TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

BỘ MÔN MẠNG MÁY TÍNH
VÀ TRUYỀN THÔNG DỮ LIỆU

Trần Trung Tín

HƯỚNG DẪN THỰC HÀNH
MÔN HỌC: TỔ CHỨC MÁY TÍNH

TP. HỒ CHÍ MINH
2023

Mục lục

Bài thực hành LAB 1

BIỂU DIỄN SỐ BẰNG TÍN HIỆU ĐIỆN

(ĐANG CẬP NHẬT)
(Vui lòng xem hướng dẫn LAB 1 phiên bản cũ và Phụ lục hướng dẫn Logism)

Bài thực hành LAB 2

MẠCH CỘNG / TRỪ NHỊ PHÂN

"Hành trình vạn dặm, bắt đầu từ một bước chân.[1] *"*

Một chương trình máy tính được hình thành bởi hàng vạn câu lệnh có nghĩa được bố trí theo giải thuật nhất định. Một câu lệnh ở ngôn ngữ cấp cao lại được biên dịch thành vài thao tác mã máy. Và một thao tác mã máy được thực thi bởi cụm mạch tổ hợp và mạch tuần tự, nơi mà các tín hiệu điện được tiếp nhận, xử lý và kết quả được truyền đến cổng ra - cũng bằng tín hiệu điện. Trong hệ thập phân, phép cộng giữa hai số tự nhiên được thực thi theo từng cặp kí số, bắt đầu từ hàng đơn vị rồi đến hàng chục và hàng trăm. Trong hệ nhị phân cũng theo quy tắc đó, và để giải quyết phép cộng hai số 32-bit với nhau, trước tiên chúng ta cần một phần cứng thực hiện phép cộng hai số 1-bit, sau đó mở rộng đến bit nhớ (carry bit) và cuối cùng là ghép nối các phần từ thành mạch cộng rộng hơn.

2.1 Mạch cộng bán phần

Tại hàng đơn vị, hai số 1-bit được cộng với nhau theo 4 tổ hợp khả hiện được liệt kê trong bảng 3.2. Mạch cộng bán phần (half adder) này có 2 đầu vào có hai đầu ra: S và C.

- S: là kết quả của phép cộng nhị phân của hai bit đầu vào a với b.
- C: là bit tràn (còn gọi là bit nhớ) mang tín hiệu 1 khi cả 2 bit đầu vào là 1.

[1] "Thiên lý chi hành, thủy vu túc hạ " - Lão Tử.

a	b	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Bảng 2.1: Bảng sự thật mạch cộng bán phần a+b

Yêu cầu 1. Thực hiện mạch con đặt tên là HA thực hiện chức năng cộng bán phần với 2 tín hiệu nhập là 2 bit a với b; và 2 tín hiệu đầu ra là S với C. Kiểm thử mạch con này sau khi hoàn tất.

Gợi ý. Tham khảo phục lục A.2 và hiện thực mạch với bảng sự thật 3.2.

2.2 Mạch cộng toàn phần

Xem xét phép cộng hai số 2-bit như sau:

$$\begin{array}{r} \overset{1}{} \\ 01 \\ +\,11 \\ \hline 0 \end{array}$$

Mạch cộng bán phần đã thực hiện được việc cộng cặp bit ở hàng đơn vị, nhưng giờ đây sẽ không thể áp dụng vào hàng chục bởi vì có một tín hiệu thứ ba xuất hiện: đó là bit tràn từ hàng đơn vị. Mạch cộng toàn phần (full adder) có 3 tín hiệu đầu vào và 2 tín hiệu đầu ra được thể hiện trong bảng sự thật 2.2. Để phân biệt bit tràn ở đầu vào và bit tràn ở kết quả đầu ra, chúng ta có thể gọi chúng lần lượt là C_{in} (Carry in) và C_{out} (Carry out).

a	b	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	
0	1	0		0
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1	1	1

Bảng 2.2: Bảng sự thật mạch cộng toàn phần C_{in}+a+b

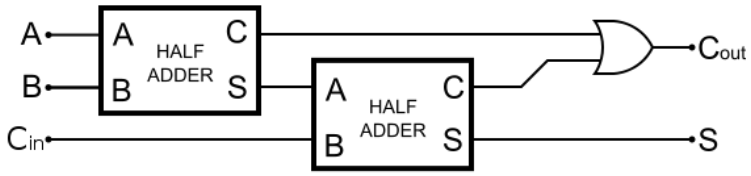
Yêu cầu 2. Hoàn tất bảng sự thật 2.2 rồi dùng nó để thực hiện mạch con đặt tên là FA thực hiện chức năng cộng toàn phần với 3 tín hiệu nhập là 2 bit a, b với bit tràn vào C_{in}; và 2 tín hiệu đầu ra là S với C_{out}. Kiểm thử mạch con này sau khi hoàn tất.

Gợi ý. Tham khảo phục lục A.2 và hiện thực mạch với bảng sự thật 2.2.

Phương pháp tạo mạch logic từ bảng sự thật không phải lúc nào cũng khả thi, đó là khi số tín hiệu đầu vào rất lớn. Với n tín hiệu đầu vào, bảng sự thật sẽ có 2^n dòng và việc hoàn tất chúng là không thể. Khi đó phương pháp thực hiện thiết kế mạch theo khối được sử dụng.

Yêu cầu 3. Thực hiện mạch con đặt tên là FA_1 thực hiện chức năng cộng toàn phần bằng cách sử dụng mạch con HA đã tạo trong yêu cầu **Yêu cầu 1.**.

Gợi ý. Tham khảo cách ghép nối như hình 2.1.



Hình 2.1: Mạch cộng toàn phần được ghép từ 2 mạch cộng bán phần

2.3 Mạch cộng nhị phân 4-bit

Xem xét phép cộng hai số 4-bit như sau:

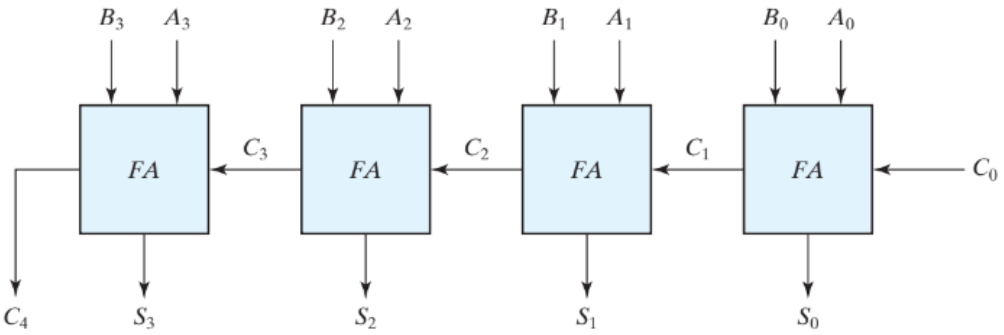
$$\begin{array}{r} \overset{1}{} \\ 1011 \\ +\,1010 \\ \hline 10100 \end{array}$$

Giờ đây, việc cộng hai số nhị phân sẽ có thể mở rộng ra nhiều bit với quy tắc: từng cặp bit tại mỗi vị trí i được cộng với nhau và cộng thêm vào số nhớ tràn từ vị trí $i-1$ sang. Tại hàng đơn vị, bit nhớ C_{in} được gán tín hiệu 0. Còn ở vị trí bit trọng số cao nhất, bit tràn được xử lý tuỳ vào mục đích sử dụng.

Yêu cầu 4. Thực hiện mạch con đặt tên là 4-BIT ADDER thực hiện chức năng cộng hai số 4-bit bằng cách sử dụng mạch con FA đã tạo trong **Yêu cầu 2.** hoặc mạch con FA_1 đã tạo ra trong **Yêu cầu 3.** Cần lưu ý rằng số chân vào/ra của

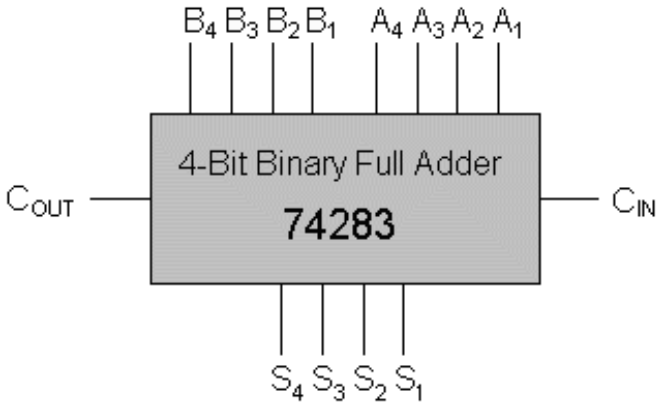
mạch con đã nhiều nên hình dạng bố trí các chân vào/ra rất quan trọng. Một bố trí tốt sẽ giúp cho việc sử dụng mạch con này thuận lợi hơn rất nhiều.

Gợi ý. Tham khảo cách ghép 4 khối FA như hình 2.2.



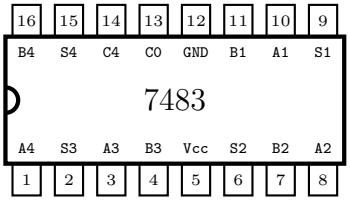
Hình 2.2: Mạch cộng 4-bit được ghép nối từ 4 mạch FA

Tham khảo bố trí chân vào/ra theo cụm từng số 4-bit như hình 2.3.



Hình 2.3: Bố trí chân vào/ra theo từng cụm số

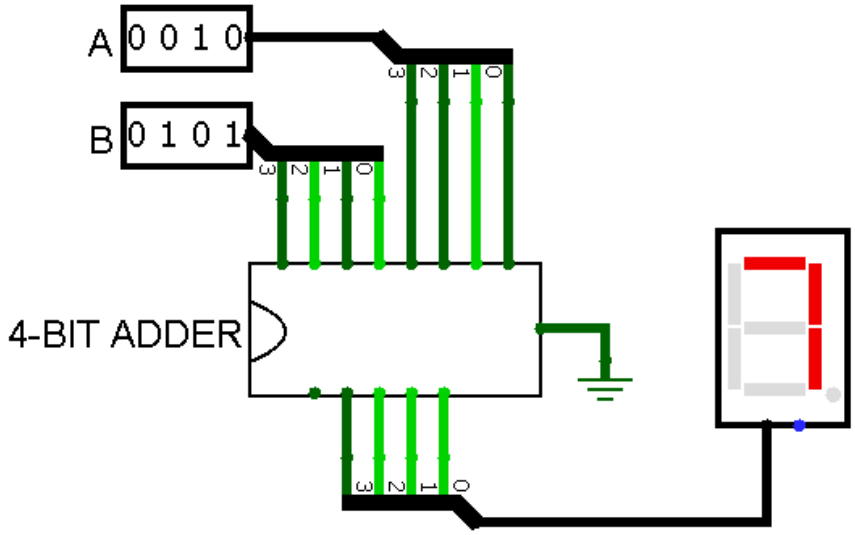
Trên thực tế, IC được đóng gói với bố trí chân như sơ đồ dưới đây.



2.4 Máy tính 4-bit đầu tiên của bạn

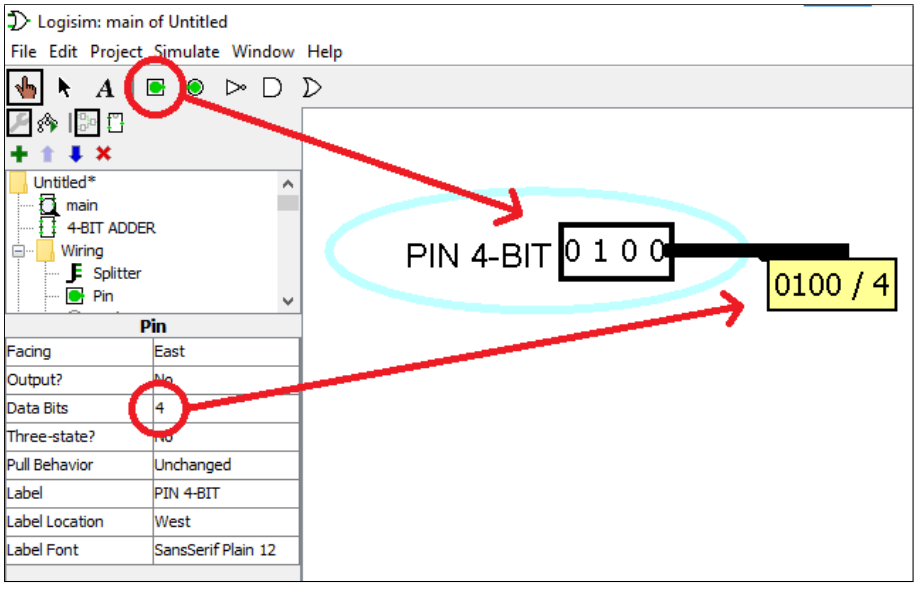
Tại mạch chính, hãy sử dụng IC 4-BIT ADDER, 2 pin 4-bit như là đầu vào của số A và B, đèn led 7 đoạn "Hex led display" và thưởng thức "máy tính" sơ khai có thể thực hiện

phép cộng hai số nguyên không dấu có giá trị lên đến 15. Một số tách/gộp dây "splitter" cần được sử dụng để nối các dây tín hiệu khác độ rộng.



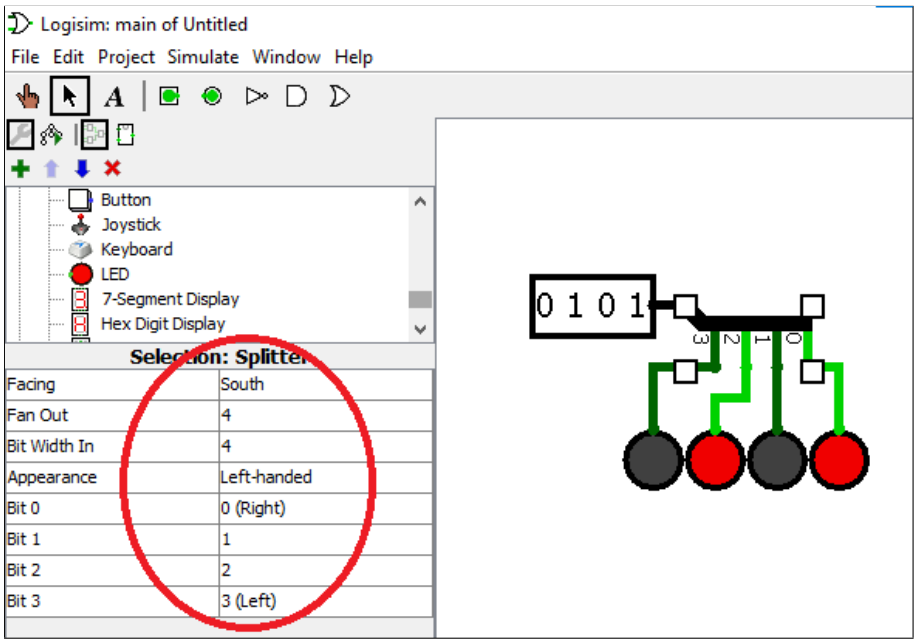
Hình 2.4: Bộ cộng 4-bit đang làm việc với 2 số đầu vào là 2 và 5

2.4.1 Pin n-bit trong Logism



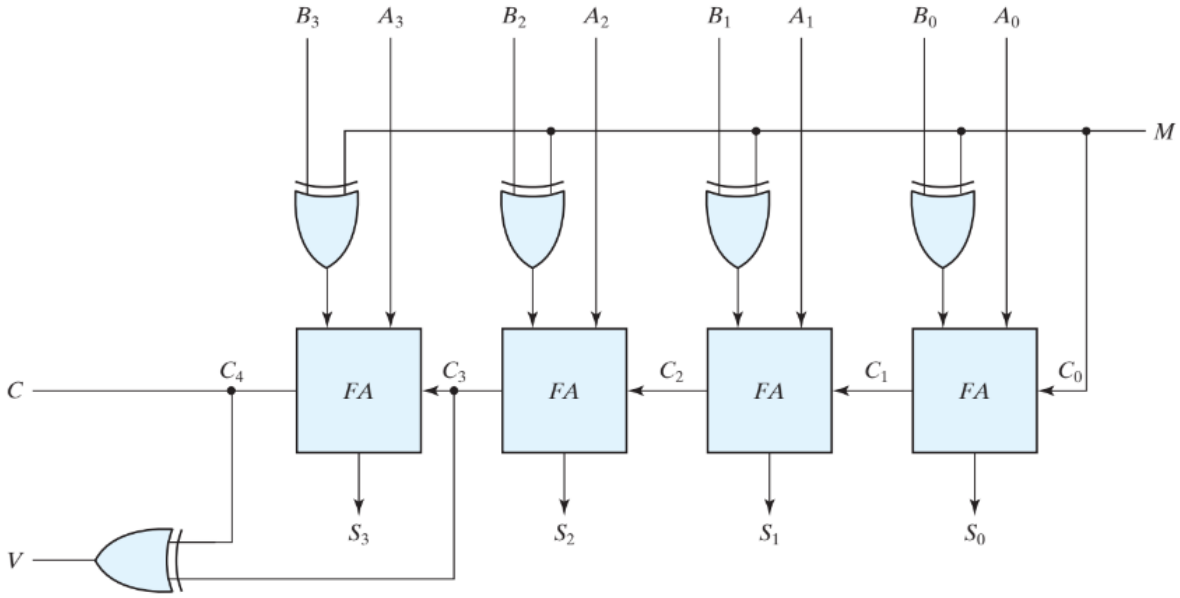
Hình 2.5: Pin đầu vào có độ rộng 4 bit.

2.4.2 Tách / gộp dây tín hiệu



Hình 2.6: Bộ chia/gộp dây tín hiệu 4 bit ra thành 4 dây 1 bit

2.5 Mạch trừ nhị phân 4-bit



Hình 2.7: Mạch trừ nhị phân 4-bit

Yêu cầu 5. Thực hiện mạch trừ 4-bit theo thiết kế ở hình 2.7. Giải thích cách hoạt động của mạch kết quả và vai trò của tín hiệu M và V.

Gợi ý. A - B = A + (-B) và -B được biểu diễn là dưới dạng số bù 2 của B.

2.6 Kết luận

Như vậy, một mạch tổ hợp phức tạp có thể được phân tích thành các mạch nhỏ hơn (thiết kế mức khối) và các mạch nhỏ có thể được hiện thực bằng bảng chân trị hay công thức đại số (thiết kế mức cổng). Tuy vậy, độ trễ khi ghép nối nhiều mạch là vấn đề cần được quan tâm vì chúng sẽ lan truyền và tích luỹ. Các vấn đề sau đây cần được thảo luận.

- Carry Propagation.
- Carry lookahead circuit.
- Overflow dectect.

Bài thực hành LAB 3

MẠCH TỔ HỢP

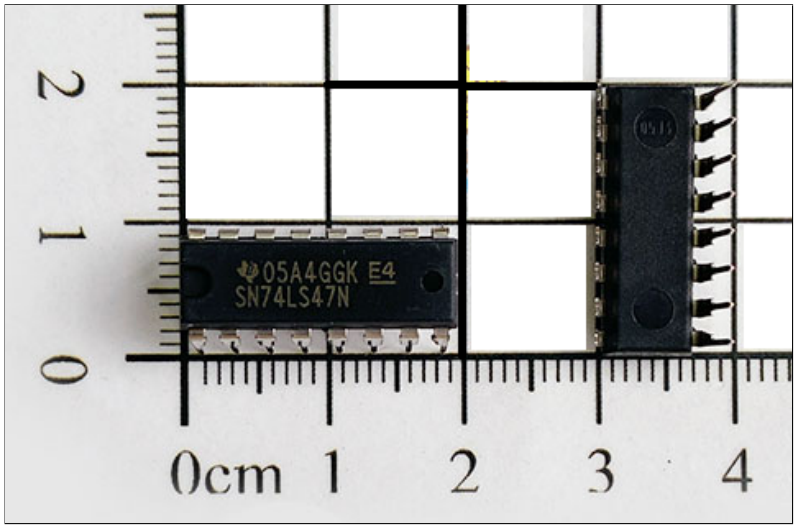
"Số lượng transistor trên mỗi đơn vị inch vuông sẽ tăng lên gấp đôi sau mỗi 18 tháng." - Gordon Earle Moore, Đồng sáng lập tập đoàn Intel.

Máy tính ngoài chức năng tính toán còn có chức năng lưu trữ và truyền tải. Trong lưu trữ, thông tin được biến đổi nhằm mục đích tiện lợi, bảo mật hoặc là nén thông tin. Trong truyền tải, thông tin cũng cần được mã hoá hoặc làm giàu với các bit kiểm lỗi. Hơn nữa, việc điều khiển các mạch điện cũng được tự động hoá bằng các tổ hợp mạch ghép kênh và tách kênh. Tất cả chức năng đó được tiến hành với những mạch tổ hợp và được đóng gói thương mại dưới các họ IC.

3.1 Mạch giải mã BCD-to-LED-7-segment

LED 7 đoạn là một loại LED có cấu tạo khá đơn giản và thường được sử dụng để hiển thị các chữ số và chữ cái ở mức đơn giản. Nó được gọi là "7 đoạn" vì gồm tối thiểu 8 đoạn LED, từ a đến g và một đoạn chấm thập phân, được kết nối với nhau để có thể hiển thị được các ký tự như "0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, b, C, d, E, F,...". Dấu "." được dùng khi có nhiều led 7 đoạn dược dùng để hiển thị một số thực. LED 7 đoạn thường được sử dụng trong các thiết bị hiển thị số như đồng hồ, bảng điện tử, thiết bị đo lường như máy đo nhiệt độ, công suất, áp suất và các ứng dụng điều khiển thời gian, đếm số, hiển thị kết quả, và thông báo lỗi.

Để điều khiển các đoạn sáng và tắt, chúng ta cần giải mã thông tin đầu vào, là một cụm 4-bit biểu diễn số nhị phân hoặc là cụm mã cần hiển thị, đến 7 đầu ra mà trong đó cạnh cần bật sáng sẽ được cấp tín hiệu điện tích cực còn cạnh không sáng sẽ được cấp tín hiệu điện đối lại. Trên thị trường linh kiện, IC 74LS47, CD4511 và CD4543 thường được sử dụng cho mục đích giải mã và điều khiển này.



Hình 3.1: IC 74LS47 với đóng gói DIP

Yêu cầu 1. Hoàn tất bảng sự thật 3.1 rồi dùng nó để thực hiện mạch con đặt tên là BCD-LED-DECODER thực hiện chức năng giải mã một số nhị phân 4-bit $a_3a_2a_1a_0$; và 7 tín hiệu đầu ra từ a đến g để điều khiển 7 đoạn trong đèn LED hiển thị số. Kiểm thử mạch con này sau khi hoàn tất.

a_3	a_3	a_3	a_3	**a**	**b**	**c**	**d**	**e**	**f**	**g**
0	0	0	0	1	1	1	1	1	1	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	1	1	0	0	X	0
1	0	0	0							
1	0	0	1							
1	0	1	0	X	X	X	X	X	X	X
1	0	1	1							
1	1	0	0							
1	1	0	1							
1	1	1	0							
1	1	1	1							

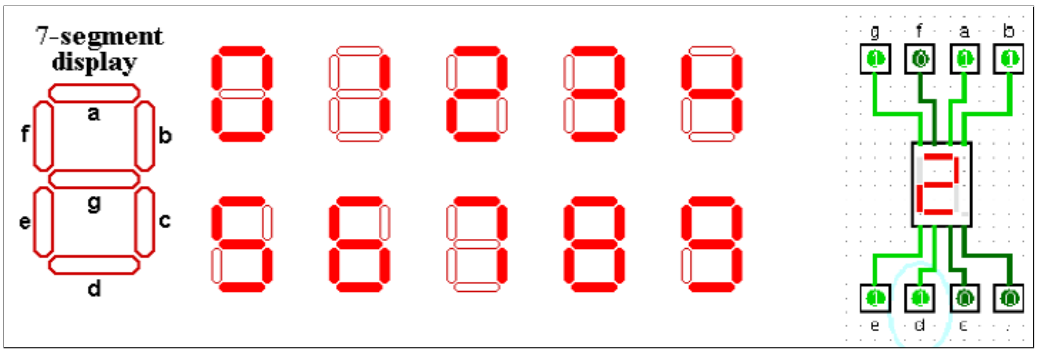
Bảng 3.1: Bảng sự thật mạch giải mã số BCD ra 7 dây điều khiển leg 7 đoạn

Một số tùy chọn khi thực hiện mạch giải mã này là:
- Giải mã BCD: thực hiện cho đầu vào từ 0000 đến 1001, các đầu vào từ

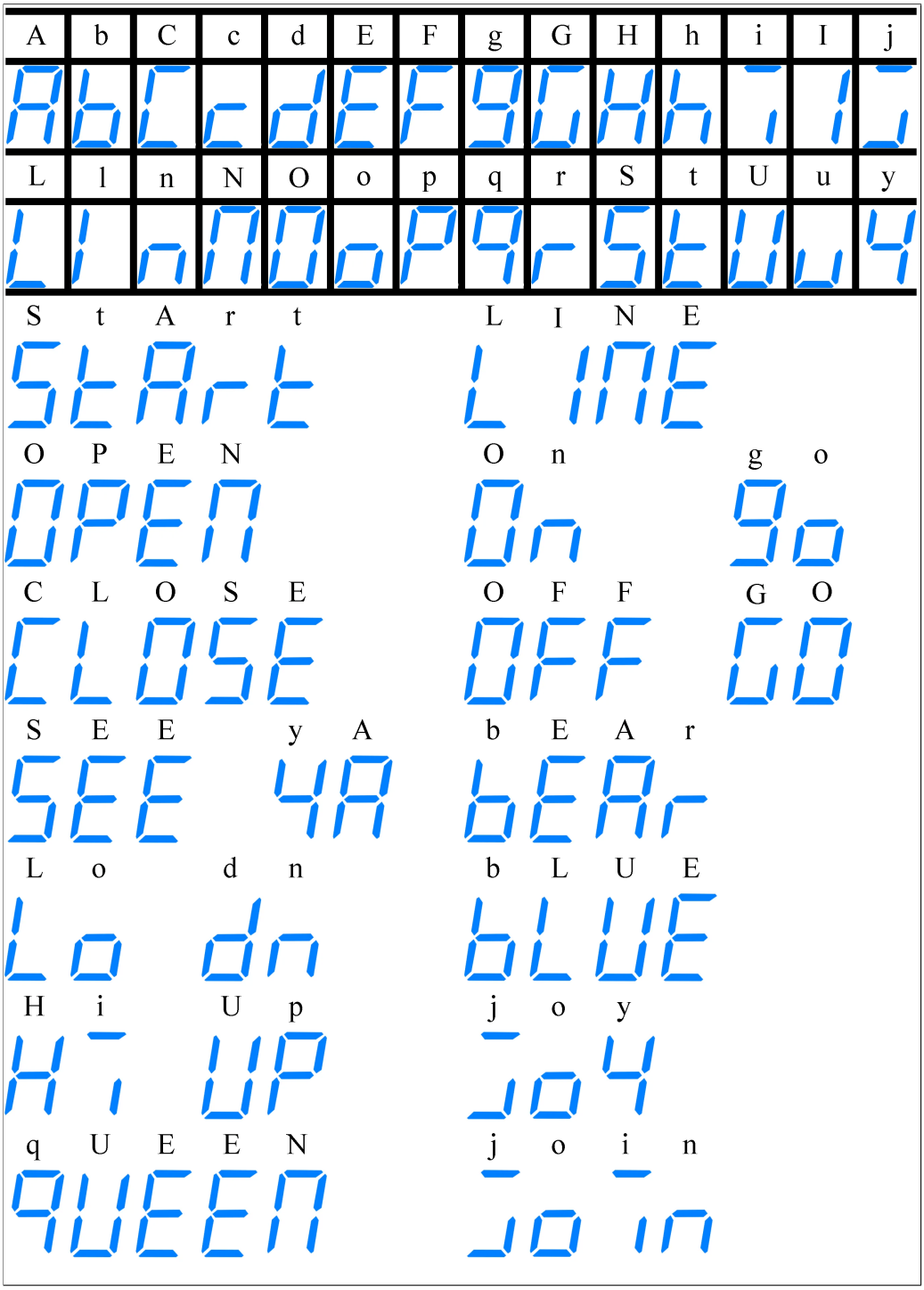
1010 đến 1111 được xem là "don't care". Trong khi đó giải mã nhị phân thực hiện cho đầy đủ 16 giá trị.

- Một vài con số như 6, 7 và 9 có một cạnh đèn dù sáng lên hay không cũng không ảnh hưởng thị giác nên cạnh đó có thể xem là don't care.
- Hiển thị chữ: không chỉ thể hiện số, một vài kí tự vẫn có thể hiển thị như trong hình 3.3. Mặc dù phương pháp này không thể hiển thị một bộ kí tự latin đầy đủ và rõ ràng, nhưng cũng đủ cho các tình huống như: báo lỗi, báo chức năng, một số trạng thái cơ bản của hệ thống máy.

Gợi ý. Hoàn tất bảng sự thật 3.1 bằng cách với mỗi tổ hợp khả hiện, chọn các cạnh sáng lên và điền tín hiệu 1 vào các cột tương ứng các cạnh đó. Sau đó hiện thực mạch với bảng sự thật vừa hoàn thành (tham khảo phục lục A.2 để xem cách thực hiện). Sau khi hoàn tất, kết nối 7 đầu ra của mạch con này vào 7 đầu vào của linh kiện "7-Segment Display" và kiểm thử.



Hình 3.2: Bảng mã hiển thị đèn led 7 đoạn



Hình 3.3: Một bảng mã chữ đơn giản

3.2 Mạch giải mã 3x8 và 4x16

Mã nhị phân của n bit có khả năng biểu thị tối đa 2^n các yếu tố riêng biệt của thông tin được mã hóa. Bộ giải mã là một mạch tổ hợp chuyển đổi thông tin nhị phân từ các dòng đầu vào thành tối đa 2^n dòng đầu ra riêng biệt. Nếu thông tin được mã hóa n bit có các kết hợp không được sử dụng thì bộ giải mã có thể có ít hơn 2^n đầu ra.

Yêu cầu 2. Dựa vào bảng sự thật 3.4 để thực hiện mạch con đặt tên là 3-8 DECODER thực hiện chức năng giải mã đầu vào bộ nhị phân 3-bit xyz đến 8 đầu ra riêng biệt từ D_0 đến D_7. Kiểm thử mạch con sau hoàn tất.

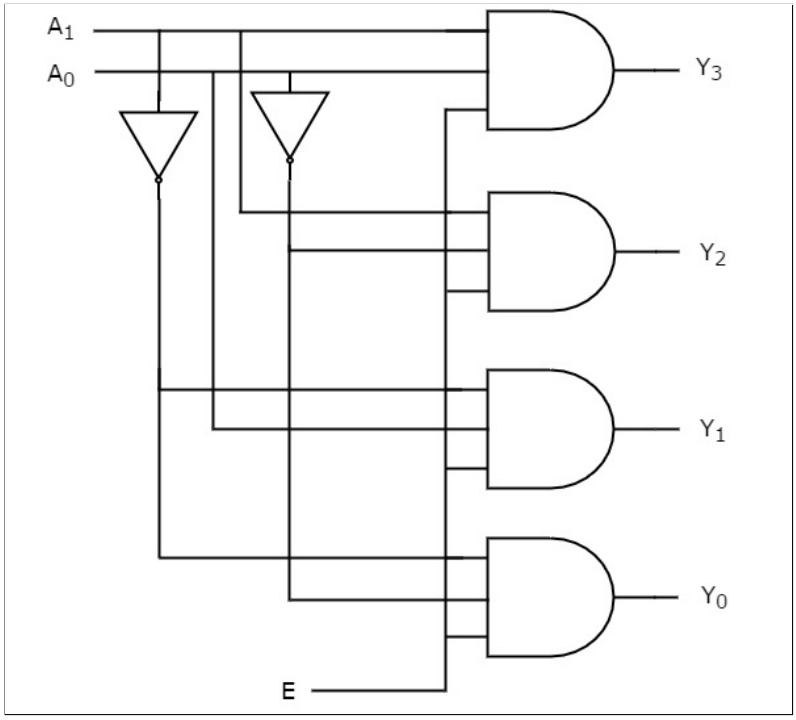
Inputs			Outputs							
x	**y**	**z**	**D_0**	**D_1**	**D_2**	**D_3**	**D_4**	**D_5**	**D_6**	**D_7**
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Hình 3.4: Bảng sự thật của bộ giải mã 3 đến 8

Yêu cầu 3. Bổ sung chân EN vào 3-8 DECODER để cho phép hoặc vô hiệu mạch giải mã.

Gợi ý.

Tham khảo cách ghép nối chân EN với từng đầu ra. Khi EN=0, tất cả đầu ra đều là 0. Khi EN=1, các đầu ra vận hành như chức năng đã định nghĩa.

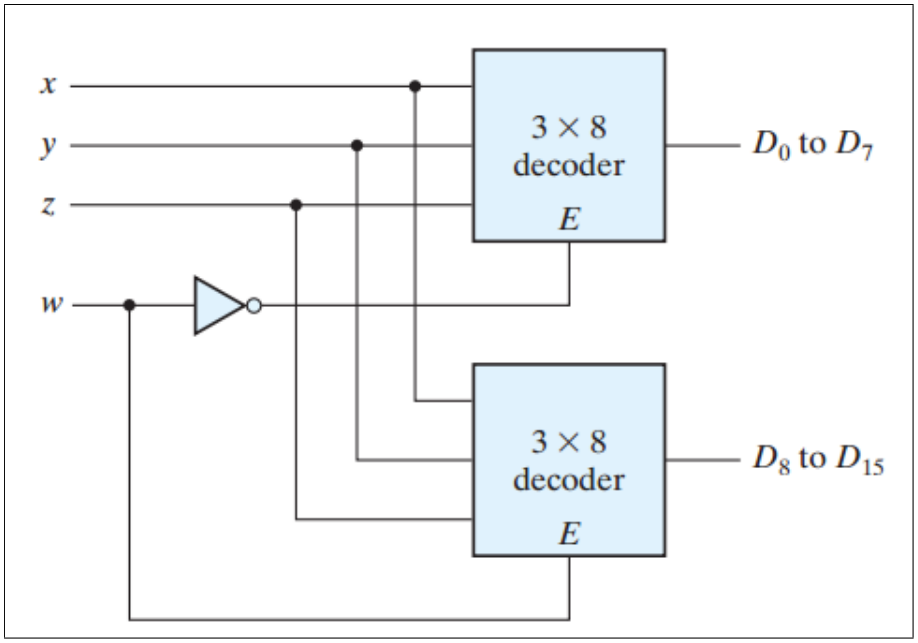


Hình 3.5: Cách ghép nối cổng EN với từng đầu ra

.

Yêu cầu 4. Thực hiện mạch con đặt tên là 4-16 DECODER thực hiện chức năng giải mã đầu vào bộ nhị phân 3-bit $wxyz$ đến 16 đầu ra riêng biệt từ D_0 đến D_{15}. Kiểm thử mạch con sau hoàn tất. Thực hiện theo 2 phương pháp: từ bảng chân trị và từ ghép nối mạch giải mã 3x8 có chân EN.

Gợi ý.
- Thiết kế từ bảng chân trị: thực hiện như **Yêu cầu 3.** với đầu vào là 4 bit $wxyz$.
- Thiết kế từ ghép nối mạch giải mã 3x8: tham khảo hình 3.6



Hình 3.6: Bảng sự thật của bộ giải mã 3 đến 8

.

3.2.1 Ứng dụng: Thực hiện mạch tổ hợp

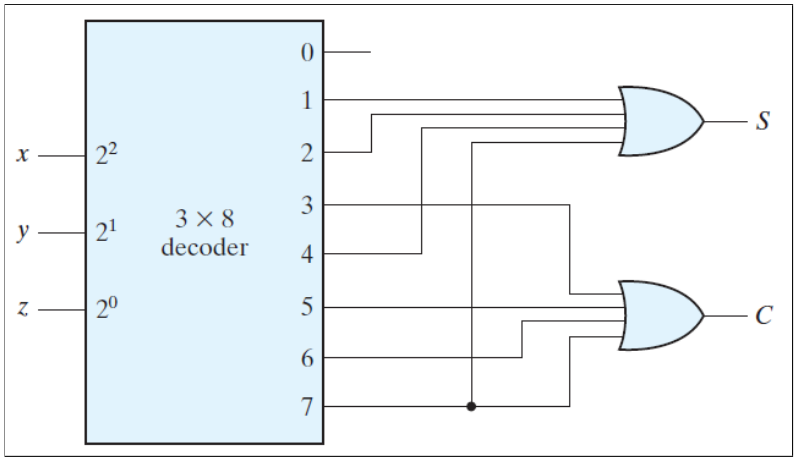
Bộ giải mã cung cấp 2^n minterm khi có n biến đầu vào. Mỗi đầu ra tích cực của bộ giải mã được liên kết với một tổ hợp bit đầu vào duy nhất. Vì bất kỳ hàm Boole nào cũng có thể được biểu diễn dưới dạng tổng các minterm, nên một bộ giải mã tạo ra các minterm của hàm này cùng với cổng OR ở cuối cùng tạo thành tổng logic của chúng. Theo cách này, bất kỳ mạch tổ hợp nào có n đầu vào và m đầu ra có thể được thực hiện bằng bộ giải mã $n - to - 2^n$ và m cổng OR. Quy trình thực hiện mạch tổ hợp bằng bộ giải mã và cổng OR yêu cầu hàm Boole của mạch được biểu thị dưới dạng tổng của minterms. Sau đó, một bộ giải mã được chọn để tạo ra tất cả các minterm của các biến đầu vào. Đầu vào của mỗi cổng OR được chọn từ đầu ra của bộ giải mã theo danh sách các minterm của từng hàm.

Yêu cầu 5. Sử dụng một mạch giải mã 3-8 DECODER và một số cổng OR cần thiết để xây dựng mạch cộng toàn phần FA.

Gợi ý.
Từ bảng sự thật 2.2 của mạch cộng toàn phần, chúng ta có hàm S và C được viết với dạng sum-of-minterms như sau:

$$S(x, y, z) = \sum(1, 2, 4, 7)$$
$$C(x, y, z) = \sum(3, 5, 6, 7)$$

Từ đó, chúng ta có thể mắc nối từ các đầu ra của mạch giải mã được liệt kê trong mỗi công thức đến một cổng OR và trở thành một cổng ra của mạch cần thiết.



Hình 3.7: Mạch cộng toàn phần hiện thực từ mạch giải mã 3x8

.

3.3 Mạch mã hoá 8x3

Một bộ mã hóa có 2^n (hoặc ít hơn) đầu vào và n đầu ra. Các tín hiệu ở đầu ra tạo thành các tổ hợp mã nhị phân tương ứng với mỗi tín hiệu ở đầu vào. Một ví dụ là bộ mã hóa bát phân sang nhị phân (8x3 Encoder) có bảng sự thật được cho tại hình 3.8. Nó có 8 đầu vào (một tín hiệu đại diện cho một kí số bát phân) và 3 đầu ra với các tín hiệu tổ hợp thành mã nhị phân tương ứng. Giả định rằng chỉ có một đầu vào được tích cực tại một thời điểm.

Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

Hình 3.8: Mạch mã hóa 8 đến 3

Bộ mã hóa 8x3 có thể hiện thực bằng các cổng OR với các công thức ở các đầu ra như sau:

$$z = D_1 + D_3 + D_5 + D_7$$
$$y = D_2 + D_3 + D_6 + D_7$$
$$x = D_4 + D_5 + D_6 + D_7$$

Yêu cầu 6. Thực hiện mạch giải mã cho một bàn phím số như hình 3.9, các phím 0 đến 9 khi được nhấn sẽ sinh mã nhị phân tương ứng từ 0000 đến 1001. Dấu '#' sinh mã 1010 và dấu '*' sinh mã 1111. Dùng linh kiện "Splitter" gom 4 đầu ra thành dây tín hiệu 4-bit rồi kết nối vào một "Hex display" để kiểm thực mạch đã thực hiện.

Hình 3.9: Một bàn phím số

3.3.1 Nút bấm ưu tiên

Ngược lại với bộ mã hóa đơn giản, nếu hai hoặc nhiều đầu vào của bộ mã hóa ưu tiên hoạt động cùng lúc thì đầu vào có mức ưu tiên cao nhất sẽ được ưu tiên. Đây là một cải tiến trên một bộ mã hóa đơn giản vì nó có thể xử lý tất cả các kết hợp đầu vào có thể có nhưng phải bổ sung một số cổng logic.

Inputs				Outputs		
D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

Hình 3.10: Bảng sự thật của bộ mã hoá ưu tiên

.

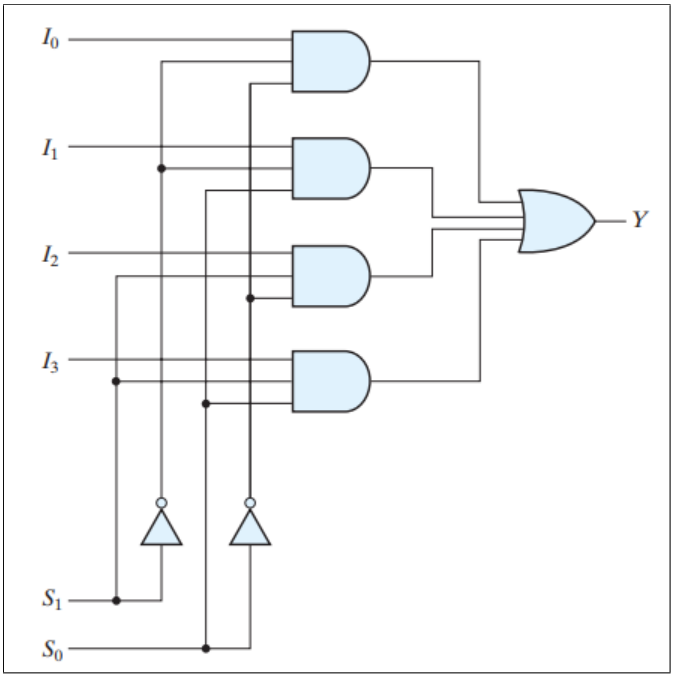
$$x = D_2 + D_3$$
$$y = D_3 + D_1.D_2'$$
$$V = D_0 + D_1 + D_2 + D_3$$

3.4 Mạch trộn kênh MUX 4 đến 1

Bộ ghép kênh (Multiplexer hay MUX) là một mạch tổ hợp chọn thông tin nhị phân từ một trong nhiều dòng đầu vào và hướng nó đến một dòng đầu ra duy nhất. Việc lựa chọn một dòng đầu vào cụ thể được điều khiển bởi một tập hợp các dòng lựa chọn. Thông thường, có 2^n dòng đầu vào và n dòng lựa chọn mà sự kết hợp bit của chúng xác định đầu vào nào được chọn.

S_1	S_0	**Y**
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Bảng 3.2: Bảng sự thật mạch cộng bán phần a+b



Hình 3.11: Bộ trộn kênh 4 đầu vào đến 1 đầu ra

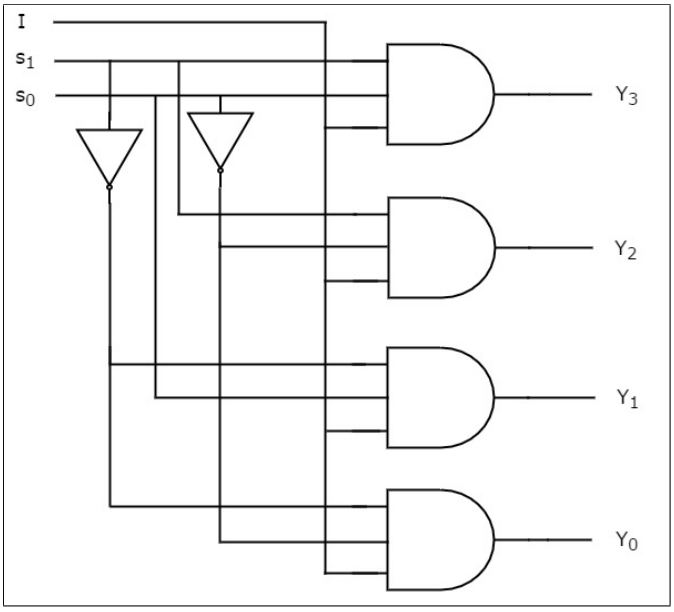
.

Yêu cầu 7. Thực hiện mạch trộn kênh 4-1 MUX với 4 đầu vào từ I_0 đến I_3 cùng 2 chân điều khiển chọn S_1 và S_0. Tham khảo hình 3.11 để thực hiện mạch con này.

Yêu cầu 8. Bổ sung chân EN để cho phép hay vô hiệu mạch trộn kênh 4-1 MUX ở **Yêu cầu 7.** Có thể ghép nối 2 mạch 4-1 MUX thành 1 mạch 8-1 MUX không?

3.5 Mạch tách kênh DeMUX 1 đến 4

Mạch tách kênh (De-Multiplexer hay DeMUX) là một mạch tổ hợp thực hiện hoạt động ngược lại của bộ gộp kênh. Nó có một đầu vào, n dây lựa chọn và tối đa 2^n đầu ra. Đầu vào sẽ được kết nối với một trong các đầu ra dựa trên giá trị của các dây lựa chọn.



Hình 3.12: Bộ tách kênh 1 đầu vào đến 4 đầu ra

.

Yêu cầu 9. Thực hiện mạch tách kênh 1-4 DeMUX với 4 đầu ra từ I_0 đến I_3 cùng 2 chân điều khiển chọn S_1 và S_0. Tham khảo hình 3.12 để thực hiện mạch con này.

S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	I
0	1	0	0	I	0
1	0	0	I	0	0
1	1	I	0	0	I

Bảng 3.3: Bảng sự thật mạch tách kênh 4 đến 1

3.6 Kết luận

Trong bài LAB này, chúng ta đã hiện thực một số chức năng cơ bản của hệ thống số, việc thiết kế ghép nối đã thể hiện ý nghĩa của hệ thống số trong cả nền tảng lý thuyết lẫn hiện thực khi không thể xây dựng mạch từng phương pháp bảng sự thật với số đầu vào rất lớn.

Phụ lục A

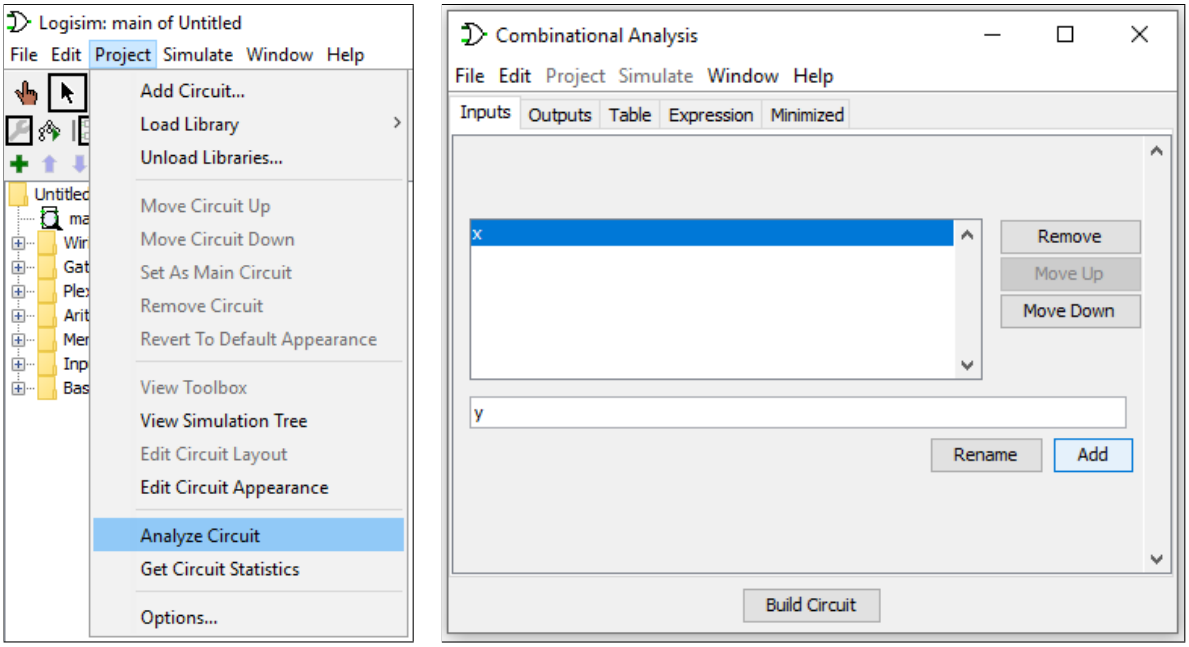
Ứng dụng Logism

A.1 Vẽ mạch logic từ biểu thức Boole

Bước 1: Từ màn hình làm việc, chọn **Project** \ **Analyze Circuit**.

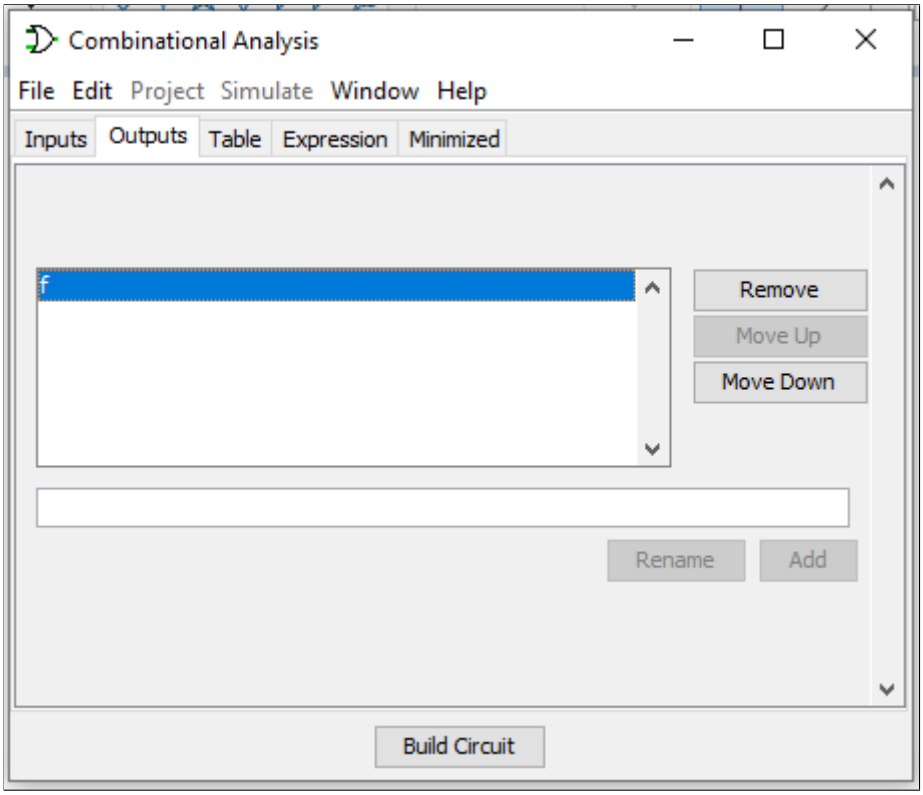
Bước 2: Nhập các biến số vào (và sắp xếp thứ tự nếu cần thiết) tại thẻ **Inputs** như hình A.1.

Bước 3: Tương tự nhập các đầu ra tại thẻ **Outputs** như hình A.2.



(a) Lựa chọn "Phân tích mạch" (b) Cửa sổ "Phân tích mạch" với thẻ "Đầu vào"

Hình A.1: Giao diện chức năng Phân tích mạch của Logism

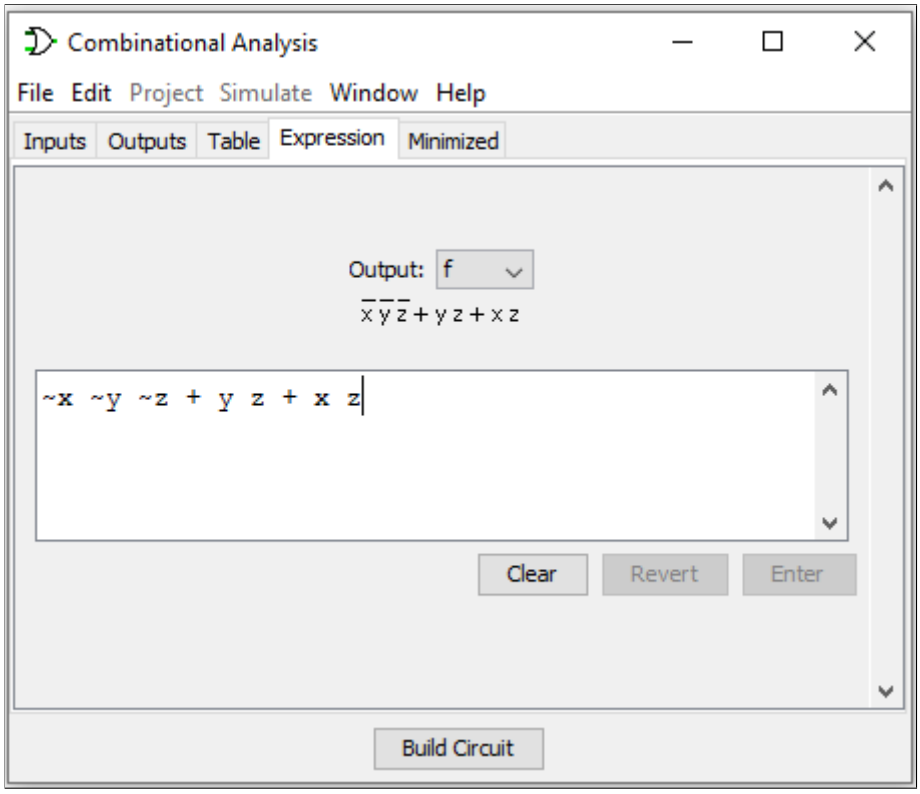


Hình A.2: Chức năng "Phân tích mạch logic"

Bước 4: Chuyển đến thẻ **Expression** và nhập biểu thức cho từng đầu ra như hình A.3 với cách viết các kí hiệu được quy ước tại bảng A.1.

Biểu thức	**Cách viết trong Logism**	**Ghi chú**
0 / 1	0 / 1	Hằng số false / true
NOT X	∼ X hoặc !X hoặc X’	Phủ định X’
X AND Y	X Y hoặc X & Y	Khoảng trắng giữa biến số
X XOR Y	X ^ Y	Phép toán XOR
X OR Y	X + Y hoặc X | Y	Phép toán HOẶC
Ví dụ 1	X ∼Y + ∼X Y	X’.Y + X.Y’
Ví dụ 2	(X ^ Y) (∼Y Z + 1)	(X ⊕ Y). (Y.Z’ + 1)
Ví dụ 3	a’ (b + c) !a & (b | c) NOT a AND (b OR c)	a’.(b+c)

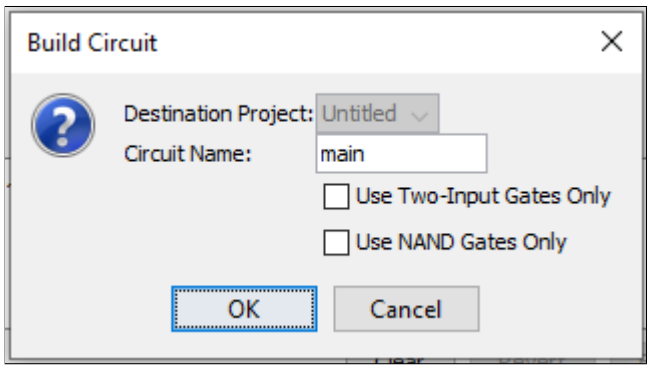
Bảng A.1: Cách viết biểu thức trong Logism



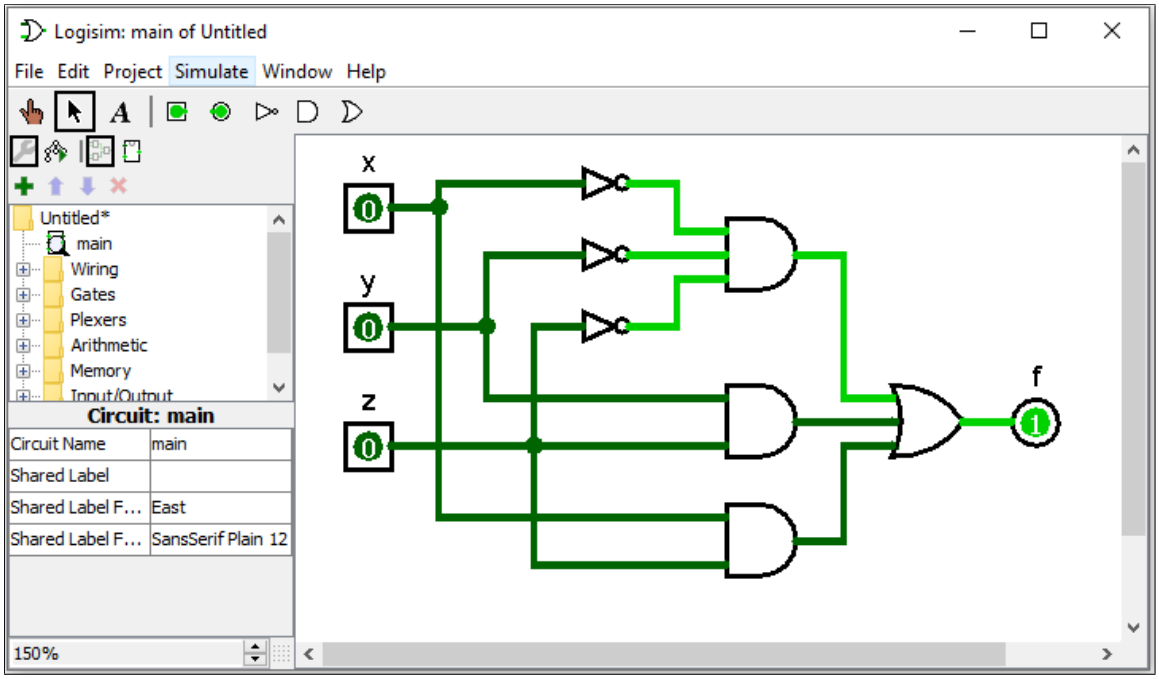
Hình A.3: Chức năng "Nhập vào biểu thức cho từng đầu ra"

Bước 5: Click chuột vào nút **Build Circuit**, tại cửa sổ **Build Circuit** như hình A.4, chúng ta có thể chọn lựa *"Chỉ sử dụng cổng có 2 đầu vào"* và/hoặc *"Chỉ sử dụng cổng đa dụng NAND"*.

Bước 6: Mạch kết quả sẽ được thể hiện trong cửa sổ làm việc chính như hình A.5.



Hình A.4: Mạch logic được vẽ từ biểu thức được nhập vào



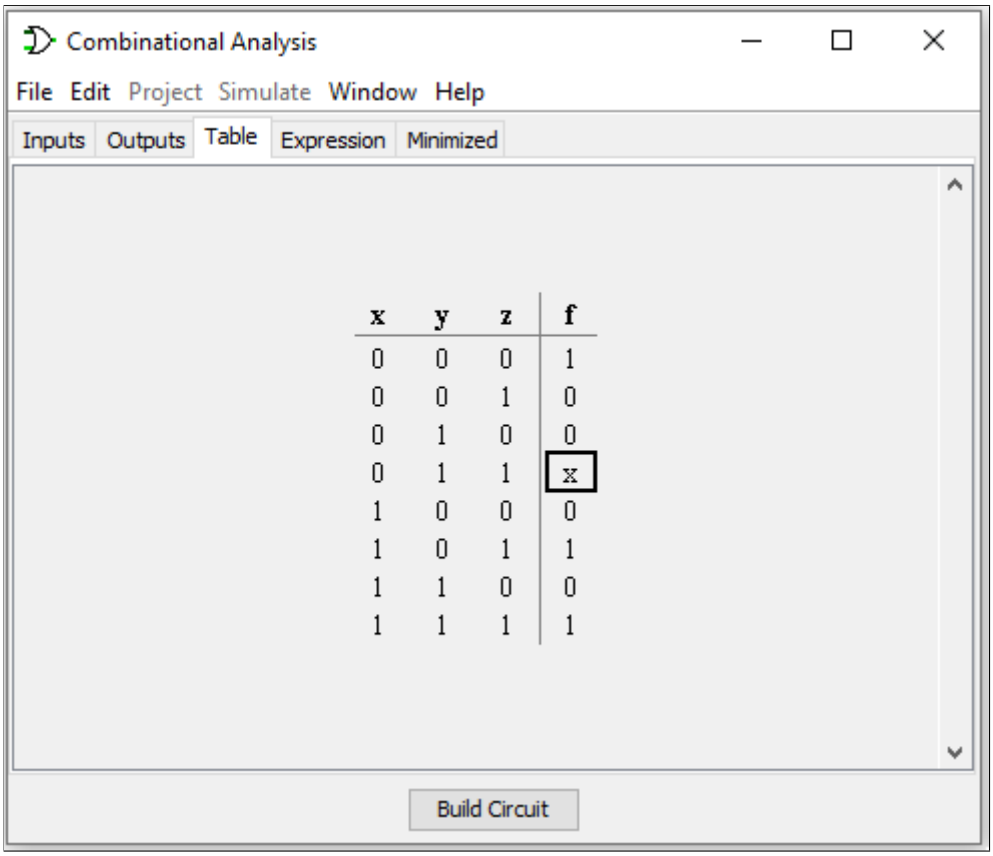
Hình A.5: Chức năng "Nhập vào biểu thức cho từng đầu ra"

A.2 Vẽ mạch logic từ bảng sự thật

Bước 1: Thực hiện như Bước 1 đến Bước 3 của phụ lục A.1.

Bước 2: Chuyển đến thẻ **Table** và click chuột lên từng bit của cột đầu ra để xác định giá trị mong muốn như hình A.6. Trong kí hiệu: 0 = false; 1 = true; X = don't care.

Bước 3: Click chuột vào nút **Build Circuit** để có kết quả mạch logic trong cửa sổ làm việc chính như hình A.5.



Hình A.6: Mạch logic được vẽ từ biểu thức được nhập vào

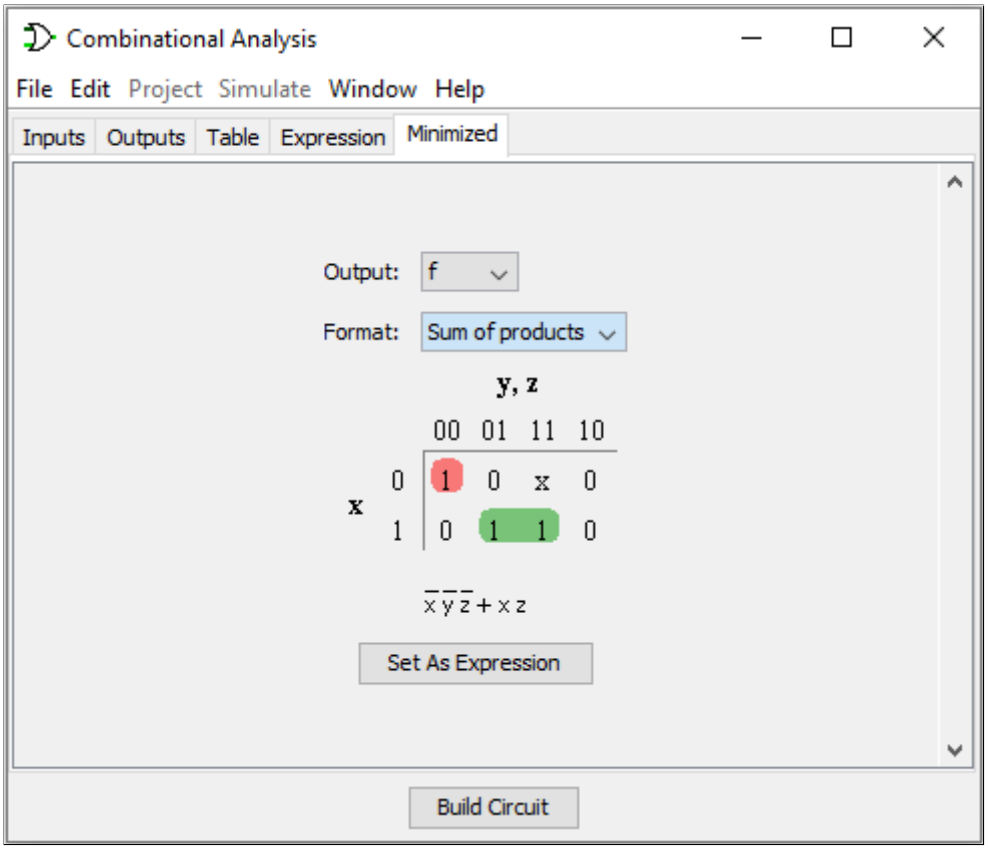
A.3 Rút gọn mạch logic và Bìa-K

Bước 1: Thực hiện như Bước 1 đến Bước 3 của phụ lục A.1.

Bước 2: Chuyển đến thẻ **Minimized** và chọn đầu ra cùng với mục tiêu rút gọn là SOP hay POS.

Bước 3: Tùy chọn **Set As Expression** sẽ chọn biểu thức rút gọn thay thế vào biểu thức ban đầu.

Bước 4: Bìa-K được thể hiện như hình A.7.



Hình A.7: Bìa-K cùng với các cụm được chọn

A.4 Tìm biểu thức từ mạch logic

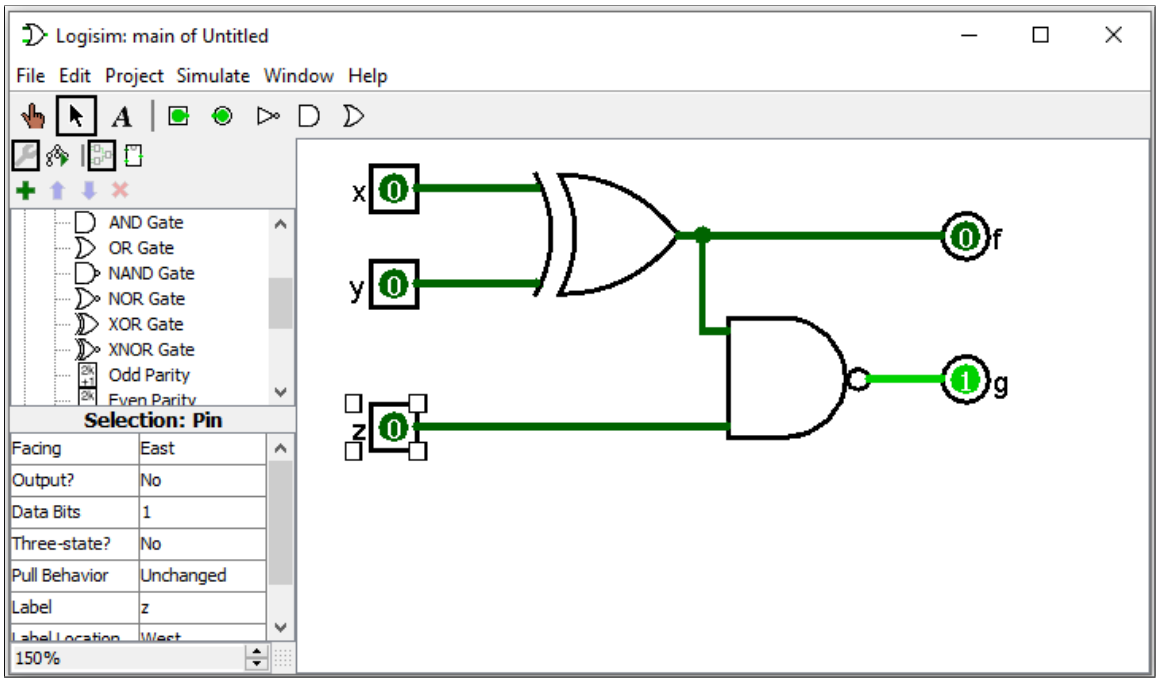
Bước 1: Tại màn hình làm việc chính, bắt đầu vẽ mạch logic cần phân tích như hình A.8.

Bước 2: Mỗi đầu vào được định nghĩa bằng một Pin input, được thêm vào bằng nút hình vuông **Add pin** (nút thứ 4 trên thanh công cụ Toolbar). Có thể thêm nhanh bằng tổ hợp phím **Ctrl + 4**.

Bước 3: Mỗi đầu ra được định nghĩa bằng một Pin output, được thêm vào bằng nút hình tròn **Add pin** (nút thứ 5 trên thanh công cụ Toolbar). Có thể thêm nhanh bằng tổ hợp phím **Ctrl + 5**.

Bước 4: Đặt các cổng cần thiết vào và kết nối dây dẫn. Tham khảo thêm Hướng dẫn của Logism cho các thao tác này.

Bước 5: Khi hoàn thành, chọn **Project \ Analyze Circuit**. Bảng sự thật thể hiện ở thẻ **Table**; Biểu thức được thể hiện ở thẻ **Expression**.



Hình A.8: Mạch logic được ghép nối từ các cổng định nghĩa sẵn