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:

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

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

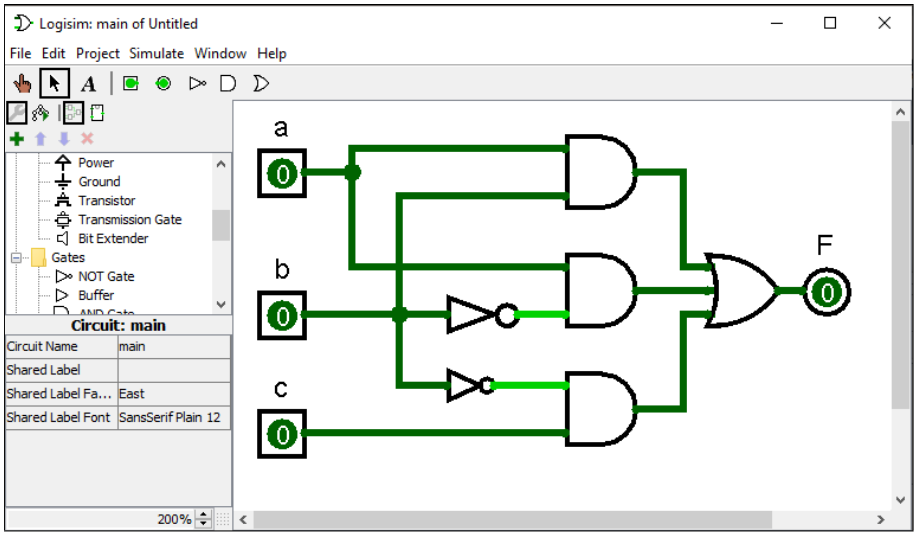
Gợi ý. Sử dụng các linh kiện cơ bản trên thanh tác vụ để hoàn tất mạch.

- **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.

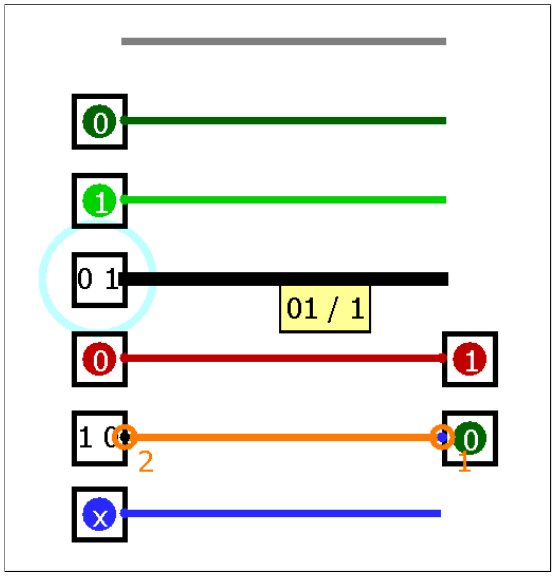


Hình 1.1: Mạch logic được xây dựng trên Logisim

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.

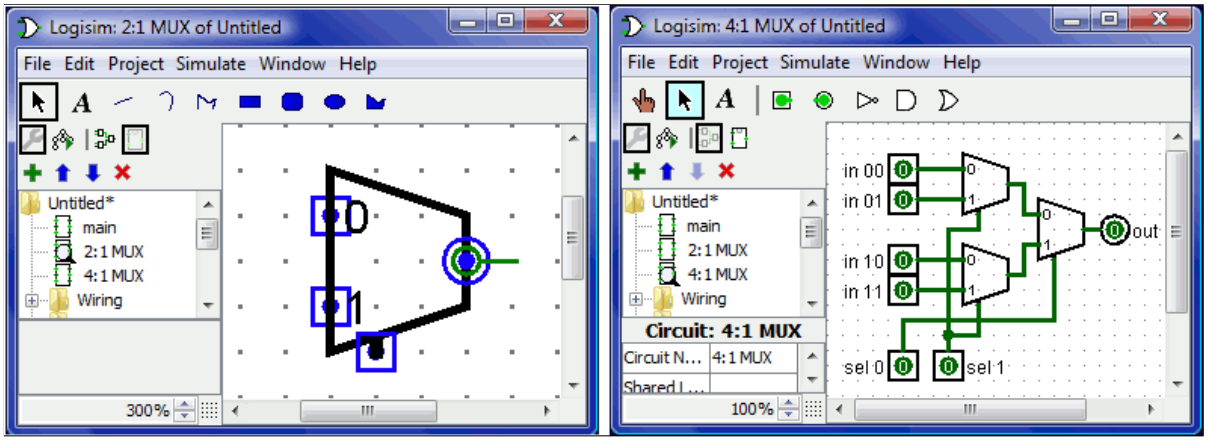
1.2 Thực hiện mạch con

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

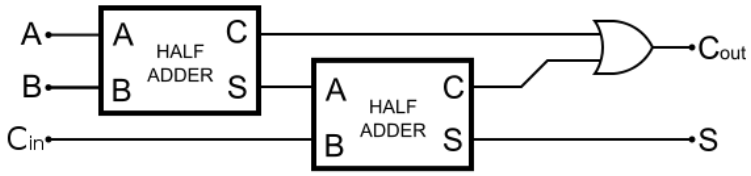
Yêu cầu 2. Hoàn tất bảng sự thật 2.2 rồi dùng nó để thực hiện mạch con đặt tên là FA thực hiện chức năng cộng toàn phần với 3 tín hiệu nhập là 2 bit a, b với bit tràn vào C_{in}; và 2 tín hiệu đầu ra là S với C_{out}. Kiểm thử mạch con này sau khi hoàn tất.

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:

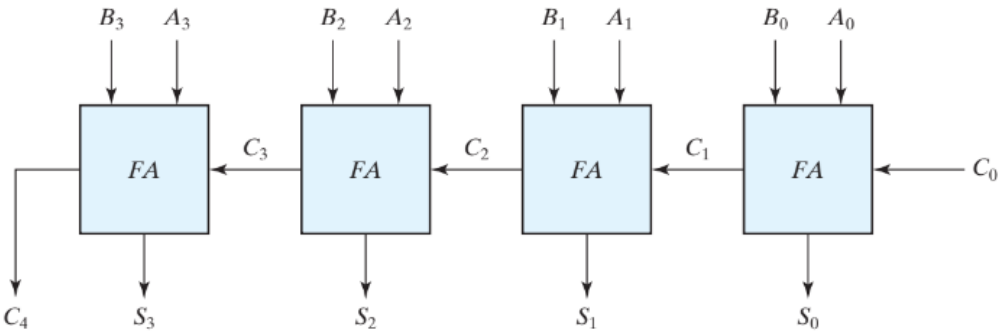
$$\begin{array}{r} \overset{1}{} \\ 1011 \\ +\,1010 \\ \hline 10101 \end{array}$$

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

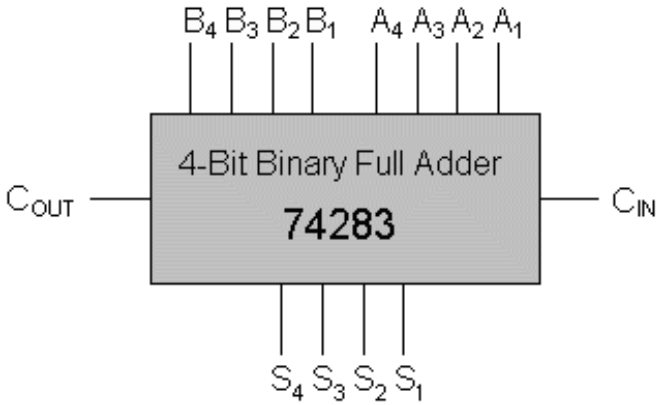
mạch con đã nhiều nên hình dạng bố trí các chân vào/ra rất quan trọng. Một bố trí tốt sẽ giúp cho việc sử dụng mạch con này thuận lợi hơn rất nhiều.

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



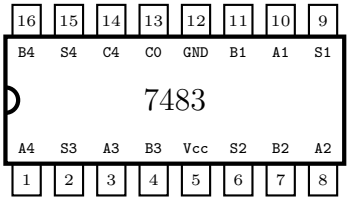
Hình 2.2: Mạch cộng 4-bit được ghép nối từ 4 mạch FA

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

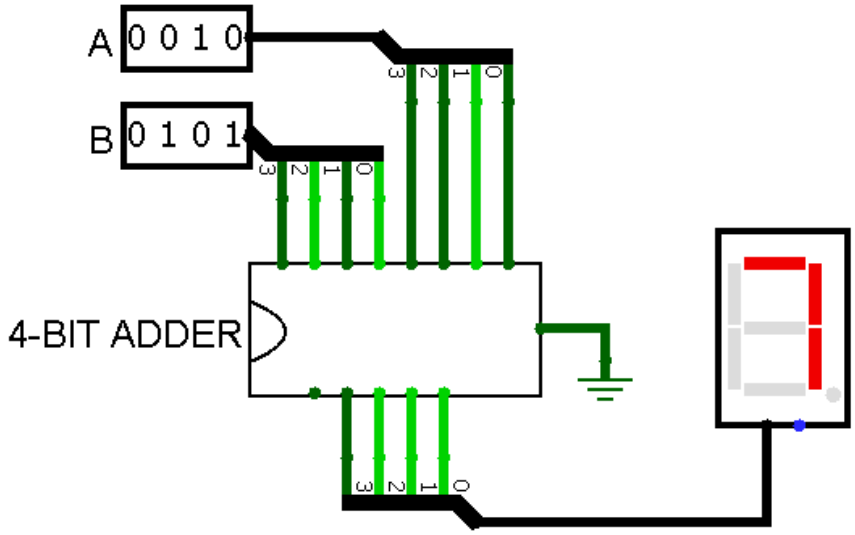
Trên thực tế, IC được đóng gói với bố trí chân như sơ đồ dưới đây.



2.4 Máy tính 4-bit đầu tiên của bạn

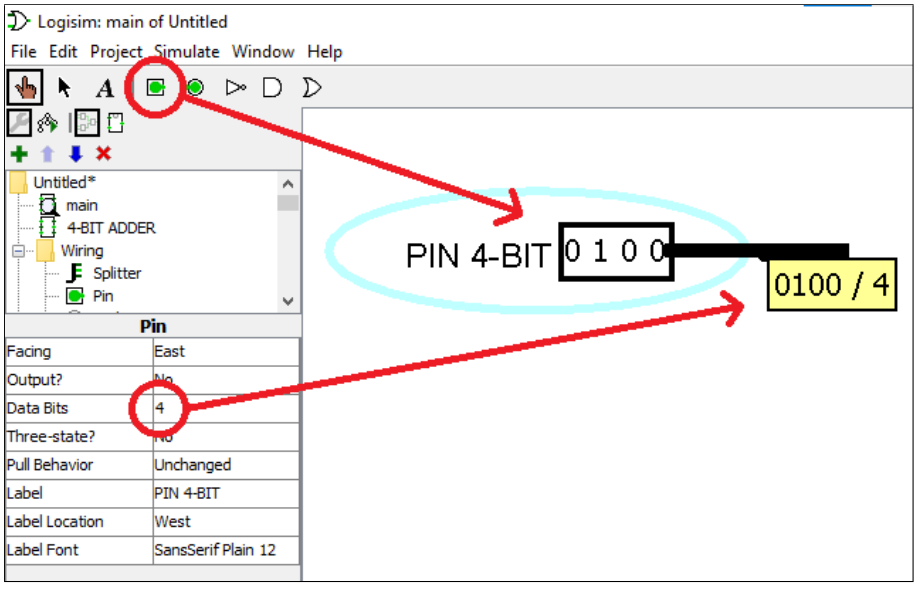
Tại mạch chính, hãy sử dụng IC 4-BIT ADDER, 2 pin 4-bit như là đầu vào của số A và B, đèn led 7 đoạn "Hex led display" và thưởng thức "máy tính" sơ khai có thể thực hiện

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.



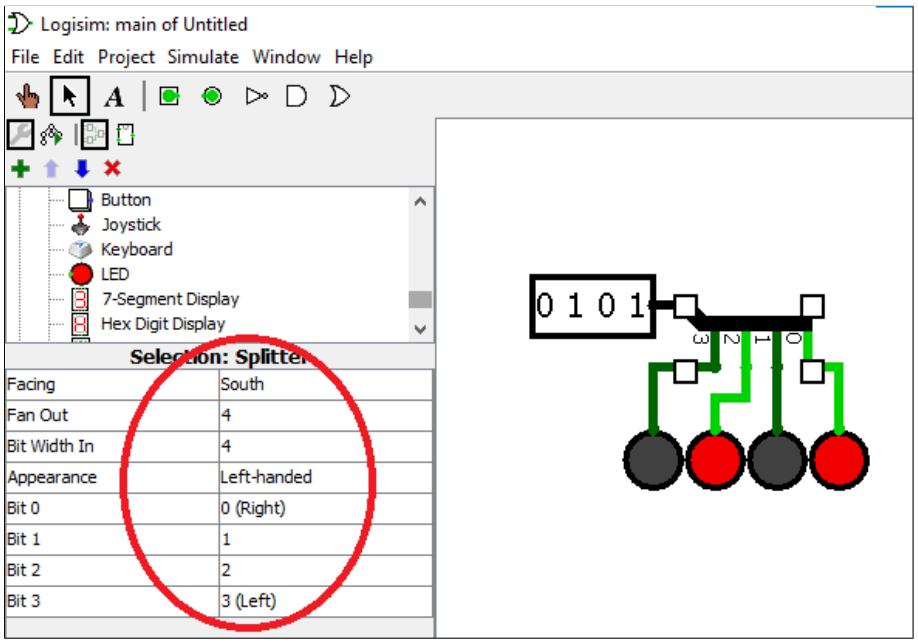
Hình 2.4: Bộ cộng 4-bit đang làm việc với 2 số đầu vào là 2 và 5

2.4.1 Pin n-bit trong Logism



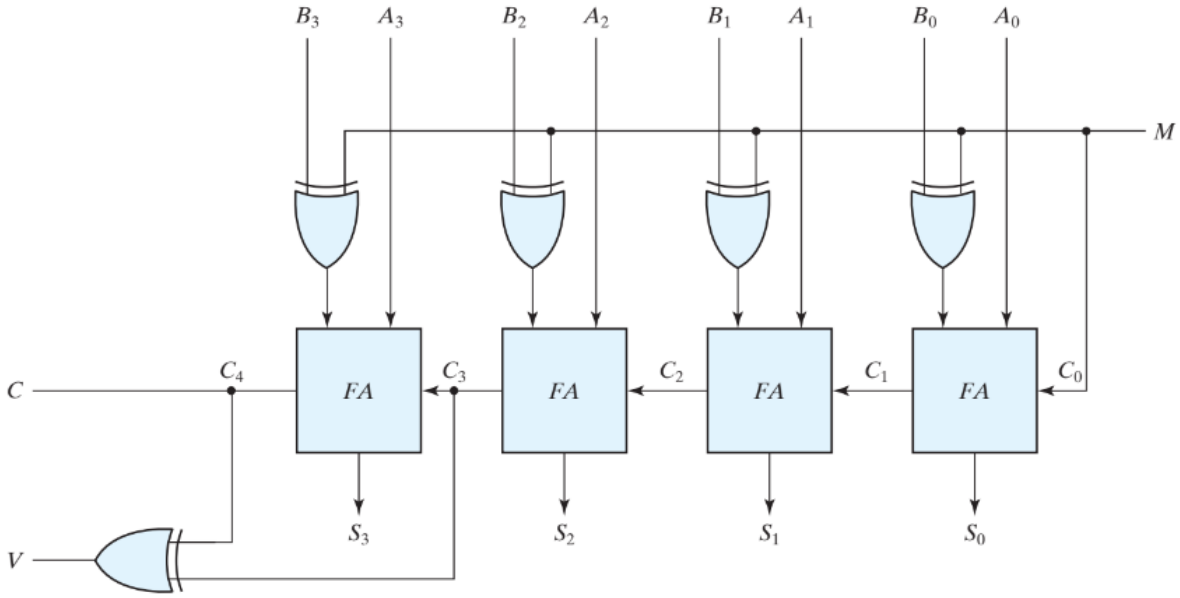
Hình 2.5: Pin đầu vào có độ rộng 4 bit.

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

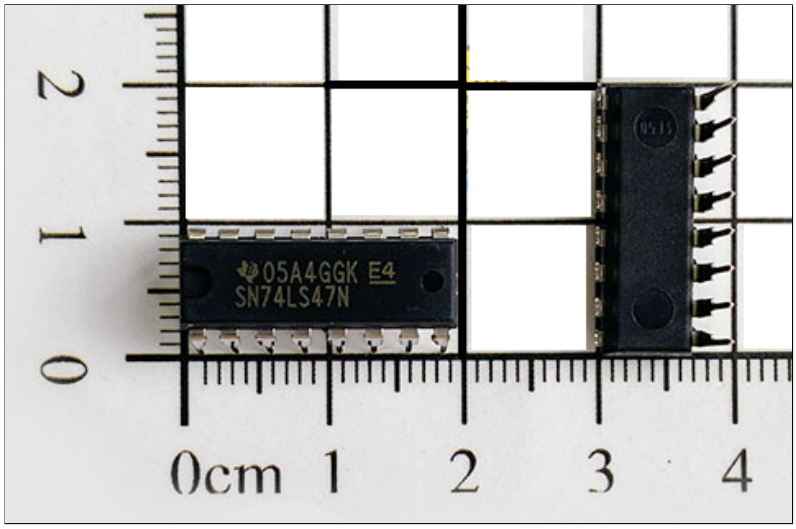
"Số lượng transistor trên mỗi đơn vị inch vuông sẽ tăng lên gấp đôi sau mỗi 18 tháng." - Gordon Earle Moore, Đồng sáng lập tập đoàn Intel.

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							

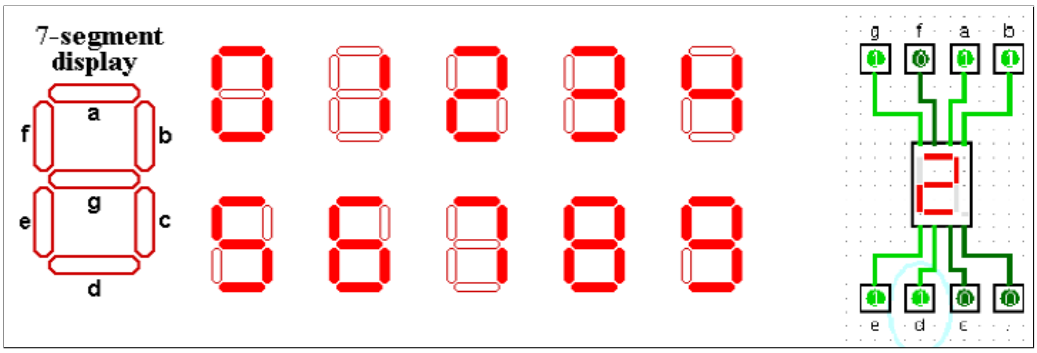
Bảng 3.1: Bảng sự thật mạch giải mã số BCD ra 7 dây điều khiển leg 7 đoạn

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

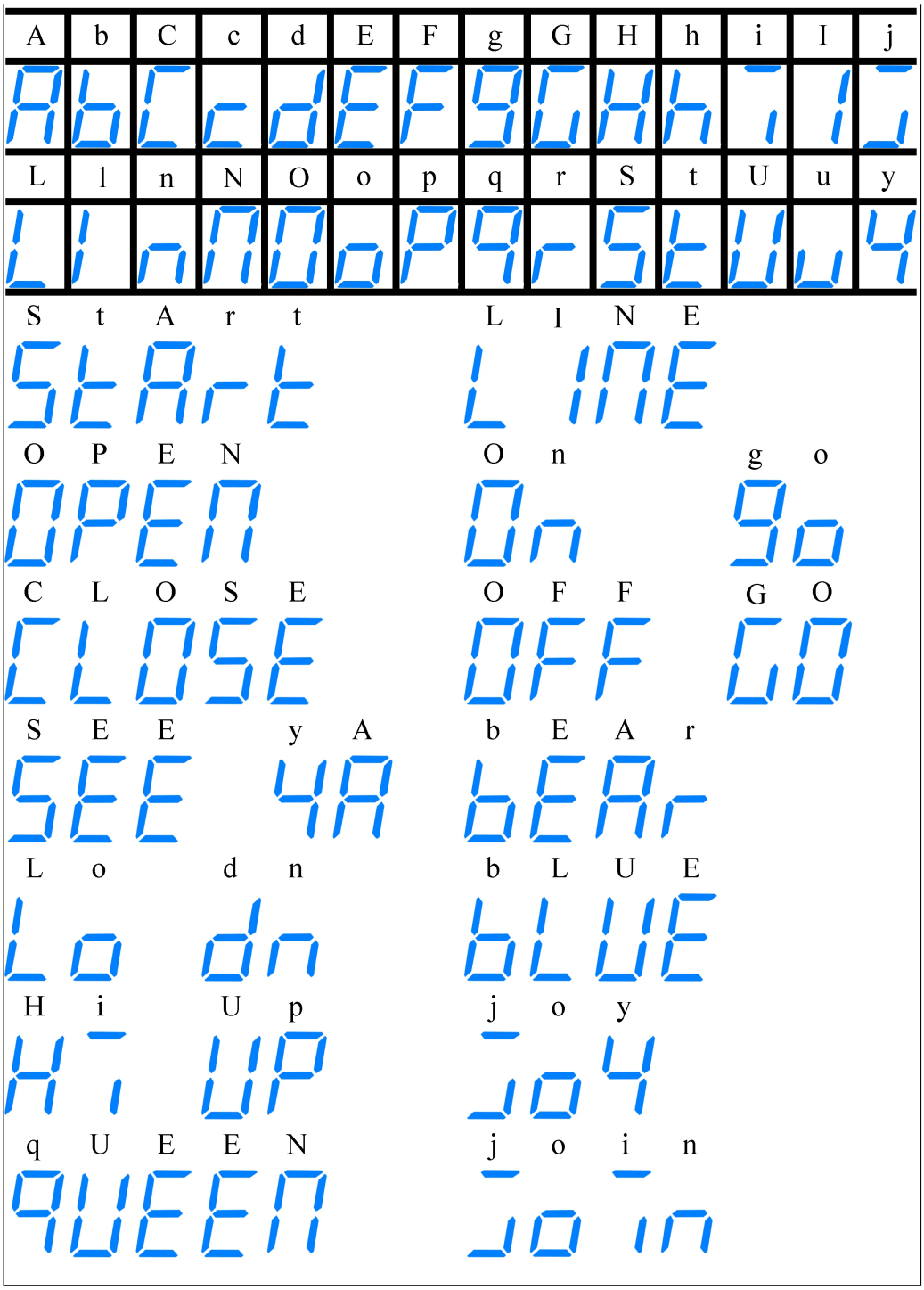
1010 đến 1111 được xem là "don't care". Trong khi đó giải mã nhị phân thực hiện cho đầy đủ 16 giá trị.

- 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.

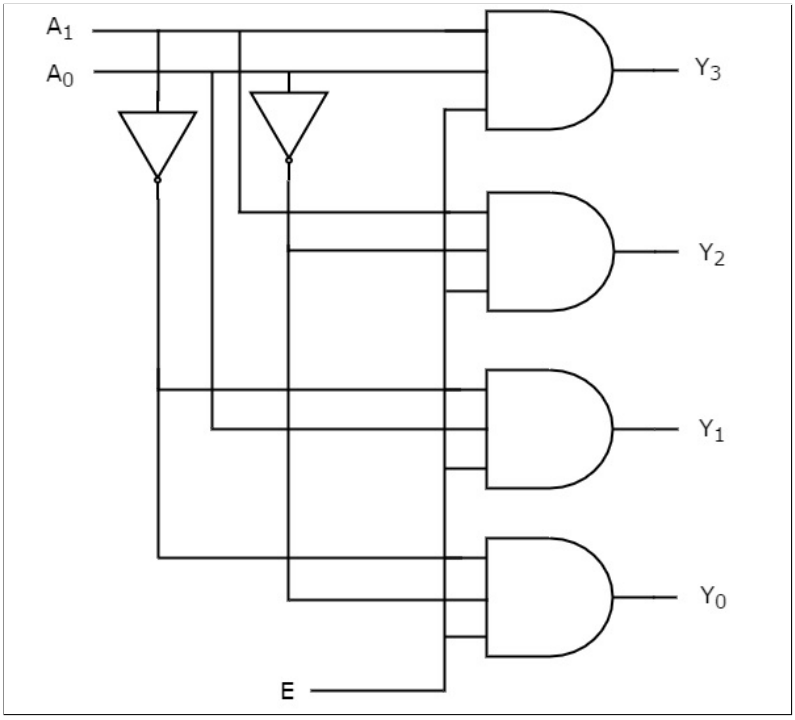
Inputs			Outputs							
x	**y**	**z**	**D_0**	**D_1**	**D_2**	**D_3**	**D_4**	**D_5**	**D_6**	**D_7**
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

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.

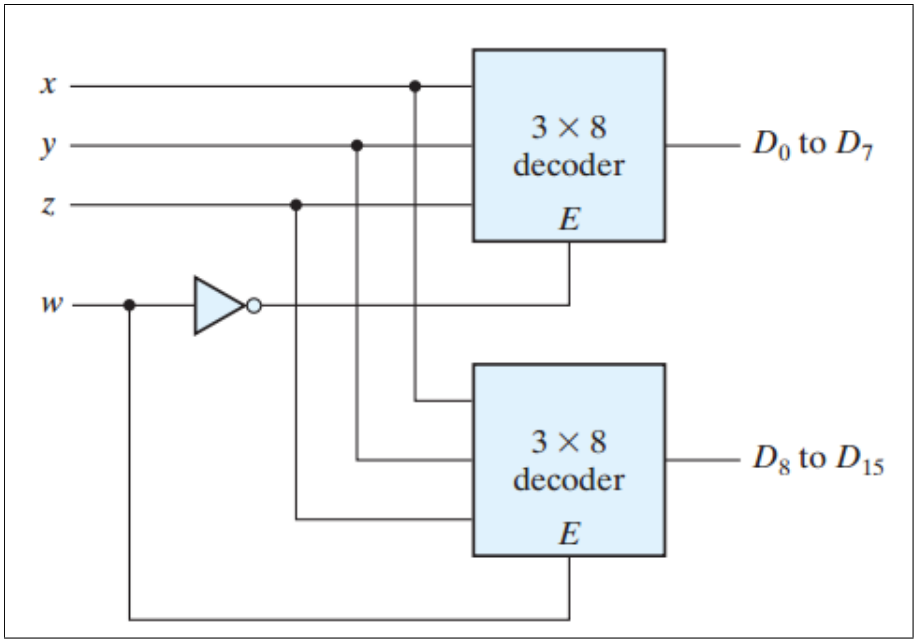


Hình 3.5: Cách ghép nối cổng EN với từng đầu ra

.

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 2.** 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

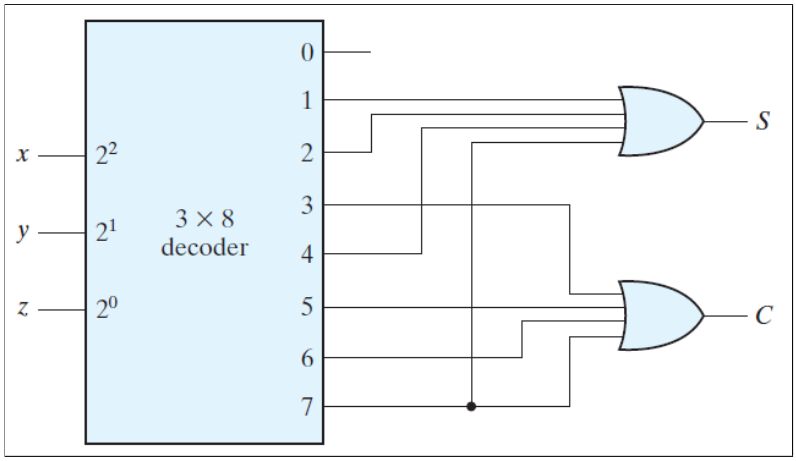
Bộ giải mã cung cấp 2^n minterm khi có n biến đầu vào. Mỗi đầu ra tích cực của bộ giải mã được liên kết với một tổ hợp bit đầu vào duy nhất. Vì bất kỳ hàm Boole nào cũng có thể được biểu diễn dưới dạng tổng các minterm, nên một bộ giải mã tạo ra các minterm của hàm này cùng với cổng OR ở cuối cùng tạo 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 có 8 đầu vào từ D_7 đến D_0 và 3 đầu ra xyz dựa trên bảng sự thật 3.8. Kiểm thử sau khi hoàn tất.

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 3.9: Bảng sự thật của bộ mã hoá ưu tiên

.

$$x = D_2 + D_3$$

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

Yêu cầu 7. Thực hiện mạch giải mã cho một bàn phím số như hình 3.10, 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.

3.3.2 Bàn phím số

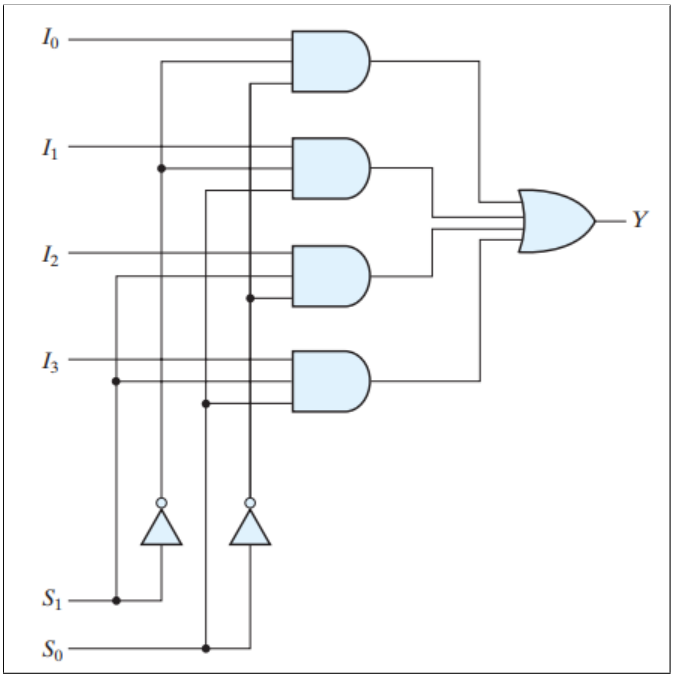
Hình 3.10: Một bàn phím số

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

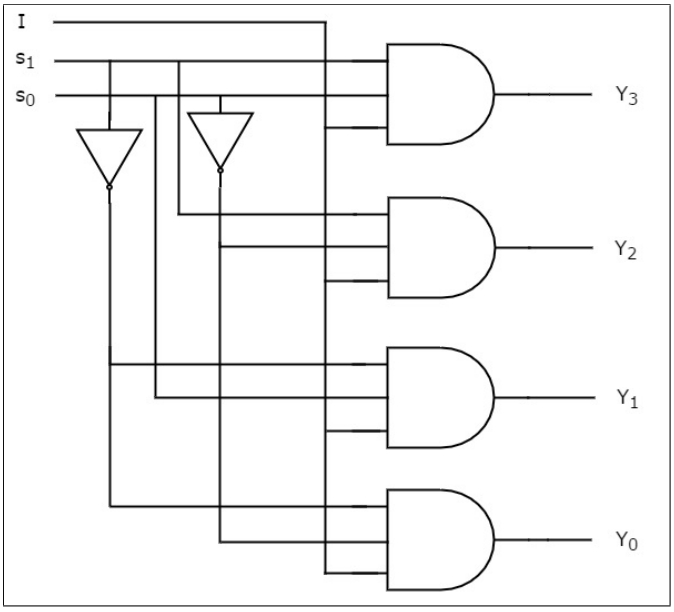
.

Yêu cầu 8. 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 9. Bổ sung chân EN để cho phép hay vô hiệu mạch trộn kênh 4-1 MUX ở **Yêu cầu 8.** 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 10. 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.

Bài thực hành LAB 4

MẠCH TUẦN TỰ

"Nếu một cỗ máy được kỳ vọng là không thể sai lầm, thì nó cũng không thể thông minh."
– Alan Turing[1].

Các mạch tổ hợp đã hoàn tất các chức năng tính toán và truyền tải các số và mã nhị phân. Tuy vậy vấn đề lưu trữ và trạng thái của máy tính không thể giải quyết bằng mạch tổ hợp, do các mạch này độc lập với thời gian. Bằng việc định nghĩa máy trạng thái hữu hạn, các mạch điện được chuyển từ trạng thái này đến trạng thái khác từ luồng tín hiệu đầu vào và đi đến một kết quả cuối cùng. Máy tính thay vì tiếp nhận tất cả tín hiệu và phải xử lý cùng lúc, đã có thể phân chia bài toán ra nhỏ hơn và tái sử dụng một mạch chức năng nhiều lần, với các dữ liệu được lưu trữ lại sau mỗi bước tính toán.

4.1 Mạch chốt (Latch) và mạch lật (Flip flop)

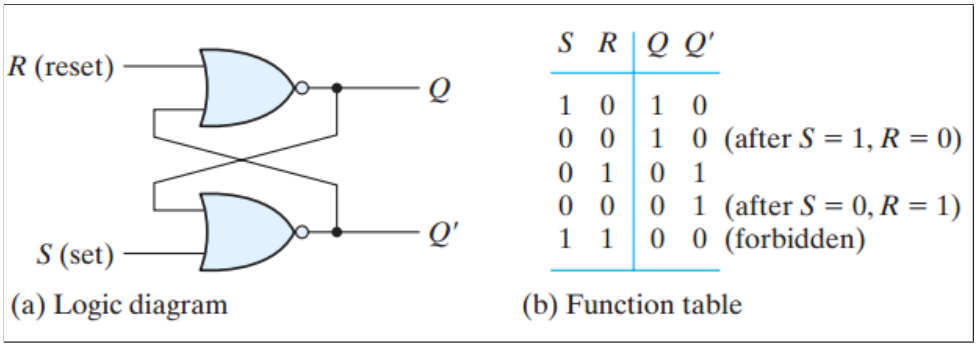
Trong điện tử kỹ thuật số, một mạch chốt (latch) còn được gọi là đa hài ổn định kép (bistable-multivibrator), bởi vì nó có hai trạng thái ổn định là mức cao và mức thấp. Nó hoạt động giống như một thiết bị lưu trữ bằng cách giữ dữ liệu thông qua một lần phản hồi. Nó lưu trữ 1 bit dữ liệu ngay khi thiết bị được kích hoạt.

Yêu cầu 1. (a) Dựa vào hình 4.1 để thực hiện một mạch chốt S-R và kiểm thử mạch, thể hiện rằng mạch chốt này đóng vai trò nhớ trạng nhớ cho đầu ra sau khi nút S hoặc R được nhấn.

(b) Dựa vào hình 4.2 để thực hiện một mạch chốt S-R có chân điều khiển EN (cho phép hoạt động hoặc vô hiệu hoá) và kiểm thử mạch. Tìm một số ví dụ trên thực tế với nút nhấn có thêm đầu vào EN.

(c) Điều gì sẽ xảy ra khi cổng vào S và R đồng thời được thiết lập tín hiệu mức cao?

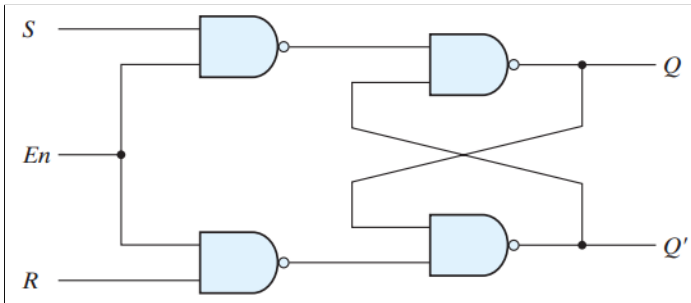
[1]Tiểu sử Alan Turing tại https://vi.wikipedia.org/wiki/Alan_Turing



S	R	Q	Q′	
1	0	1	0	
0	0	1	0	(after $S = 1, R = 0$)
0	1	0	1	
0	0	0	1	(after $S = 0, R = 1$)
1	1	0	0	(forbidden)

(b) Function table

Hình 4.1: Sơ đồ mạch chốt S-R



En	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	$Q = 0$; reset state
1	1	0	$Q = 1$; set state
1	1	1	Indeterminate

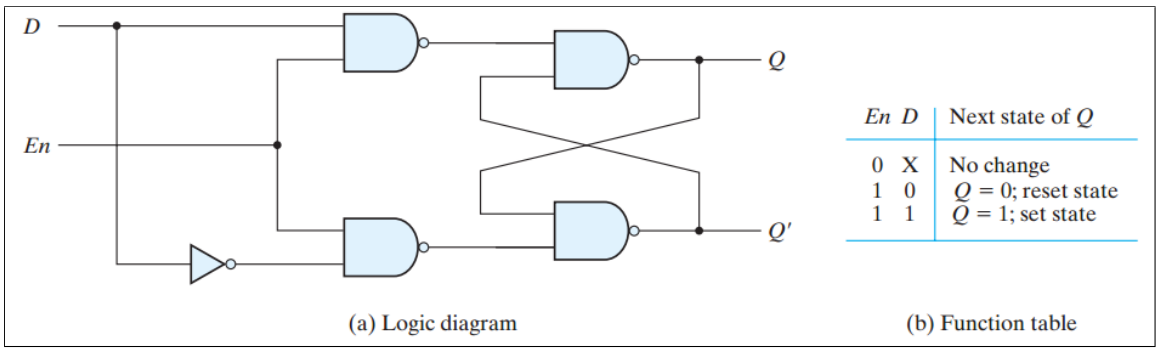
(b) Function table

Hình 4.2: Sơ đồ mạch chốt S-R có chân EN

4.1.1 Thanh ghi

Trong kiến trúc máy tính, một thanh ghi (registers) là một bộ nhớ dung lượng nhỏ và rất nhanh được sử dụng để tăng tốc độ xử lý của các chương trình máy tính bằng cách cung cấp các truy cập trực tiếp đến các giá trị cần dùng. Thanh ghi thường được đo bằng các bit nó có thể chứa, ví dụ, một thanh ghi "8-bit" hay thanh ghi "32-bit". Hầu hết các máy tính hiện đại hoạt động theo nguyên lý chuyển dữ liệu từ bộ nhớ chính vào các thanh ghi, tính toán trên chúng, sau đó chuyển kết quả vào bộ nhớ chính.

Yêu cầu 2. (a) Dựa vào hình 4.3 để thực hiện một mạch chốt D và kiểm thử mạch.
(b) Kết nối một pin 8-bit với 8 mạch chốt D và đầu ra cũng là một pin 8-bit rồi mô tả quá trình ghi dữ liệu lên thanh ghi 8-bit này.



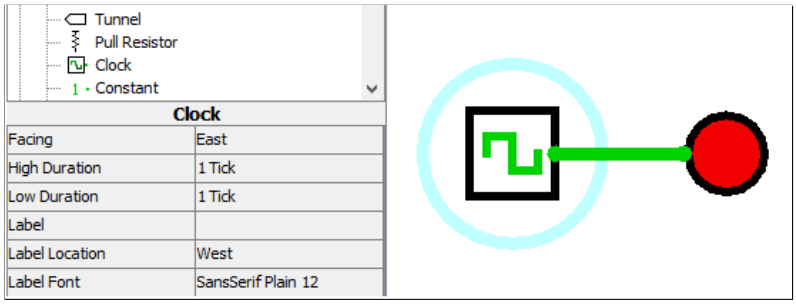
Hình 4.3: Sơ đồ mạch chốt D

Yêu cầu 3. Tạo ra một cổng cấp xung đồng hồ có chu kỳ 0.5 giây với thời gian tích cực mức cao chiếm $^3/_4$ chu kỳ.

Gợi ý. Tham khảo Phục lục A.5 để đặt một chân cấp xung đồng hồ như hình 4.4. Để có 4 tíc cho 1 chu kỳ trong 0.5 giây, tần số cần được thiết lập:

$$f = \frac{1}{0.5/_4} = 8Hz \tag{4.1}$$

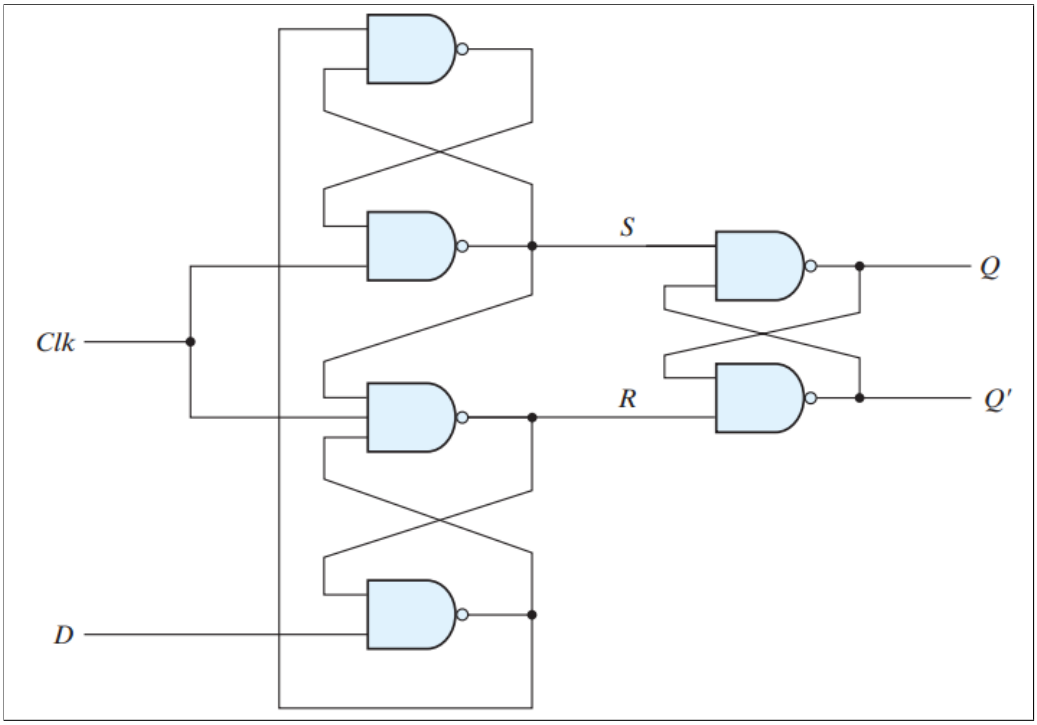
Sau đó, thiết lập **High Duration** và **Low Duration** của đầu cấp xung đồng hồ này lần lượt là **3 Tick** và **1 Tick**.



Hình 4.4: Một ví dụ cho cổng cấp xung đồng hồ

Cũng giống như mạch chốt, mạch lật cũng là một linh kiện kỹ thuật số đa hài ổn định kép và lưu trữ được 1 bit dữ liệu. Tuy nhiên, không giống như mạch chốt, việc kích hoạt mạch lật chỉ làm thay đổi đầu ra khi có sự thay đổi của tín hiệu xung đồng hồ ở đầu vào, có thể là một cạnh lên hoặc một cạnh xuống.

Yêu cầu 4. (a) Dựa vào hình 4.3 để thực hiện một mạch lật D và kiểm thử mạch.
(b) Phân biệt sự khác nhau giữa mạch chốt D và mạch lật D.



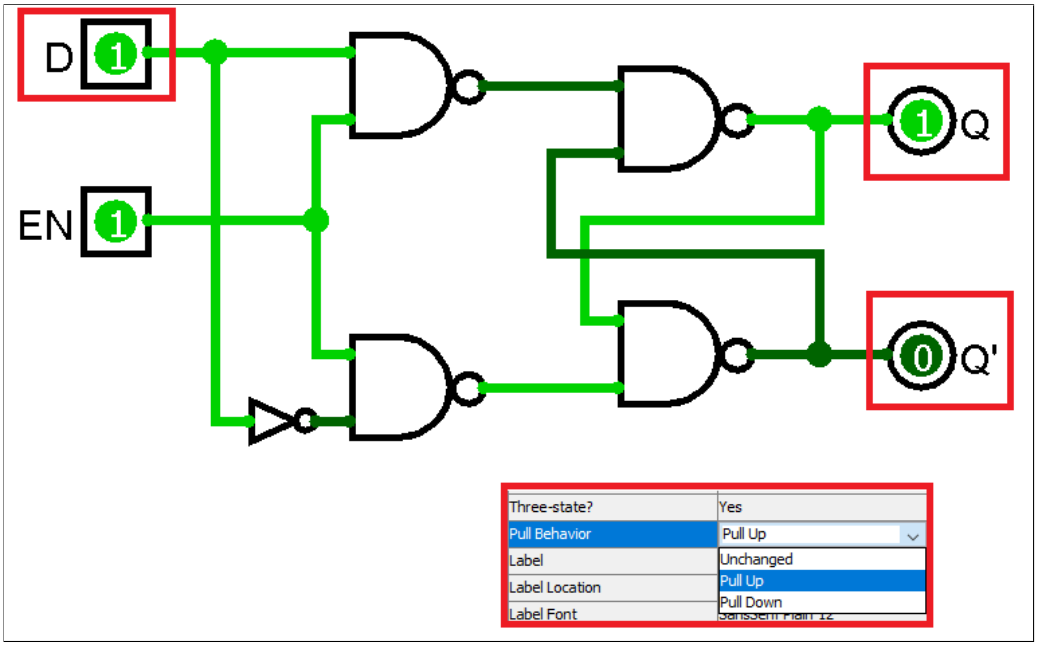
Hình 4.5: Sơ đồ mạch lật D

4.1.2 Điện trở kéo lên

Khi mà đầu ra của một IC được đấu nối ngược lại mạch điện trước IC đó, tín hiệu không xác định có thể bị lan truyền và làm mạch điện rơi vào trạng thái lỗi. Lỗi này có thể tránh khỏi bằng cách đặt một điện trở kéo lên (Pull Up) hoặc kéo xuống (Pull Down) và đầu vào / ra thay vì để cho thả nổi (Unchanged). Thao tác này thiết lập trạng thái "khởi động" cho mạch điện.

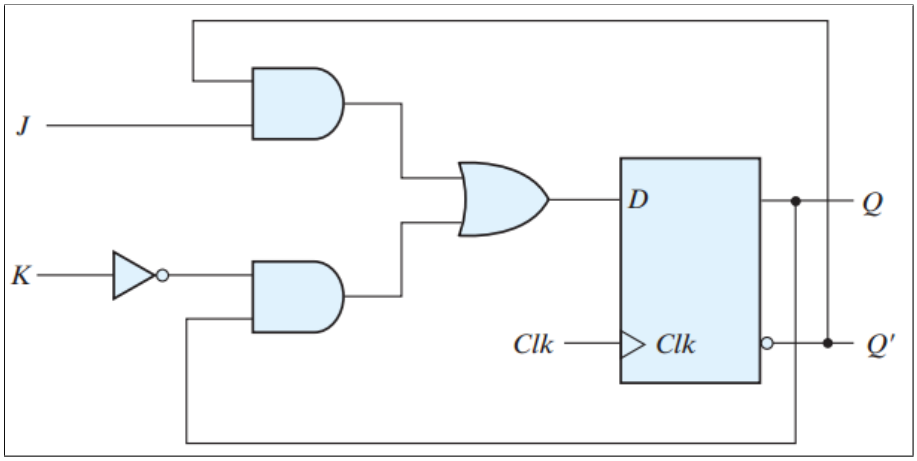
Ngõ vào/ra	**Thả nổi**	**Kéo lên**	**Kéo xuống**
x	x	1	0
0	0	0	0
1	1	1	1

Bảng 4.1: Bảng so sánh đầu ra với các tình huống mạch đầu vào



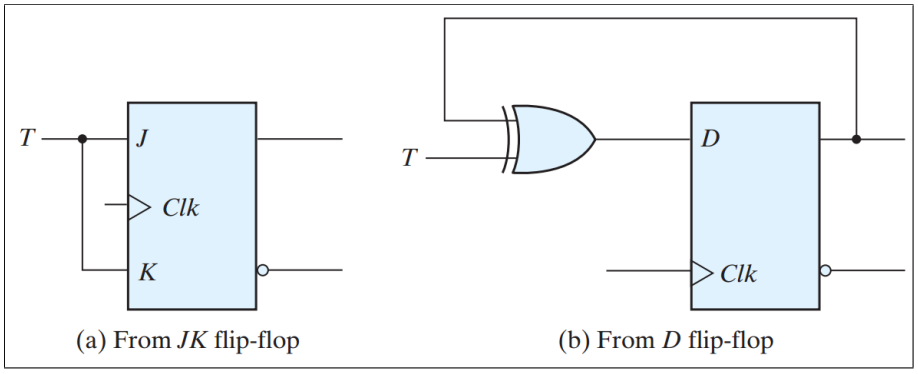
Hình 4.6: Thiết lập điện trở pull-up cho các cổng ra

Yêu cầu 5. (a) Dựa vào hình 4.7 để thực hiện một mạch lật J-K và kiểm thử mạch.
(b) Thay đổi tần số xung clock để đầu ra nhấp nháy nhanh hơn hoặc chậm đi. Tần số bằng bao nhiêu thì mắt người không nhận ra sự nhấp nháy?



Hình 4.7: Sơ đồ mạch lật J-K

Yêu cầu 6. Dựa vào hình 4.8 để thực hiện hai mạch lật T bằng hai cách ghép nối và kiểm thử các mạch.



Hình 4.8: Sơ đồ mạch lật T

4.2 Mạch đếm (Counter)

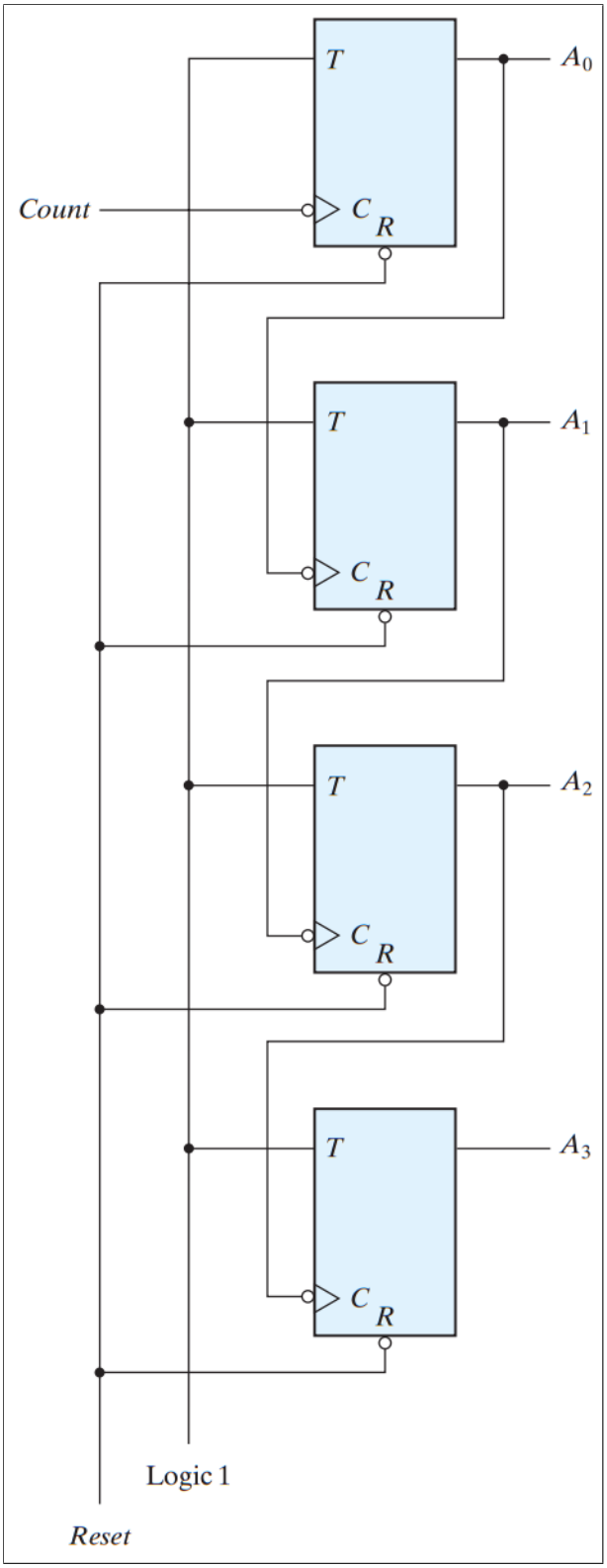
Trong logic và điện toán kỹ thuật số, bộ đếm là một thiết bị lưu trữ (và hiển thị) số lần xảy ra một sự kiện hoặc quá trình đếm thứ tự cụ thể, thường liên quan đến đồng hồ. Phổ biến nhất là mạch tuần tự với một đầu vào đồng hồ và nhiều đầu ra. Các giá trị trên các đầu ra đại diện cho một số trong hệ thống số nhị phân hoặc số BCD. Mỗi xung cấp cho đầu vào đồng hồ sẽ làm tăng hoặc giảm số trong bộ đếm.

Một mạch đếm thường được cấu tạo từ nhiều mạch lật nối thành tầng. Bộ đếm là một thành phần được sử dụng rất rộng rãi trong các mạch kỹ thuật số và được sản xuất dưới dạng các mạch tích hợp riêng biệt và cũng được tích hợp như một phần của các mạch chức năng lớn hơn.

4.2.1 Mạch đếm bất đồng bộ

Bộ đếm bất đồng bộ (asynchronous or ripple counter) là một "chuỗi" các mạch lật T trong đó mạch lật có trọng số thấp nhất (bit 0) được thay đổi theo chu kỳ của tín hiệu đồng hồ ở đầu vào; và từng mạch lật tiếp theo được thay đổi theo chu kỳ của đầu ra mạch lật liền trước nó. Mỗi mạch lật tạo ra một độ trễ từ cạnh đồng hồ đến chuyển đổi đầu ra, do đó làm cho các bit bộ đếm thay đổi vào các thời điểm khác nhau và tạo ra hiệu ứng gợn sóng khi đồng hồ đầu vào truyền qua chuỗi. Mạch lật J-K cũng có thể sử dụng cho bộ đếm bất đồng bộ bằng cách thiết lập cả hai đầu vào J và K ở mức điện thế cao.

Yêu cầu 7. Dựa vào hình 4.9 để thực hiện mạch đếm bất đồng bộ (Ripple counter) và kiểm thử mạch. Đầu ra A_i nào là bit trọng số thấp nhất? Hãy ghép 4 đầu ra và dẫn đến một đèn **Hex display** để hiển thị giá trị đếm.

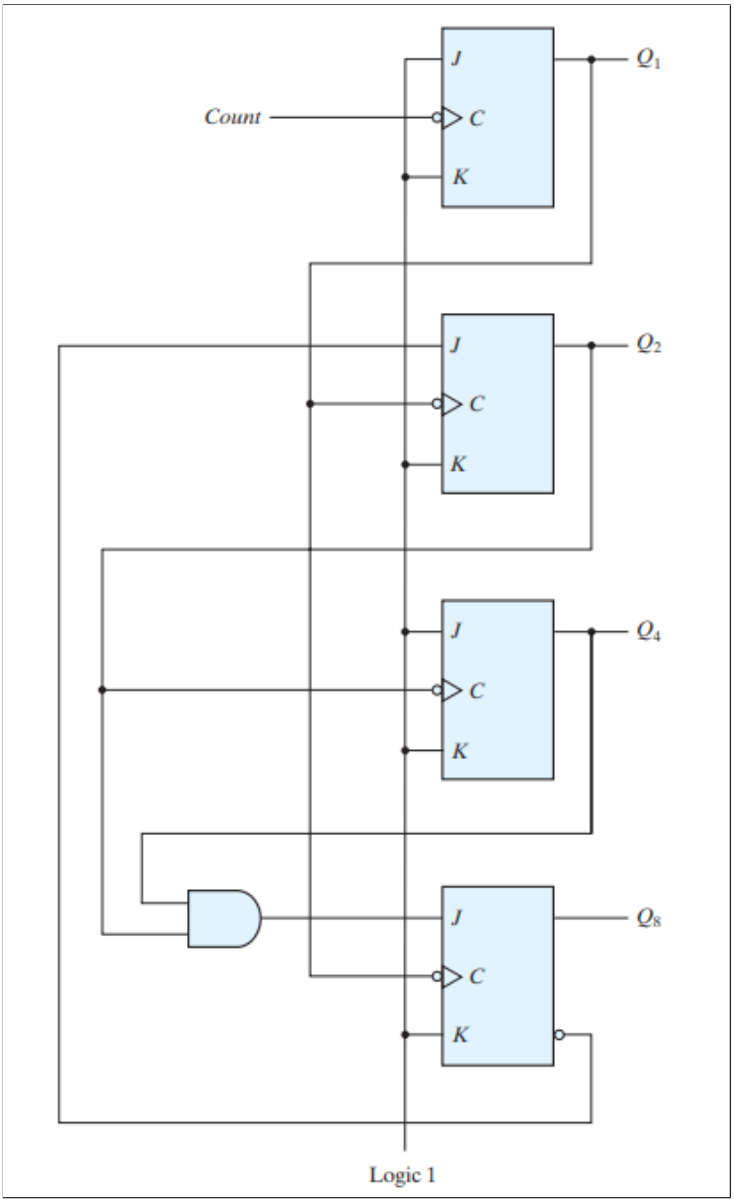


Hình 4.9: Sơ đồ mạch đếm bất đồng bộ xây dựng từ mạch lật T

4.2.2 Bộ đếm thập phân

Bộ đếm thập phân tuân theo chuỗi 10 trạng thái và trở về 0 sau số đếm là 9. Bộ đếm như vậy phải có ít nhất bốn mạch lật để biểu thị mỗi chữ số thập phân. Trình tự các trạng

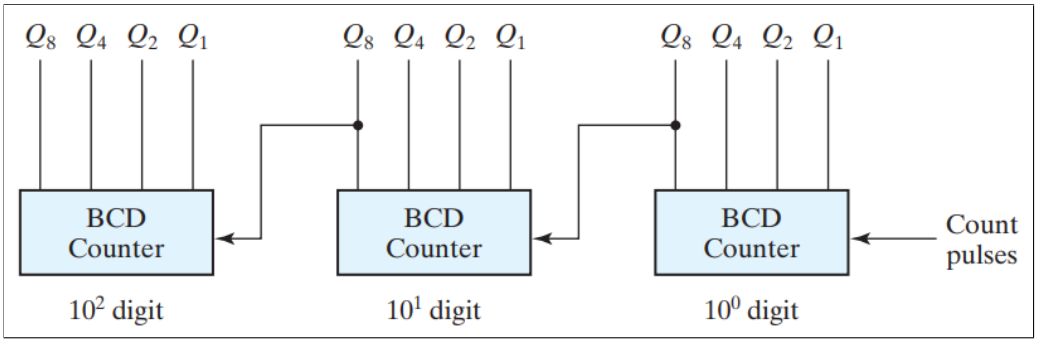
thái trong bộ đếm thập phân được quyết định bởi mã nhị phân dùng để biểu thị một chữ số thập phân. Bộ đếm thập phân tương tự như bộ đếm nhị phân, ngoại trừ trạng thái sau 1001_2 (9) là 0000_2 (0).



Hình 4.10: Sơ đồ mạch đếm BCD bất đồng bộ xây dựng từ mạch lật J-K

Yêu cầu 8. (a) Dựa vào hình 4.10 để thực hiện mạch đếm BCD bất đồng bộ và kiểm thử mạch. Hãy giải thích vì sao khi mạch đếm ở trạng thái 1001_2 thì chuyển về 0000_2 thay vì chuyển đến 1010_2?

(b) Ghép 3 mạch đếm BCD đã hoàn thành theo hình 4.11 và kết hợp các đèn **Hex display** để hiển thị giá trị đếm từ 000 đến 999.

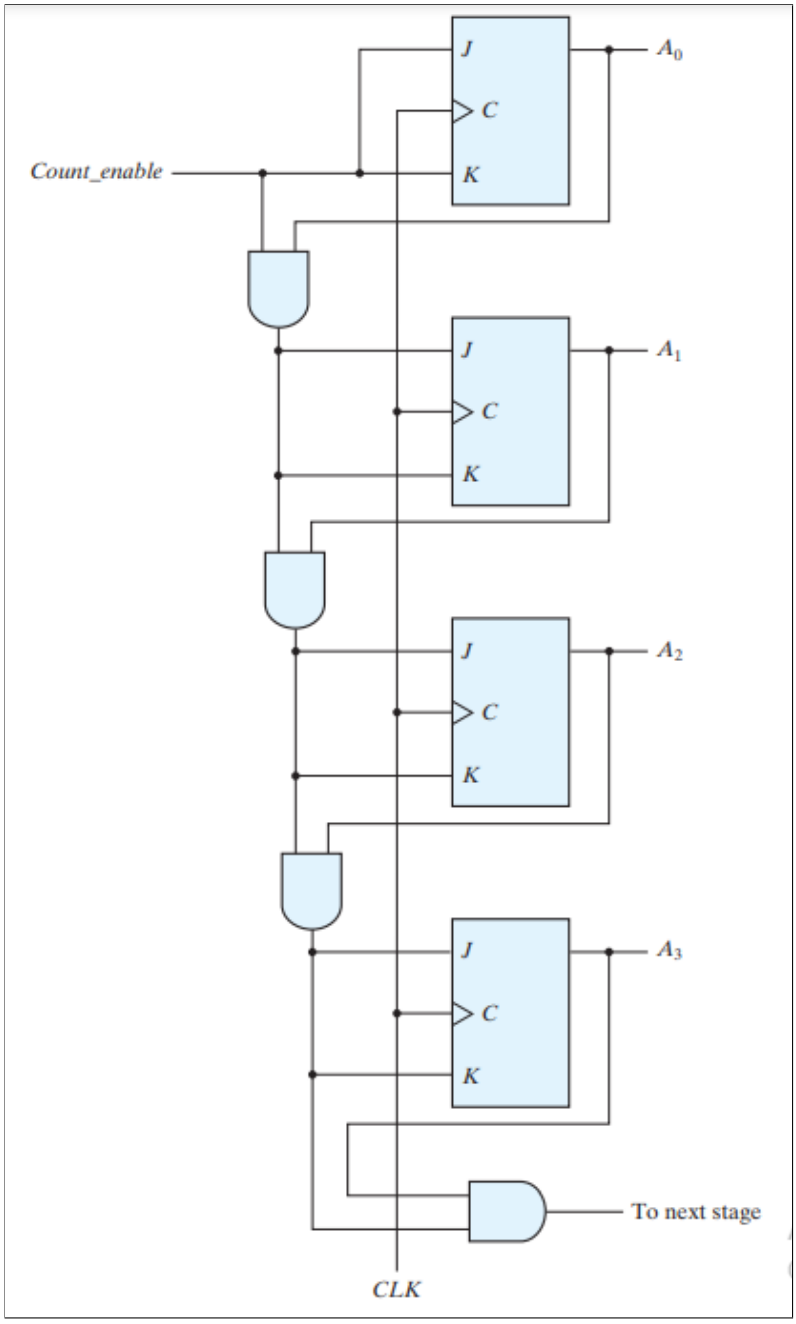


Hình 4.11: Bộ đếm BCD được mở rộng ra nhiều kí số

4.3 Mạch đếm đồng bộ (Synchronous Counter)

Trong bộ đếm đồng bộ, đầu vào đồng hồ của tất cả mạch lật được cùng kết nối chung và đồng thời được kích hoạt bởi cùng một dao động đồng hồ. Do đó, tất cả các mạch lật thay đổi trạng thái cùng một lúc.

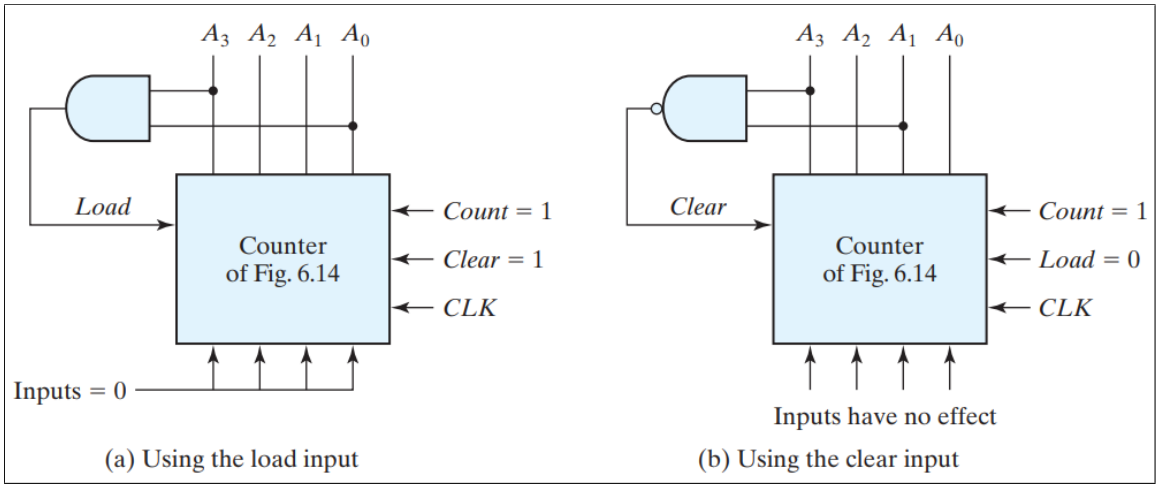
Yêu cầu 9. (a) Dựa vào hình 4.12 để thực hiện mạch đếm đồng bộ và kiểm thử mạch bằng cách ghép thêm đèn **Hex display** để hiển thị giá trị đếm.



Hình 4.12: Sơ đồ mạch đếm đồng bộ xây dựng từ mạch lật J-K

4.3.1 Cổng nạp song song

Các bộ đếm được sử dụng trong các hệ thống kỹ thuật số thường yêu cầu khả năng nạp song song để truyền số nhị phân ban đầu vào bộ đếm trước khi thực hiện thao tác đếm. Các bộ đếm này cần thêm cụm đầu vào cho giá trị khởi tạo và tín hiệu nạp. Khi tín hiệu bằng 1, điều khiển nạp đầu vào sẽ vô hiệu hóa thao tác đếm và truyền dữ liệu từ cụm đầu vào dữ liệu đến cụm mạch lật.



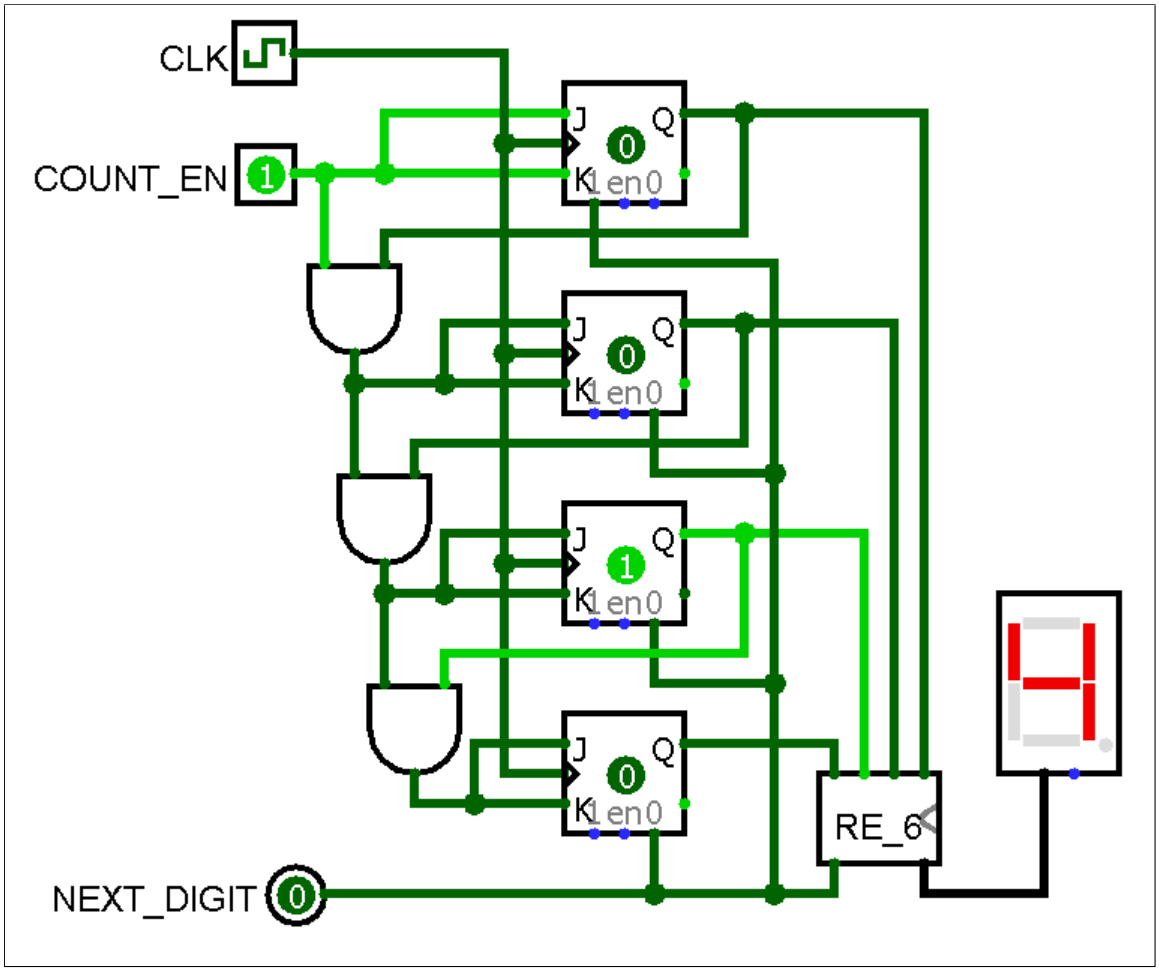
Hình 4.13: Hai cách tạo thành mạch đếm BCD đồng bộ sử dụng điều khiển nạp (trái) hoặc điều khiển xoá (phải)

Yêu cầu 10. (a) Dựa vào hình 4.13 và sử dụng linh kiện J-K để thực hiện mạch đếm đồng bộ từ 0 đến 9 và kiểm thử mạch.

(b) Để thực hiện việc đếm trong đoạn giá trị bất kỳ thì tiến hành như thế nào? Ví dụ hãy thực hiện bộ đếm tạo giá trị tăng từ 1 đến 5.

Gợi ý. (a) Sử dụng các mạch lật J-K đã thực hiện trong các yêu cầu trước, hoặc linh kiện **Memory** → **J-K Flip flop** của thư viện Logisim để thực hiện như mô tả tại hình 4.13. Kết nối các đầu ra thành cụm 4-bit và hiển thị bằng đèn **Hex display**.

(b) Sử dụng linh kiện **J-K Flip flop** của thư viện Logisim, kết nối tín hiệu khởi tạo vào các chân 0 hoặc 1 của mỗi mạch lật để xác lập giá trị khởi tạo. Tín hiệu khởi tạo có được khi bộ đếm tràn qua số giới hạn, vì vậy chúng ta cần thực hiện một mạch tổ hợp có đầu vào là cụm bit đang đếm và đầu ra bằng 1 khi và chỉ khi giá trị thập phân của cụm bit đầu vào là số liền sau của số giới hạn.



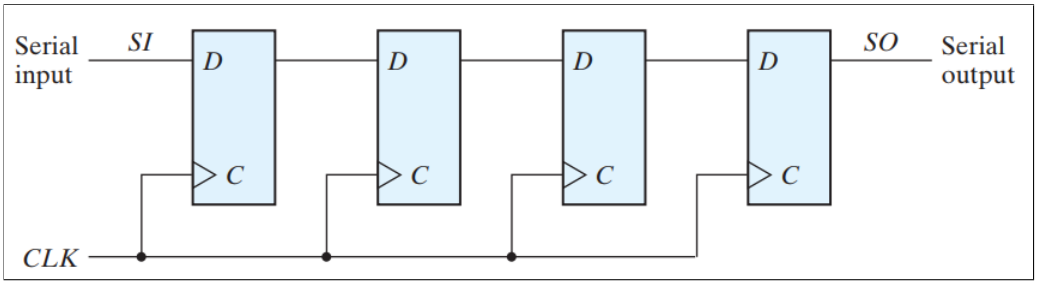
Hình 4.14: Một mạch đếm từ 1 đến 5

4.4 Các bộ đếm khác

Bộ đếm có thể được thiết kế để tạo ra bất kỳ chuỗi trạng thái mong muốn nào. Bộ đếm chia cho N (còn được gọi là bộ đếm modulo-N) là bộ đếm trải qua một chuỗi N trạng thái lặp lại. Trình tự có thể theo số nhị phân hoặc có thể là bất kỳ trình tự tùy ý nào khác. Bộ đếm được sử dụng để tạo tín hiệu định thời nhằm điều khiển chuỗi hoạt động trong hệ thống kỹ thuật số. Bộ đếm cũng có thể được xây dựng bằng các thanh ghi dịch chuyển. Trong phần này, chúng tôi trình bày một vài ví dụ về bộ đếm phi nhị phân.

4.4.1 Bộ đếm vòng

Các tín hiệu định thời điều khiển chuỗi hoạt động trong hệ thống kỹ thuật số có thể được tạo ra bởi một thanh ghi dịch hoặc bằng một bộ đếm có bộ giải mã. Bộ đếm vòng (Ring counter) là một thanh ghi dịch chuyển vòng chỉ có một mạch lật có đầu ra là 1 tại bất kỳ thời điểm cụ thể nào; và những mạch lật khác cùng có đầu ra là 0. Bit đơn được chuyển từ mạch lật này sang mạch lật tiếp theo để tạo ra chuỗi tín hiệu định thời.



Hình 4.15: Sơ đồ ghép nối và khởi tạo bộ đếm vòng

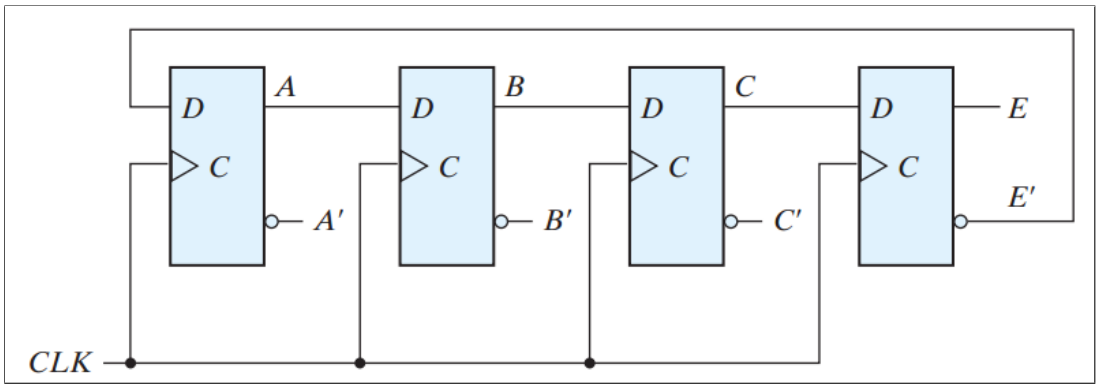
Yêu cầu 11. (a) Dựa vào hình 4.15 để thực hiện mạch đếm vòng với 4 mạch lật D và kiểm thử mạch bằng cách cài đặt Input là một Pin và thay đổi giá trị Pin này tùy ý sau mỗi chu kỳ, đặt các đèn LED vào mỗi đầu ra của từng mạch lật D để quan sát sự dịch chuyển.

4.4.2 Bộ đếm Johnson

Bộ đếm Johnson (còn có các tên gọi khác: bộ đếm vòng xoắn hay bộ đếm Möbius) là bộ đếm vòng được sửa đổi, trong đó đầu ra từ linh kiện cuối được đảo ngược và đưa trở lại đầu vào cho linh kiện đầu tiên. Thanh ghi quay vòng qua một chuỗi bit có độ dài bằng hai lần độ dài của thanh ghi dịch, và lặp lại vô tận.

Yêu cầu 12. (a) Dựa vào hình 4.16 để thực hiện mạch Johnson counter với 4 mạch lật D và kiểm thử mạch.

(b) Mạch Johnson có ưu điểm gì so với mạch đếm vòng?



Hình 4.16: Sơ đồ mạch đếm Johnson

Bài thực hành LAB 5

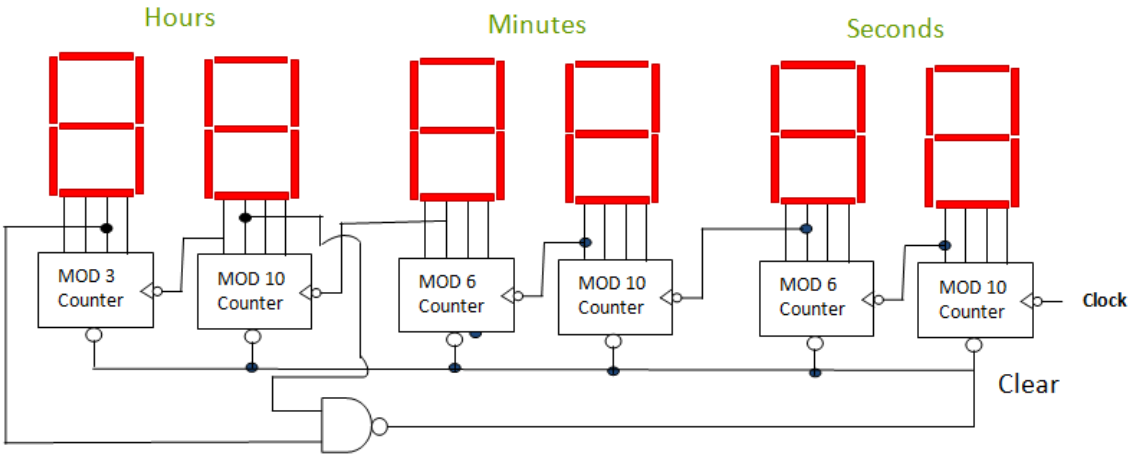
TÍNH TOÁN - LƯU TRỮ - TRUYỀN TẢI

"Mạng máy tính là công nghệ tiên tiến nhất của thời đại chúng ta. Nó đã mở ra cánh cửa cho việc chia sẻ thông tin, kết nối con người và thúc đẩy sự phát triển toàn cầu." - Bill Gates, nhà sáng lập Microsoft.

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 chọn kênh và trộn kênh. Tất cả chức năng đó được tiến hành với những mạch tổ hợp và được đóng gói thương mại dưới các họ IC.

5.1 Đồng hồ kỹ thuật số

Đồng hồ kỹ thuật số là một sự thay thế cho đồng hồ kim truyền thống. Loại đồng hồ này hiển thị số để hiển thị thời gian ở dạng kỹ thuật số như trên đồng hồ, điện thoại hay đồng hồ báo thức. Hiển thị này có thể ở cả định dạng 12 và 24 giờ. Đồng hồ kỹ thuật số có thể hiện thực bằng các bộ đếm phù hợp cho từng kí số ở giờ, phút và giây.



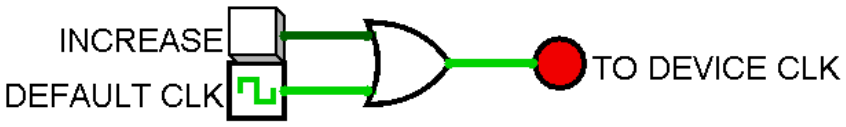
Hình 5.1: Đồng hồ từ các bộ đếm

Yêu cầu 1. (a) Hiện thực một đồng hồ từ các bộ đếm đồng bộ có cổng nạp song song.
(b) Hiện thực chức năng điều chỉnh giờ / phút.

Gợi ý. (a) Các linh kiện cần hiện thực bao gồm:

- Bộ đếm BCD cho kí số hàng đơn vị ở phút và giây.
- Bộ đếm MOD-6 cho kí số hàng chục ở phút và giây.
- Bộ đếm MOD-2 và MOD-4 cho giờ (nếu sử dụng hiển thị 24H).
- Các bộ giải mã BCD đến LED 7 đoạn và các đèn LED 7 đoạn; hoặc là các đèn LED 7 đoạn đã tích hợp giải mã như **Hex display**.
- Bộ sinh xung đồng hồ tần số 1Hz.

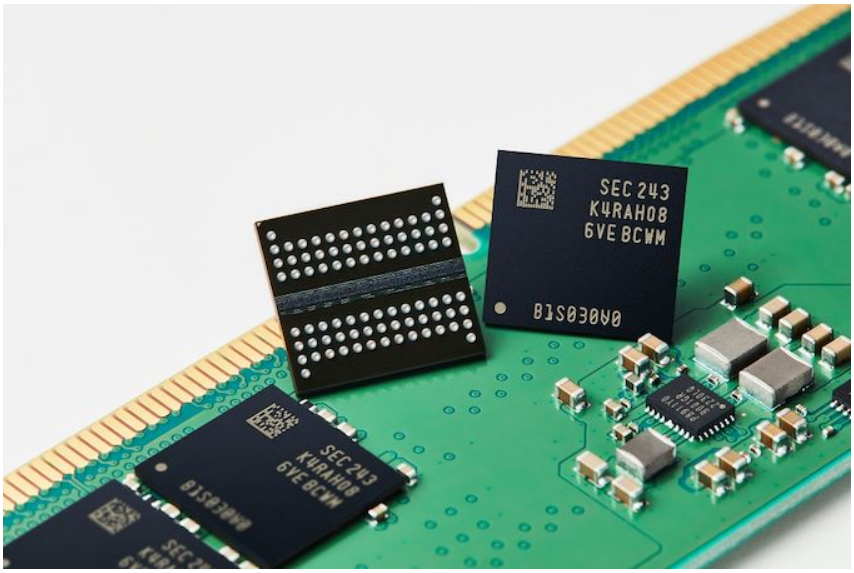
(b) Việc điều chỉnh giờ / phút có thể thực hiện bằng cách bổ sung xung đồng hồ có tần số nhanh để các bộ đếm này nhanh chóng thay đổi giá trị (thay vì phải chờ đúng thời điểm đếm). Tín hiệu bổ sung và xung đồng hồ mặc định được tổng hợp bằng cổng OR như hình dưới đây:



5.2 Bộ nhớ máy tính

Bộ nhớ máy tính đóng vai trò quan trọng khi chứa chương trình đang thực thi cùng với dữ liệu liên quan, những thông tin mà bộ vi xử lý sẽ lần lượt truy cập để đọc lệnh hoặc

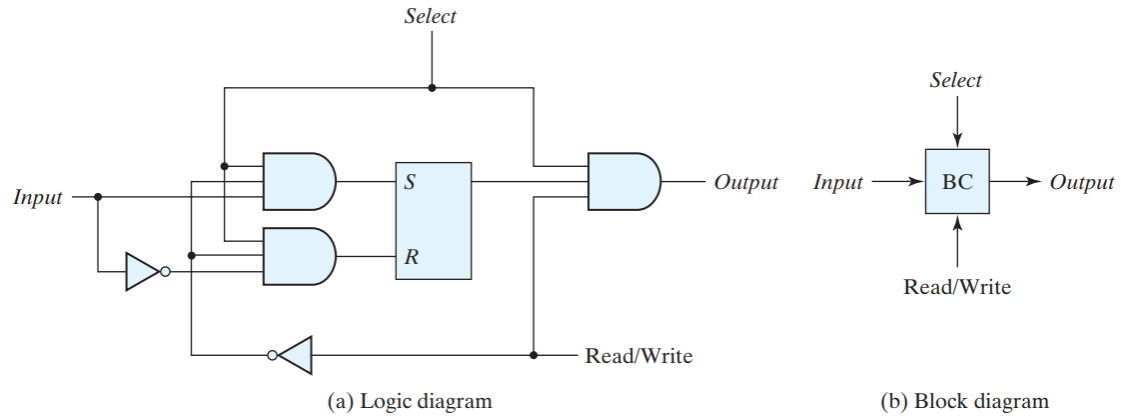
dữ liệu cho quá trình giải mã và thực thi trong một chu kỳ lệnh. Kết quả tính toán sau mỗi chu kỳ lệnh cũng được ghi lại vào bộ nhớ. Bởi vì bộ vi xử lý chỉ thao tác trên vùng nhớ 4 byte trong một lần đọc hoặc ghi, nên các byte đó cần định vị bằng địa chỉ trong một không gian bộ nhớ có lên lớn đến hàng chục Gigabye trong các máy tính cá nhân hiện nay. Thời gian cần thiết để truyền thông tin đến hoặc từ bất kỳ vị trí ngẫu nhiên mong muốn nào cũng luôn giống nhau - do đó bộ nhớ chính còn có tên gọi bộ nhớ truy cập ngẫu nhiên (random access memory, viết tắt là RAM).



Hình 5.2: Một khối nhớ DDR5 DRAM do Samsung sản xuất với công nghệ 12nm

Bộ nhớ được hiện thực bằng cách ghép nhiều khối nhớ đơn lẻ với nhau, mỗi khối nhớ lại được tổ chức từ nhiều mạch lật. Khi ghi một dữ liệu lên bộ nhớ, nội dung ghi được gửi trên đường truyền dữ liệu đồng thời vị trí ghi được gửi đi trên đường truyền địa chỉ. Ngược lại, thao tác đọc sẽ được tiến hành khi bộ vi xử lý cung cấp địa chỉ cần truy cập và dữ liệu đang được lưu trữ tại địa chỉ đó sẽ được truyền lại về cho CPU. Đường truyền địa chỉ được các bộ giải mã xử lý và chỉ định một khối nhớ chứa địa chỉ đó được kích hoạt, các khối nhớ khác sẽ bị vô hiệu quá.

Yêu cầu 2. Thiết kế một ô nhớ 1 bit có các cổng điều khiển READ/WRITE và SELECT để điều khiển ghi dữ liệu từ cổng vào Input hoặc đọc dữ liệu từ cổng ra Output. Tham khảo hình 5.3.

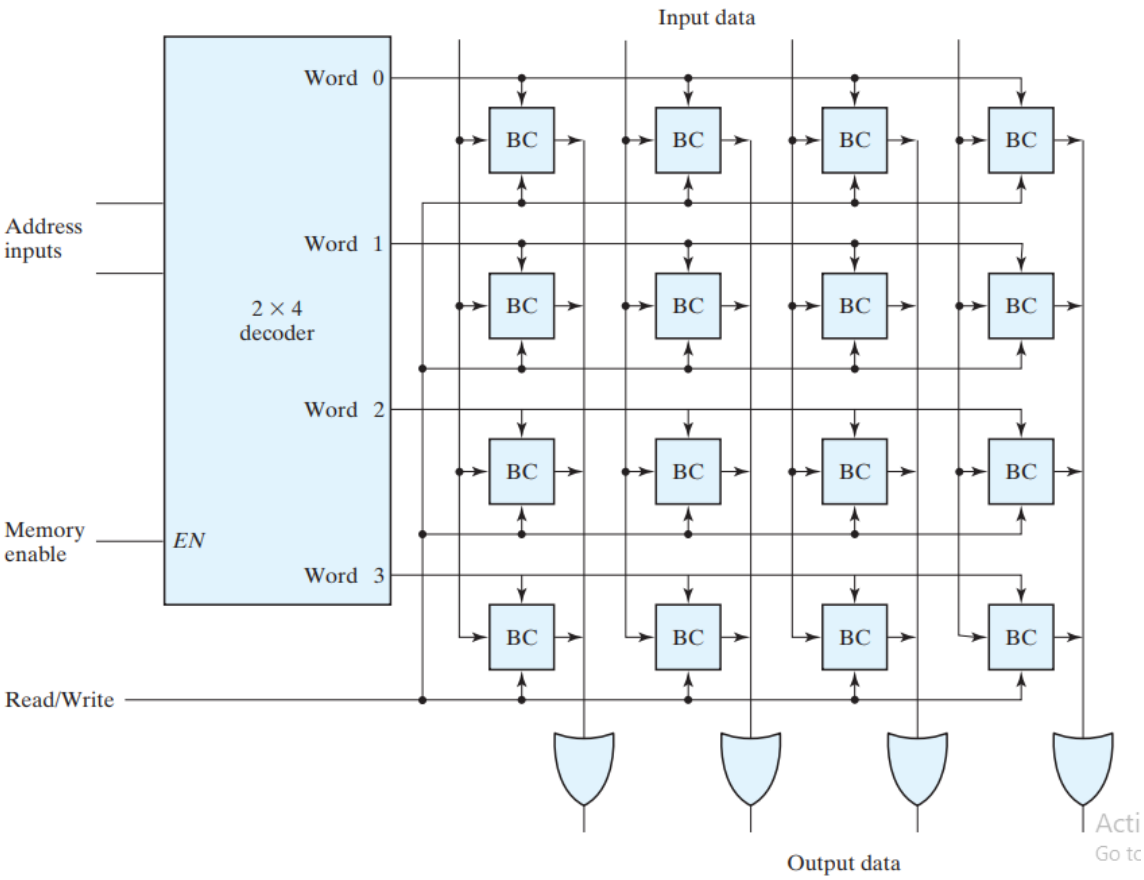


Hình 5.3: Một ô nhớ 1 bit

Yêu cầu 3. (a) Thiết kế một khối nhớ có kích thước 4x4 bằng cách sử dụng 4 ô nhớ cho một Word và bộ nhớ có 4 Word. Sử dụng bộ giải mã Decoder 2x4 để giải mã địa chỉ.

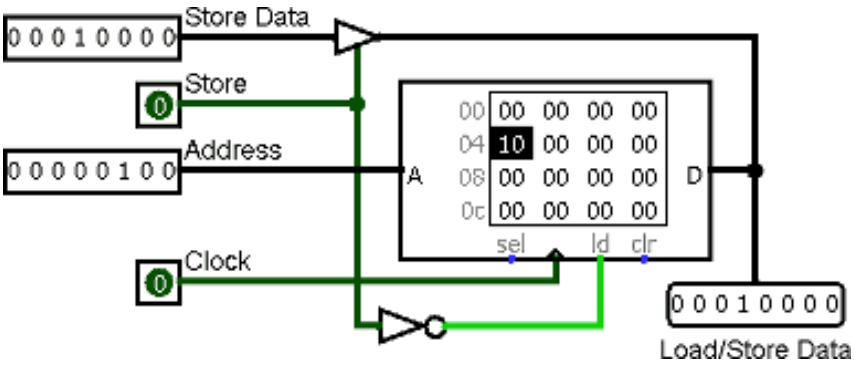
(b) Mô tả quá trình ghi giá trị $0b1001$ vào địa chỉ $0x1$.

(d) Mô tả quá trình đọc giá trị tại địa chỉ $0x3$.



Hình 5.4: Bộ nhớ RAM 4x4

Yêu cầu 4. Khảo sát linh kiện **RAM** trong Logisim và hiện thực một bộ nhớ RAM 4x4.



Hình 5.5: Bộ nhớ RAM 256x8 trong thư viện Logisim

5.3 Bộ truyền tuần tự

5.4 Kết luận

- Mạch cộng BCD.

Phụ lục A

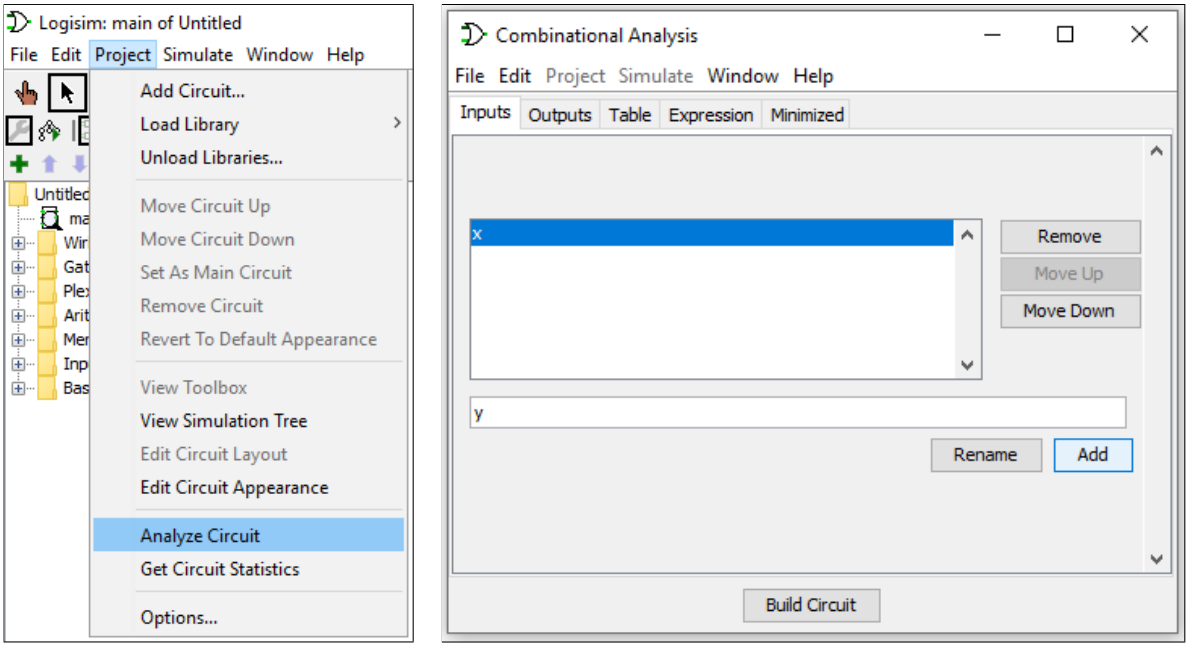
Ứng dụng Logisim

A.1 Vẽ mạch logic từ biểu thức Boole

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

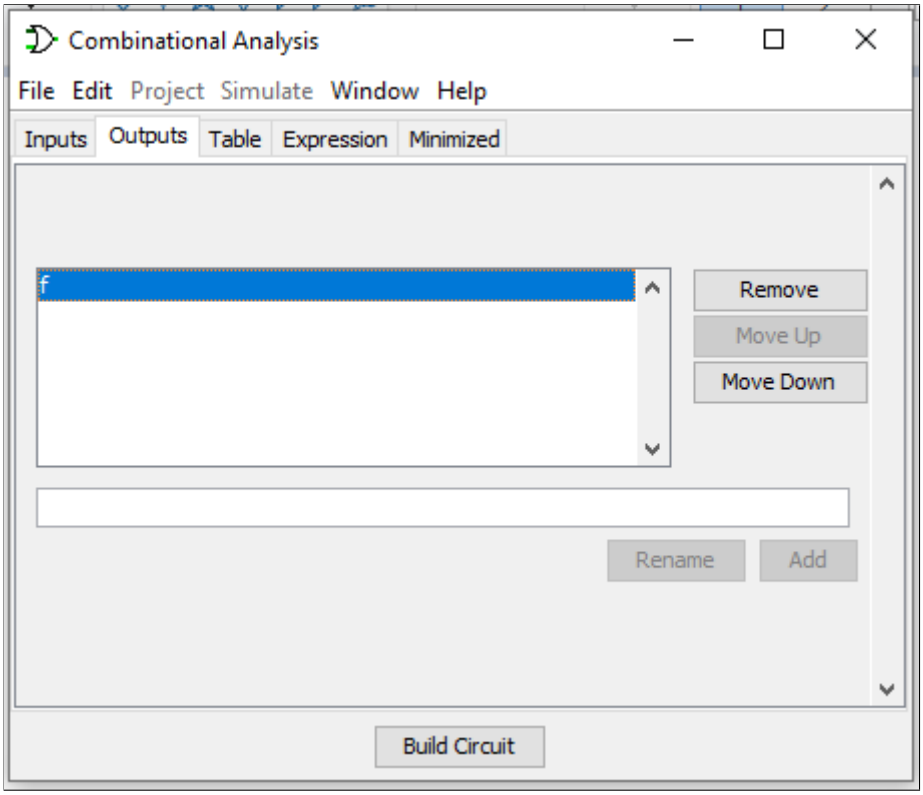
Bước 2: Nhập các biến số vào (và sắp xếp thứ tự nếu cần thiết) tại thẻ **Inputs** như hình A.1.

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

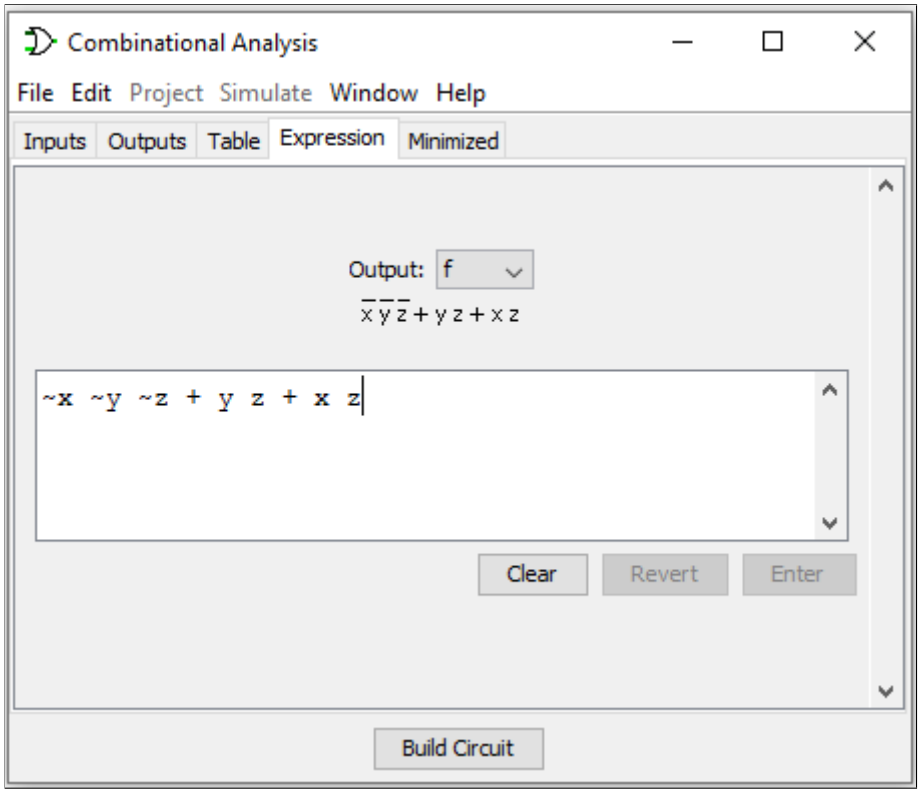


Hình A.2: Chức năng "Phân tích mạch logic"

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)

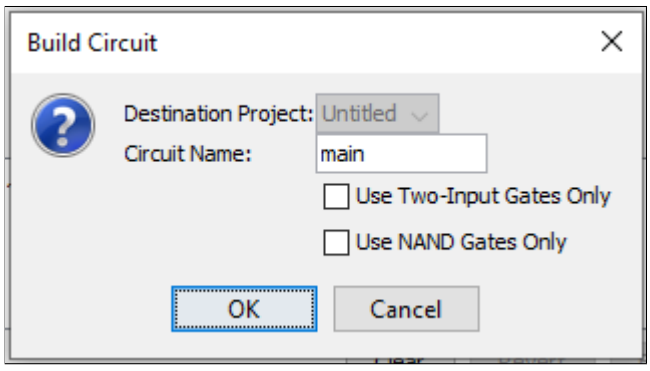
Bảng A.1: Cách viết biểu thức trong Logisim



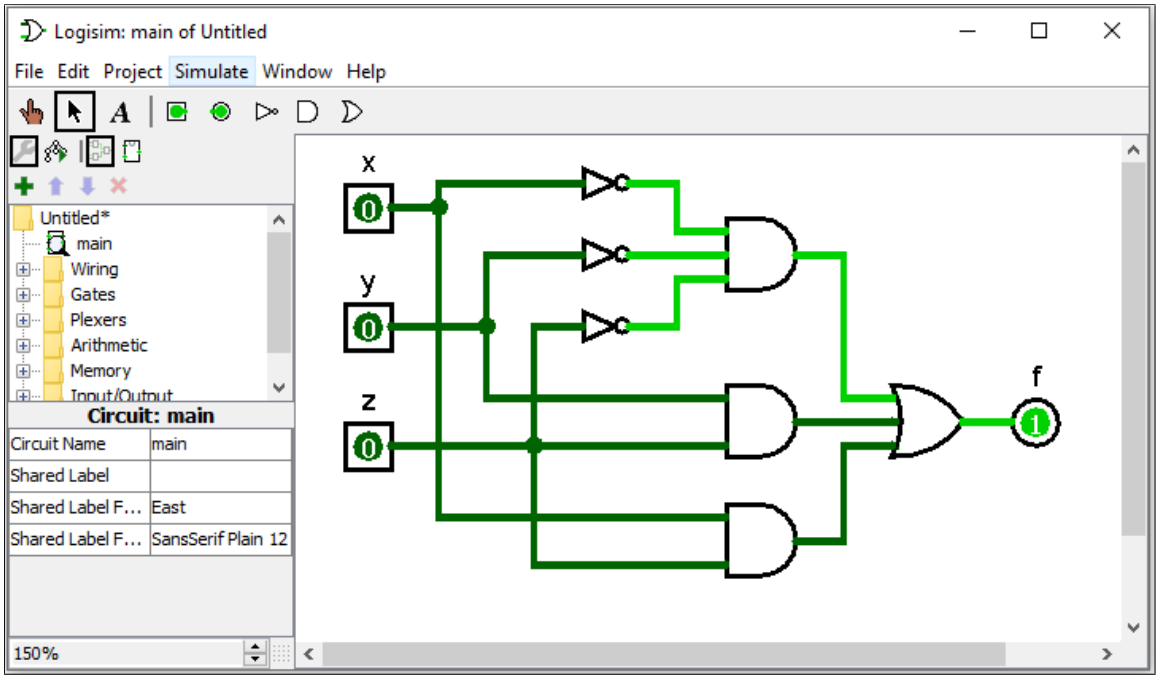
Hình A.3: Chức năng "Nhập vào biểu thức cho từng đầu ra"

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



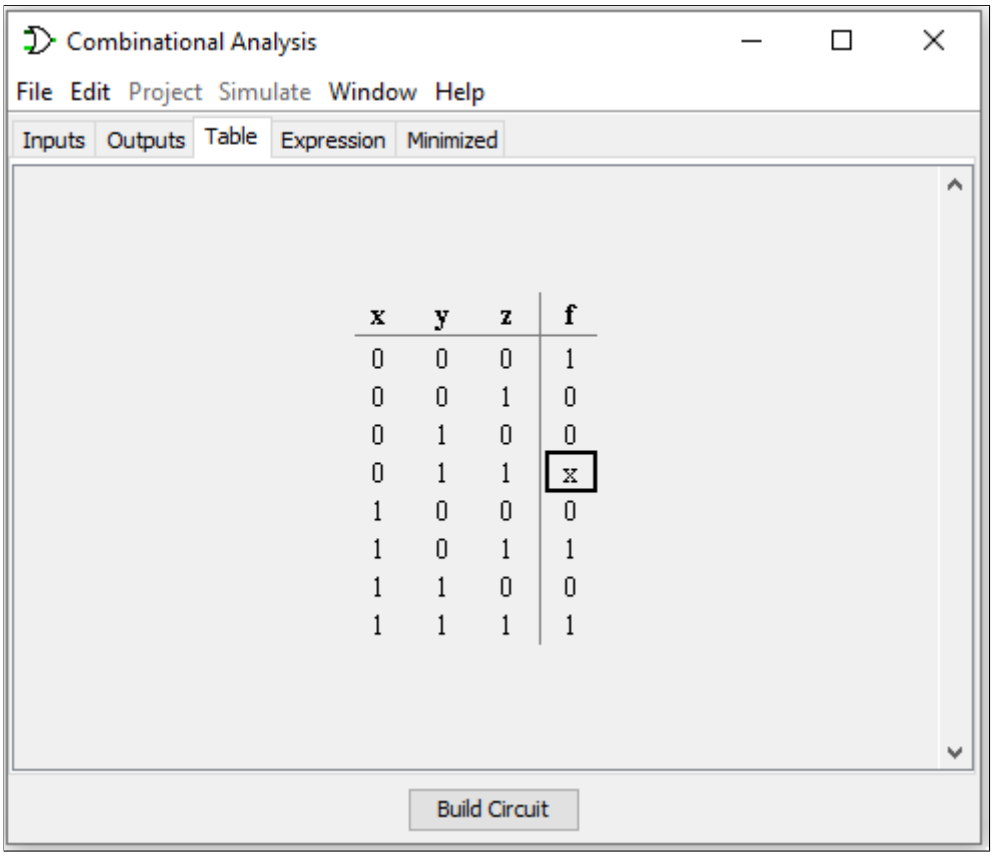
Hình A.5: Chức năng "Nhập vào biểu thức cho từng đầu ra"

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

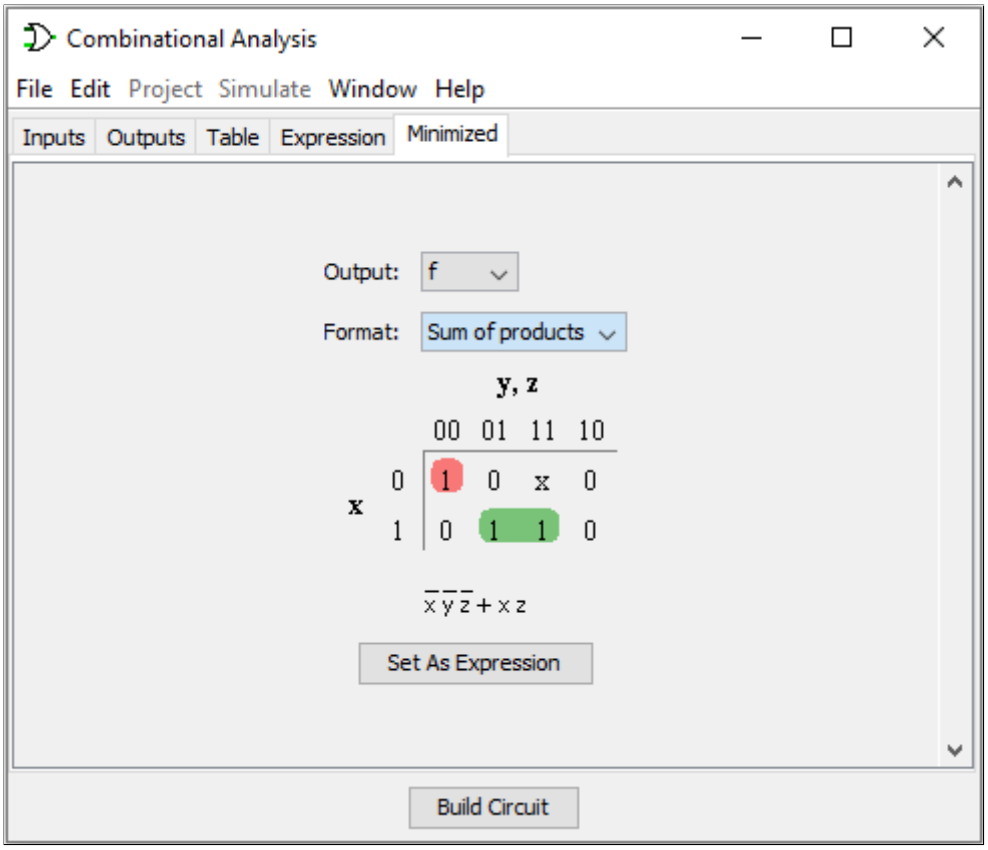
A.3 Rút gọn mạch logic và Bìa-K

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

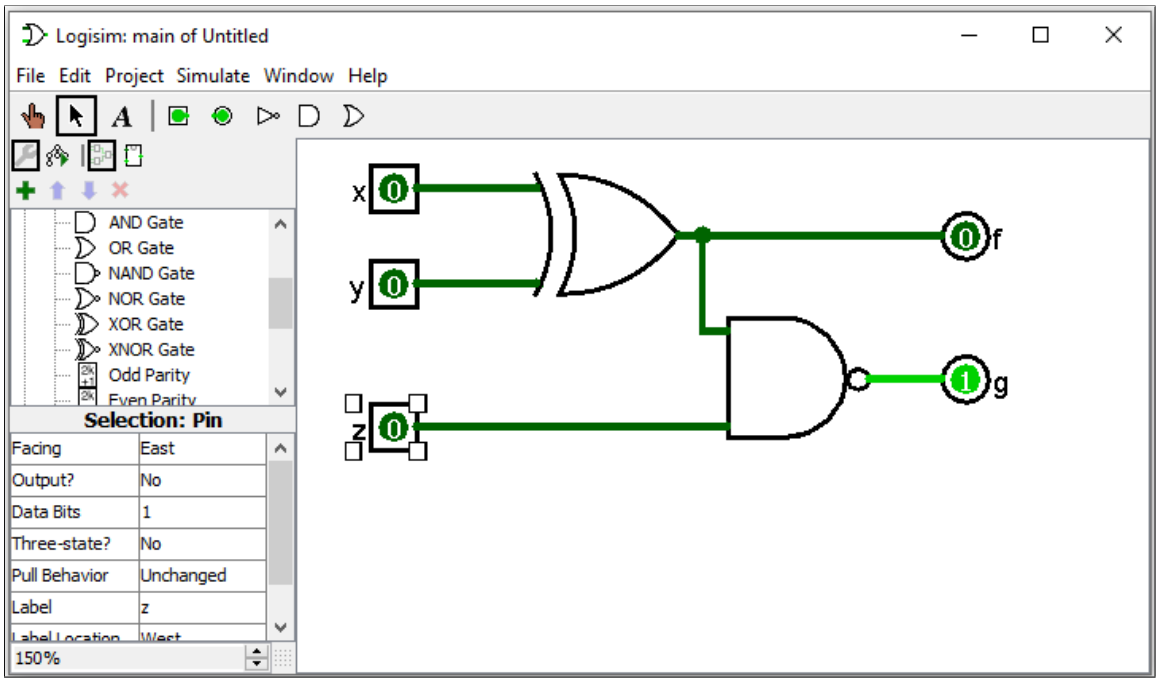
Bước 1: Tại màn hình làm việc chính, bắt đầu vẽ mạch logic cần phân tích như hình A.8.

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**.

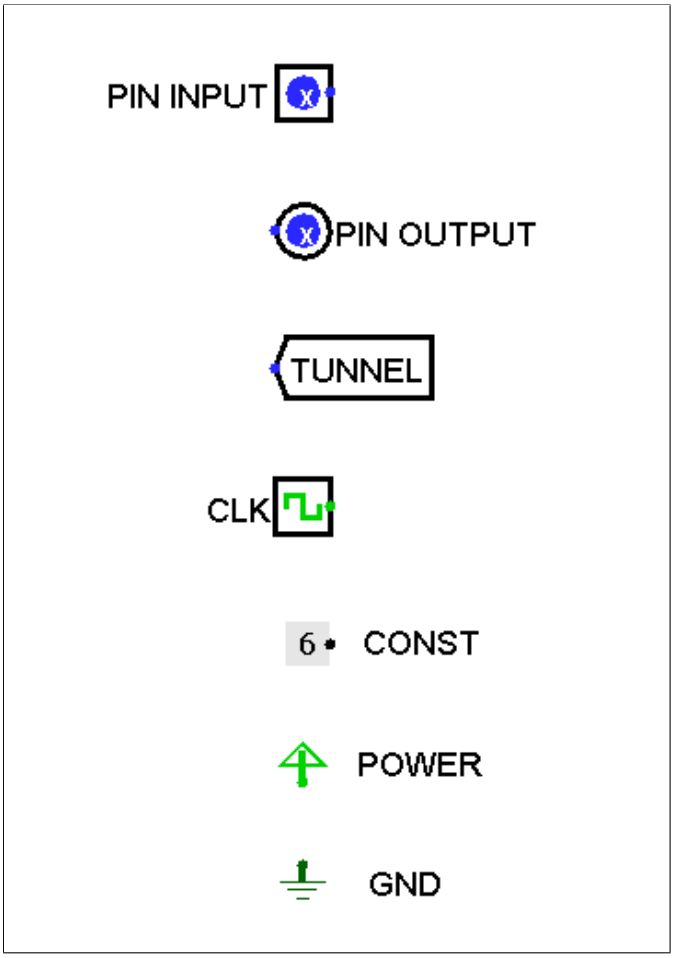
Bước 4: Đặt các cổng cần thiết vào và kết nối dây dẫn. Tham khảo thêm Hướng dẫn của Logisim cho các thao tác này.

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.