên vẽ cách thức lắp mạch nối với GPIO\_0 của Kit DE2 tới LED (lưu ý tính thẩm mỹ)

#### Phần V: LED 7 đoạn 5161BS

Sinh viên tham khảo datasheet IC led 7 đoạn 5161BS (Cho LED Common Anode)

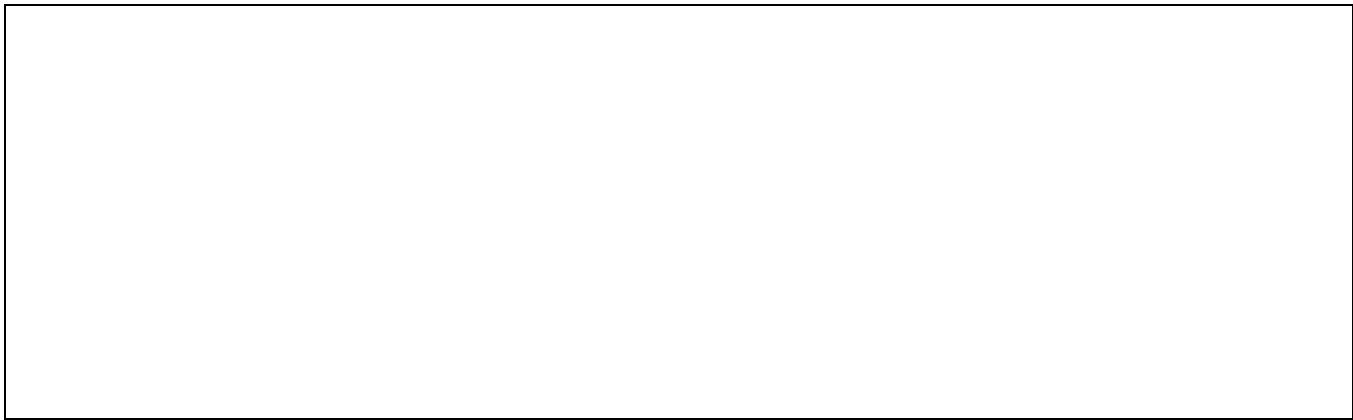
**Yêu cầu:** Sinh viên vẽ cách thức lắp mạch nối với GPIO\_0 của Kit DE2 tới LED 7 đoạn (lưu ý tính thẩm mỹ)



## Phần VI: IC giải mã led 7 đoạn 74LS47

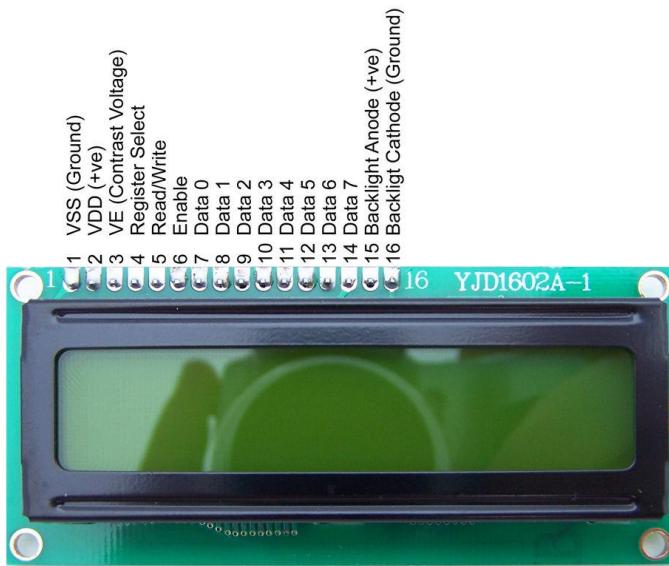
Sinh viên tham khảo datasheet IC giải mã led 7 đoạn 74LS47 (Cho LED Common Anode)

**Yêu cầu:** Sinh viên vẽ cách thức lắp mạch nối với GPIO\_0 của Kit DE2 tới IC 74LS47 tới LED 7 đoạn (lưu ý tính thẩm mỹ)



## Phần VII: Giao tiếp LCD 16x2

Sinh viên cần tham khảo datasheet LCD 16x2, đa phần sử dụng chip điều khiển HD44780 với sơ đồ chân cắm như sau:



Hình 5.1 LCD 16x2

LCD yêu cầu sinh viên gửi các mã Hex code dưới dạng các command, trong đó bao gồm mã HEX code khởi động và mã HEX Code để thực hiện ghi / xóa nội dung

Table 4 Correspondence between Character Codes and Character Patterns (ROM Code: A00)

	Upper 4 Bits Lower 4 Bits	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
xxxx0000	CG RAM (1)	Ø	ø	P	¶	P	¶	-	¶	Ξ	ø	p	¶	Ø	ø	p	¶
xxxx0001	(2)	!	!	A	Q	a	q	.	Ø	¶	¶	¶	¶	ä	q	¶	¶
xxxx0010	(3)	"	Z	B	R	b	r	¶	¶	¶	¶	¶	¶	¶	¶	¶	¶
xxxx0011	(4)	#	3	C	S	c	s	¶	¶	¶	¶	¶	¶	¶	¶	¶	¶
xxxx0100	(5)	\$	4	D	T	d	t	¶	¶	¶	¶	¶	¶	¶	¶	¶	¶
xxxx0101	(6)	%	5	E	U	e	u	¶	¶	¶	¶	¶	¶	¶	Ü	¶	¶
xxxx0110	(7)	&	6	F	V	f	v	¶	¶	¶	¶	¶	¶	¶	¶	¶	¶
xxxx0111	(8)	'	7	G	W	g	w	¶	¶	¶	¶	¶	¶	¶	¶	¶	¶
xxxx1000	(1)	(	8	H	X	h	x	¶	¶	¶	¶	¶	¶	¶	¶	¶	¶
xxxx1001	(2)	)	9	I	Y	i	y	¶	¶	¶	¶	¶	¶	¶	¶	¶	¶
xxxx1010	(3)	*	:	J	Z	j	z	¶	¶	¶	¶	¶	¶	¶	¶	¶	¶
xxxx1011	(4)	+	;	K	L	k	l	¶	¶	¶	¶	¶	¶	¶	¶	¶	¶
xxxx1100	(5)	,	<	L	¥	1	l	¶	¶	¶	¶	¶	¶	¶	¶	¶	¶
xxxx1101	(6)	-	=	M	J	m	j	¶	¶	¶	¶	¶	¶	¶	¶	¶	¶
xxxx1110	(7)	.	>	N	^	n	†	¶	¶	¶	¶	¶	¶	¶	¶	¶	¶
xxxx1111	(8)	/	?	O	_	o	‡	¶	¶	¶	¶	¶	¶	¶	¶	¶	¶

Note: The user can specify any pattern for character-generator RAM.

**Hình 5.2:** Mã ký tự và dạng hiển thị tương ứng

### Các lệnh cho mô-đun LCD 16x2

Mô-đun LCD  $16 \times 2$  có một bộ hướng dẫn lệnh đặt trước. Mỗi lệnh sẽ làm cho mô-đun thực hiện một nhiệm vụ cụ thể. Các lệnh thường được sử dụng và chức năng của chúng được đưa ra trong bảng dưới đây

Lệnh	Chức năng
0F	LCD bật, con trỏ bật, con trỏ nháy bật
01	Xoá toàn màn hình
02	Quay về màn hình chính
04	Giảm con trỏ

06	Tăng con trỏ
0E	Màn hình bật, con trỏ nhấp nháy tắt
80	Bắt con trỏ trở về vị trí đầu tiên của hàng 1
C0	Bắt con trỏ trở về vị trí đầu tiên của hàng 2
38	Sử dụng 2 hàng và ma trận 5x7
83	Con trỏ hàng 1 vị trí 3
3C	Kích hoạt dòng 2
08	Tắt màn hình hiển thị và con trỏ
C1	Nhảy đến dòng 2 vị trí 1
OC	Bật màn hình hiển thị, tắt con trỏ
C2	Nhảy đến hàng 2, vị trí 2

## Khởi tạo LCD

Các bước phải được thực hiện để khởi tạo màn hình LCD được đưa ra dưới đây và các bước này là phổ biến cho hầu hết các ứng dụng.

B1: Gửi 38H đến dòng dữ liệu 8 bit để khởi tạo

B2: Gửi 0FH để bật LCD, con trỏ BẬT và con trỏ nhấp nháy ON.

B3: Gửi 06H để tăng vị trí con trỏ.

B4: Gửi 01H để xóa màn hình và trả về con trỏ.

## Đưa dữ liệu vào LCD

Các bước để gửi dữ liệu đến mô-đun LCD được đưa ra dưới đây. Mô-đun LCD có các chân RS, R / W và E. Chính trạng thái logic của các chân này làm cho mô-đun xác định xem đầu vào dữ liệu đã cho là lệnh hay dữ liệu được hiển thị.

Đặt R / W mức thấp.

Đặt RS = 0 nếu byte dữ liệu là lệnh và tạo RS = 1 nếu byte dữ liệu là dữ liệu sẽ được hiển thị.



Đặt byte dữ liệu trên thanh ghi dữ liệu.

Xung E từ cao xuống thấp.

Lặp lại các bước trên để gửi dữ liệu khác.

Dữ liệu chữ cái và chữ số, ký hiệu sinh viên có thể tham khảo bảng mã ASCII

Dec	Hx	Char	Dec	Hx	HTML	Char	Dec	Hx	HTML	Char	Dec	Hx	HTML	Char
0	0	NUL (null)	32	20	&#32;	Space	64	40	&#64;	�	96	60	&#96;	�
1	1	SOH (Start of heading)	33	21	&#33;	!	65	41	&#65;	A	97	61	&#97;	a
2	2	STX (Start of text)	34	22	&#34;	"	66	42	&#66;	B	98	62	&#98;	b
3	3	ETX (End of text)	35	23	&#35;	#	67	43	&#67;	C	99	63	&#99;	c
4	4	EOT (End of transmission)	36	24	&#36;	\$	68	44	&#68;	D	100	64	&#100;	d
5	5	ENQ (Enquiry)	37	25	&#37;	%	69	45	&#69;	E	101	65	&#101;	e
6	6	ACK (Acknowledge)	38	26	&#38;	&	70	46	&#70;	F	102	66	&#102;	f
7	7	BEL (Bell)	39	27	&#39;	*	71	47	&#71;	G	103	67	&#103;	g
8	8	BS (Backspace)	40	28	&#40;	{	72	48	&#72;	H	104	68	&#104;	h
9	9	TAB (Horizontal tab)	41	29	&#41;	}	73	49	&#73;	I	105	69	&#105;	i
10	A	LF (NL line fd, new line)	42	2A	&#42;	*	74	4A	&#74;	J	106	6A	&#106;	j
11	B	VT (Vertical tab)	43	2B	&#43;	+	75	4B	&#75;	K	107	6B	&#107;	k
12	C	FF (NP form fd, new page)	44	2C	&#44;	,	76	4C	&#76;	L	108	6C	&#108;	l
13	D	CR (Carriage return)	45	2D	&#45;	-	77	4D	&#77;	M	109	6D	&#109;	m
14	E	SO (Shift out)	46	2E	&#46;	.	78	4E	&#78;	N	110	6E	&#110;	n
15	F	SI (Shift in)	47	2F	&#47;	/	79	4F	&#79;	O	111	6F	&#111;	o
16	10	DLE (Data link escape)	48	30	&#48;	0	80	50	&#80;	P	112	70	&#112;	p
17	11	DCL (Device control 1)	49	31	&#49;	1	81	51	&#81;	Q	113	71	&#113;	q
18	12	DC2 (Device control 2)	50	32	&#50;	2	82	52	&#82;	R	114	72	&#114;	r
19	13	DC3 (Device control 3)	51	33	&#51;	3	83	53	&#83;	S	115	73	&#115;	s
20	14	DC4 (Device control 4)	52	34	&#52;	4	84	54	&#84;	T	116	74	&#116;	t
21	15	NAK (Negative acknowledge)	53	35	&#53;	5	85	55	&#85;	U	117	75	&#117;	u
22	16	SYN (Synchronous idle)	54	36	&#54;	6	86	56	&#86;	V	118	76	&#118;	v
23	17	ETB (End of trans. block)	55	37	&#55;	7	87	57	&#87;	W	119	77	&#119;	w
24	18	CAN (Cancel)	56	38	&#56;	8	88	58	&#88;	X	120	78	&#120;	x
25	19	EM (End of medium)	57	39	&#57;	9	89	59	&#89;	Y	121	79	&#121;	y
26	1A	SUB (Substitute)	58	3A	&#58;	:	90	5A	&#90;	Z	122	7A	&#122;	z
27	1B	ESC (Escape)	59	3B	&#59;	:	91	5B	&#91;	[	123	7B	&#123;	{
28	1C	FS (File separator)	60	3C	&#60;	<	92	5C	&#92;	\	124	7C	&#124;	
29	1D	GS (Group separator)	61	3D	&#61;	=	93	5D	&#93;	]	125	7D	&#125;	)
30	1E	RS (Record separator)	62	3E	&#62;	>	94	5E	&#94;	^	126	7E	&#126;	-
31	1F	US (Unit separator)	63	3F	&#63;	?	95	5F	&#95;	-	127	7F	&#127;	DEL

Hình 5.3: Bảng mã ASCII

**Yêu cầu:** Sinh viên tìm hiểu ý nghĩa chân cắm của LCD. Sinh viên viết mã phần cứng SystemVerilog có chức năng nạp vào LCD dòng chữ “Chuan bi TN5”

Tham khảo module LCD:



```
//////////  
// Ho Chi Minh University of Technology  
// Faculty of Electrical & Electronics Engineering  
// Department of Electronics  
// EE1016 - EE1010 : Ky Thuat So  
//  
// Lab 5 : Thiet ke he thong den giao thong  
// Thuc hien giao tiep DE2 - GPIO - Breadboard - ICs/LEDs/LCDs  
// module simple lcds  
//  
//////////  
  
module lcddisp_ex(  
    input clk,  
    output reg lcd_e, lcd_rs,  
    output reg [7:0] data  
);  
//////////  
//preset lcd write  
integer j = 1;  
  
reg [7:0] Datas [1:15];  
  
always @(posedge clk) begin  
    Datas[1] = 8'h38;    //--- control instruction : configure - 2 lines, 5x7 matrix --  
    Datas[2] = 8'h0C;    //--- control instruction : Display on, cursor off --  
    Datas[3] = 8'h06;    //--- control instruction : Increment cursor : shift cursor to right --  
    //
```



```
Datas[4] = 8'h01;    //-- control instruction : clear display screen --
Datas[5] = 8'h80;    //-- control instruction : force cursor to begin at first
line --

Datas[6] = 8'h54;    //-- T --
Datas[7] = 8'h68;    //-- h --
Datas[8] = 8'h69;    //-- i --
Datas[9] = 8'h20;    //--   --
Datas[10] = 8'h4E;   //-- N --
Datas[11] = 8'h67;   //-- g --
Datas[12] = 8'h68;   //-- h --
Datas[13] = 8'h69;   //-- i --
Datas[14] = 8'h65;   //-- e --
Datas[15] = 8'h6D;   //-- m --
end

///////////////////////////////
integer i = 0; //

always @(posedge clk) begin

//-- Delay for writing data

if (i <= 1000000) begin
  i = i + 1; lcd_e = 1;
  data = Datas[j];
end

else if (i > 1000000 & i < 2000000) begin
```



```
i = i + 1; lcd_e = 0;  
end  
  
else if (i == 2000000) begin  
j = j + 1; i = 0;  
end  
else i = 0;  
  
//-- LCD_RS signal should be set to 0 for writing commands and to 1 for writing data  
  
if (j <= 5 )  
    lcd_rs = 0;  
else if (j > 5 & j< 18)  
    lcd_rs = 1;  
else if (j == 18)  
    lcd_rs = 0;  
else if (j > 27) begin  
    lcd_rs = 1;  
    j = 5;  
end  
else  
    lcd_rs = 1;  
//else if (j > 27) begin  
//    lcd_rs = 1;  
///  j = 5;  
//end  
end  
endmodule
```





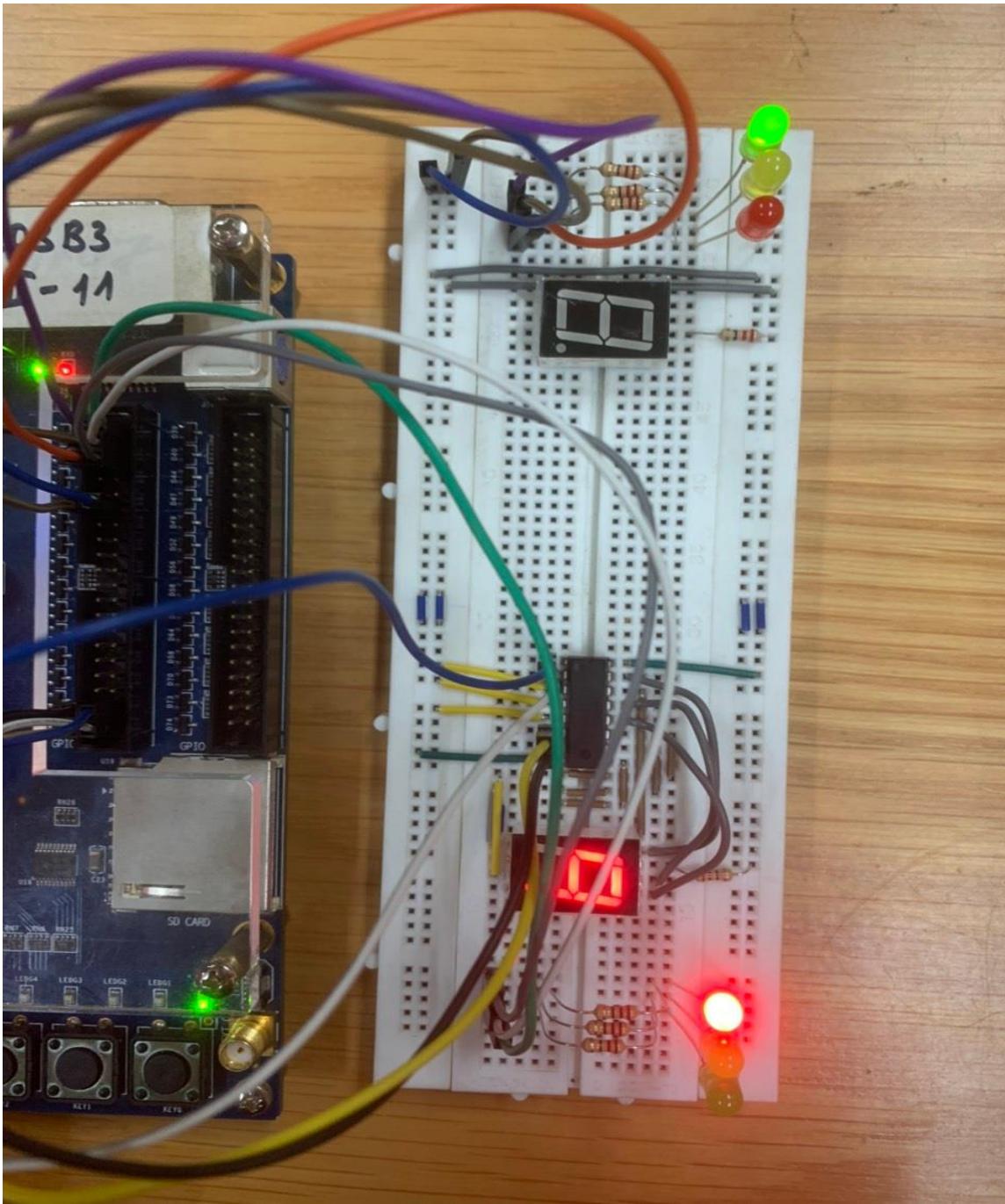
Module LCD Sinh viên viết:

Ngoài ra, với chân Contrast Voltage, sinh viên cần lắp biến trở để điều chỉnh. Sinh viên vẽ lại sơ đồ mạch kết nối GPIO với LCD

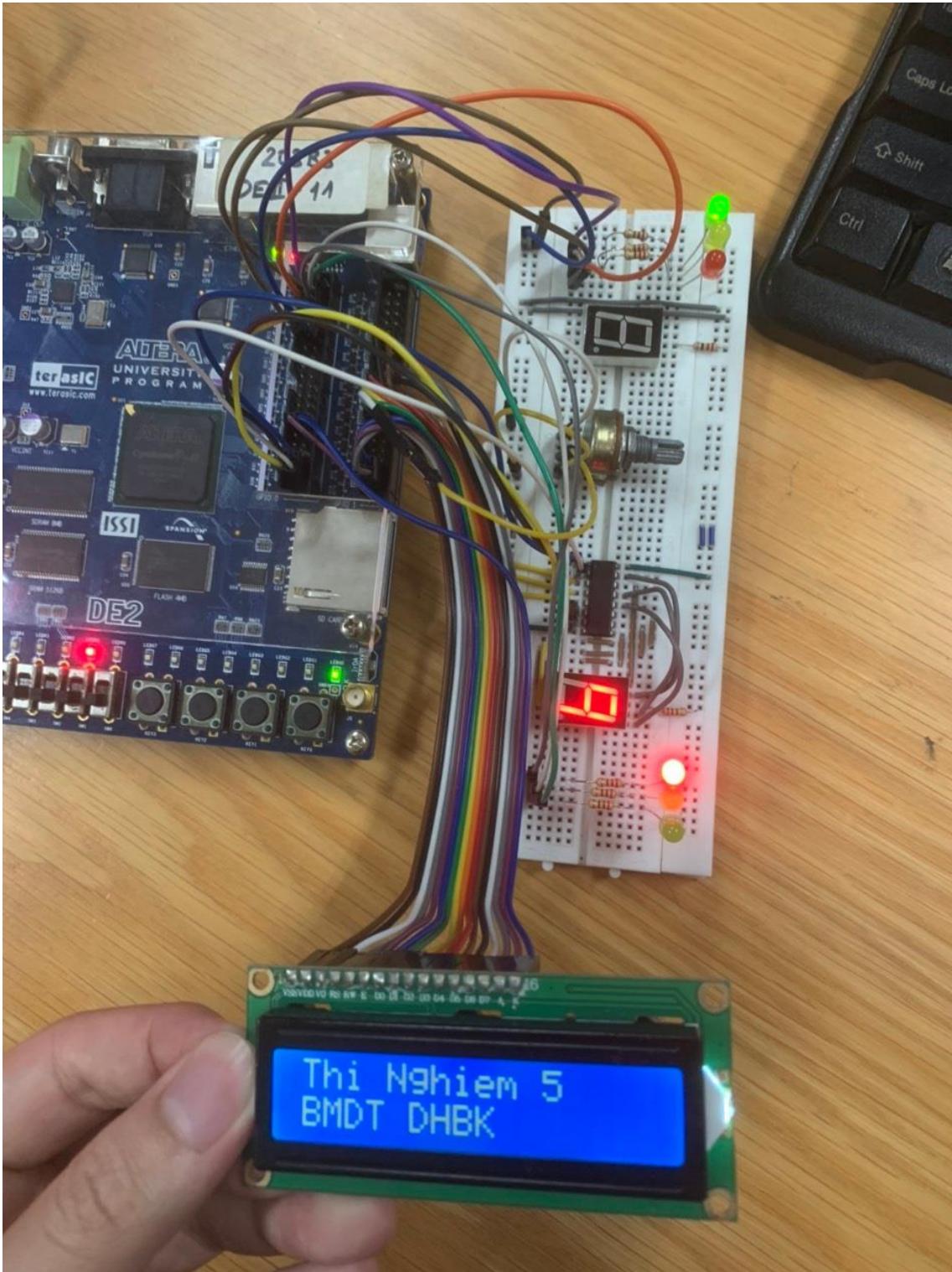


### Phần VIII: Mạch lắp tham khảo

Sinh viên tham khảo hình mạch lắp mẫu dưới đây để có thể làm sơ đồ lắp mạch tương tự cho thí nghiệm 1, 2 và 3



Hình 5.4 Sơ đồ lắp mạch ví dụ - 1



**Hình 5.5** Sơ đồ lắp mạch ví dụ - 2

## B. HƯỚNG DẪN THÍ NGHIỆM

### I. MỤC TIÊU

- Áp dụng lý thuyết vào thiết kế trong thực tế.
- Kết hợp kiến thức ngôn ngữ mô tả phần cứng và mạch kỹ thuật số.

### II. CHUẨN BI:

- Hoàn thành đầy đủ các bài thí nghiệm 1, 2, 3, 4.
- Sinh viên phải hoàn thành và nộp PRELAB 5 trước khi vào lớp.

### III. HƯỚNG DẪN THÍ NGHIỆM

#### THÍ NGHIỆM 1

**Mục tiêu:** Thiết kế hệ thống đèn giao thông điều khiển thủ công

**Yêu cầu:** Thiết kế một hệ thống đèn giao thông cho một ngã tư có các thông số sau:

Đầu vào (input):

- Các nút nhấn bao gồm:
  - o 1 nút **RESET**. (Sử dụng nút Key0 trên Kit De 2)
  - o 1 Switch thay đổi hướng xe được di chuyển (Sử dụng nút SW1 trên Kit De2)

Ngõ ra (output):

- Sinh viên sử dụng ngõ ra Expansion Header trên Kit De 2 (dùng GPIO\_0) để kết nối tới các linh kiện sau:
- Hai bộ đèn (Hai bộ còn lại đối diện ở mỗi hướng là tùy chọn) bao gồm:
  - o 1 Led 7 đoạn đếm giây
  - o 1 Led Đỏ
  - o 1 Led Xanh lá
  - o 1 Led Vàng





Mô tả hoạt động:

- Sau khi RESET, hệ thống đèn giao thông tự động hoạt động theo chế độ
- Ở chế độ điều khiển:
  - Hướng xe được cho phép di chuyển sẽ sáng đèn xanh lá
  - Hướng còn lại sáng đèn đỏ
  - Tất cả đèn đếm hiển thị 9 giây và không thay đổi giá trị
- Khi hướng xe thay đổi (ở chế độ điều khiển)
  - Hướng xe được di chuyển trước đó chuyển từ đèn xanh lá sang đèn vàng, đèn đếm hiển thị 3 giây và đếm ngược xuống 0 rồi chuyển sang đèn đỏ
  - Hướng còn lại giữ nguyên đèn đỏ cho đến khi hướng trước kia chuyển sang đèn đỏ, thời gian đếm ngược từ 3 giây đếm xuống 0.
  - Lúc này, hướng được chuyển sang đèn xanh. Tất cả đèn đếm hiển thị 9 giây và không thay đổi giá trị

**Kiểm tra:**

Sinh viên sử dụng kit De 2 với ngôn ngữ mô tả phần cứng SystemVerilog cho mạch mô tả phần cứng hoạt động như trên, kết nối từ GPIO 0 đến breadboard để thể hiện hệ thống đèn. Hệ thống nút nhấn, switch sử dụng trên kit như yêu cầu đề bài.

- Sinh viên vẽ lại sơ đồ khối của thiết kế phần cứng



- Sinh viên vẽ lại sơ đồ nguyên lý mạch đã lắp (Từ GPIO đến các linh kiện trên Breadboard)



- Sinh viên vẽ lại FSM của hệ thống điều khiển thủ công



- Viết lại đoạn mã SystemVerilog wrapper (top\_level) của thiết kế





- Sinh viên trình bày kết quả hoạt động khi nạp lên Kit DE 2 (hình ảnh)



## THÍ NGHIỆM 2

**Mục tiêu:** Thiết kế hệ thống đèn giao thông tự động, hai chế độ

### Yêu cầu:

Thiết kế một hệ thống đèn giao thông cho một ngã tư có các thông số sau:

Đầu vào (input):

- Các nút nhấn bao gồm:
  - o 1 nút **RESET**. (Sử dụng nút Key0 trên Kit De 2)
  - o 1 Switch chuyển giữa chế độ tự động và điều khiển (Sử dụng SW0 trên Kit De2)
  - o 1 Switch thay đổi hướng xe được di chuyển (Sử dụng nút SW1 trên Kit De2)

Ngõ ra (output):

- Thay thế kết nối giữa GPIO\_0 Kit De 2 đến Led 7 đoạn thành Kết nối từ GPIO\_0 Kit De 2 đến IC 74LS47 đến Led 7 đoạn.

Mô tả hoạt động:

- Sau khi RESET, hệ thống đèn giao thông tự động hoạt động theo chế độ cài ở SW0.
- Chế độ điều khiển như thí nghiệm 1
- Ở chế độ tự động:
  - o Mỗi hướng xe thay nhau di chuyển
  - o Hướng được di chuyển có đèn xanh trong thời hạn 5 giây đếm xuống 0, chuyển sang đèn vàng và từ 2 giây đếm xuống 0. Lúc này hướng chuyển sang đèn đỏ, thời gian đếm ngược 9 đếm xuống 0.
  - o Hướng còn lại hiển thị đèn đỏ và thời gian đếm ngược 9 đếm xuống 0 (Trùng thời điểm hướng còn lại vừa đèn đỏ được 1 giây) và chuyển sang





đèn xanh trong thời hạn 5 giây đếm xuống 0, chuyển sang đèn vàng và từ 2 giây đếm xuống 0. Và cứ thế tiếp tục.

**Lưu ý:**

Thời gian trong yêu cầu có tính chất tham khảo, sinh viên sử dụng và áp dụng cách đếm sao cho hệ thống hoạt động hợp lý.

**Kiểm tra:**

Sinh viên sử dụng kit De 2 với ngôn ngữ mô tả phần cứng SystemVerilog cho mạch mô tả phần cứng hoạt động như trên, kết nối từ GPIO 0 đến breadboard để thể hiện hệ thống đèn. Hệ thống nút nhấn, switch sử dụng trên kit như yêu cầu đề bài.

- Sinh viên vẽ lại sơ đồ khối của thiết kế phần cứng





## Lab 5: Bài tổng hợp thực hiện mạch số trên breadboard – giao tiếp với FPGA.

- Sinh viên vẽ lại sơ đồ nguyên lý mạch đã lắp (Từ GPIO đến các linh kiện trên Breadboard)

- Sinh viên vẽ lại FSM của hệ thống điều khiển tự động





- Viết lại đoạn mã SystemVerilog wrapper (top\_level) của thiết kế



- Sinh viên trình bày kết quả hoạt động khi nạp lên Kit DE 2 (hình ảnh)

### THÍ NGHIỆM 3

**Mục tiêu:** Thiết kế hệ thống đèn giao thông tự động, hai chế độ như trên sử dụng IC 74Ls47 kèm theo bảng quảng cáo LCD đặt ở góc ngã tư

**Yêu cầu:**

- Sinh viên kết nối đến module LCD 16 x 2 dùng GPIO\_1 tới breadboard, hiển thị đoạn quảng cáo “Thi Nghiêm 5 KTS” ở hàng 1, “BM Dien Tu DHBK” ở hàng 2.



**Hình 5.6: LCD 2x16**

**Kiểm tra:**

Sinh viên sử dụng kit De 2 với ngôn ngữ mô tả phần cứng SystemVerilog cho mạch mô tả phần cứng hoạt động như trên, kết nối từ GPIO 0 đến breadboard để thể hiện hệ thống đèn. Hệ thống nút nhấn, switch sử dụng trên kit như yêu cầu đề bài.

- Sinh viên vẽ lại sơ đồ khói của thiết kế phần cứng



- Sinh viên vẽ lại sơ đồ nguyên lý mạch đã lắp (Từ GPIO đến các linh kiện trên Breadboard)





## Lab 5: Bài tổng hợp thực hiện mạch số trên breadboard – giao tiếp với FPGA.

- Viết lại đoạn mã SystemVerilog wrapper (top\_level) của thiết kế

- Giao tiếp LCD bằng phần cứng mô tả với SystemVerilog được thực hiện như thế nào? Viết lại đoạn mã SystemVerilog giao tiếp LCD.





- Sinh viên trình bày kết quả hoạt động khi nạp lên Kit DE 2 (hình ảnh)



## PHỤ LỤC 1: HƯỚNG DẪN CÀI ĐẶT QUARTUS VÀ UBUNTU TRONG WINDOWS

### A. HƯỚNG DẪN CÀI ĐẶT UBUNTU TRONG WINDOWS:

#### I. Download và cài đặt công cụ:

Tải và cài đặt Xming:

Xming là X Server trên hệ điều hành Windows, được sử dụng để chạy các ứng dụng Linux trên Windows.

Link download Xming: <https://sourceforge.net/projects/xming/>

Tải WSL2 Kernel: (chỉ thực hiện bước này nếu máy tính sử dụng hệ điều hành Windows 11)

WSL2 Kernel là công cụ cho phép người dùng chạy Linux trong Windows.

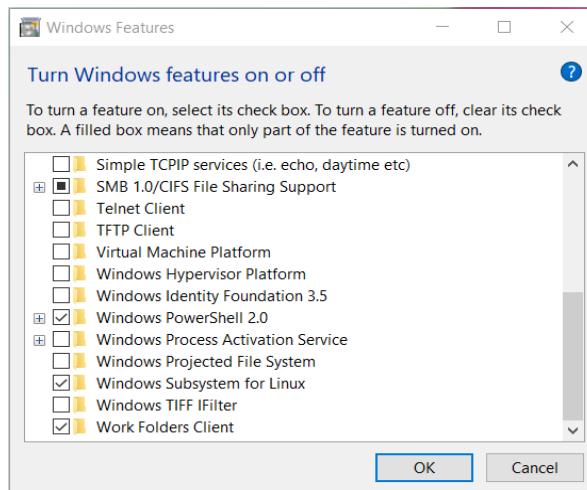
Link download WSL2 Kernel:

[https://wslstorestorage.blob.core.windows.net/wslblob/wsl\\_update\\_x64.msi](https://wslstorestorage.blob.core.windows.net/wslblob/wsl_update_x64.msi)

#### II. Cài đặt Ubuntu trong Windows:

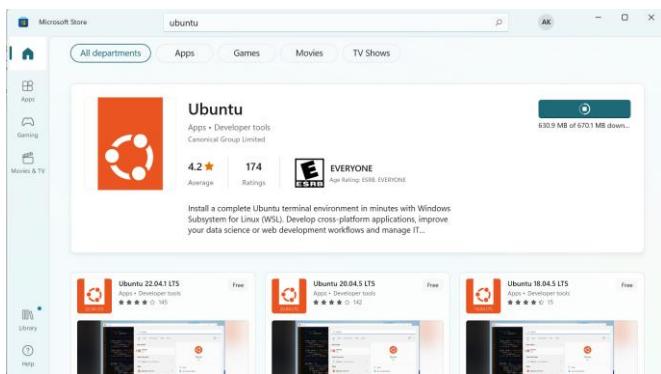
Bước 1: Bật Windows Subsystem for Linux

- Mở Control Panel → Programs and Features → “Turn Windows features on or off”.
- Chọn “Windows Subsystem for Linux, Virtual Machine Platform”
- Đối với Windows 11, cài đặt WSL2 Kernel.
- Khởi động lại Windows.

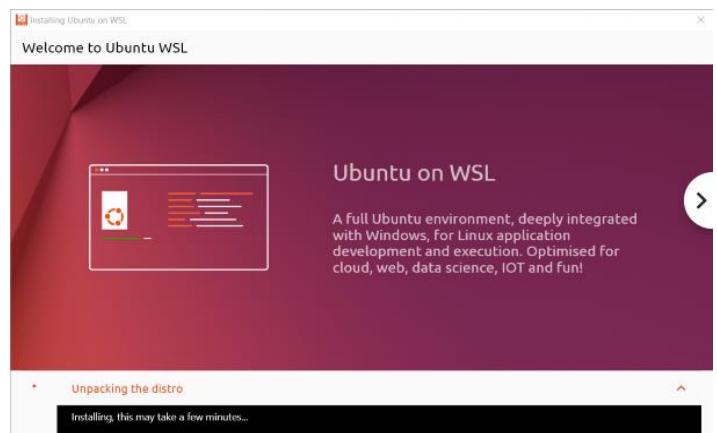


Hình phụ lục 1.1 “Turn Windows features on or off”

## Bước 2: Mở Windows Store → Tìm và cài đặt Ubuntu 22.04.1 LTS



Hình phụ lục 1.2 Ubuntu trong Windows Store



Hình phụ lục 1.3 Khởi động Ubuntu

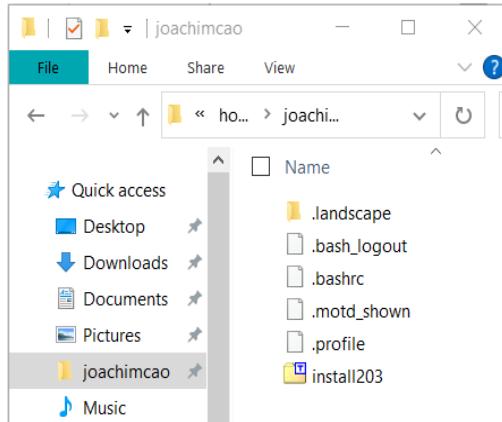
- Khởi động Ubuntu 22.04.1 LTS
- Đặt Username và Password. Lưu ý: password sẽ KHÔNG hiển thị trên màn hình.

```
Ubuntu 22.04.1 LTS
Please create a default UNIX user account. The username does not need to match your
For more information visit: https://aka.ms/wslusers
Enter new UNIX username: joachimcao
New password:
```

Hình phụ lục 1.4 Đặt UNIX username và password

- Xác định thư mục làm việc của Ubuntu trong Windows: nhập command **cd ~; explorer.exe .**. Trong thư mục **home**, tìm thư mục có tên là username đã đặt; click chuột phải vào thư mục này, chọn **Pin to Quick Access**. Tất cả các file và thư mục được tạo ra trong Ubuntu đều được lưu tại thư mục **username**.

**Bước 3:** Download file sau (*install203.tar*) và lưu trong thư mục có tên là **username** đã đặt:  
<https://drive.google.com/file/d/1F2aKS83WR8D6xWmu1k4xSWetmcbcmeue/view?usp=sharing>



**Hình phụ lục 1.5 Lưu file install203**

Kiểm tra: Sau khi hoàn thành các bước trên, khi khởi động Ubuntu → gõ “**ls**” (liệt kê các file có trong thư mục) → xuất hiện file **install203.tar**

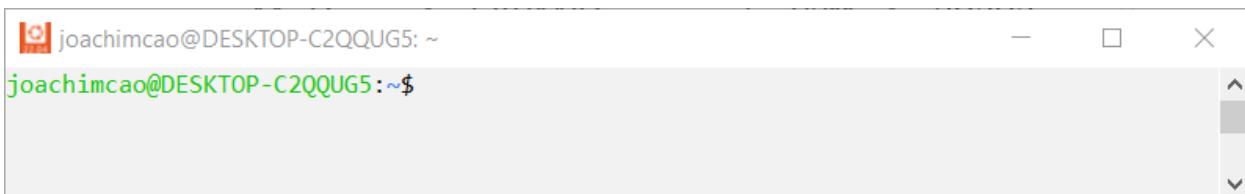
#### Cài đặt file **install203.tar**:

Mở Ubuntu 22.04.1 LTS và chạy command sau :

```
$ sudo apt -y install make  
$ chmod 755 install203.tar  
$ tar xvf install203.tar  
$ cd install203/  
$ make
```

### **III. Một số lệnh cơ bản trong Linux**

Trong terminal Ubuntu, phía trước dấu **\$** hiển thị tên của thư mục hiện tại.



**Hình phụ lục 1.6 Terminal Ubuntu**

Một số lệnh thông dụng trong Linux:

**ls** List (liệt kê) files trong thư mục

**tree** Tương tự như **ls** như ở dạng cây thư mục



<b>mkdir abc</b>	Make directory (tạo thư mục) tên “abc”
<b>cd abc</b>	Change directory (đi vô thư mục) <b>abc</b>
<b>cd ..</b>	Change directory về thư mục trước
<b>micro file1.sv</b>	Tạo/soạn thảo file tên “ <b>file1.sv</b> ” bằng <b>micro</b>
<b>cp file1.sv file2.sv</b>	Copy <b>file1.sv</b> vô thư mục hiện tại và đặt tên là <b>file2.sv</b>
<b>cp file1.sv abc/</b>	Copy <b>file1.sv</b> vô thư mục <b>abc</b>
<b>cp file1.sv abc/file3.sv</b>	Copy <b>file1.sv</b> vô thư mục <b>abc</b> và đặt tên là <b>file3.sv</b>
<b>mv file1.sv file0.sv</b>	Move <b>file1.sv</b> thành <b>file0.sv</b> (đổi tên)
<b>mv file0.sv abc/</b>	Move <b>file0.sv</b> vô thư mục <b>abc</b>
<b>mv abc/file3.sv .</b>	Move <b>file3.sv</b> trong thư mục <b>abc</b> ra thư mục hiện tại
<b>cp -r abc/ xyz/</b>	Copy directory <b>abc</b> và đặt tên là <b>xyz</b>
<b>pwd</b>	Cho biết đường dẫn thư mục hiện tại
<b>rm file1.sv</b>	Remove (xoá) <b>file1.sv</b>
<b>rm file2.sv xyz/file1.sv</b>	Remove (xoá) <b>file2.sv</b> và <b>file1.sv</b> trong thư mục <b>xyz</b>
<b>rm -rf xyz</b>	Remove directory <b>xyz</b>

Khi cần sử dụng nhiều lệnh, các lệnh nên được thực hiện theo thứ tự trên. Sau mỗi lệnh, sinh viên nên thực hiện lệnh **ls** hoặc **tree** và **pwd** để hiểu hoạt động của các lệnh.

(Sinh viên có thể tham khảo các tổ hợp phím tắt bằng cách sử dụng tổ hợp phím **Alt+G**).



## B. HƯỚNG DẪN CÀI ĐẶT QUARTUS 13.0SP1

**Bước 1:** Download và cài đặt Quartus 13.0sp1.

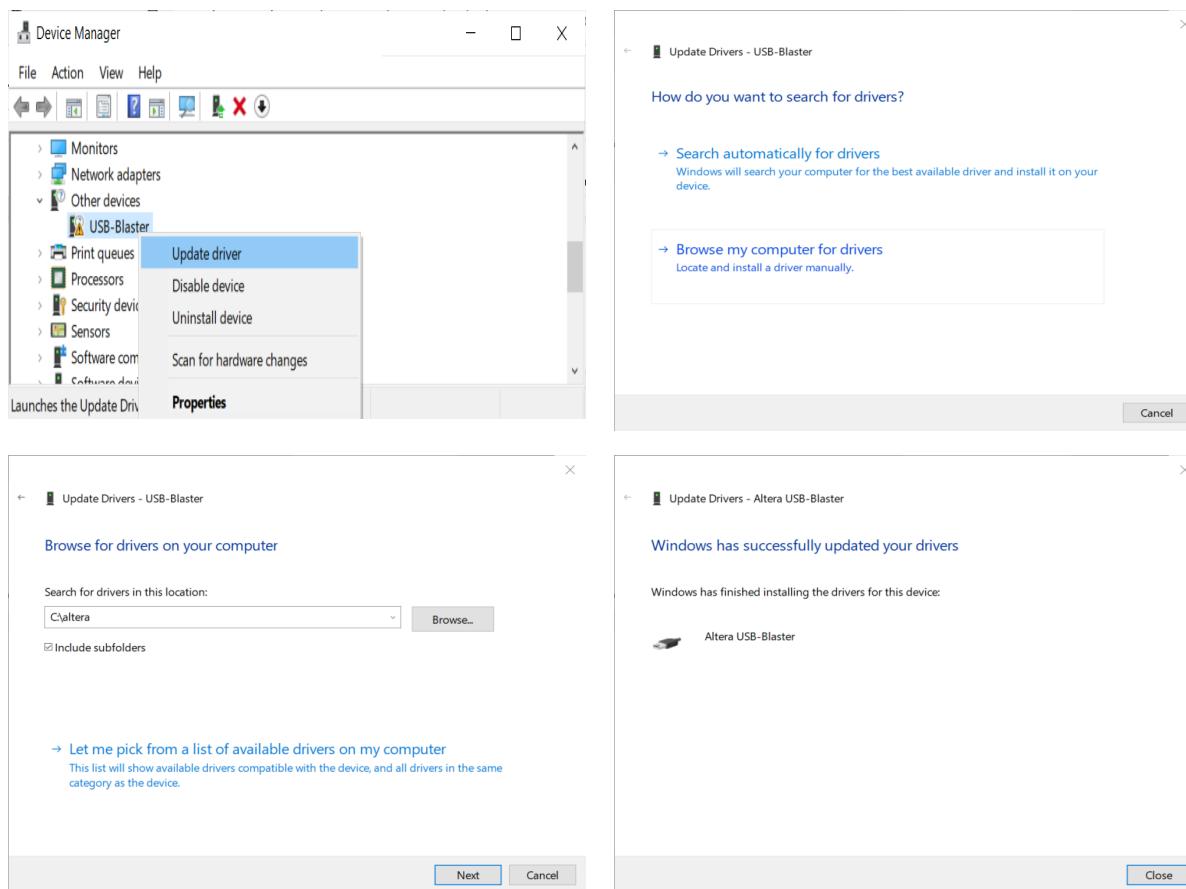
Link download: <https://www.intel.com/content/www/us/en/software-kit/711791/intel-quartus-ii-web-edition-design-software-version-13-0sp1-for-windows.html>

Giải nén file cài đặt. Click phải file **Setup** → chọn **Run as Administrator**. Chọn địa chỉ cài đặt là **C:\altera**.

Lưu ý: Khi cài đặt Quartus, phần chọn components : chọn các devices Cyclone để giao tiếp với kit DE2 và DE10.

**Bước 2:** Cài đặt driver giao tiếp với kit FPGA

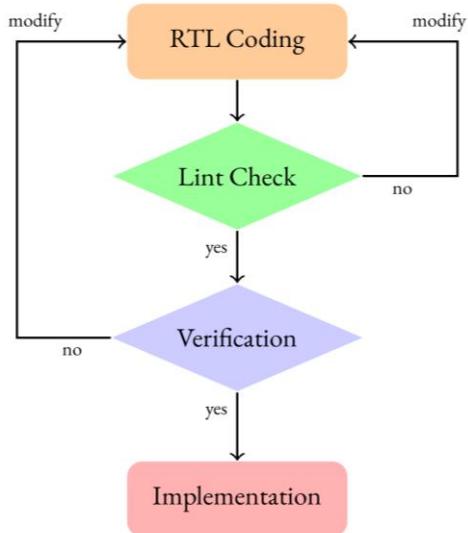
- Kết nối USB giữa kit FPGA và máy tính. Cáp nguồn cho kit FPGA.
- Mở Control Panel → Device Manager → Other devices, click chuột phải USB-Blaster → chọn Update Driver → Browse for drivers on your computer → chọn đường dẫn tới thư mục chứa USB – Blaster.



**Hình phụ lục 1.7** Cài đặt driver USB – Blaster

## PHỤ LỤC 2: QUY TRÌNH THIẾT KẾ MẠCH SỐ SỬ DỤNG NGÔN NGỮ SYSTEMVERILOG

### A. QUY TRÌNH THIẾT KẾ



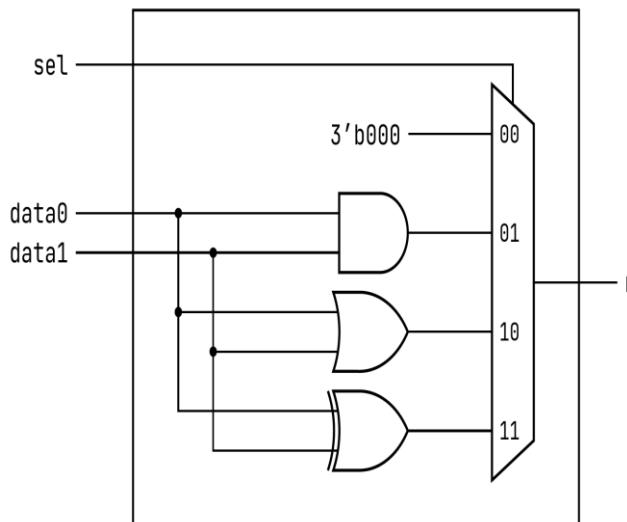
*Hình phụ lục 2.1 Quy trình thiết kế sử dụng ngôn ngữ SystemVerilog*

- RTL Coding:** Thiết kế thuật toán, FSM,... để giải quyết các yêu cầu được đặt ra, ngôn ngữ thiết kế ở đây là SystemVerilog HDL.
- Lint Check:** Kiểm tra những lỗi về cú pháp/syntax hoặc những cách viết code có khả năng gây ra lỗi hoặc bug. Lint cần được chạy sau khi viết code xong. Lưu ý, Lint Check là quy trình kiểm tra code tĩnh - tức là code sẽ không được chạy, nên tính đúng đắn và hoạt động của code sẽ không được kiểm tra.  
→ Nếu Lint Check báo lỗi, cần đọc lỗi để biết lỗi gì, ở dòng nào → Sửa code
- Verification:** Kiểm tra hoạt động của code - dựa trên yêu cầu thiết kế. Verification là giai đoạn chạy thử chương trình đã viết. Kết quả của quá trình verification là tập ngõ ra tương ứng với tập ngõ vào cho trước, và so sánh ngõ ra này với giá trị kỳ vọng. Đây là quy trình kiểm tra tính đúng đắn của code.  
→ Nếu Verification có lỗi, cần đọc code để biết lỗi gì, ở dòng nào → Sửa code → Lint Check
- Implementation:** Thiết kế sau khi Verification hoàn tất đã đạt yêu cầu thì sẽ được nạp lên KIT (DE2 hoặc DE10 hay những KIT FPGA khác).

## B. VÍ DỤ THIẾT KẾ HỆ TỔ HỢP

### I. Yêu cầu:

Thiết kế hệ tổ hợp sau



Hệ có các ngõ vào:

data0, data1 : 3 bit

sel : 2 bit

Ngõ ra của hệ:

result : 3 bit

Bảng hoạt động của hệ:

sel	result
0	0
0	data0 & data1
1	data0   data1
1	data0 ^ data1

### II. Thiết kế:

#### Tạo Project

Tại ~ (~ nếu cần thiết), tạo directory **projects**, trong đó tạo directory **lab0**. Trong **lab0**, sử dụng template trên GitHub, đặt tên là **ex01**.

```
gettemplate ex01
```

Gõ **ls** để xem các files và thư mục có trong thư mục hiện tại, hoặc **tree** nếu muốn xem cây thư mục.

```
$ tree
:
└── ex01
    ├── filelist
    ├── makefile
    └── quartus
        ├── de10_pin_assign.qsf
        └── de2_pin_assign.qsf
    └── test
        ├── driver.cpp
        └── tb_top.cpp
3 directories, 6 files
```

Một project cần được tổ chức như sau:

- Chuyển thư mục hiện hành thành **ex01** (`cd ex01`), tạo thư mục **src**: `mkdir src`
- Tất cả các file design **.sv** cần đặt trong directory **src**.
- Soạn thảo: tạo file hoặc sửa file (nếu file đã tồn tại): `micro <filename>`
- Các file design cần được khai báo trong **filelist**.

```
joachimcao@DESKTOP-C2QQUG5:~/lab0/ex01
joachimcao@DESKTOP-C2QQUG5:~$ mkdir lab0
joachimcao@DESKTOP-C2QQUG5:~$ tree
└── lab0

1 directory, 0 files
joachimcao@DESKTOP-C2QQUG5:~$ cd lab0
joachimcao@DESKTOP-C2QQUG5:~/lab0$ gettemplate ex01
A   ex01/filelist
A   ex01/makefile
A   ex01/quartus
A   ex01/quartus/de10_pin_assign.qsf
A   ex01/quartus/de2_pin_assign.qsf
A   ex01/test
A   ex01/test/driver.cpp
A   ex01/test/tb_top.cpp
Checked out revision 1.
joachimcao@DESKTOP-C2QQUG5:~/lab0$ tree
└── ex01
    ├── filelist
    ├── makefile
    └── quartus
        └── de10_pin_assign.qsf
            └── de2_pin_assign.qsf
    └── test
        ├── driver.cpp
        └── tb_top.cpp

3 directories, 6 files
joachimcao@DESKTOP-C2QQUG5:~/lab0$ cd ex01
joachimcao@DESKTOP-C2QQUG5:~/lab0/ex01$ mkdir src
joachimcao@DESKTOP-C2QQUG5:~/lab0/ex01$ tree
.
├── filelist
├── makefile
├── quartus
│   └── de10_pin_assign.qsf
│       └── de2_pin_assign.qsf
└── src
    └── test
        ├── driver.cpp
        └── tb_top.cpp

3 directories, 6 files
joachimcao@DESKTOP-C2QQUG5:~/lab0/ex01$
```

Hình phụ lục 2.2 Tạo project



## Code

Tạo file design có tên **design\_1.sv** đặt trong **src**.

**micro src/design\_1.sv** hoặc **cd src** rồi **micro design\_1.sv**

**src/design\_1.sv**

```
1  module design_1 (
2      // input
3      input logic [2:0] data0_i,
4      input logic [2:0] data1_i,
5      input logic [1:0] sel_i,
6
7      // output
8      output logic [2:0] result_o
9  );
10
11     // local declaration
12     logic [2:0] and_tmp; // temporary for and result
13     logic [1:0] or_tmp; // temporary for or result
14     logic [2:0] xor_tmp; // temporary for xor result
15
16     assign and_tmp = data0_i & data1_i;
17     assign or_tmp = data0_i | data1_i;
18     assign xor_tmp = data0_i ~^ data1_i;
19
20     always_comb begin : proc_mux
21         case (sel_i)
22             2'b00: result_o =      '0;
23             2'b01: result_o = and_tmp;
24             2'b11: result_o = or_tmp;
25             2'b10: result_o = xor_tmp;
26         endcase
27     end
28
29 endmodule : design_1
```

Tại **ex01**, mở và thêm tên file trong **filelist**. Nhập **micro filelist** hoặc nếu đang ở trong **src** thì ra ngoài **ex01** bằng cách nhập **cd ..**

**filelist**

```
1  src/design_1.sv
```

## Lint Check

Bước 1: Chạy Lint Check để kiểm tra lỗi, nhập **make lint**.

```
$ make lint
-----> LINT CHECK <-----
%Error: src/design_1.sv:23:25: Can't find definition of variable: 'and_tnp'
          : ... Suggested alternative: 'and_tmp'
 23 |     2'b01: result_o = and_tnp;
      |           ^
%Error: Exiting due to 1 error(s)
make: *** [makefile:50: lint] Error 1
$
```

Bước 2: Phân tích và debug

Sau khi chạy Lint, đoạn code trên có bug ở dòng 23, lỗi không tìm thấy **and\_tnp** (sai so với tên biến khai báo là **and\_tmp**). Sửa lại ở dòng 23, lặp lại **Bước 1, make lint**.

```
$ make lint
-----> LINT CHECK <-----
%Warning-WIDTH: src/design_1.sv:17:18: Operator ASSIGNW expects 2 bits on the Assign
RHS, but Assign RHS's OR generates 3 bits.
          : ... In instance design_1
 17 |     assign or_tmp = data0_i | data1_i;
      |           ^
          ... For warning description see
https://verilator.org/warn/WIDTH?v=4.225
          ... Use "/* verilator lint_off WIDTH */" and lint_on around source to
disable this message.
%Warning-WIDTH: src/design_1.sv:24:23: Operator ASSIGN expects 3 bits on the Assign
RHS, but Assign RHS's VARREF 'or_tmp' generates 2 bits.
          : ... In instance design_1
 24 |     2'b11: result_o = or_tmp;
      |           ^
%Warning-CASEOVERLAP: src/design_1.sv:25:7: Case values overlap (example pattern 0x3)
 25 |     2'b11: result_o = xor_tmp;
      |           ^
%Warning-CASEINCOMPLETE: src/design_1.sv:21:5: Case values incompletely covered
(example pattern 0x2)
 21 |     case (sel_i)
      |           ^
%Error: Exiting due to 4 warning(s)
make: *** [makefile:50: lint] Error 1
$
```

Tiếp tục thực hiện phân tích và debug:

1. Lỗi đầu tiên, **%Warning-WIDTH: src/design\_1.sv:17** → lỗi về độ dài bit không phù hợp ở dòng 17. Đọc thêm ta thấy biến **or\_tmp** dài 2 bit, nhưng kết quả của phép gán



bên phải dài 3 bit (RHS: Right Hand Side - bên phải). Đây là lỗi sai khi khai báo **or\_tmp** có 2 bit ([1:0]). → Sửa dòng khai báo **or\_tmp** (dòng 13) thành 3 bit: [2:0]

2. Lỗi phía trên kéo theo lỗi tiếp theo ở dòng 24.
3. Lỗi ở dòng 25, **%Warning-CASEOVERLAP: src/design\_1.sv:25** → lỗi về việc trùng lắp giá trị so sánh case (đoạn code thấy có 2 trường hợp 2'b11).
4. Lỗi ở dòng 21, **%Warning-CASEINCOMPLETE: src/design\_1.sv:21** → lỗi về việc thiếu giá trị so sánh case. Lỗi này do thiếu trường hợp 2'b10. Như vậy lỗi dòng 21 và 25 liên quan tới nhau, sửa dòng 24 thành 2'b10.

**Bước 3:** Sau khi debug, lắp lại **Bước 1, make lint.**

```
$ make lint
-----> LINT CHECK <-----
$
```

Kết quả thông báo không có lỗi, chuyển qua phần tiếp theo là Verification.

## Verification

Set Up: Để verify thiết kế trên, cần tạo driver để lái input và đặt điều kiện ở output ứng với input cụ thể.

Các giá trị input, hay còn gọi là test transactions, được tạo ngẫu nhiên, vì thế ta không biết được chính xác giá trị inputs nào được đưa vào và tính toán trước để so sánh. Thêm nữa, việc tính toán thủ công theo cách đó, đối với thiết kế lớn với hàng ngàn mẫu test sẽ gây khó khăn cho người thiết kế. Do đó phải kiểm tra outputs bằng cách tự động, ở đây sử dụng assertions để kiểm tra.

**Bước 1:** Tạo file **top.sv** ở trong directory **ex01**, gọi file **design\_1.sv**.

1. Khai báo input và output như trong **design\_1**, thêm xung clock. Cứ mỗi xung clock là một test transaction.

**top.sv**

```
1 module top (
2     // input
3     input logic clk_i,
4     input logic [2:0] data0_i,
5     input logic [2:0] data1_i,
6     input logic [1:0] sel_i ,
7     // output
```



```
8     output logic [2:0] result_o
9 );
10
```

2. Gọi module **design\_1**. Lưu ý, sau dấu chấm là tên port của design, trong dấu ngoặc là tên signal/data nối vô port đó.

**top.sv**

```
11   design_1 dut (
12     .data0_i (data0_i ),
13     .data1_i (data1_i ),
14     .sel_i   (sel_i  ),
15     .result_o(result_o)
16   );
17
```

3. Đặt điều kiện cho output. Sử dụng keyword **assert** để ràng buộc output như đoạn code bên dưới theo yêu cầu của thiết kế, ví dụ: nếu **sel = 2'b11**, thì **result** phải bằng **xor** hai biến data, nếu không, simulation sẽ bị lỗi vì design không thỏa điều kiện assertion. Ở đây có tổng là 4 điều kiện ứng với 4 giá trị **sel**.

**top.sv**

```
18   always @(posedge clk_i) begin : proc_assertions
19     if (sel_i == 2'b00)
20       assert (result_o == '0);
21     if (sel_i == 2'b01)
22       assert (result_o == (data0_i & data1_i));
23     if (sel_i == 2'b10)
24       assert (result_o == (data0_i | data1_i));
25     if (sel_i == 2'b11)
26       assert (result_o == (data0_i ^ data1_i));
27   end
28
29 endmodule : top
```

**Bước 2:** Tạo driver để lái input cách ngẫu nhiên (để phát hiện bug hơn đặt sẵn input và kiểm tra output, vì ngõ ra output chỉ cần ràng buộc để kiểm tra tính đúng đắn)

1. Tạo file **driver.cpp** trong **test**: **micro test/driver.cpp** hoặc **cd test** rồi **micro driver.cpp**

**test/driver.cpp**

```
1 #define MAX_SIM 20
2
```



```
3 void set_random(Vtop *dut, vluint64_t sim_unit) {
4     dut->data0_i = rand()%8;
5     dut->data1_i = rand()%8;
6     dut->sel_i    = rand()%4;
7 }
```

2. Lưu ý 1: input gồm **data0\_i**, **data1\_i**, và **sel\_i**, trình bày dưới dạng **dut->data0\_i**, ... . Tín hiệu **clk\_i** luôn được cấp tự động, cho nên **top.sv** luôn luôn phải có **input logic clk\_i**
3. Lưu ý 2: đây là file C++, để tạo random, sử dụng hàm **rand()**. **sel\_i** có 2 bit, do đó số ngẫu nhiên **sel\_i** được tạo ra bằng cách lấy số dư của phép chia 4, và các giá trị **data** có 3 bit nên được tạo thành bằng cách lấy số dư của phép chia 8. Nếu muốn giữ giá trị cố định, ví dụ như **data0\_i** là **3'b101**, chỉ cần gán **dut->data0\_i = 0b101** (Cách gán của C++).
4. Lưu ý 3: **MAX\_SIM** là số mẫu ngẫu nhiên được tạo.

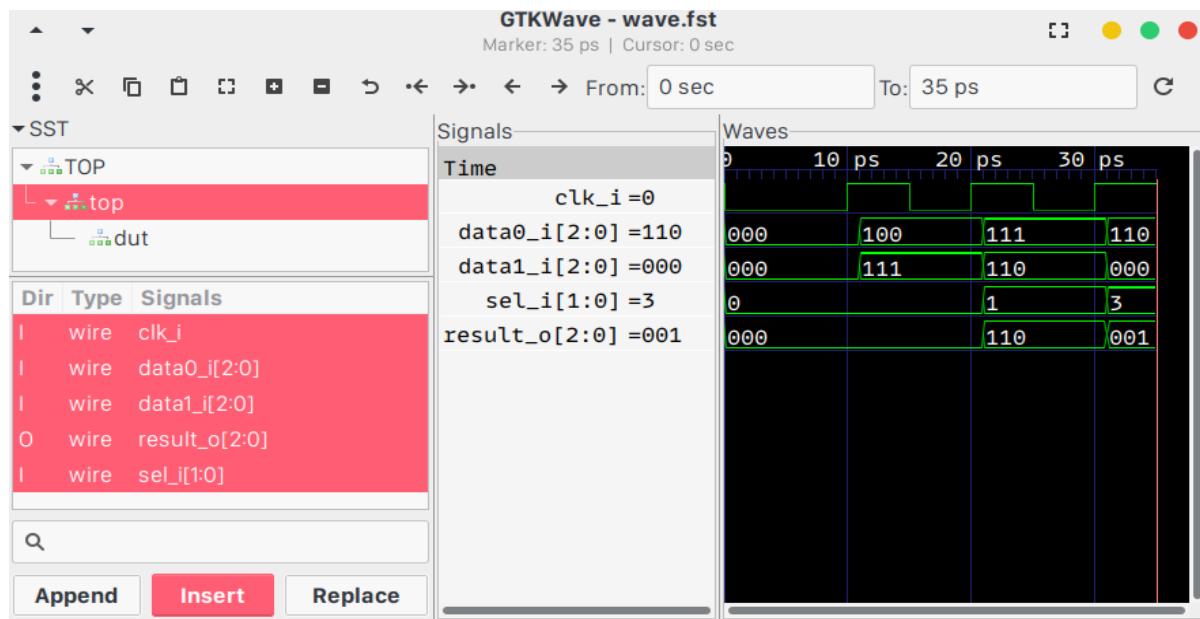
#### Simulating:

**Bước 1:** Chạy Simulating để kiểm tra lỗi, nhập **make sim**.

```
$ make sim
-----> BUILD <-----
...
...
...
-----> SIMULATING <-----
[0] %Error: top.sv:26: Assertion failed in TOP.top.proc_assertions: 'assert' failed.
%Error: top.sv:26: Verilog $stop
Aborting...
make: *** [makefile:61: sim] Aborted (core dumped)
$
```

Xuất hiện lỗi **Assertion failed** ở dòng 26 của **top.sv**. Kiểm tra dòng 26, assertion này ràng buộc ngõ ra của **sel = 2'b11**, chứng tỏ **design\_1.sv** sai khi **sel = 2'b11**.

**Bước 2:** Mở waveform để quan sát, nhập **make wave**. Chọn **top** → Chọn tất cả tín hiệu trong ô dưới → Chọn **Insert**. Output là **001** thay vì **110**.



*Hình phụ lục 2.3 Kết quả mô phỏng bước Verification*

Có thể mở menu và lưu việc chọn tín hiệu này: **File → Write Save File**

**Bước 3:** Kiểm tra lại code **design\_1.sv** thấy:

*src/design\_1.sv*

```
18 assign xor_tmp = data0_i ^ data1_i;
```

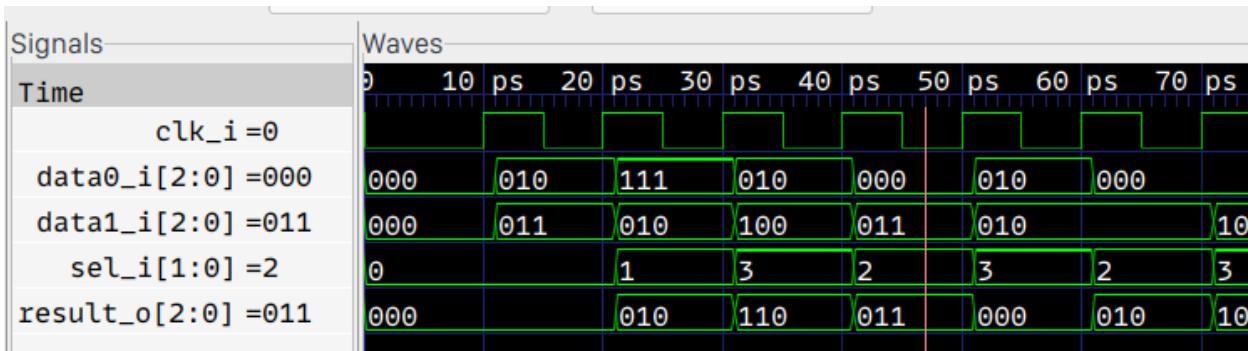
Bug xuất hiện ở dòng 18 vì phép toán đúng là  $\wedge$  nhưng là  $\sim\wedge$  → Sửa code.

**Bước 4:** Lặp lại **Bước 1**. Chạy sim: **make sim**.

```
$ make sim
-----> BUILD <-----
...
...
...
-----> SIMULATING <-----
$
```

Không có lỗi. Lặp lại **Bước 2**, quan sát waveform: **make wave**.

```
$ make wave
-----> WAVEFORMS <-----
```



*Hình phụ lục 2.4 Kết quả mô phỏng sau khi hoàn tất bước Verification*

**Bước 5:** Hoàn tất.

## Implementation

### Set Up

**Bước 1:** Tạo file **wrapper.sv**.

File wrapper sẽ gọi **design\_1** giống như file **top.sv**, tuy nhiên các ngõ vào và ngõ ra sẽ lấy từ file sau:

<https://www.terasic.com.tw/cgi-bin/page/archive.pl?CategoryNo=53&No=30&PartNo=4>

(tải file *DE2\_UserManual\_1.6* hoặc *DE2 Pin Table*)

Các ngõ vào **data0\_i**, **data1\_i**, và **sel\_i**: sử dụng switches để làm input (bảng 4.1 trong file trên). Gán **data0\_i** là **SW2**, **SW1**, và **SW0**; **data1\_i** là **SW5**, **SW4**, và **SW3**; và **sel\_i** là **SW7** và **SW6**. Vậy input sẽ khai báo như dòng 3 và gán như 3 dòng 9, 10, 11.

Ngõ ra **result\_o** có 3 bits: sử dụng LED làm output (bảng 4.3 trong file trên). Chọn LED màu đỏ **LEDR** 2, 1, và 0. Vậy output sẽ khai báo như dòng 5 và gán như dòng 12.

*quartus/wrapper.sv*

```

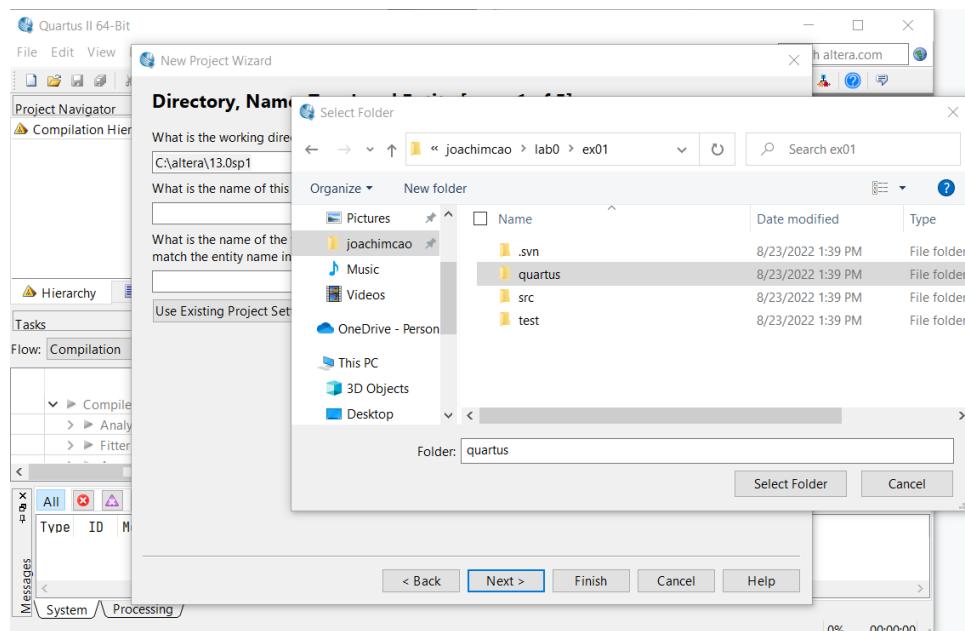
1  module wrapper (
2    // input
3    input logic [7:0] SW,
4    // output
5    output logic [2:0] LEDR
6  );
7
8  design_1 dut (
9    .data0_i (SW[2:0] ),
10   .data1_i (SW[5:3] ),
11   .sel_i   (SW[7:6] ),

```

```
12      .result_o(LEDR[2:0])
13  );
14
15 endmodule : wrapper
```

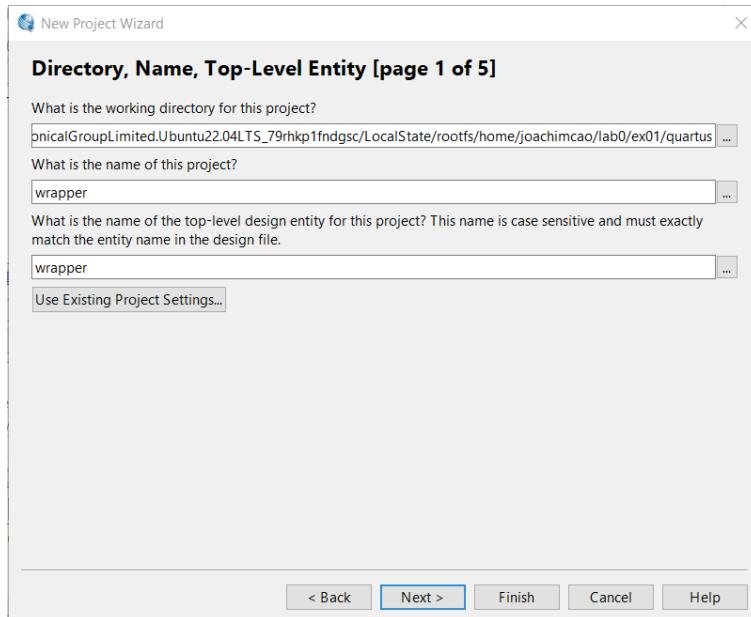
## Bước 2: Tạo project trong Quartus.

1. Mở Quartus.
2. Chọn **File → New Project Wizard**.
  - a. Dẫn tới directory **quartus** trong **ex01**.



Hình phụ lục 2.5 Xác định thư mục chứa project

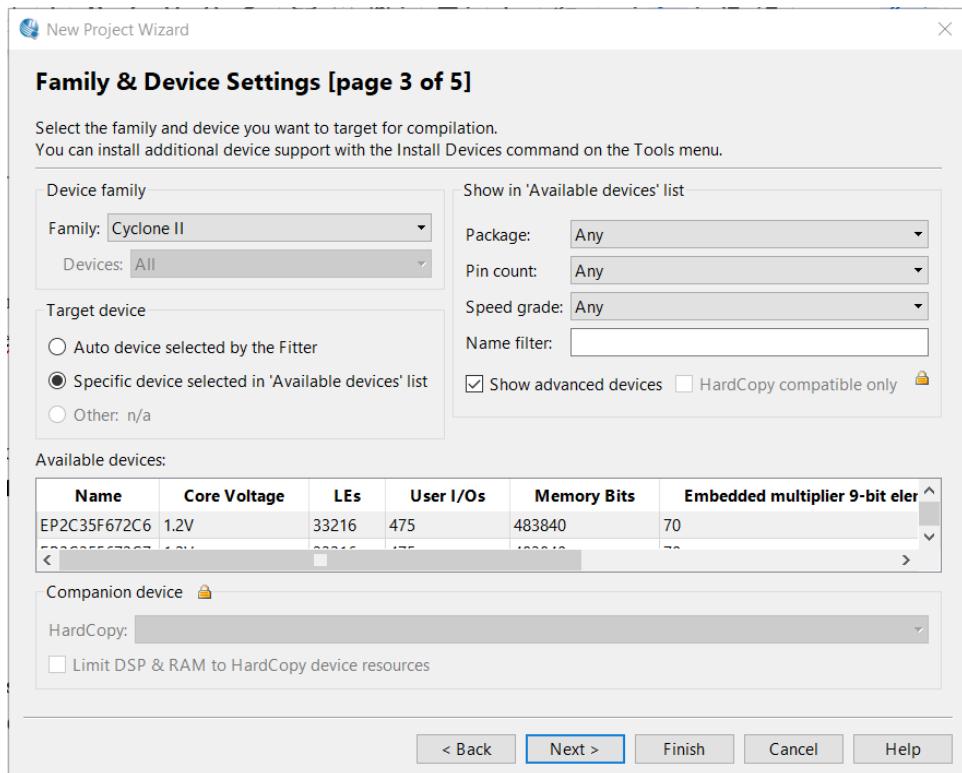
- b. Đặt tên project là **wrapper**. Vì project thực tế là thư mục **ex01**, nên cần đặt tên đúng như tên file nối pin là file **wrapper.sv**.



Hình phụ lục 2.6 Đặt tên project

- c. **Next** để đi tiếp.
3. Chọn ... để lấy source code
- Trỏ tới directory **wrapper.sv** là **quartus**, chọn hết tất cả các files → **Open** → **Add**.
  - Trỏ tới directory chứa source code là **src**, chọn hết tất cả các files → **Open** → **Add**.
  - Chọn cột **Type** là **SystemVerilog HDL File**.
  - Next** để đi tiếp.
4. Đối với DE2
- Device family** chọn **Cyclone II**.
  - Available devices** chọn **EP2C35F672C6**.

## c. Finish để hoàn tất.



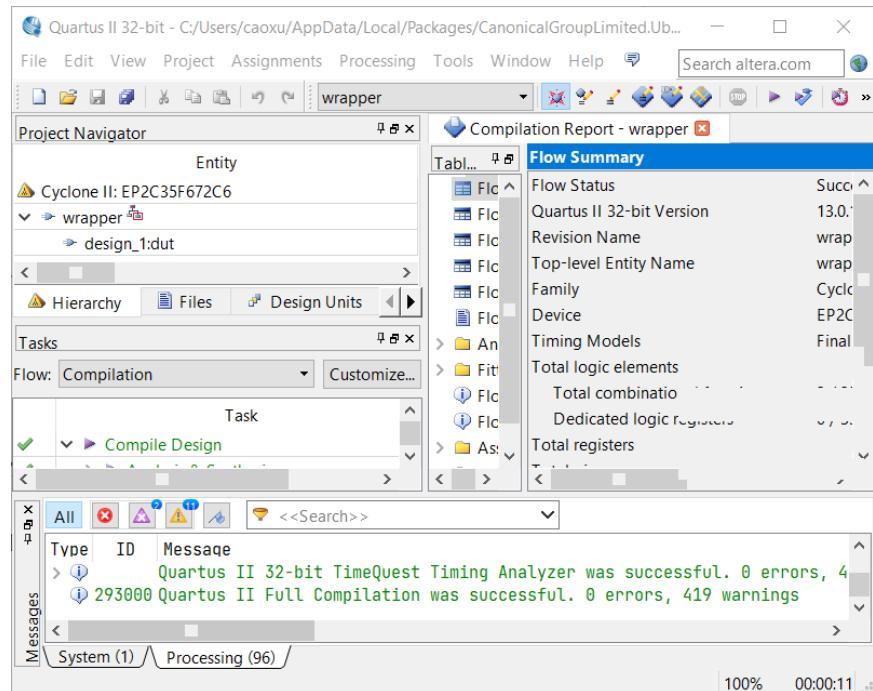
**Hình phụ lục 2.7 Chọn kit DE2**

**Bước 3:** Import pin assignment của DE2.

1. Chọn **Assignments → Import Assignments**.
2. Chọn ... , trả lời **quartus** và chọn **de2\_pin\_assign.qsf** → **Open** → **OK**.

**Bước 4:** Compilation

**Ctrl+L** hoặc **Processing → Start Compilation**



*Hình phụ lục 2.8 Kết quả biên dịch*

Kiểm tra cửa sổ thông báo không có errors → tiến hành nạp lên KIT.

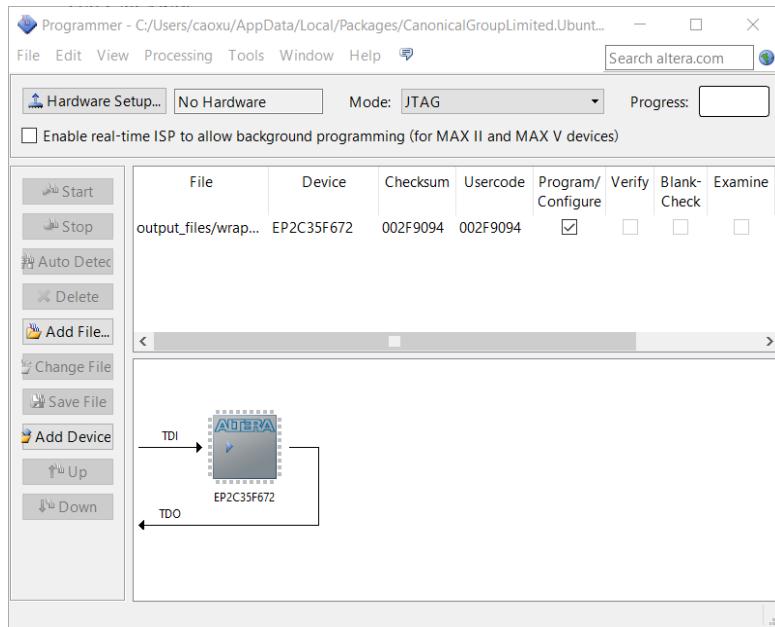
### Thực hiện trên KIT

1. Kết nối DE2 bằng dây USB ở cổng Blaster



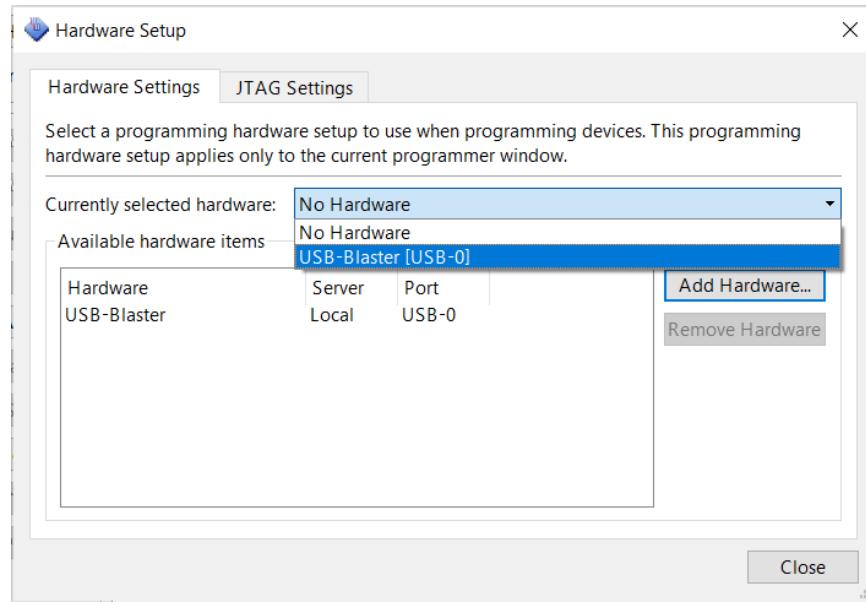
*Hình phụ lục 2.9 Kết nối kit DE2*

## 2. Chọn Tools → Programmers



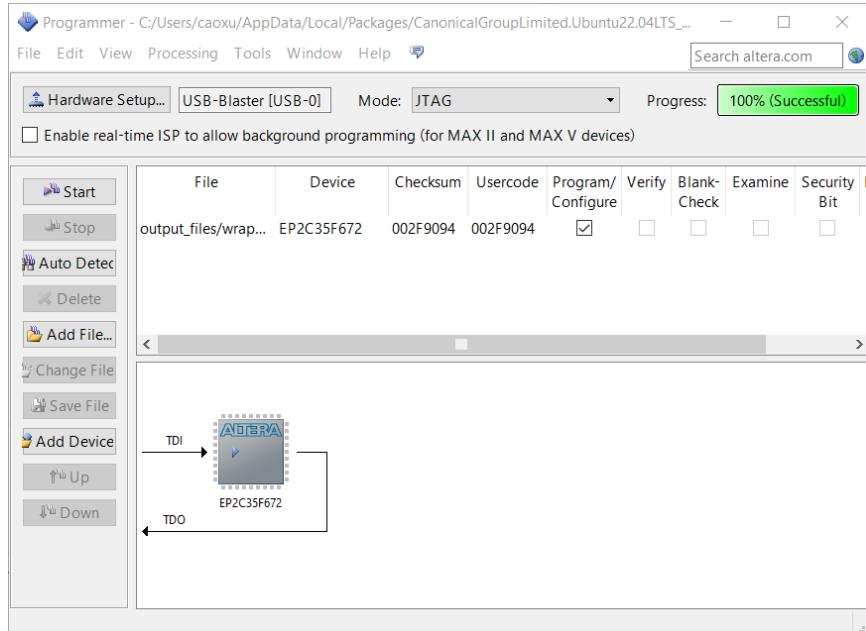
*Hình phụ lục 2.10 Cửa sổ Programmer*

## 3. Kiểm tra đã kết nối với KIT: USB-Blaster X, nếu hiện No Hardware, nhấn **Hardware Setup** và chọn **USB-Blaster X → Close**



*Hình phụ lục 2.11 Hardware setup*

## 4. Chọn **Start** để tiến hành nạp. Ô **Progress** hiển thị **100% (successful)** tức là đã nạp thành công.



*Hình phụ lục 2.12 Nạp kit thành công*

## C. VÍ DỤ THIẾT KẾ HỆ TUẦN TỰ

### I. Yêu cầu:

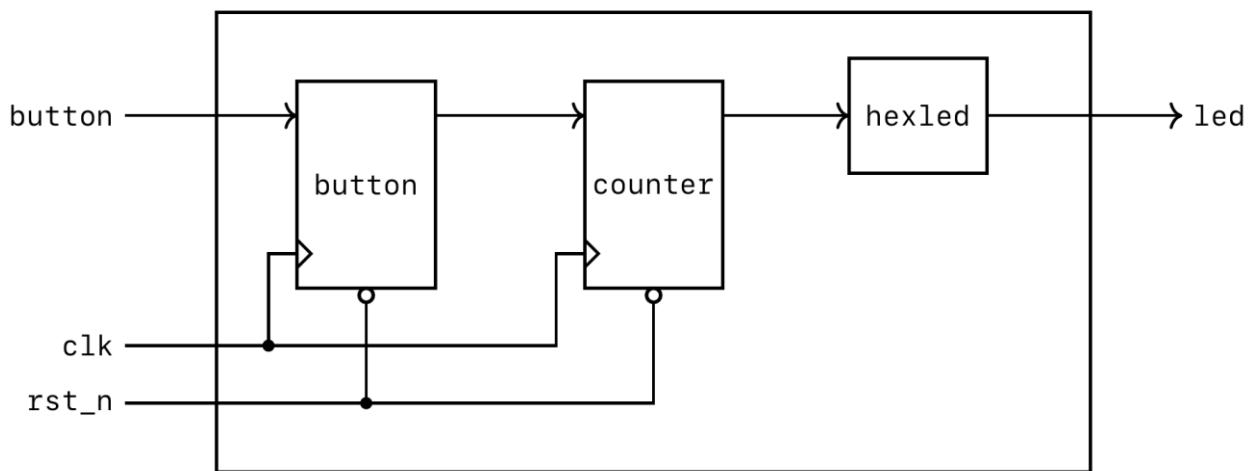
Thiết kế mạch đếm số lần nhấn nút bấm (nếu nút được nhấn và giữ thì vẫn tính là một lần nhấn), hiển thị kết quả lên LED 7 đoạn.

### II. Phân tích:

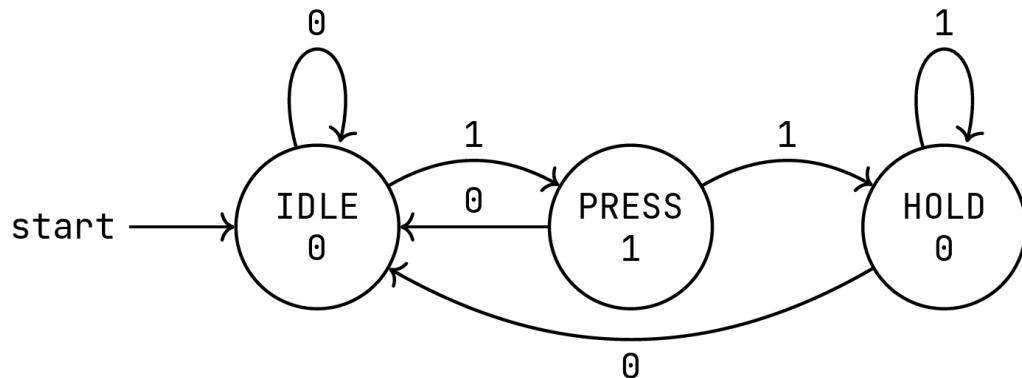
Hệ nhận ngõ vào từ một nút nhấn (khi nút được nhấn, hệ nhận được bit 1, ngược lại, hệ nhận được bit 0) và hiển thị ngõ ra trên LED 7 đoạn (ngõ ra nhỏ hơn hoặc bằng 9). Ngoài ra, hệ tuần tự còn có ngõ vào clock và reset (chọn reset tích cực thấp). Khi nhấn giữ nút nhấn, hệ tuần tự nhận được tín hiệu 1 chỉ trong 1 chu kỳ.

Mạch gồm một bộ đếm, trạng thái của bộ đếm tăng thêm 1 khi nhận tín hiệu ngõ vào là 1. Ngoài ra, hệ cần có một mạch chuyển mã BCD sang mã LED 7 đoạn, hiển thị trạng thái của bộ đếm lên LED 7 đoạn.

Khối button là một FSM (Finite State Machine – máy trạng thái) với 3 trạng thái: IDLE (chờ nhận), PRESS (khi nút được nhấn trong cycle đầu), HOLD (trạng thái đang nhấn giữ nút).



Hình phụ lục 2.13 Sơ đồ khói của hệ tuần tự



Hình phụ lục 2.14 Máy trạng thái của khói button

### III. Thiết kế:

#### Tạo Project

Source template từ GitHub để sử dụng, đặt tên là ex02

```
gettemplate ex02
```

#### Code

1. Tạo file design có tên **button.sv**.

*src/button.sv*

```
1  module button (
2      // input
3      input logic clk_i,
4      input logic rst_ni,
5
6      input logic button_i,
7
8      // output
9      output logic stable_o
10 );
11
12 // local declaration
13 typedef enum logic [1:0] {
14     IDLE,
15     PRESS,
16     HOLD
17 } state_e;
18
19 state_e state_d;
20 state_e state_q;
21
22 always_comb begin : proc_next_state
23     case (state_q)
24         IDLE: state_d = button_i ? PRESS : IDLE;
25         PRESS: state_d = button_i ? PRESS : IDLE;
26         HOLD: state_d = button_i ? HOLD : IDLE;
27         default: state_d = IDLE;
28     endcase
29 end
30
31 always_ff @(posedge clk_i) begin : proc_state_register
32     if (!rst_ni)
33         state_q <= IDLE;
34     else
35         state_q <= state_d;
36 end
37
38 // output combinational logic
39 assign stable_o = (state_q == PRESS) ? 1'b1 : 1'b0;
40
41 endmodule : button
```

2. Tạo file design có tên **counter.sv**.*src/counter.sv*

```
1  module counter (
2      // input
3      input logic clk_i,
4      input logic rst_ni,
```



```
5      input logic      inc_i,
6
7      // output
8      output logic [3:0] counter_o
9
10 );
11
12 always_ff @(posedge clk_i) begin : proc_counter
13     if (!rst_ni)
14         counter_o <= '0;
15     else begin
16         if ((counter_o < 4'h9) && inc_i)
17             counter_o <= counter_o + 4'h1;
18     end
19 end
20
21 endmodule : counter
```

### 3. Tạo file design có tên **hexled.sv**.

*src/hexled.sv*

```
1 module hexled (
2     // input
3     input logic [3:0] data_i,
4
5     // output
6     output logic [6:0] hex_o
7 );
8
9     always_comb begin : proc_7seg_decoder
10         case (data_i)
11             4'h0: hex_o = 7'b100_0000;
12             4'h1: hex_o = 7'b111_1001;
13             4'h2: hex_o = 7'b010_0100;
14             4'h3: hex_o = 7'b011_0000;
15             4'h4: hex_o = 7'b001_1001;
16             4'h5: hex_o = 7'b001_0010;
17             4'h6: hex_o = 7'b000_0010;
18             4'h7: hex_o = 7'b101_1000;
19             4'h8: hex_o = 7'b000_0000;
20             4'h9: hex_o = 7'b001_1000;
21             4'ha: hex_o = 7'b000_1000;
22             4'hb: hex_o = 7'b000_0011;
23             4'hc: hex_o = 7'b100_0110;
24             4'hd: hex_o = 7'b010_0001;
25             4'he: hex_o = 7'b000_0110;
26             4'hf: hex_o = 7'b000_1110;
27         endcase
28     end
29
30 endmodule : hexled
```



4. Tạo file design có tên **design\_2.sv**.*src/design\_2.sv*

```
1  module design_2 (
2      // input
3      input logic      clk_i,
4      input logic      rst_ni,
5
6      input logic      button_i,
7
8      // output
9      output logic [6:0] led_o
10 );
11
12     // local declaration
13     logic      inc;
14     logic [3:0] counter;
15
16     button button0 (
17         .clk_i (clk_i),
18         .rst_ni (rst_ni),
19         .button_i(button_i),
20         .stable_o(inc)
21     );
22
23     counter counter0 (
24         .clk_i (clk_i),
25         .rst_ni (rst_ni),
26         .inc_i (inc),
27         .counter_o(counter)
28     );
29
30     hexled hexled0 (
31         .data_i(counter),
32         .hex_o (led_o)
33     );
34
35 `ifdef VERILATOR
36     /*verilator lint_off UNUSED*/
37     logic pastvld;
38     always @ (posedge clk_i) begin
39         pastvld <= 1'b1;
40
41         if (pastvld && $past(inc))
42             assert (!inc);
43
44             assert (counter <= 4'h9);
45         end
46     /*verilator lint_on UNUSED*/
47 `endif
48
49 endmodule : design_2
```



Lưu ý: Khi thiết kế hệ tổ hợp, mạch không có tín hiệu clock, nhưng assertion thì yêu cầu cần có clock nên phải tạo assertions (trong file **top.sv** ở ví dụ phía trên). Khi thiết kế hệ tuần tự trong ví dụ này, clock là tín hiệu ngõ vào của hệ, cung cấp tới các khối **button** và **counter**, nên assertions cần đặt ở trong **design\_2.sv**, như vậy clock có thể tác động đến hoạt động của khối **button** – với 1 lần nhấn chỉ lái output lên 1 trong 1 cycle, tức là output của button không thể bằng 1 trong 2 cycles liên tiếp (dòng 42) – và khối **counter** – chỉ đếm từ 0 tới 9 (dòng 44).

Ở đây có thêm tín hiệu **pastvld**, bởi vì để sử dụng được **\$past** để lấy giá trị tại cycle trước đó thì giá trị đó cần phải valid.

Vì **pastvld** là tín hiệu sử dụng cho bước verification, và các **assert** cũng tương tự, cần bỏ qua lúc đỗ lên KIT, nên đặt những dòng đó trong khối sau:

```
35 `ifdef VERILATOR
36   /*verilator lint_off UNUSED*/
...
46   /*verilator lint_on UNUSED*/
47 `endif
```

Vậy, khi chạy Lint Check và khi chạy Quartus để đỗ lên KIT, phần cho verification này sẽ bị bỏ qua.

5. Tại **ex02**, mở và thêm tên file trong **filelist**.

#### filelist

```
1 src/button.sv
2 src/counter.sv
3 src/hexled.sv
4 src/design_2.sv
```

6. Tại **ex02**, tạo file design có tên **top.sv**.

#### top.sv

```
1 module top (
2   // input
3   input logic      clk_i,
4   input logic      rst_ni,
5
6   input logic      button_i,
7
8   // output
9   output logic [6:0] led_o
10 );
11
12   design_2 dut (
```



```

13      .clk_i  (clk_i  ),
14      .rst_ni (rst_ni ),
15      .button_i(button_i),
16      .led_o  (led_o  )
17  );
18
19 endmodule : top

```

## 7. Tạo file **driver.cpp** trong directory **test**

*test/driver.cpp*

```

1 #define MAX_SIM 200
2
3 void set_random(Vtop *dut, vluint64_t sim_unit) {
4     dut->rst_ni = (sim_unit > 4) && (rand()%30 != 0);
5     dut->button_i = (rand()%8 > 2);
6 }

```

- a. Lưu ý 1: input gồm **rst\_ni** và **button\_i**.
- b. Để counter có thể đếm tới 9 cần chạy đoạn test này nhiều lần, do đó cần để **MAX\_SIM** ở giá trị lớn, ví dụ là 200.
- c. Lưu ý 2: **dut->button\_i = (rand()%8 > 2);** giá trị của **button\_i** sẽ là kết quả của **rand()%8**, tức là lấy số ngẫu nhiên từ 0 tới 7, sau đó là so sánh với 2. Điều này có nghĩa là **button\_i** sẽ bằng 1 khi số dư của phép chia 8 ngẫu nhiên có giá trị từ 3 tới 7, và **button\_i** bằng 0 trong trường hợp còn lại. Như vậy, ta có xác suất:  $P(button_i = 1) = 5/8 = 62.5\%$ ,  $P(button_i = 0) = 3/8 = 37.5\%$ . Việc này làm tăng xác suất **button\_i** nhận được 1 trong 2 cycles liên tiếp, nghĩa là trường hợp nhấn giữ nút nhấn xảy ra nhiều hơn.
- d. Lưu ý 3: **dut->rst\_ni = (sim\_unit > 4) && (rand()%30 != 0);** giá trị của **rst\_ni** sẽ là luôn bằng 0, tức là reset, khi **sim\_unit <= 4** (**sim\_unit** là đơn vị simulating, mỗi xung clock ứng với 1 **sim\_unit**). Như vậy, mạch sẽ reset trong 4 chu kỳ đầu tiên, sau đó thì xác suất xuất hiện tín hiệu reset bằng xác suất của **(rand()%30 != 0)** là  $29/30 = 96.7\%$ , hay  $P(rst_ni = 0) = 3.3\%$ . Việc này làm xác suất **rst\_ni** tích cực thấp nhõ, giúp trường hợp đếm lên tới 9 (và không thể đếm được nữa) dễ xảy ra hơn.

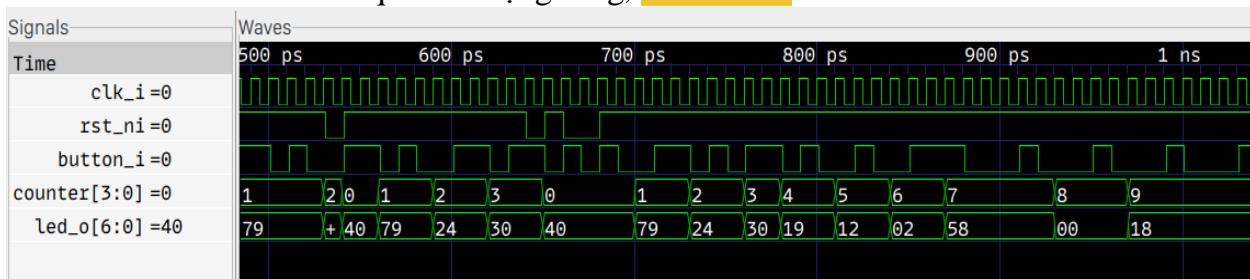
## Lint Check & Verification

**Bước 1:** Chạy Lint Check để kiểm tra lỗi, **make lint**.



**Bước 2:** Chạy Simulating để verify, **make sim** → Debug

**Bước 3:** Mở waveform để quan sát dạng sóng, **make wave**.



*Hình phụ lục 2.15 Dạng sóng mô phỏng*

**DEBUG:** có bug ở dòng 25 trong **button.sv**

PRESS: state\_d = button\_i ? PRESS : IDLE;  
 → PRESS: state\_d = button\_i ? HOLD : IDLE;

Khuyến khích sinh viên sửa thêm một vài chỗ trong 2 modules **button** và **counter** để tạo bug, từ đó làm quen với thông báo WARNING và ERROR, và biết cách để debug.

### Implementation

**Bước 1:** Tạo file **wrapper.sv**.

Các nút nhấn KEY được dùng để điều khiển các ngõ vào **button\_i**, và **rst\_ni** (tên pin như bảng 4.2 trong DE2 Manual - coi lại ví dụ trước). Gán **button\_i** là **KEY 2**, **rst\_ni** là **KEY 0**. Vậy input sẽ được khai báo ở dòng 4 và gán ở 2 dòng 12 và 13. Vì các KEY trên kit DE2 khi không được nhấn cho giá trị là 1, để phù hợp với thiết kế, ngõ vào **button\_i** của hệ chính là NOT của **KEY 2**, còn **rst\_ni** là ngõ vào tích cực thấp nên không cần thực hiện phép toán NOT.

Đối với ngõ vào clock, chọn clock 50MHz (pin clock của DE2, được liệt kê trong bảng 4.5). Khai báo clock như dòng 3 và nối dây như dòng 10.

Ngõ ra **led\_o** là led 7 đoạn gồm 7 bits (LED 7 đoạn trên kit DE2 được trình bày trong bảng 4.4 của file DE2 Manual). Chọn led 7 đoạn số 0, **HEX0**, vậy output sẽ được khai báo như dòng 7 và gán như dòng 14.

*quartus/wrapper.sv*

```

1  module wrapper (
2      // input
3      input logic      CLOCK_50,
4      input logic [2:0] KEY,
5
6      // output

```



```
7      output logic [6:0] HEX0
8  );
9
10   design_2 dut (
11     .clk_i  (CLOCK_50),
12     .rst_ni (KEY[0]  ),
13     .button_i(~KEY[2] ),
14     .led_o  (HEX0    )
15   );
16
17 endmodule : wrapper
```

**Bước 2:** Tạo project trong Quartus (tương tự ví dụ thiết kế hệ tổ hợp).

**Bước 3:** Import pin assignment của DE2 (tương tự ví dụ thiết kế hệ tổ hợp).

**Bước 4:** Compilation (tương tự ví dụ thiết kế hệ tổ hợp).

**Bước 5:** Nạp KIT (tương tự ví dụ thiết kế hệ tổ hợp).

---HẾT---