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

THỰC HÀNH MẠCH SỐ

"Máy tính ra đời để giải quyết những vấn đề trước đây chưa từng tồn tại" - Bill Gates, người sáng lập Microsoft.

Trong đại số trừu tượng, đại số Boole làm việc với các đại lượng chỉ nhận giá trị Đúng hoặc Sai và có thể thể hiện hệ thống số nhị phân, hoặc các mức điện thế trong mạch điện logic. Do đó đại số Boole có nhiều ứng dụng trong kỹ thuật điện và khoa học máy tính, cũng như trong logic toán học.

Để hiện thực trên nền tảng đại số Boole, các mạch số vận hành với hai mức điện thế cao và thấp cùng với các cổng cơ bản. Các tín hiệu đi qua các cổng này và đến ngõ ra để biểu diễn cho kết quả mong đợi. Các mạch số này được sản xuất và tích hợp lại thành một linh kiện[1] và được dùng như một thành phần của một mạch số khác có chức năng phức tạp hơn.

1.1 Biểu diễn hàm Boole bằng mạch logic

Yêu cầu 1. Thực hiện các hàm Boole sau đây thành mạch điện:

$F_{(x,y,z)} = (x \text{ OR } y) \text{ AND } (\text{NOT } z)$

$F_{(a,b,c)} = a.b + a.b' + b'.c$

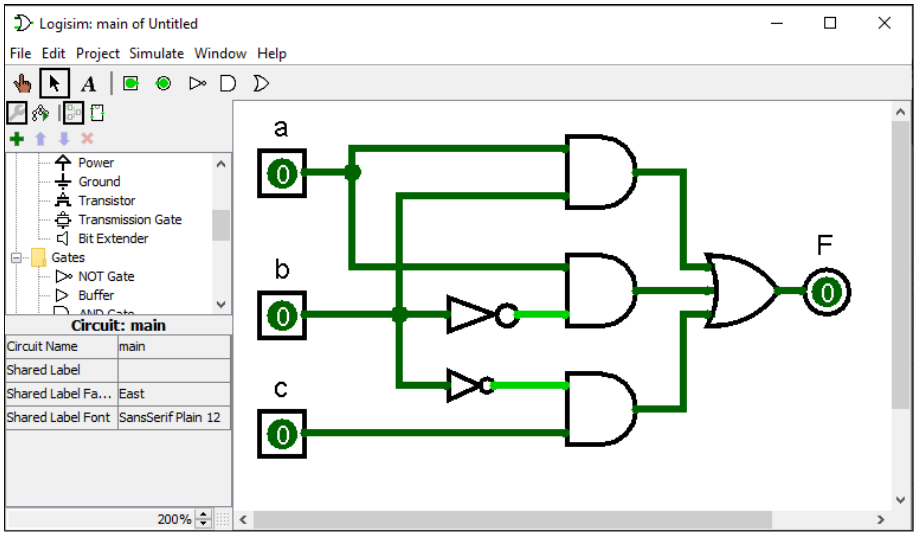
Gợi ý. Sử dụng các linh kiện cơ bản trên thanh tác vụ để hoàn tất mạch.

- **Pin input** là đầu vào cho mạch logic. Khi được thiết lập kích thước 1-bit, cổng tương đương với một biến số trong công thức đại số Boole. Khi được thiết lập n-bit với $n > 1$ thì cổng là ngõ vào của cụm n-bit. Mỗi bit của đầu vào có thể thiết lập giá trị HIGH, LOW và FLOATING lần lượt tương ứng với logic 1, 0 và thả nổi.
- **Pin output** là đầu ra của mạch logic, tương đương với giá trị của hàm số Boole khi cổng có thiết lập 1-bit. Đầu ra này cũng có thể tăng độ rộng lên $n - bit$ để biểu

[1] Thường gọi là IC, viết tắt từ Integrated Circuit

diễn kết quả nhiều bit. Các đầu vào, đầu ra đều có thể đặt tên (Label) trong phần thuộc tính của từng linh kiện.

- **Cổng logic** AND, OR và NOT là 3 cổng cơ bản tương ứng với 3 phép toán của đại số Boole. Mỗi cổng đều có thể thiết lập số chân vào, cũng như hướng đặt linh kiện và thuộc tính điện của các đầu vào và đầu ra.
- **Dây nối** kết nối dữ liệu giữa hai đầu nối của hai hoặc nhiều linh kiện. Tất cả đầu nối tham gia vào một dây dẫn cần tương đồng về độ rộng.

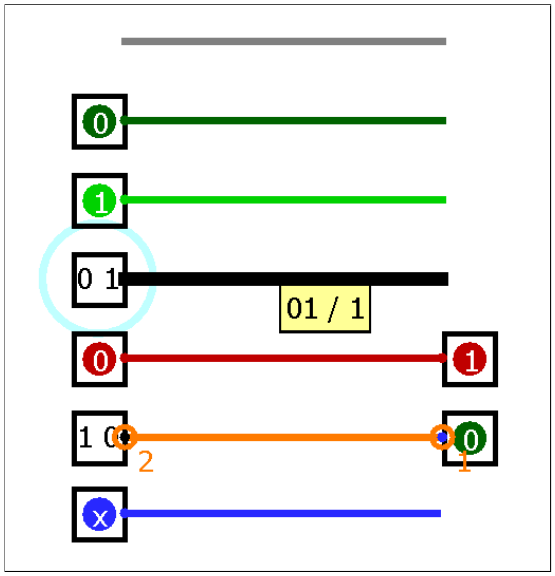


Hình 1.1: Mạch logic được xây dựng trên Logisim

1.1.1 Vận hành giả lập mạch logic

Mạch điện có thể vận hành giả lập khi được kích hoạt tại chọn lựa **Simulate** → **Simulation Enable**. Các dây dẫn sẽ có màu sắc như hình 1.2. Trong đó:

- màu xám là không có kết nối;
- màu xanh lục tối/sáng thể hiện tín hiệu 0/1;
- màu đen đậm là dây tín hiệu n-bit;
- màu cam là không đồng nhất độ rộng bit giữa các đầu kết nối;
- màu đỏ là có lỗi mạch.



Hình 1.2: Các thể hiện của dây dẫn trong Logisim

1.1.2 Kiểm thử mạch con

Với một mạch điện đã hoàn thành và đã kích hoạt giả lập, các đầu ra sẽ thể hiện giá trị tùy thuộc vào bộ tín hiệu đầu vào. Để thay đổi các tổ hợp tín hiệu đầu vào khác, chúng ta sử dụng công cụ **Toolbar** → **Poker tool** (hoặc nhấn Ctrl+1). Mỗi lần chọc vào một đầu vào, trạng thái của đầu vào sẽ thay đổi. Sau khi đã kiểm thử tất cả tổ hợp có thể có của các đầu vào và đầu ra đều cho kết quả mong muốn, chúng ta có thể kết luận rằng mạch logic đã hiện thực hoàn hảo.

Yêu cầu 2. Xem xét hàm Boole sau đây:

$F_{(x,y,z)} = (x \text{ OR } y) \text{ AND } (\text{NOT } z)$

(a) Với tổ hợp giá trị (x=true; y=false; z=true) thì đầu ra F có giá trị nào?

(b) Có bao nhiêu tổ hợp giá trị (x,y,z) khác nhau?

Gợi ý. (a) - Trên lý thuyết, một hàm đại số Boole có thể được lượng giá khi chúng ta thay toàn bộ các biến số thành giá trị nhị phân 0 hoặc 1. Các cụm nhiều biến số được lượng giá bởi định nghĩa của phép toán NOT, AND và OR.

- Trên thực hành, một mạch điện trong tình trạng chạy giả lập sẽ mang trạng thái được xác lập bởi các đầu vào và cấu trúc mạch logic. Sử dụng công cụ **Poker** nhấp lên các đầu vào x, y và z để chuyển giá trị chúng thành 1, 0 và 1 tương ứng theo thứ tự. Giá trị đầu ra F sẽ xác lập sau đó.

(b) Mỗi đầu vào đơn (là một biến số) sẽ có giá trị là 0 hay 1. Vậy n đầu vào đơn sẽ có 2^n tổ hợp khả dĩ.

1.1.3 Kiểm thử tự động

Khi số tổ hợp đầu vào có số lượng lớn, việc kiểm thử từng bộ tổ hợp sẽ tốn nhiều thời gian và công sức. Chương trình Logisim không cung cấp công cụ kiểm thử tự động, tuy nhiên có một vài phương pháp thay thế:

- Sử dụng dự án CENG232 Logisim TestBench được cung cấp tại `https://github.com/ozansz/logisim-testbench`.

- Sử dụng Bộ hiện thực và giả lập trực tuyến tại `https://circuitverse.org/`.

Vấn đề kiểm thử tự động sẽ được trình bày trong các chương sau.

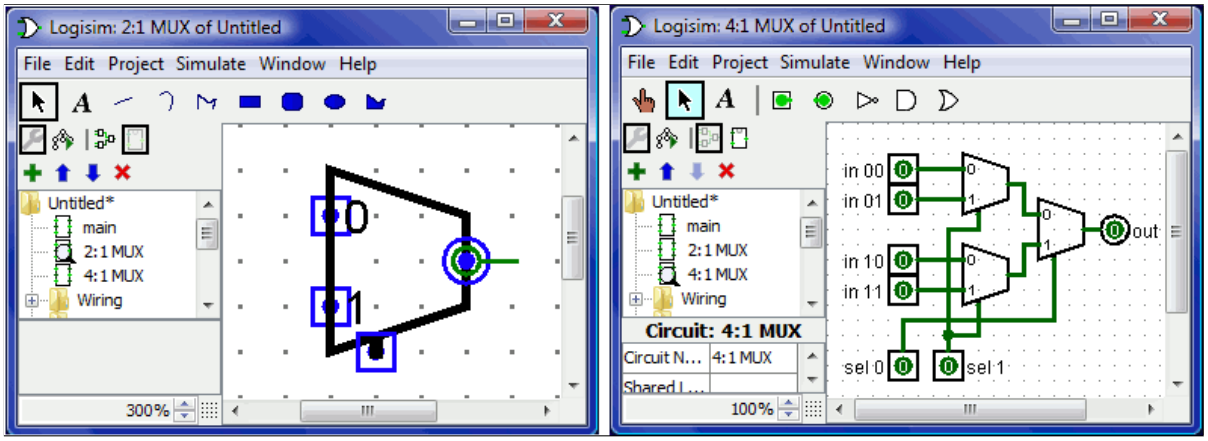
1.2 Thực hiện mạch con

Khi xây dựng các mạch ngày càng phức tạp hơn, chúng ta cần xây dựng các mạch nhỏ hơn mà có thể sử dụng nhiều lần dưới dạng mô-đun được lồng trong các mạch lớn. Trong Logisim, mạch nhỏ hơn được sử dụng trong mạch lớn hơn được gọi là mạch con (sub circuit).

Điều này tương tự như tạo ra một hàm con và gọi nó nhiều lần trong hàm main, hoặc các hàm con khác. Việc mô-đun giúp cho mạch điện đơn giản, dễ thiết kế và gỡ lỗi, cũng như thể hiện ưu điểm nổi trội của mạch số là khả năng mở rộng và thiết kế hệ thống. Có 2 phần quan trọng: hiện thực mạch con và đóng gói mạch con.

- **Hiện thực mạch con** Một mạch con cần xác định số đầu vào và số đầu ra cũng với chức năng của nó.
  - Nếu chức năng của mạch con được biểu diễn bởi một hàm số Boole, tham khảo **Phụ lục A.1** để hiện thực mạch.
  - Nếu chức năng của mạch con được xác định bằng bảng sự thật, tham khảo **Phụ lục A.2** để hiện thực mạch.
  - Hoặc mạch con có thể hiện thực bằng cách sử dụng các cổng và linh kiện trong thư viện của Logisim.

- **Đóng gói mạch con** Các mạch con khi được sử dụng trong một mạch khác với một hình dạng nhất định, thông thường là một hình chữ nhật mà xung quanh các các chân điện vào và ra, một rãnh khuyết được vẽ để tránh nhầm lẫn khi linh kiện được xoay hướng. Tuy vậy, người thiết kết có thể bố trí lại các chân vào ra và hình dạng của mạch con bằng cách chọn lựa **Project** → **Edit Circuit Appearance**, rồi bố trí các chân và vẽ các hình dạng cần thiết.



Hình 1.3: Hình dạng của mạch con (trái) và thể hiện của chúng ở mạch khác (phải)

Yêu cầu 3. (a) Thực hiện mạch con IC_1, IC_2, IC_3 lần lượt theo thứ tự cho mỗi hàm Boole sau đây:

$$F_{(x,y,z)} = (x \text{ OR } y) \text{ AND } (\text{NOT } z)$$
$$F_{(x,y,z)} = (x \text{ OR } y) \text{ AND } (\text{NOT } z)$$
$$F_{(x,y,z)} = (x \text{ OR } y) \text{ AND } (\text{NOT } z)$$

(b) Thực hiện hàm con IC_4 có 3 đầu vào (x, y, z) và một đầu ra là XOR 3 tín hiệu của 3 mạch con đã thực hiện tại yêu cầu (a).

1.3 Phân tích mạch logic

- Rút gọn mạch logic: tham khảo **Phụ lục A.3**.
- Tìm biểu thức từ mạch logic: tham khảo **Phụ lục A.4**.

1.4 Kết luận

Chương trình Logisim đã được giới thiệu với một số chức năng đầu tiên và các cổng cơ bản. Bằng việc ghép nối các cổng thành một mạch con, ghép nối các mạch con thành một mạch con lớn hơn, chúng ta có thể hiện thực ý tưởng "Trừu tượng hóa" của máy tính điện tử. Các mô tả của quá trình phân tích, hiện thực và ghép nối được trình bày trong các chương tiếp theo.

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 2.1. 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 2.1.

2.2 Mạch cộng toàn phần

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

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

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

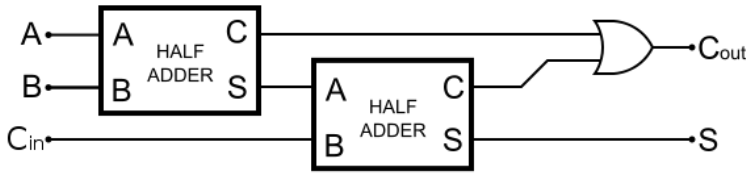
Yêu cầu 2. Hoàn tất bảng sự thật 2.2 rồi dùng nó để thực hiện mạch con đặt tên là FA thực hiện chức năng cộng toàn phần với 3 tín hiệu nhập là 2 bit a, b với bit tràn vào C_{in}; và 2 tín hiệu đầu ra là S với C_{out}. Kiểm thử mạch con này sau khi hoàn tất.

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:

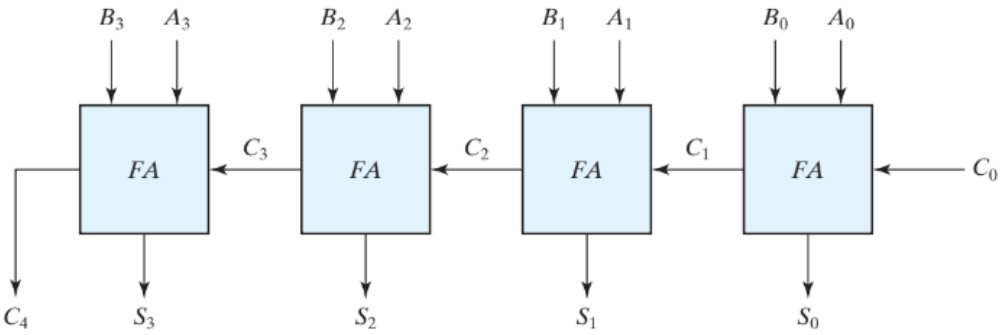
$$\begin{array}{r} \overset{1}{} \\ 1011 \\ +\,1010 \\ \hline 10100 \end{array}$$

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

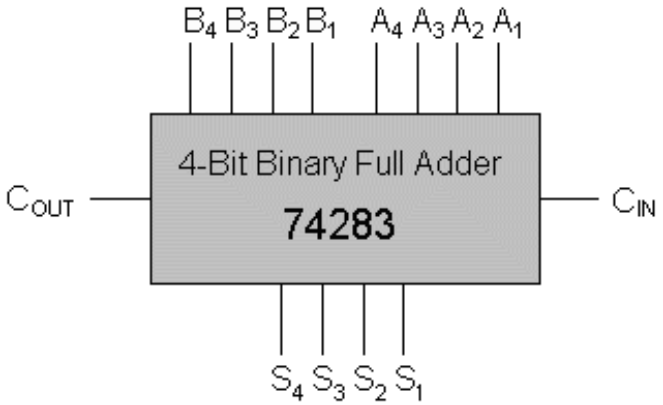
mạch con đã nhiều nên hình dạng bố trí các chân vào/ra rất quan trọng. Một bố trí tốt sẽ giúp cho việc sử dụng mạch con này thuận lợi hơn rất nhiều.

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



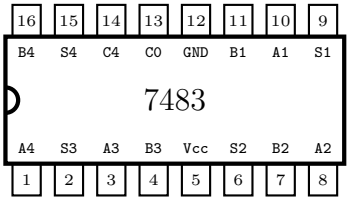
Hình 2.2: Mạch cộng 4-bit được ghép nối từ 4 mạch FA

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

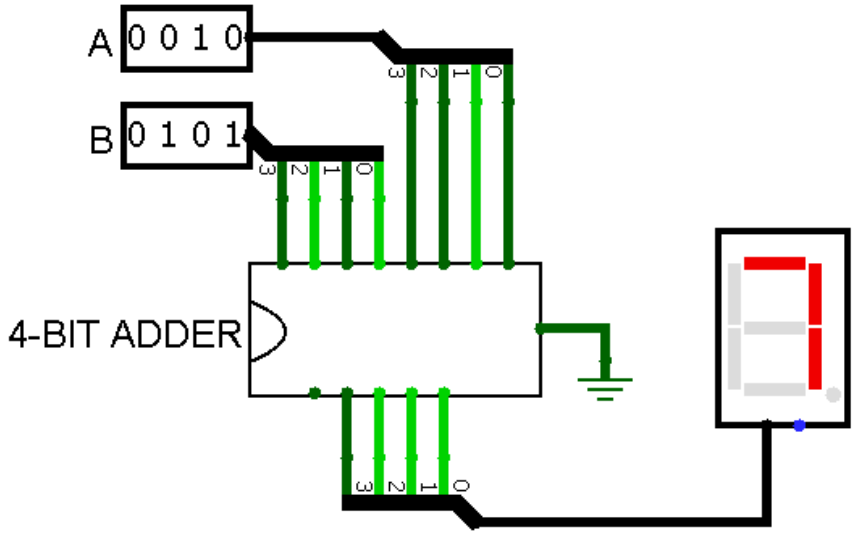
Trên thực tế, IC được đóng gói với bố trí chân như sơ đồ dưới đây.



2.4 Máy tính 4-bit đầu tiên của bạn

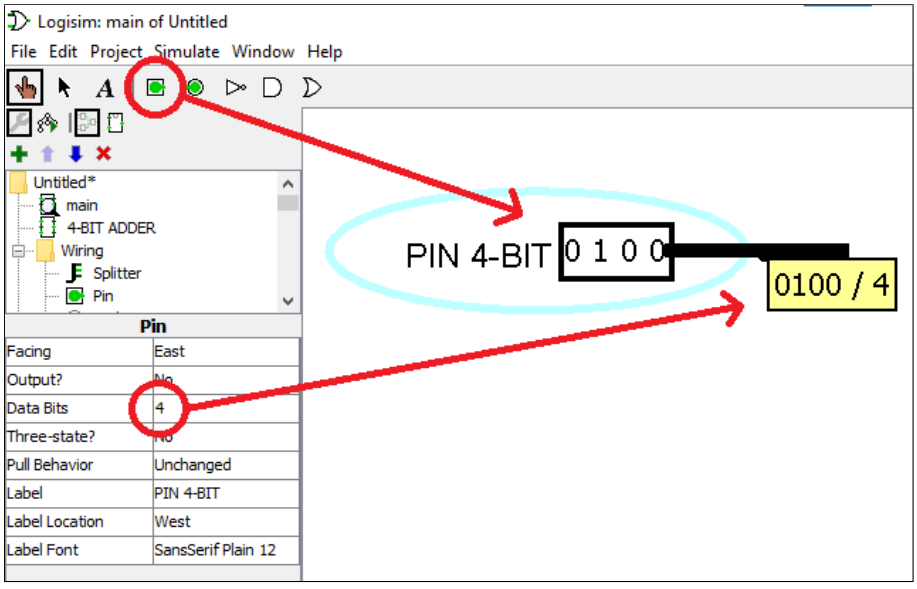
Tại mạch chính, hãy sử dụng IC 4-BIT ADDER, 2 pin 4-bit như là đầu vào của số A và B, đèn led 7 đoạn "Hex led display" và thưởng thức "máy tính" sơ khai có thể thực hiện

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.



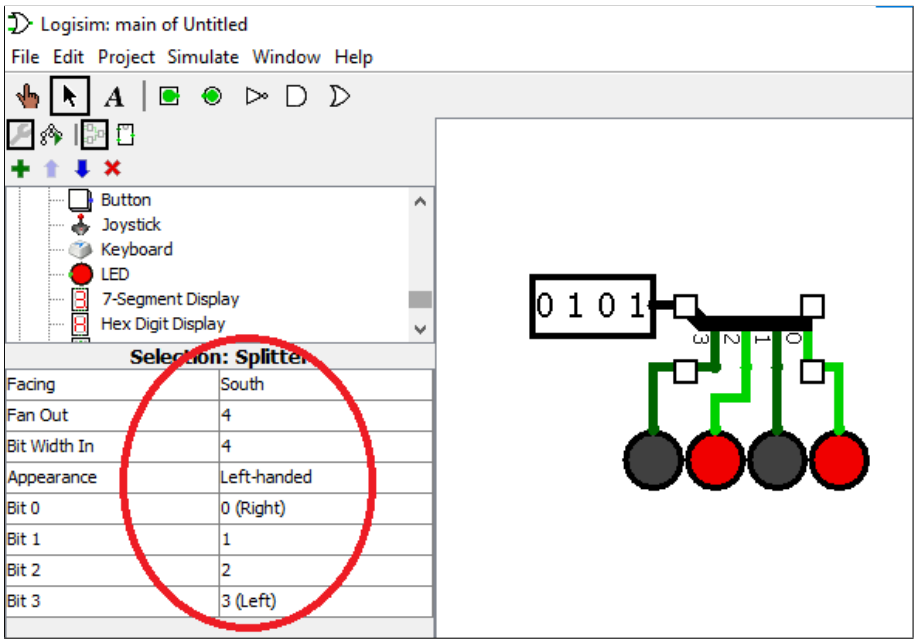
Hình 2.4: Bộ cộng 4-bit đang làm việc với 2 số đầu vào là 2 và 5

2.4.1 Pin n-bit trong Logism



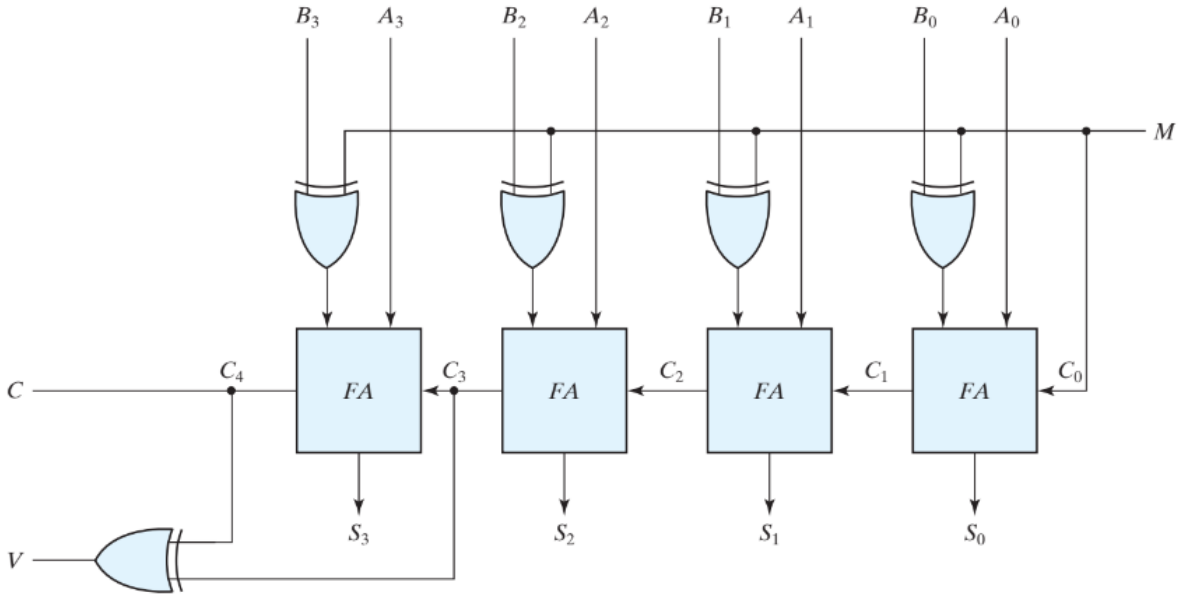
Hình 2.5: Pin đầu vào có độ rộng 4 bit.

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.

Phụ lục A

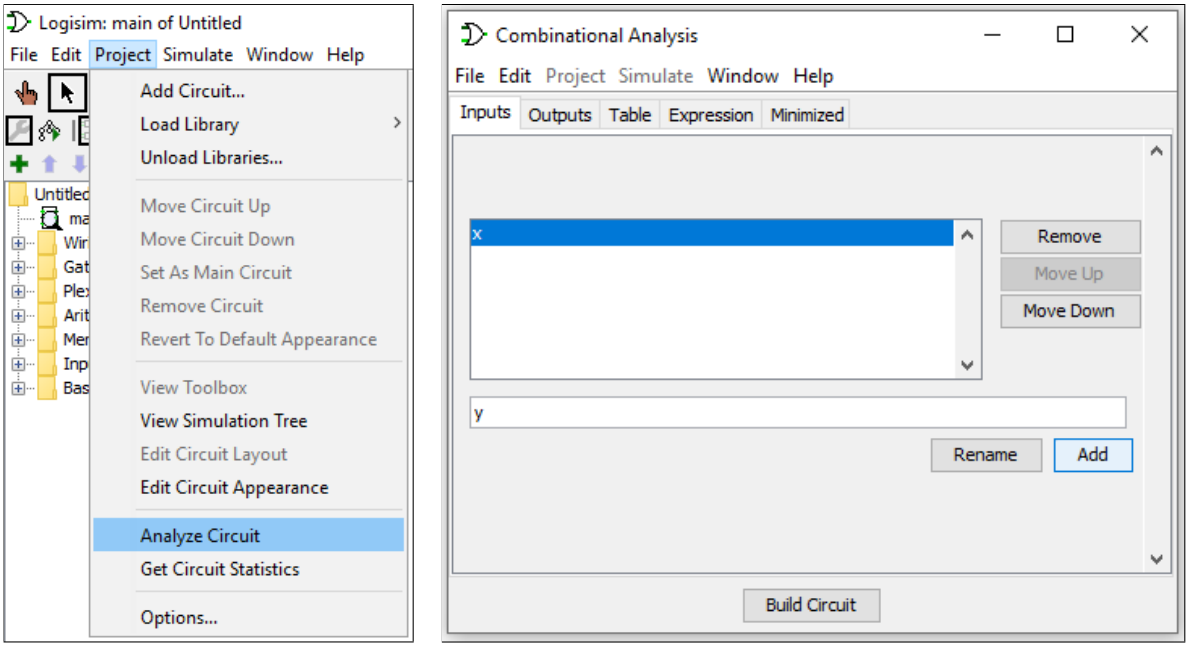
Ứng dụng Logisim

A.1 Vẽ mạch logic từ biểu thức Boole

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

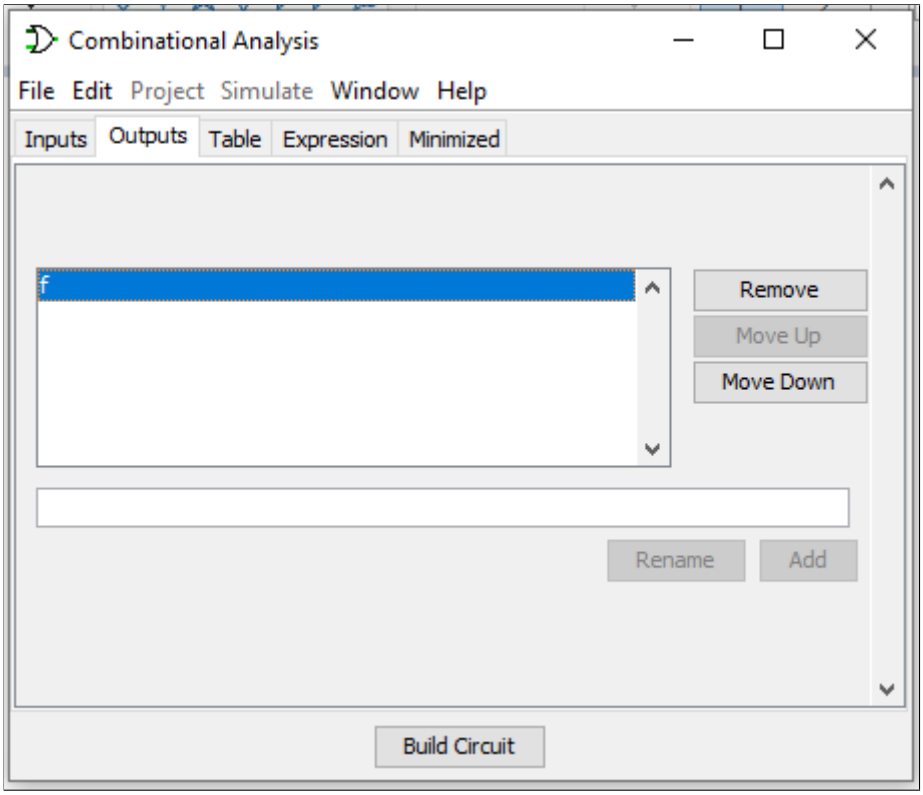
Bước 2: Nhập các biến số vào (và sắp xếp thứ tự nếu cần thiết) tại thẻ **Inputs** như hình A.1.

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 Logisim

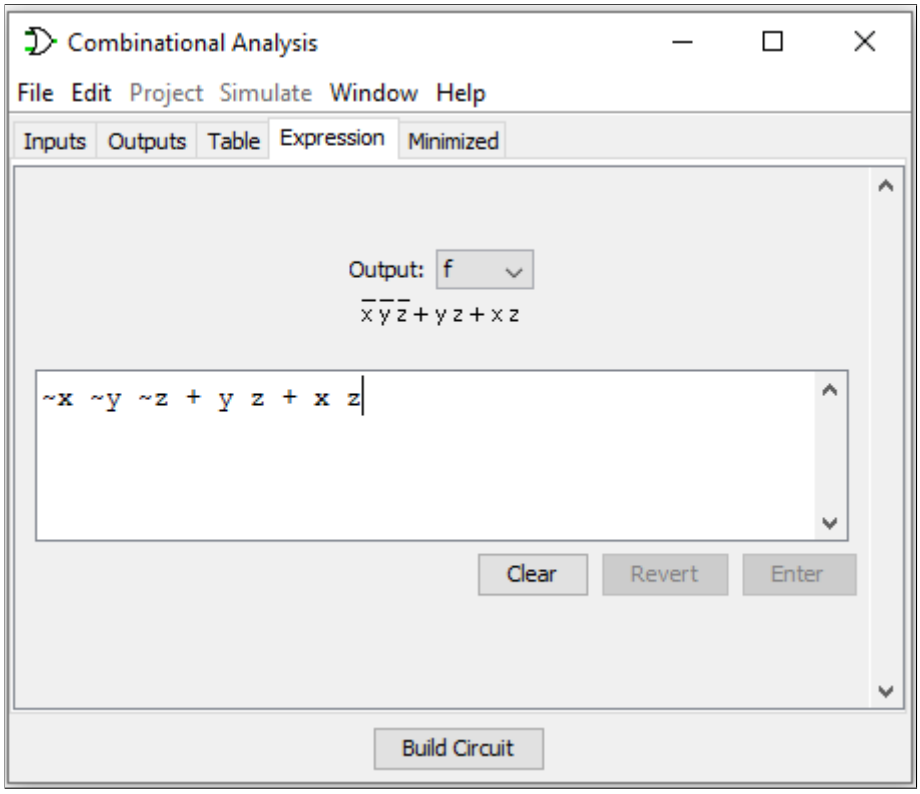


Hình A.2: Chức năng "Phân tích mạch logic"

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 Logisim**	**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)

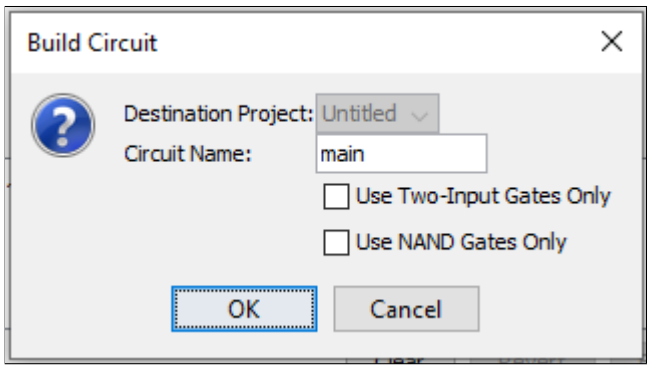
Bảng A.1: Cách viết biểu thức trong Logisim



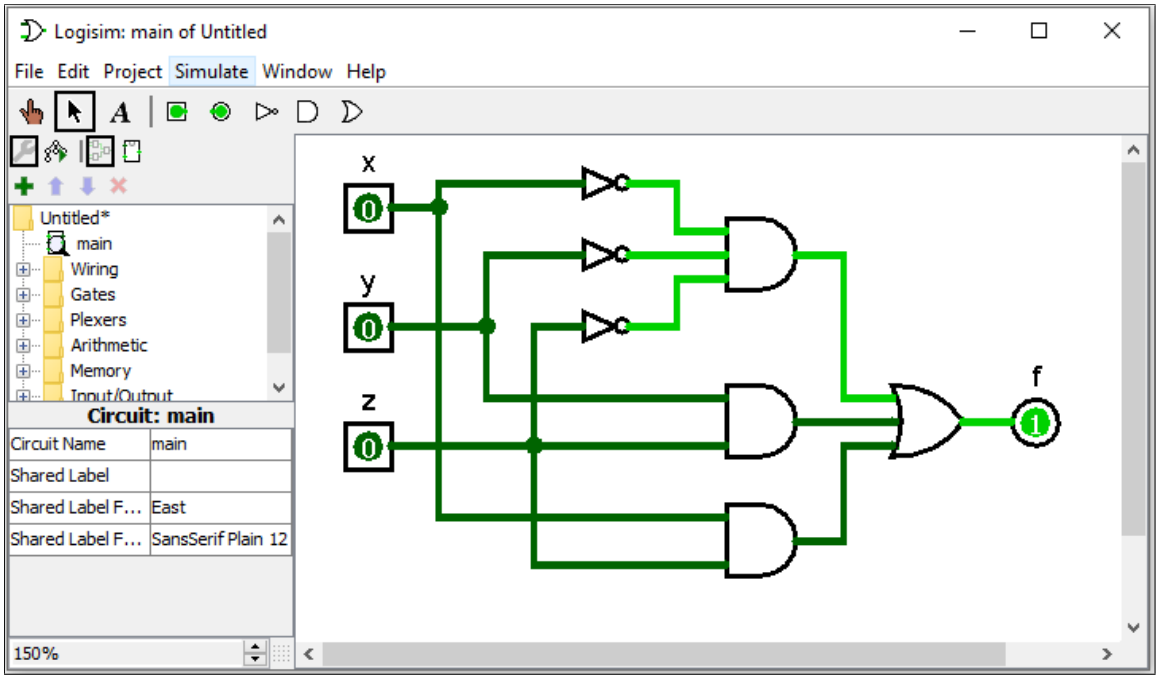
Hình A.3: Chức năng "Nhập vào biểu thức cho từng đầu ra"

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



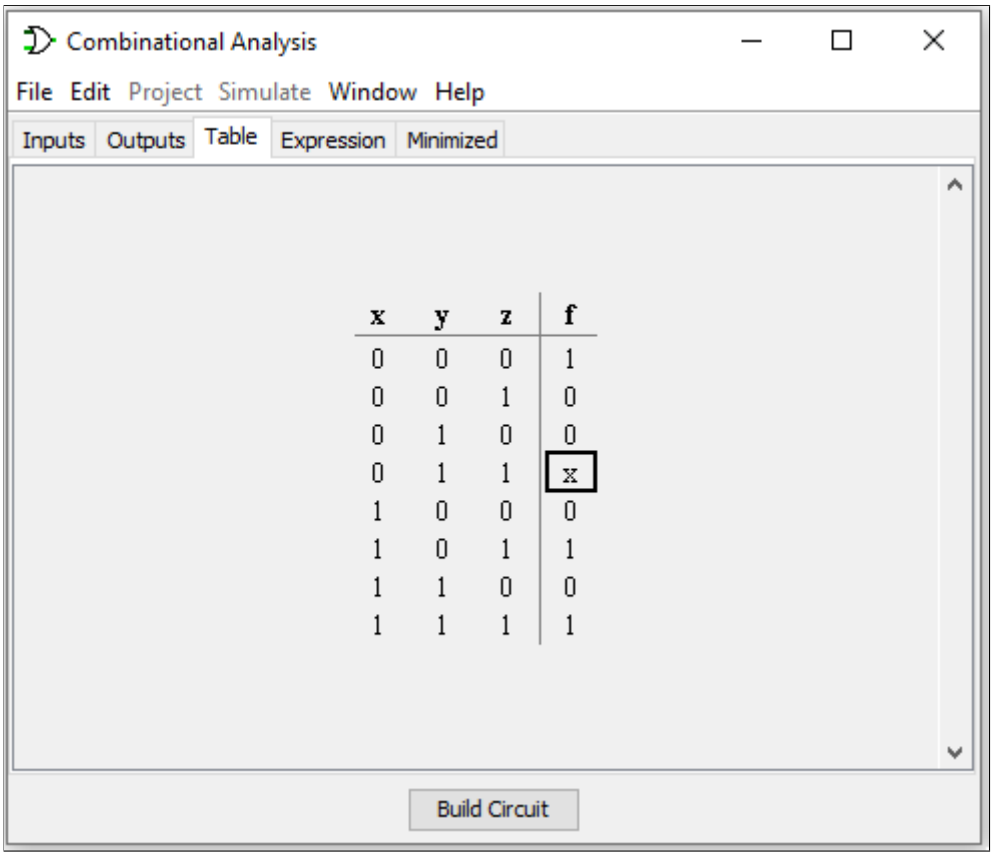
Hình A.5: Chức năng "Nhập vào biểu thức cho từng đầu ra"

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

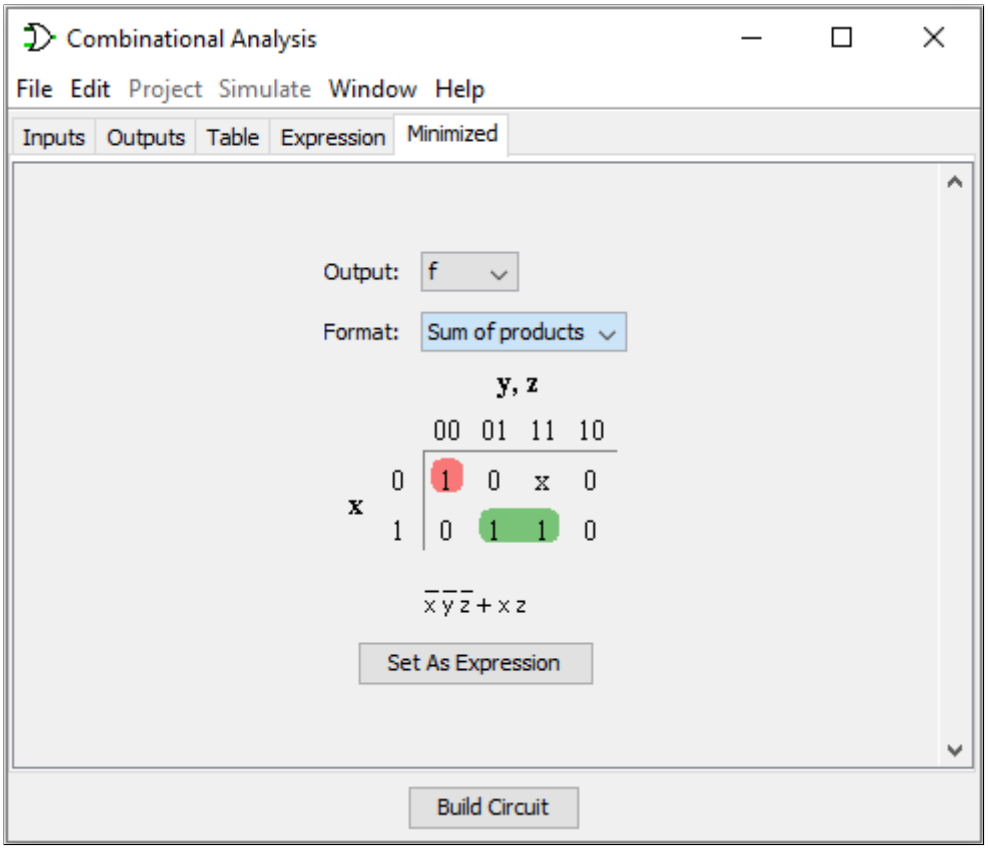
A.3 Rút gọn mạch logic và Bìa-K

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

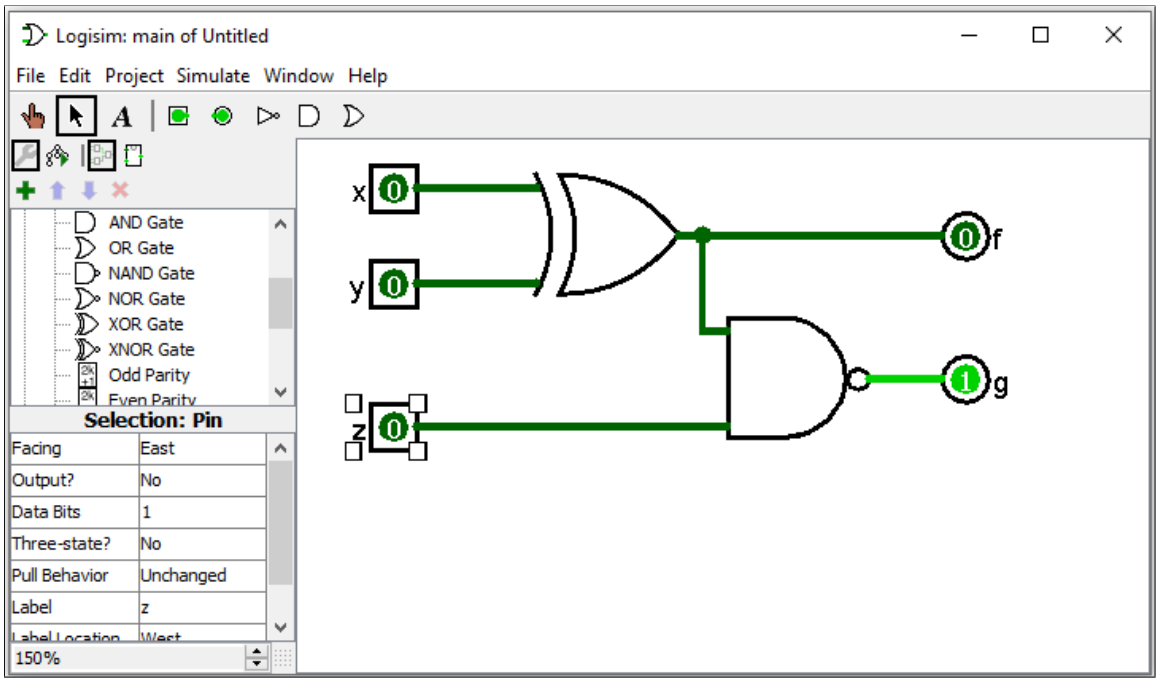
Bước 1: Tại màn hình làm việc chính, bắt đầu vẽ mạch logic cần phân tích như hình A.8.

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**.

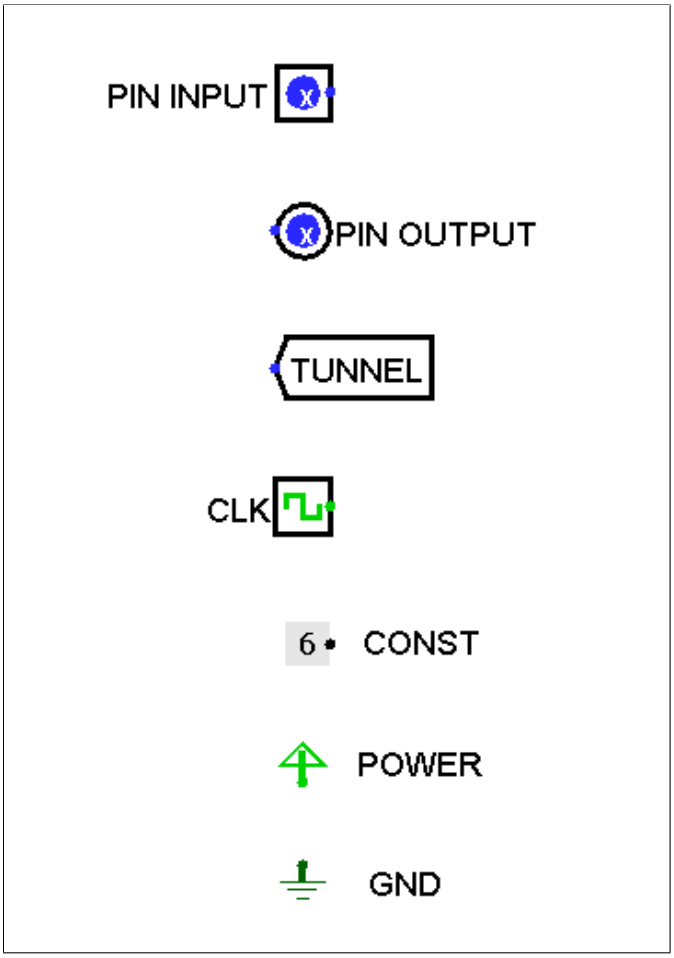
Bước 4: Đặt các cổng cần thiết vào và kết nối dây dẫn. Tham khảo thêm Hướng dẫn của Logisim cho các thao tác này.

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

A.5 Các đầu nút trong Logisim



Hình A.9: Các chân điện thường sử dụng trong Logisim

1. **PIN INPUT**: Là chân đầu vào n bit của một mạch điện. Có dạng hình vuông và thuộc tính **Output?** được thiết lập *No*. Chức năng phân tích mạch sẽ không làm việc với các PIN nhiều bit.

2. **PIN OUTPUT**: Là chân đầu ra có n bit của một mạch điện. Có dạng hình tròn và thuộc tính **Output?** được thiết lập *Yes*.

3. **TUNNEL**: các nút có cùng **Label** được gán trong thuộc tính sẽ được xem là kết nối với nhau. Các chân TUNNEL được dùng để hạn chế dây dẫn cắt nhau hoặc khi cần truyền đi xa.

4. **CLK**: Khi lựa chọn **Simulate Tick enable (Ctrl + K)** được chọn, chân CLK sẽ sinh tín hiệu nhấp nháy với mức cao trong **High Duration** tíc và theo sau đó là mức thấp trong **Low Duration** tíc. Tần số của mỗi tíc (Tick) được thiết lập trong lựa chọn **Simulate \ Tick Frequency** từ 0.25 Hz đến 4.1 kHz.

5. **CONST**: Đầu vào hằng số n bit.

6. **POWER**: Nguồn điện, tương đương logic với đầu vào hằng số -bit mang giá trị TRUE hay 1.

7. **GND**: Nối đất, tương đương logic với đầu vào hằng số 1-bit mang giá trị FASLE hay 0.