**Lab 1: Multisim Circuit Simulation**

Trong Lab1, chúng ta học cách sử dụng phần mềm mô phỏng để mô phỏng các mạch số. Sử dụng môi trường Multisim, chúng thể mô phỏng các mạch để học các khái niệm cơ bản về điện tử số mà không cần các linh kiện vật lý.

Bên cạnh đó, chúng sẽ tạo mạch số và triển khai chúng lên bảng phát triển Hệ thống Số NI. Trong bài học sau, chúng sẽ tìm hiểu về Programmable Logic Devices (PLD) và một loại PLD gọi là FiledProgrammable Gate Array (FPGA). FPGA là công nghệ cho phép các nhà thiết kế mạch xây dựng mạch trên phần cứng có thể cấu hình lại. Khi thiết kế mạch cần thay đổi, nhà thiết kế có thể thay đổi mạch trên phần cứng và triển khai lại lên FPGA thay vì hàn mạch mới. Trong loạt bài thí nghiệm này, chúng ta sẽ sử dụng FPGA trên Digital Electronics Board để triển khai các mạch thiết kế trong Multisim.

Trong lab1 sẽ:

1. Mở các mạch và thiết kế PLD đã được cấu hình sẵn trong Multisim.
2. Cấu hình và xây dựng các mạch và thiết kế PLD trong Multisim.
3. Mô phỏng các mạch số đơn giản.
4. Triển khai thiết kế PLD lên FPGA.
   1. Simulate: Circuits in Multisim

AND gate

|  |  |  |
| --- | --- | --- |
|  |  |  |

Để “A” là 1, “B” là 0 (Ques1: Khi này đèn không sáng), Để “A” là 0, “B” là 1 (Ques2: Đèn không sáng), Để “A” là 1, “B” là 1 (Ques3: đèn sáng)

Ques4: Đèn sáng khi cả hai cổng “A” và “B” đều là “1”, đèn không sáng khi một trong hai cổng có trạng thái là “0”.

* 1. Exercise: Creating a New Multisim Circuit

OR gate

|  |  |  |
| --- | --- | --- |
|  |  |  |

Để “A” là 1, “B” là 0 (Ques5: Đèn sáng), để “A” là 0, “B” là 1 (Ques6: Đèn sáng), Để “A” và “B” đề bằng 1 (Ques7: Đèn sáng).

Ques8: Đèn sáng khi một trong hai cổng đầu vào “A” hoặc “B” có trạng thái là 1 hoặc cả hai cổng “A” và “B” có trạng thái là 1.

* 1. Exercise: Building Circuits on the Digital Electronics Board
  2. Exercise: Creating a New PLD Design

(Trong tài liệu hướng dẫn, không trình bày lại)

* 1. Building and Testing an OR Gate on the Digital Electronics Board

Từ Ques9-12: mạch chạy đúng như mô phỏng cổng OR

* 1. Conclusion

Ques13: Việc mô phỏng mạch trong Multisim thường nhanh hơn so với chạy mạch trên FPGA, vì mô phỏng không cần thời gian để cấu hình và triển khai phần cứng.

Ques14: Nên mô phỏng mạch trong Multisim khi cần kiểm tra, thiết kế, hoặc sửa lỗi mạch một cách nhanh chóng mà không cần triển khai trên phần cứng.

Chạy mạch trên FPGA khi cần kiểm tra hoạt động thực tế của mạch trên phần cứng.

Ques15: Môi trường NI multisim tạo các mạch ảo mà không cần các linh kiện vật lý

Ques16: Ưu điểm của việc tạo mạch trong PLD là chúng có thể triển khai trên Digital Electronics Board

Ques17: interactive digital constant trong mạch cho phép tạo ra các mức logic cao (1) và thấp (0)

Ques18: Có thể thay đổi giá trị cao và thấp thống qua interactive digital constant

Ques19: Lợi ích của FPGA là có thể cấu hình lại phần cứng.

**Lab 2: Truth Tables and Basic Logic Gates**

Phần cứng số xuất hiện trong hầu hết mọi khía cạnh của cuộc sống hàng ngày, là một phần của máy tính cá nhân, thiết bị gia dụng, robot, mạng truyền hình,...

Các mạch logic là nền tảng của phần cứng số. Chúng thực hiện các phép toán trên tín hiệu số và thường được triển khai dưới dạng mạch điện tử, trong đó giá trị tín hiệu bị giới hạn ở một vài giá trị rời rạc. Phổ biến nhất là các mạch logic nhị phân, nơi các giá trị chỉ là 0 và 1.

Ba phép toán logic cơ bản bao gồm:

* Phép toán AND logic
* Phép toán OR logic
* Phép toán NOT (đảo ngược)

Các phép toán logic được thực hiện bằng các cổng logic. Một cổng logic là một mạch điện tử được tạo thành từ các transistor. Thông tin liên quan đến cổng logic và các hàm logic có thể được mô tả bằng bảng chân lý (truth table).

Trong Lab2:

1. Các cấu hình khác nhau của cổng logic.
2. Cấu hình và xây dựng các mạch và thiết kế PLD trong Multisim.
   1. Theory and Background

Bảng Chân Lý (truth table)  
Cách phổ biến để biểu diễn chức năng cụ thể của một mạch logic là sử dụng bảng chân lý. Bảng chân lý hiển thị tất cả các tổ hợp của đầu vào với các giá trị đầu ra tương ứng dưới dạng các mức logic. Các mức logic thường được biểu diễn dưới dạng:

* 1 và 0
* Cao (HIGH) và Thấp (LOW)
* Đúng (True) và Sai (False)

Ví dụ với truth table với 2 biến đầu vào:



Cổng Logic  
Cổng logic là các thiết bị vật lý thực hiện các hàm Boolean của bảng chân lý. Hai cổng logic cơ bản nhất là "AND" và "OR".

Với cổng "AND", đầu ra sẽ là 1 nếu cả hai đầu vào A và B đều là 1. Nếu một hoặc cả hai đầu vào A và B là 0, thì đầu ra sẽ là 0.

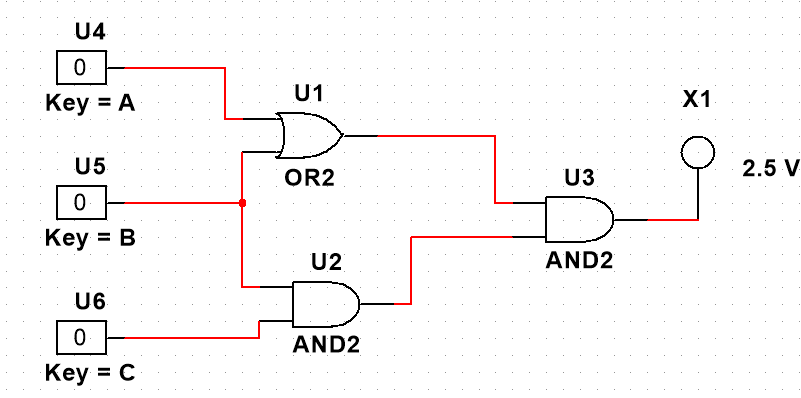
|  |  |
| --- | --- |
|  |  |

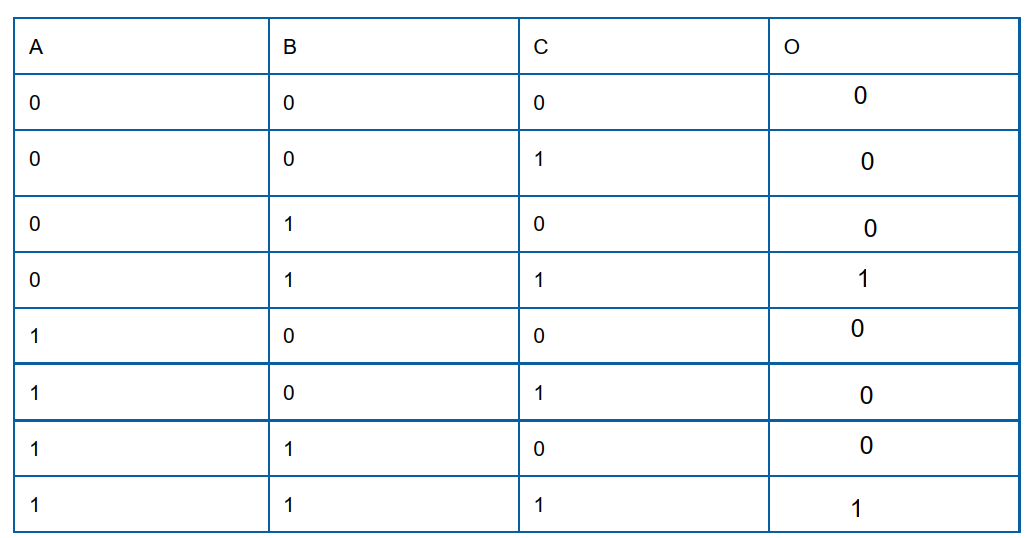
Trong cổng "OR", đầu ra sẽ là 0 nếu cả hai đầu vào A và B đều là 0. Nếu một hoặc cả hai đầu vào A và B là 1, thì đầu ra sẽ là 1.

|  |  |
| --- | --- |
|  |  |

* 1. Simulate: Building a Circuit with Multiple Gates

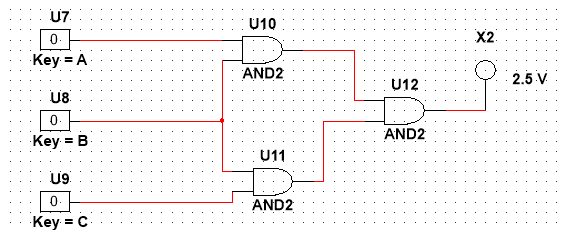
Circuit Example 1

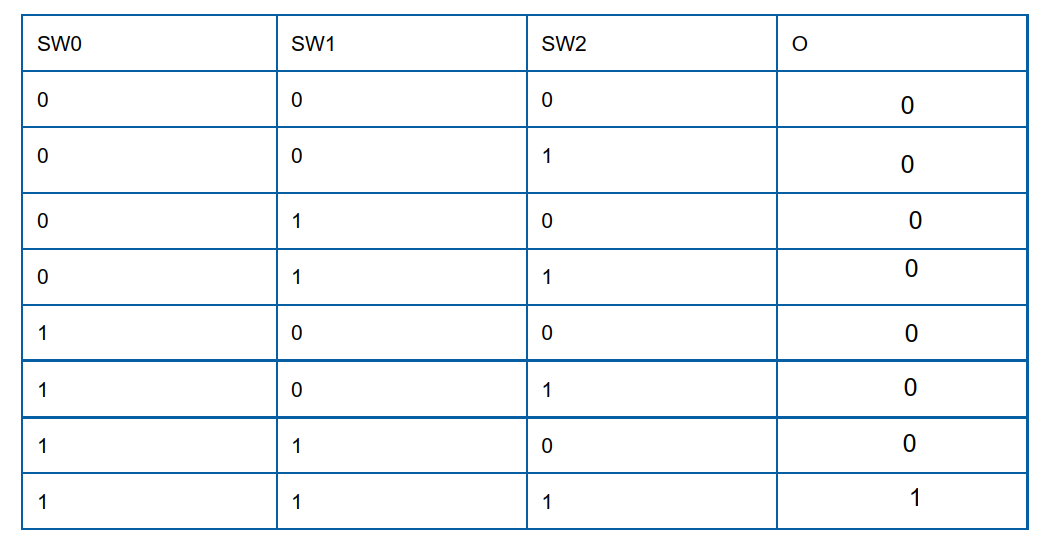




* 1. Simulate: Building a Circuit with Multiple Gates

Circuit Example 2





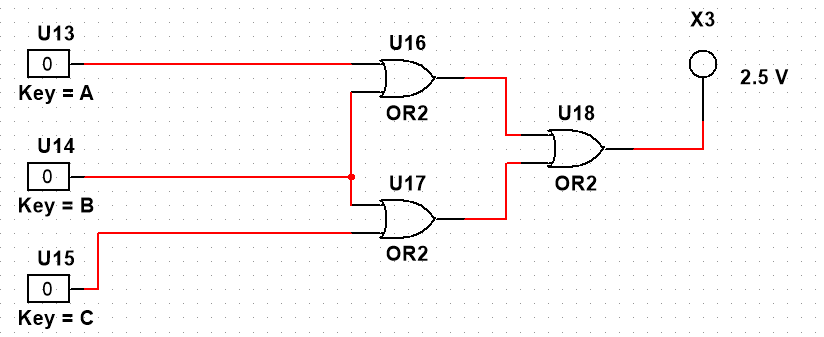
* 1. Exercise: Determining a Circuit from a Truth Table

Từ kết quả của phần trên *Circuit Example 2* mạch chạy đúng như mô phỏng, mạch 2 có thể mô tả là

Ví dụ khác: Với truth table sau



Bảng này có thể mô tả là:



Ques7: Mạch chạy đúng

* 1. Conclusion

Ques9: Dùng hay truth table vì bảng chân lý cho thấy mọi tổ hợp đầu vào và kết quả đầu ra tương ứng, giúp xác định chính xác chức năng của mạch và phát hiện lỗi nếu có.

Ques10: Sự khác nhau giữa “AND” và “OR” là:

Cổng logic "AND" chỉ xuất ra 1 khi tất cả các đầu vào đều là 1; nếu một hoặc nhiều đầu vào là 0, đầu ra sẽ là 0.

Cổng logic "OR" sẽ xuất ra 1 nếu ít nhất một trong các đầu vào là 1; chỉ khi tất cả các đầu vào đều là 0, đầu ra mới là 0.

Ques11: Các cổng logic cơ bản là “AND”, “OR” và “NOT”

Ques12: Truth table: Thể hiện chức năng cụ thể của mạch logic; Hiển thị tất cả các hoán vị có thể có của đầu vào và các giá trị đầu ra tương ứng; Được định lượng dưới dạng trạng thái mức logic.

Ques13: Truth table có 3 biến đầu vào có thể có khả năng đầu ra

Ques14: Cổng “AND” có 2 đầu vào và 1 đầu ra

Ques15: Cổng “OR”: thể hiện giá trị lớn nhất giữa hai đầu vào

**Lab 3: Logic Gates Explored and Boolean Algebra**

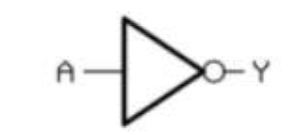
Trong Lab2, chúng ta đã được tìm hiểu về hai cổng logic cơ bản – AND và OR. Dựa trên những cổng này, chúng ta có thể tạo ra một số loại cổng logic khác, bao gồm: bộ đảo (NOT), NAND, NOR, XOR và XNOR.

Trong Lab3 sẽ:

1. Tìm hiểu chức năng của các cổng logic khác nhau.
2. Thiết kế mạch sử dụng các cổng logic đa dạng cả trong lý thuyết và thực hành.
3. Tính toán và xây dựng các mạch logic tổ hợp từ dạng tổng của tích (Sum-of-Products) và tích của tổng (Product-of-Sums) dựa trên truth table.
   1. Theory and Background

Cổng NOT

Hàm:

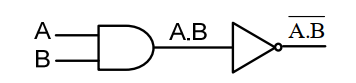
Khí hiệu: 

Truth table:

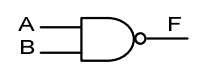
|  |  |
| --- | --- |
| A | f |
| 0 | 1 |
| 1 | 0 |

Cổng NAND

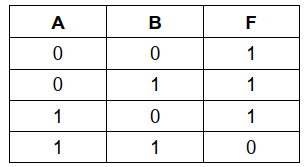
Hàm:



Khí hiệu:



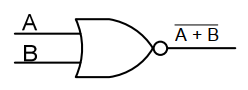
Truth table:



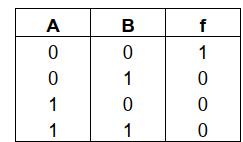
Cổng NOR

Hàm:

Kí hiệu:



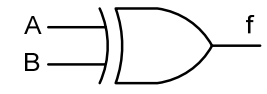
Truth table:



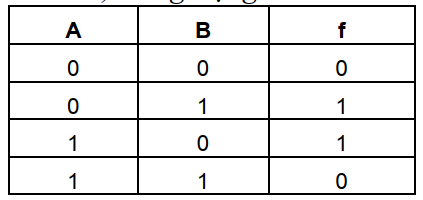
Cổng XOR

Hàm:

Khí hiệu:

**

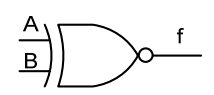
Truth table:



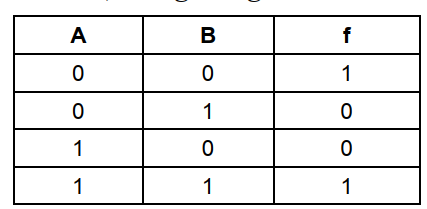
Cổng XNOR

Hàm:

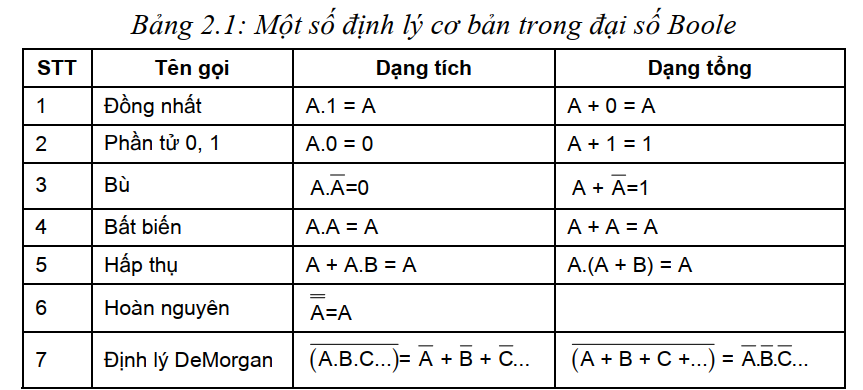
Kí hiệu:



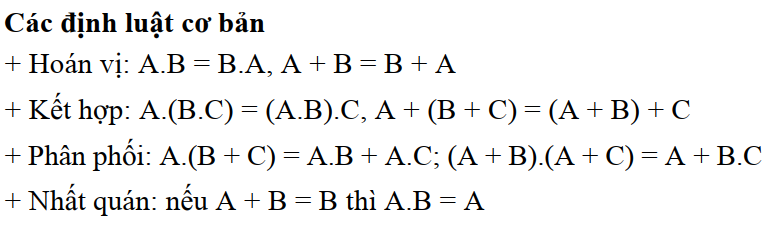
Truth table:



Đại số Boolean



Các định luật cơ bản



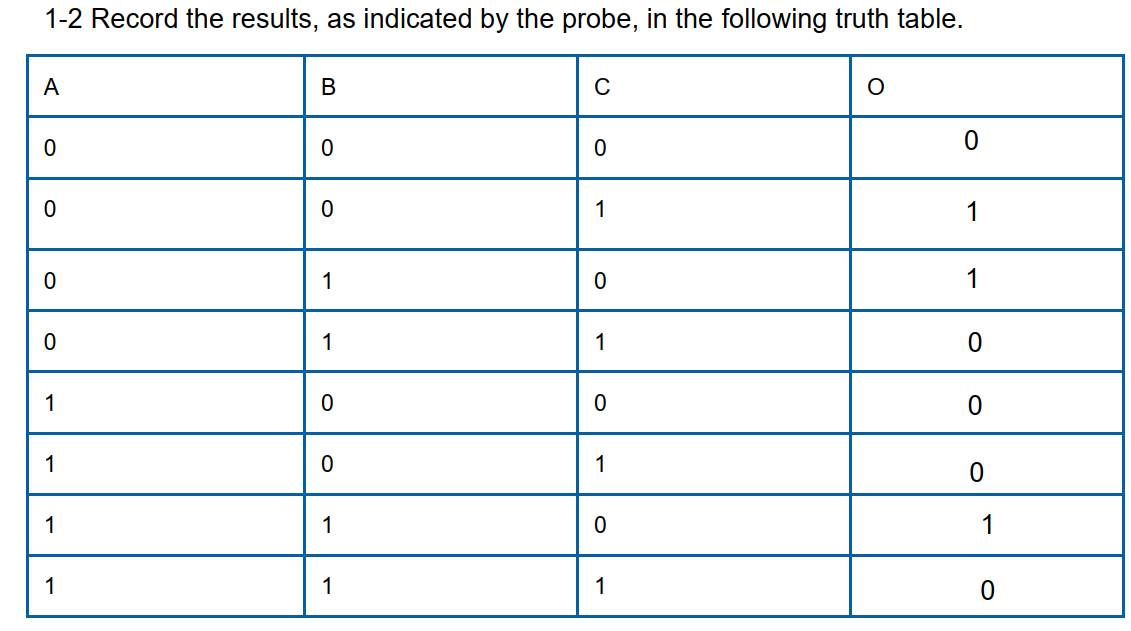
|  |  |
| --- | --- |
|  |  |
| hoặc  *O2​=(A+B+C)(A′+B+C)* |

* 1. Simulate: Building a CLC Circuit - Example 1

Có



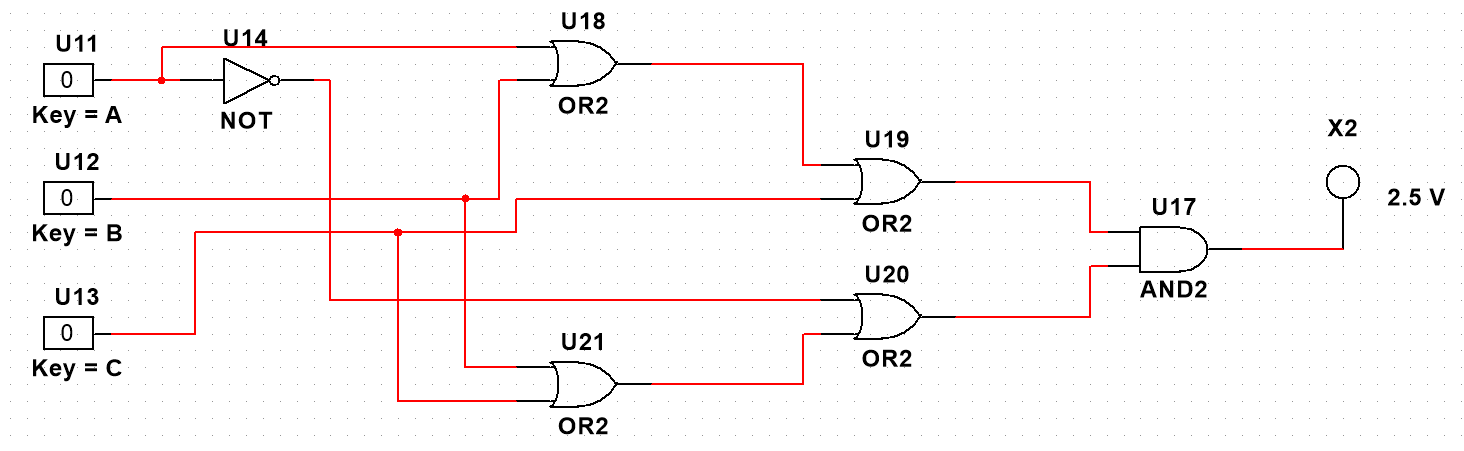
Kết quả:



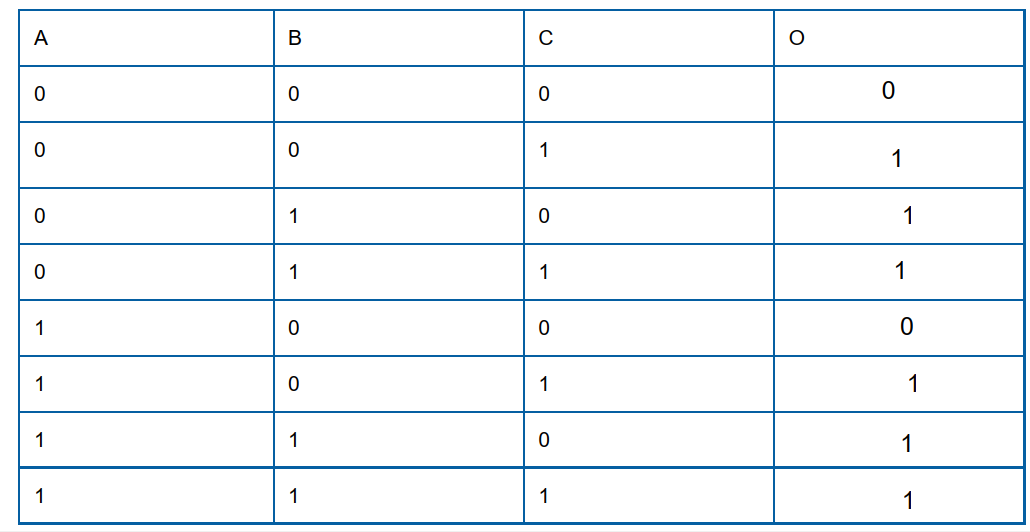
Ques3: Kết quả chạy mô phỏng mạch giống với trạng thái đầu ra của

* 1. Simulate: Building a CLC Circuit - Example 2

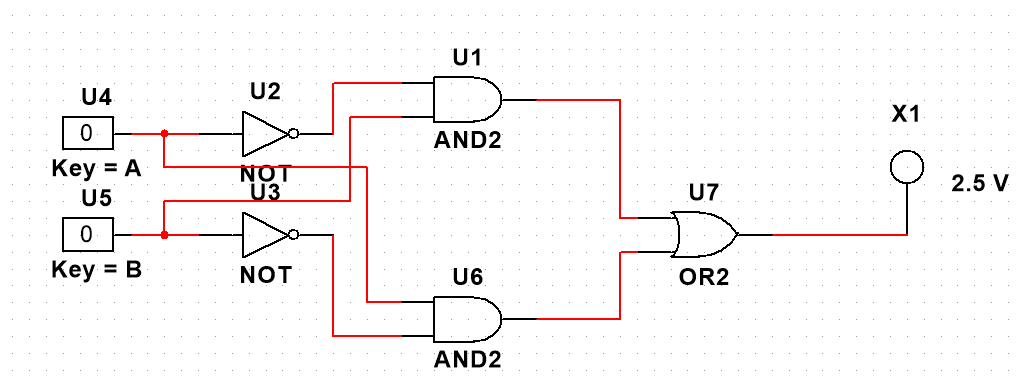
Có



Kết quả:



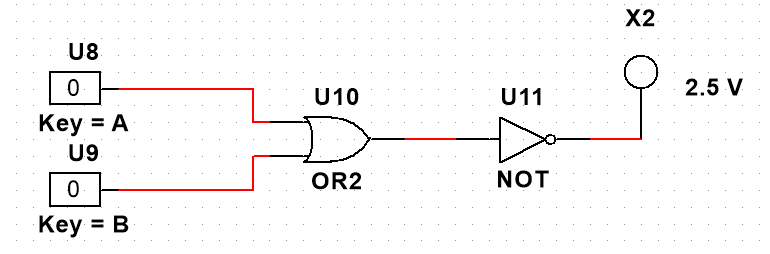
* 1. Exercise: Building an XOR Logic Gate in Multisim



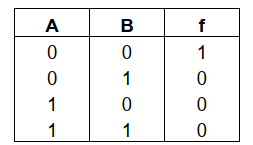
Kết quả:



* 1. Exercise: Building a NOR Logic Gate on the Digital Electronics Board



Kết quả:



* 1. Conclusion

1-15. Sự khác biệt giữa SOP và POS?

SOP (Sum of Products - Tổng của các tích): Là dạng biểu thức Boolean được tạo thành từ các minterm (tích của các biến) được cộng lại với nhau. SOP được thực hiện bằng cách sử dụng các cổng AND kết nối vào một cổng OR.

POS (Product of Sums - Tích của các tổng): Là dạng biểu thức Boolean được tạo thành từ các maxterm (tổng của các biến) được nhân lại với nhau. POS được thực hiện bằng cách sử dụng các cổng OR kết nối vào một cổng AND.

1-16. nhiều thiết kế mạch có thể tạo ra cùng một bảng truth table.

1-17. Do các quy tắc của đại số Boolean, một số biến và thuật toán có thể được rút gọn.

Ví dụ:

Luật hấp thụ: A+AB=AA + AB = AA+AB=A

Luật bù trừ: A+A′B=A+BA + A'B = A + BA+A′B=A+B

Luật kết hợp: AB+AC=A(B+C)AB + AC = A(B + C)AB+AC=A(B+C)

Việc rút gọn giúp giảm số lượng cổng logic cần thiết, tối ưu hóa thiết kế mạch và giảm chi phí phần cứng.

1-18. Mạch logic tổ hợp (CLC) là Một mạch sử dụng đại số Boolean và chỉ phụ thuộc vào đầu vào hiện tại

Mạch logic tổ hợp là mạch mà đầu ra chỉ phụ thuộc vào giá trị đầu vào hiện tại và không có trạng thái nhớ.

1-19. Các minterm được sử dụng để xác định tổng của các tích (Sum of Products - SOP

1-20. Khi tạo một mạch từ dạng Tích của Tổng (POS), nó sẽ được xây dựng bằng cách: Các cổng OR đầu ra nối vào một cổng AND

1-22. Khi tạo một mạch trong PLD, các hằng số số học tương tác được thay thế bằng Công tắc (Switches)

Khi triển khai mạch trong PLD (Programmable Logic Device), các giá trị hằng số số học được thay thế bằng công tắc (switches) để có thể điều chỉnh đầu vào.

**Lab 4: Binary Conversion and Adders**

Trong lab4, chúng ta đã tìm hiểu về bảng chân lý với hai đầu vào và cách thiết kế các mạch tương ứng. Hai đầu vào nhị phân, cụ thể là 1 và 0, là các mạch đơn giản nhất. Các mạch phức tạp hơn có nhiều tổ hợp số nhị phân hơn, điều này khiến việc tạo bảng chân lý cho tất cả các tổ hợp và hoán vị có thể trở nên không thực tế.

Thay vì tạo bảng chân lý, chúng ta hướng tới một hệ thống chuyển đổi số nhị phân thành số thập phân mã hóa nhị phân (BCD - Binary-Coded Decimal). Tùy thuộc vào thứ tự và vị trí của từng số trong dãy, chúng được gán một giá trị. Tổng của các giá trị này sẽ tạo ra số BCD.

Trong Lab4 sẽ:

* Xây dựng Half Adder và bộ cộng toàn phần (Full Adder) bằng cách sử dụng các cổng logic và tạo bảng chân lý từ chúng.
* Kết nối nhiều bộ cộng toàn phần lại với nhau để tìm hiểu hiệu ứng lan truyền (ripple effect).
* Xác nhận bảng chân lý của bộ cộng toàn phần.
  1. Theory and Background

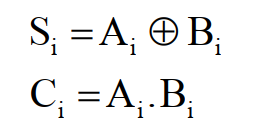
Bộ cộng bán phần (Half- adder - HA)

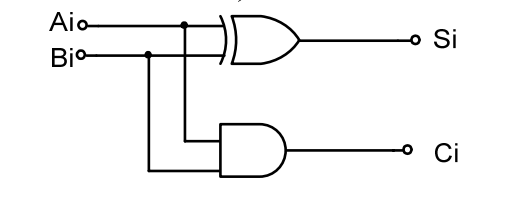
Mạch công bán phần có hai đầu vào : là chữ số cột thứ của số ; là chữ số cột thứ của số . Mạch có 2 đầu ra: là kết quả phép cộng ở cột thứ ; giá trị nhớ sang cột có trọng số cao hơn kế tiếp.

Bảng trạng thái của mạch HA

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

Từ bảng trạng thái, suy ra biểu thức logic của HA



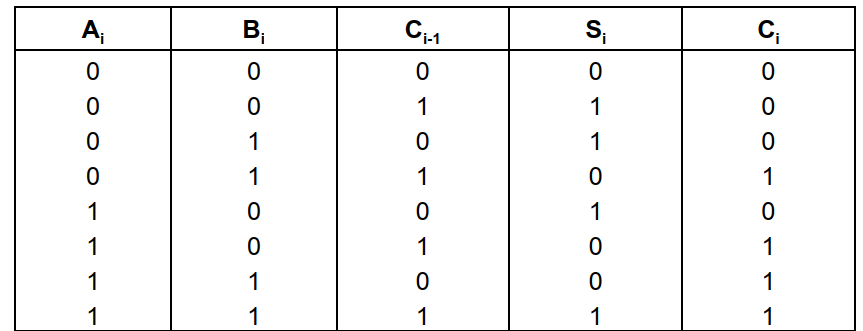


Bộ cộng toàn phần (Full adder -FA)

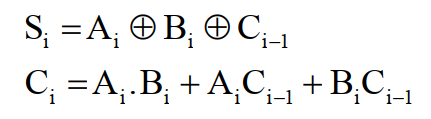
Mạch có ba đầu vào: là chữ số ở cột thứ của số ; là chữ số ở cột thứ của số B; là bít nhớ của trọng số nhỏ hơn liền kề chuyển đến.

Mạch có 2 đầu ra: là kết quả của phép cộng thứ ; là bít nhớ sang trọng số lớn hơn kế tiếp.

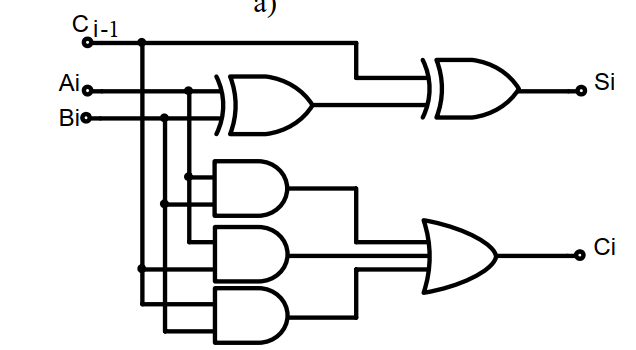
Bảng trạng thái của FA:



Từ bảng trạng thái suy ra hàm logic của FA



Mạch logic của FA



Ques1: Sử dụng bảng chân lý trở nên khó khăn khi số lượng chữ số nhị phân tăng lên vì:

Khi số lượng chữ số nhị phân tăng lên, số lượng tổ hợp đầu vào cũng tăng theo cấp số nhân (, với n là số bit đầu vào). Điều này làm cho bảng chân lý trở nên rất lớn và khó xử lý. Khi số bit quá nhiều, việc xây dựng và kiểm tra toàn bộ bảng chân lý trở nên không thực tế, đặc biệt đối với các mạch logic phức tạp.

Ques2: Điểm giống và khác nhau giữa bộ cộng nửa (Half Adder) và bộ cộng toàn phần (Full Adder):

Giống nhau: Cả hai đều là mạch số học dùng để thực hiện phép cộng nhị phân; cả hai đều có hai đầu ra: tổng (Sum) và bit nhớ (Carry-out).

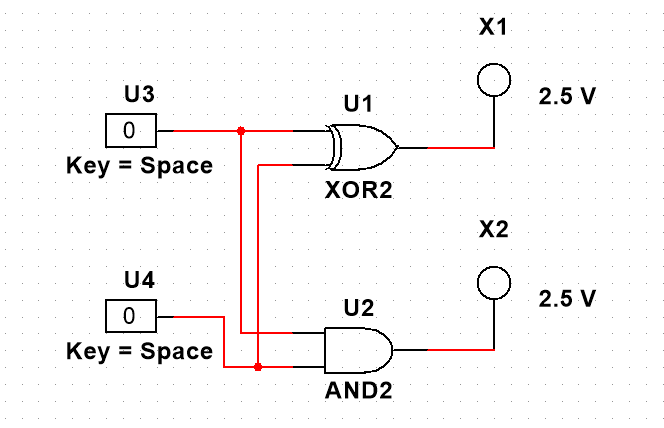
Khác nhau: Bộ cộng nửa (Half Adder) chỉ có hai đầu vào (A và B) và không có đầu vào cho bit nhớ (Carry-in), do đó nó chỉ có thể cộng hai bit đơn lẻ; Bộ cộng toàn phần (Full Adder) có ba đầu vào (A, B, và Carry-in), cho phép nó cộng thêm bit nhớ từ phép cộng trước đó, giúp cộng nhiều bit với nhau một cách chính xác; Bộ cộng toàn phần có thể được kết nối với nhau để tạo thành bộ cộng nhiều bit (Ripple Carry Adder), trong khi bộ cộng nửa không thể thực hiện điều này một cách trực tiếp.

Ques3: Khi các bộ cộng toàn phần (Full Adders) được kết nối với nhau sao cho (Carry-out) của bộ cộng trước được đưa vào (Carry-in) của bộ cộng tiếp theo, quá trình này được gọi là Ripple Carry.

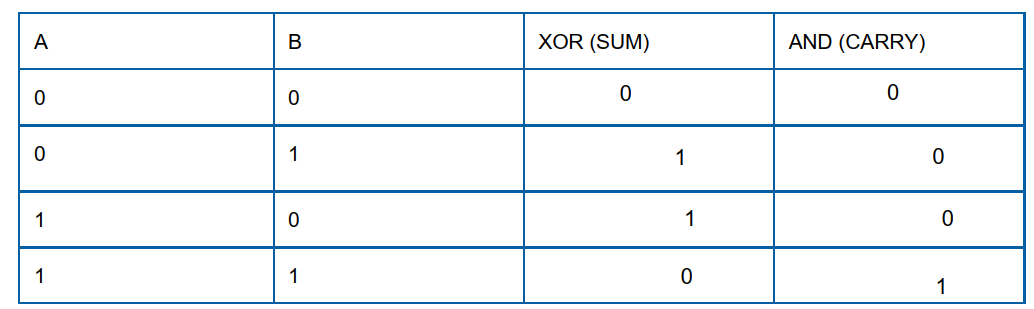
Điều này có nghĩa là bit nhớ (Carry) được truyền tuần tự từ bit thấp nhất đến bit cao hơn. Nhược điểm của phương pháp này là thời gian tính toán sẽ bị chậm lại, vì mỗi bộ cộng phải đợi kết quả từ bộ cộng trước đó trước khi có thể đưa ra kết quả chính xác.

* 1. Simulate: Building a Half-Adder Circuit

Mạch mô phỏng:

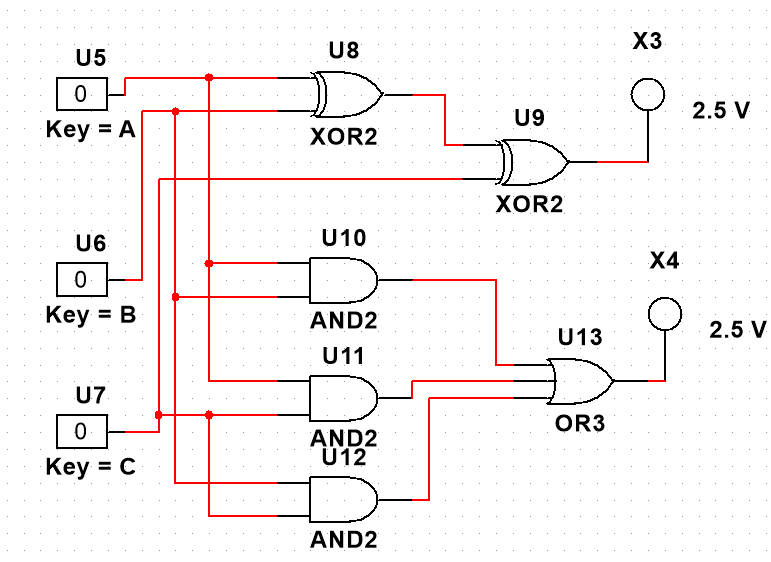


Ques4 Kết quả:

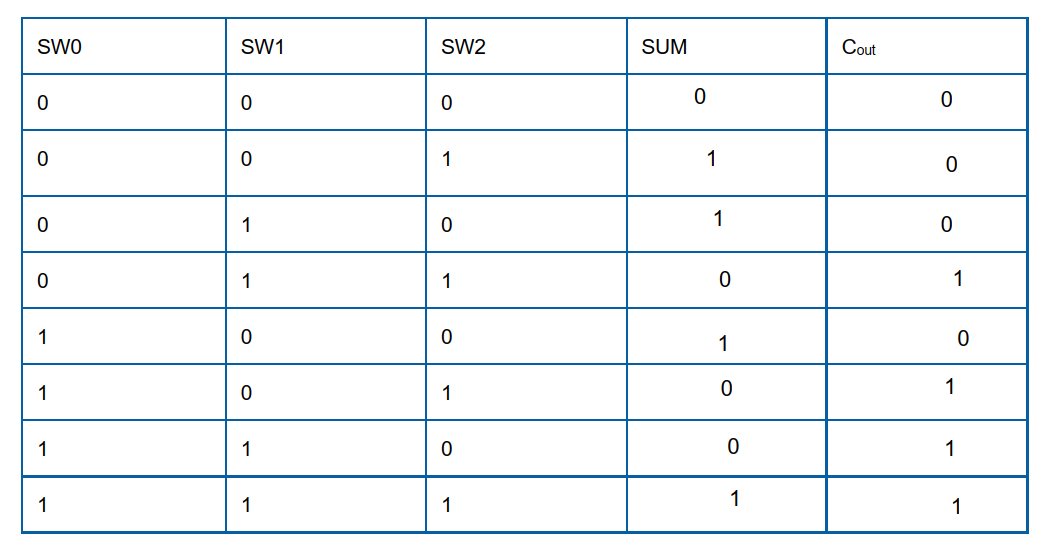


* 1. Simulate: Building a Full-Adder Circuit

Mạch mô phỏng:

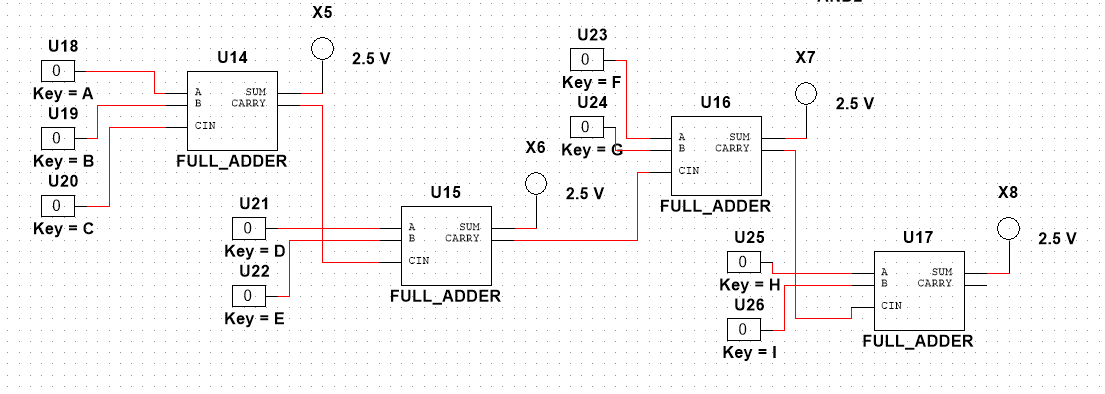


Ques5 Kết quả:

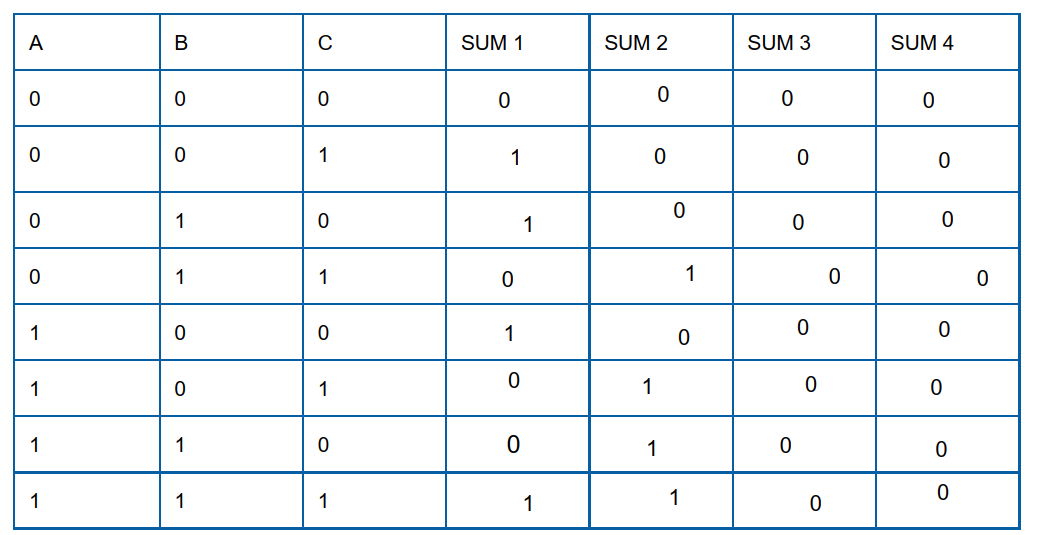


* 1. Building a Circuit Using Built-In Full Adders

Mạch mô phỏng:



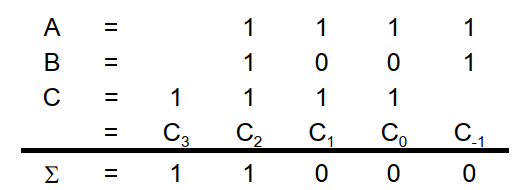
Ques6: thay đổi A, B, C kết quả được ghi lại ở bảng dưới đây

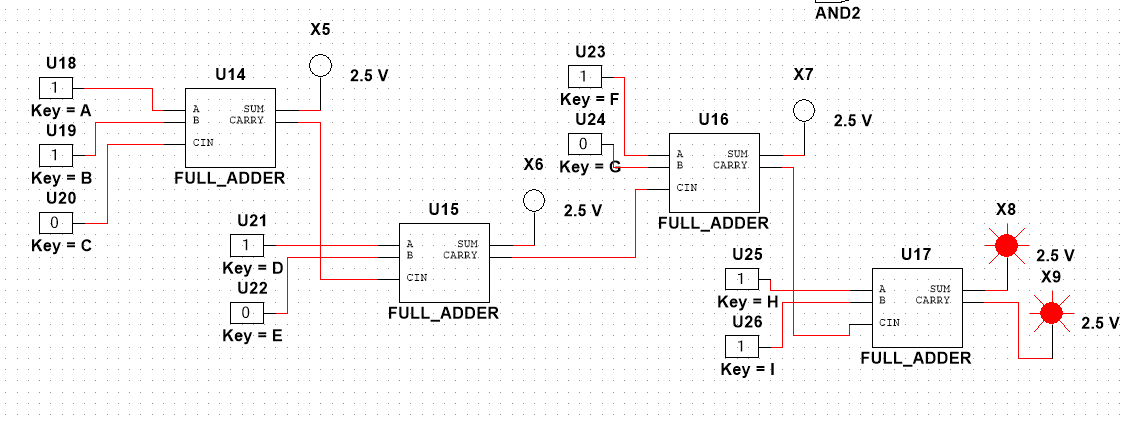


Ques7: Kết quả bảng ở Ques6 hoàn toàn phù hợp với bảng ở Ques5

Mạch trên là mạch cộng hai số nhị phân 4 bit:

Giả sử A = 1111 và B = 1001





* 1. Conclusion

Câu trả lời:

Ques8. Trong điều kiện nào sử dụng Half Adder và bộ cộng toàn phần (Full Adder)

Half Adder được sử dụng khi chỉ cần cộng hai bit đầu vào mà không có bit nhớ từ phép cộng trước đó. Nó chỉ bao gồm hai đầu vào (A, B) và cho ra hai đầu ra (Sum, Carry).

Bộ cộng toàn phần (Full Adder) được sử dụng khi cần cộng ba bit đầu vào: hai bit dữ liệu (A, B) và một bit nhớ (Cin) từ phép cộng trước đó. Nó quan trọng trong việc xây dựng mạch cộng nhiều bit (Ripple Carry Adder).

Ques9. Bộ cộng toàn phần trong Multisim giúp đơn giản hóa việc thiết kế mạch số bằng cách cung cấp một mô-đun có sẵn thay vì phải tự thiết kế bằng các cổng logic riêng lẻ.

Nếu chỉ dùng cổng XOR và AND, ta sẽ phải tự xây dựng lại toàn bộ logic của bộ cộng, làm tăng độ phức tạp và dễ dẫn đến sai sót.

Bộ cộng toàn phần có thể được kết nối trực tiếp với nhau để tạo bộ cộng nhiều bit, giúp việc thiết kế hiệu quả hơn.

Ques10. Khi chuyển đổi số nhị phân sang BCD: Số được gán giá trị cụ thể theo vị trí của chúng trong dãy; Giá trị của số bên trái tăng gấp đôi so với giá trị của số bên phải; Các số có một chữ số đã ở dạng BCD.

Ques11. Half Adder thực hiện phép cộng nhị phân trên hai đầu vào bằng hai cổng logic XOR và AND

Cổng XOR dùng để tính tổng (Sum) của hai bit đầu vào; Cổng AND dùng để tính bit nhớ (Carry) cho phép cộng tiếp theo.

Ques12. Đầu ra của Half Adder (HA) là: Sum và Carry; HA có hai đầu vào (A, B) và hai đầu ra: Sum (Tổng): Kết quả của phép cộng hai bit đầu vào; Carry (Nhớ): Giá trị nhớ khi tổng lớn hơn 1.

Ques13. Đặc điểm của bộ cộng toàn phần (Full Adder) là:

Bộ cộng toàn phần có ba đầu vào: A, B, và Cin; Có hai đầu ra: Sum và Carry Out (Cout); Chúng có thể được kết nối nối tiếp, trong đó Cout của một bộ cộng là Cin của bộ cộng tiếp theo, tạo thành một mạch cộng nhiều bit (Ripple Carry Adder).

Ques14. Khi kết nối các bộ cộng toàn phần trong Multisim, Carry của bộ này được nối với đầu vào của bộ tiếp theo, giúp thực hiện phép cộng nhiều bit theo chuỗi.

**Lab 5: Karnaugh Maps**

Bảng Karnaugh (K-map) là một công cụ và phương pháp được sử dụng để tối thiểu hóa các hàm Boolean. Được sử dụng để thiết kế thủ công các hàm logic đơn giản có số lượng biến nhỏ. Phương pháp lập bản đồ Karnaugh thường yêu cầu ít bước hơn so với đơn giản hóa đại số và luôn cho ra một biểu thức tối thiểu.

Bảng Karnaugh của một hàm thực chất là bảng giá trị của hàm đó được viết dưới dạng lưới. Các hàng và cột của bản đồ tương ứng với các giá trị có thể có của các đầu vào và mỗi ô biểu diễn đầu ra của hàm ứng với các đầu vào liên quan.

Các biểu thức đơn giản hóa luôn có một trong hai dạng chuẩn:

Tổng của tích (Sum-of-Products)

Tích của tổng (Product-of-Sums)

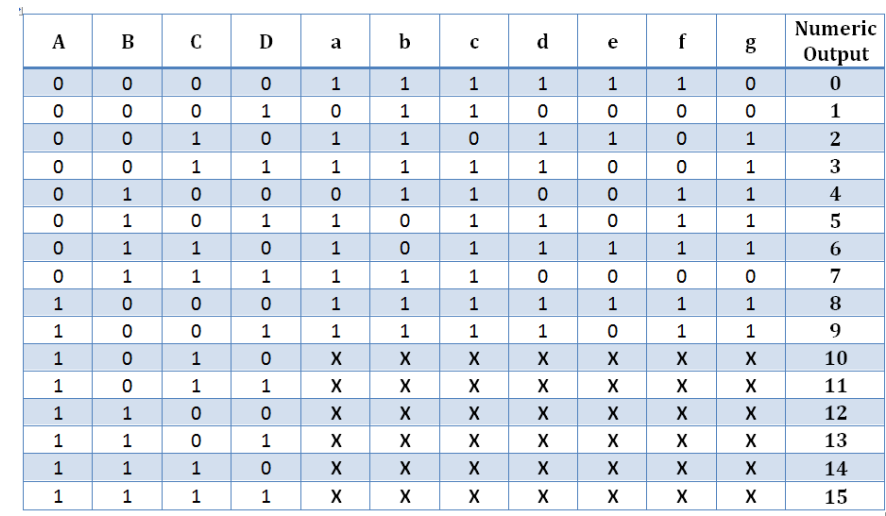
Các ô trong bản đồ được sắp xếp theo hình vuông hoặc hình chữ nhật sao cho các ô liền kề chỉ khác nhau một biến duy nhất, điều này được gọi là Gray code ordering (sắp xếp theo mã Gray). Để đơn giản, các giá trị đầu vào được đặt làm nhãn cho cột và hàng. Mỗi ô trong bản đồ tương ứng với một hàng trong bảng giá trị.

Trong Lab5 sẽ:

1. Đơn giản hóa một biểu thức Boolean bằng cách sử dụng bản đồ Karnaugh.
2. Sử dụng một mạch với các đầu vào để xác định:
   * Đầu ra thông qua thực nghiệm và sử dụng đại số Boolean.
   * Bản đồ Karnaugh.
   * Mạch logic tổ hợp đã được đơn giản hóa.
   1. Theory and Background

<Bảng Karnaugh không trình bày lại>

Có bảng Truth table của 7-segment



* 1. Exercise: Creating a Karnaugh Map from a Combinational Logic Circuit
  2. Implement: Using Karnaugh Maps in Seven Segment Displays
  3. Exercise: Simplified Circuits
  4. Conclusion

**Lab 6: Encoders and Decoders**

Trong Lab 4, các cổng logic được sắp xếp để thực hiện một chức năng cụ thể, chẳng hạn như cộng nhị phân, có thể được biểu diễn dưới dạng một vi mạch (chip). Các ứng dụng khác của khái niệm này bao gồm bộ mã hóa (encoder) và bộ giải mã (decoder). Bộ mã hóa (encoder) là mạch logic có nhiệm vụ giảm kích thước của đầu vào. Bộ giải mã (decoder) thực hiện thao tác ngược lại, tăng kích thước của đầu vào.

Trong lab6, sinh viên sẽ:

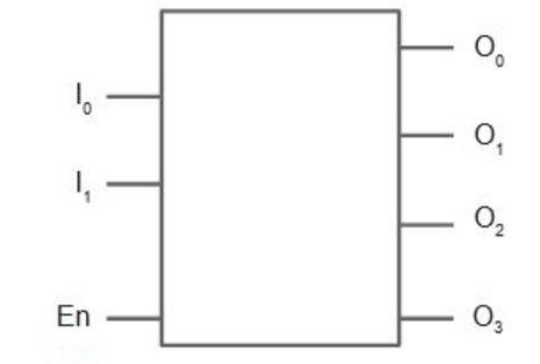
* Giải thích cách hoạt động của bộ giải mã (decoder), đặc biệt là trong màn hình LED 7 đoạn (SSD - Seven Segment Display).
* Tạo một mạch sử dụng bộ giải mã từ BCD sang màn hình LED 7 đoạn và kiểm tra bảng chân lý của nó.

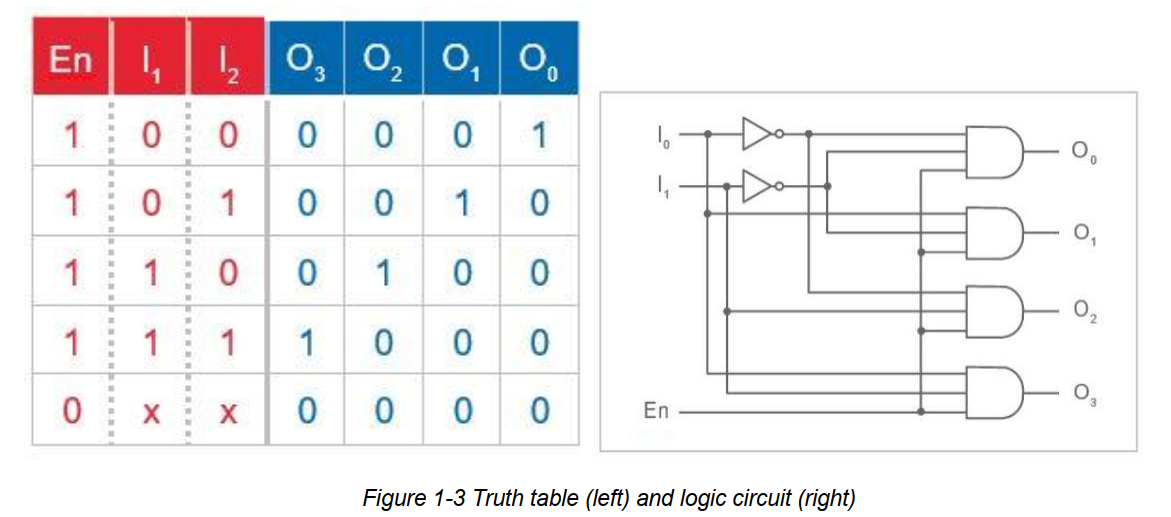
1.1 Theory and Background

Bộ giải mã (Decoders)

* Quá trình chuyển đổi thông tin mơ hồ thành một dạng dễ hiểu đối với thiết bị nhận dữ liệu được gọi là giải mã (decoding).  
  Do đó, thiết bị thực hiện chức năng này được gọi là bộ giải mã (decoder).
* Bộ giải mã nhận mã nhị phân có n bit và tạo ra 2ⁿ đầu ra.
* Đầu ra của một bộ giải mã nhị phân được gọi là one-hot encoded, vì với bất kỳ tổ hợp tín hiệu đầu vào nào, chỉ có một đầu ra nhận giá trị 1.
* Bộ giải mã có thể bao gồm tín hiệu cho phép (enable signal) để kiểm soát hoạt động của mạch.
* Tín hiệu cho phép có thể là active-low (mạch chỉ hoạt động khi enable = 0) hoặc active-high (mạch hoạt động khi enable = 1).
* Các bộ giải mã có đầu vào enable có thể được sử dụng để xây dựng các bộ giải mã lớn hơn.
* Một trong những ứng dụng quan trọng nhất của bộ giải mã là truy cập bộ nhớ, nơi chúng được sử dụng để giải mã địa chỉ của các hàng trong các khối bộ nhớ.

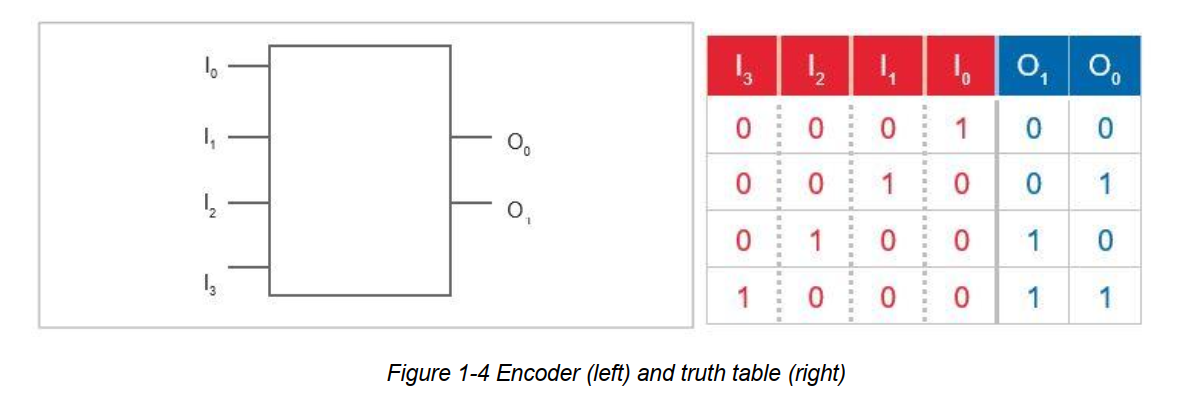
Giải mã vào 2 ra 4



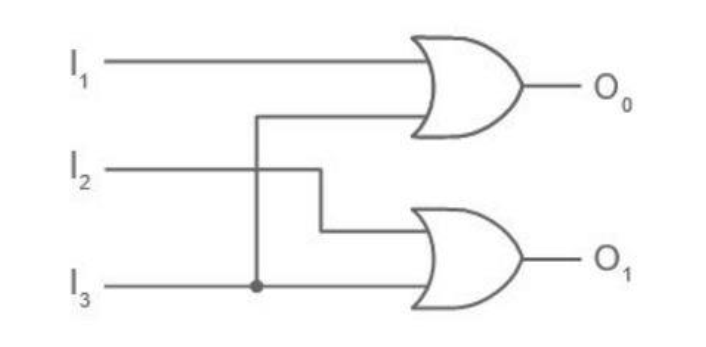


Bộ mã hóa (Encoders)

* Bộ mã hóa (Encoders) là các mạch logic thực hiện chức năng ngược lại so với bộ giải mã. Bộ mã hóa nhị phân (Binary encoders) mã hóa thông tin từ 2ⁿ đường đầu vào, tạo ra một mã nhị phân n-bit.
* Tại một thời điểm bất kỳ, chỉ có một trong số 2ⁿ đầu vào có thể nhận giá trị 1.
* Mã hóa được sử dụng để giảm số bit cần thiết để biểu diễn thông tin. Chúng thường được ứng dụng trong truyền dữ liệu và lưu trữ dữ liệu.
* Ký hiệu đồ họa của bộ mã hóa nhị phân 4 sang 2 (4-to-2 binary encoder) được trình bày bên dưới. Các trường hợp có nhiều hơn một đầu vào bằng 1 không được hiển thị trong bảng chân lý vì chúng được xem như điều kiện không quan trọng (don’t care conditions).



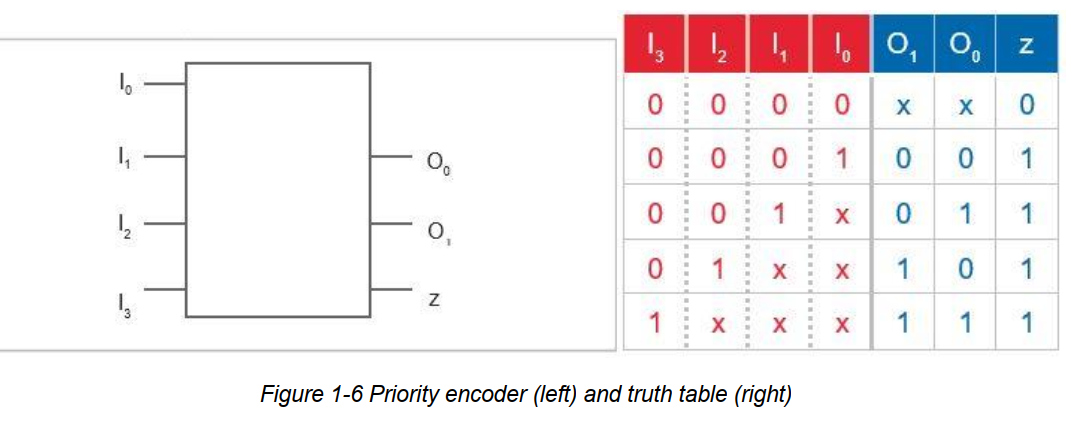
* Có thể thấy trong bảng chân lý rằng đầu ra O₁ bằng 1 khi I₃ hoặc I₂ bằng 1 và đầu ra O₀ bằng 1 khi I₃ hoặc I₁ bằng 1.
* Cũng có thể thấy rằng đầu vào I₀ có thể được bỏ qua.
* Các bộ mã hóa đã trình bày cho đến nay được coi là có đầu vào được mã hóa one-hot.



Một loại bộ mã hóa thường được sử dụng khác là bộ mã hóa ưu tiên:

* Bộ mã hóa ưu tiên có thể xác định thứ tự ưu tiên cho các đầu vào.
* Điều này quan trọng vì các bộ mã hóa thông thường có thể tạo ra đầu ra sai khi có nhiều hơn một đầu vào ở mức logic 1.
* Loại bộ mã hóa này có một đầu ra bổ sung, z, dùng để chỉ trường hợp không có đầu vào nào bằng 1.

Ký hiệu đồ họa của bộ mã hóa ưu tiên được trình bày bên dưới. Bảng chân lý mô tả hành vi của bộ mã hóa ưu tiên 4-đến-2. Có thể thấy ở dòng cuối cùng của bảng chân lý rằng nếu đầu vào I₃ bằng 1, tất cả các đầu ra đều bằng 1 và các giá trị của các đầu vào khác không quan trọng, được ký hiệu là ‘x’.



1.2 Exercise: Seven Segment Display Diagram

Ques1: Bộ giải mã (decoder) là một mạch logic nhận vào một mã nhị phân và chuyển đổi nó thành một đầu ra duy nhất trong số nhiều đầu ra có thể có. Ví dụ, bộ giải mã 3-đến-8 (3-to-8 decoder) có ba đầu vào và tám đầu ra, chỉ có một đầu ra được kích hoạt tại một thời điểm tương ứng với giá trị nhị phân của đầu vào. Bộ giải mã thường được sử dụng trong các hệ thống hiển thị như bộ giải mã BCD-7 đoạn hoặc trong bộ nhớ để chọn các ô nhớ cụ thể.

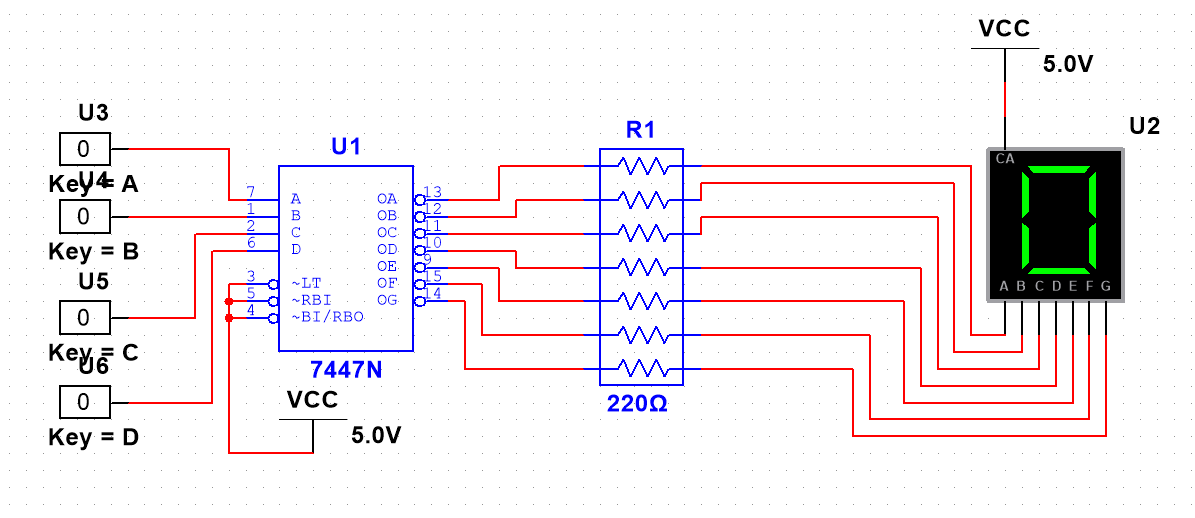
Ques2: Sự khác biệt giữa bộ mã hóa (encoder) và bộ mã hóa ưu tiên (priority encoder).

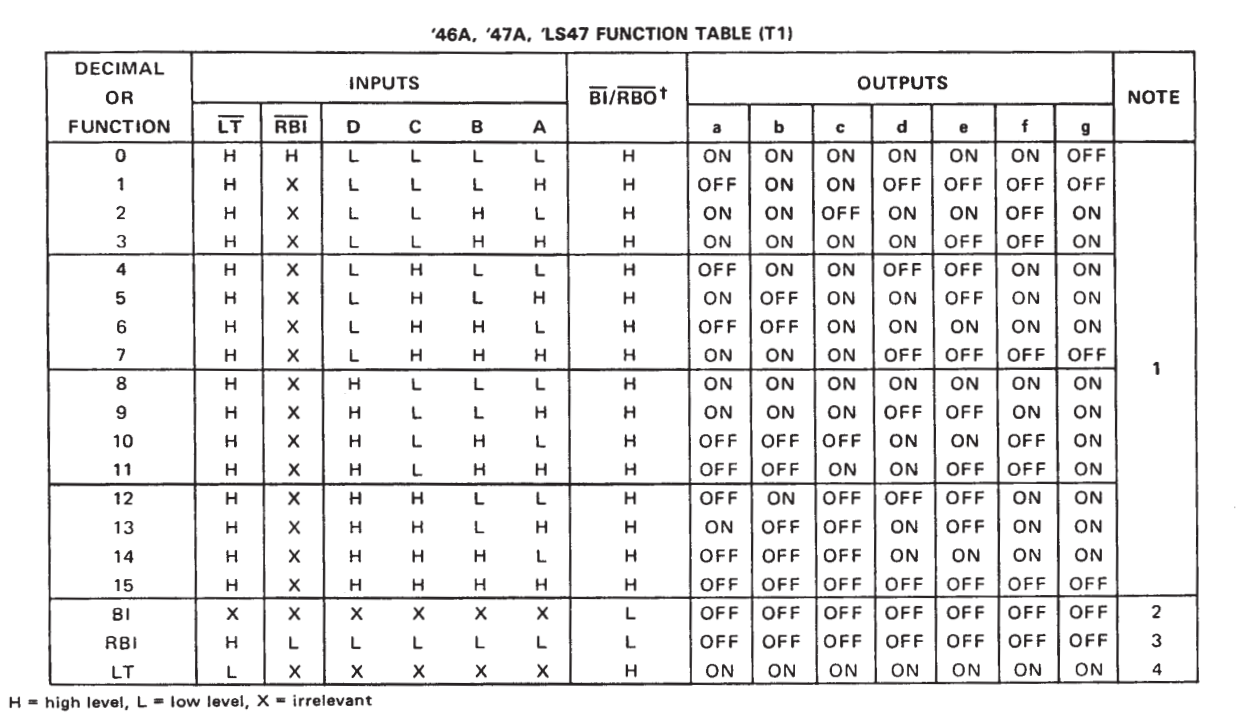
Bộ mã hóa (encoder): chuyển đổi một đầu vào có 2ⁿ đường thành một mã nhị phân có n bit. Chỉ một đầu vào có thể ở mức logic 1 tại một thời điểm để tránh nhầm lẫn trong đầu ra.

Bộ mã hóa ưu tiên (priority encoder): bộ mã hóa, có thể xử lý trường hợp nhiều đầu vào ở mức logic 1 cùng lúc. Khi có nhiều đầu vào cùng bật, bộ mã hóa ưu tiên sẽ chọn đầu vào có mức độ ưu tiên cao nhất (thường là giá trị lớn nhất) để tạo ra đầu ra chính xác. Ngoài ra, nó có thêm đầu ra chỉ báo khi không có đầu vào nào được kích hoạt.

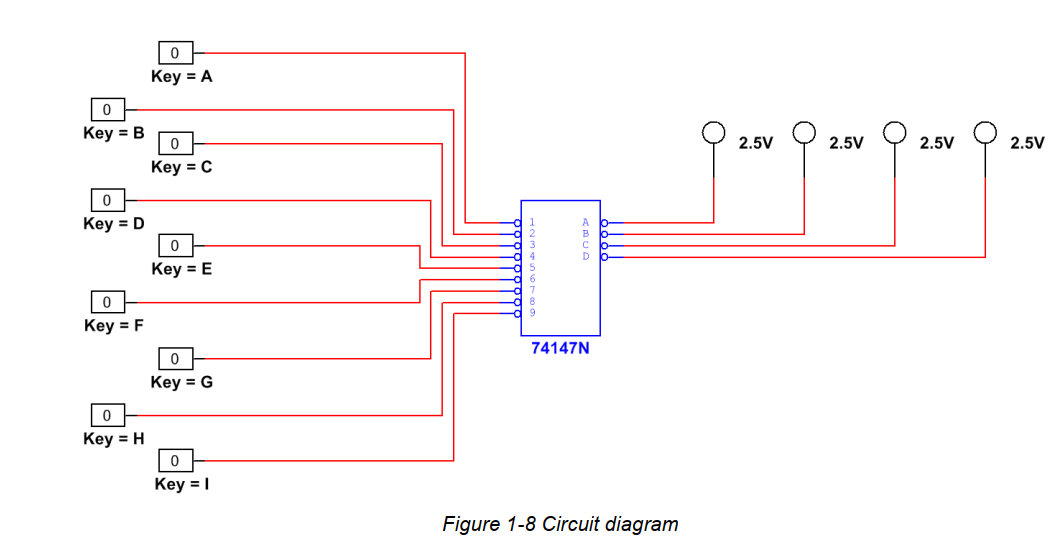
1.3 Implement: Building a BDC to Seven Segment Display Decoder

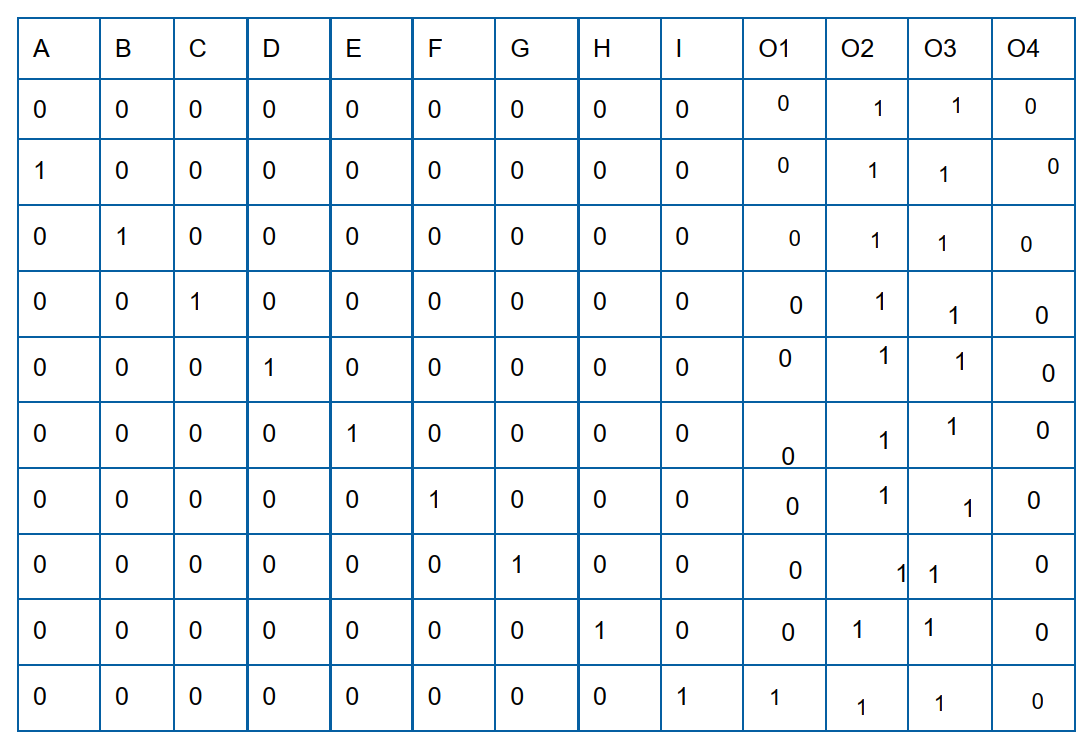
BCD to Seven Segment Display Decoder





1.4 Implement: Building a Circuit for a 10 to 4 Priority Encoder





* 1. Conclusion

Ques8: bộ mã hóa và bộ giải mã có tỷ lệ đầu vào và đầu ra khác nhau

Bộ mã hóa (encoder) có số đầu vào lớn hơn số đầu ra vì nó thực hiện quá trình mã hóa dữ liệu, tức là chuyển đổi một tập hợp lớn các tín hiệu đầu vào thành một số lượng nhỏ hơn các bit đầu ra đại diện cho trạng thái của đầu vào. Ngược lại, bộ giải mã (decoder) nhận một số lượng nhỏ các bit đầu vào và chuyển đổi chúng thành một tập hợp lớn hơn các đầu ra riêng biệt, thực hiện quá trình giải mã dữ liệu.

Ques9: bộ mã hóa có một đường đầu vào có thể bị bỏ qua khi xác định hành vi với các cổng logic

Một số bộ mã hóa có một đầu vào có thể bị bỏ qua vì nó có thể đại diện cho một trạng thái không quan trọng hoặc một tín hiệu mặc định khi không có tín hiệu đầu vào hợp lệ. Điều này giúp giảm độ phức tạp trong thiết kế mạch số và tối ưu hóa việc sử dụng các cổng logic.

Ques10: Tín hiệu Enable kiểm soát cách thức hoạt động của bộ giải mã

* Tín hiệu Enable quyết định đầu vào nào sẽ được xử lý trước.
* Bộ giải mã chuyển đổi n bit đầu vào thành 2ⁿ đầu ra.
* Tín hiệu Enable xác định liệu mạch có hoạt động hay không (bật hoặc tắt).

Ques11: Những cổng logic nào cần thiết để tạo bộ giải mã 2 sang 4

Cần 2 cổng NOT và 4 cổng AND

Bộ giải mã 2-to-4 cần:

2 cổng NOT để tạo các tín hiệu đảo ngược của đầu vào.

4 cổng AND để tạo ra 4 đầu ra duy nhất tương ứng với các tổ hợp đầu vào khác nhau.

**Lab 7: Multiplexers and Demultiplexers**

Bộ ghép kênh (Multiplexer) là các mạch logic tổ hợp có nhiều đầu vào tiềm năng nhưng luôn chỉ có một đầu ra. Bộ phân kênh (Demultiplexer) thì ngược lại, chúng chỉ có một đầu vào nhưng có nhiều đầu ra tiềm năng. Cả bộ ghép kênh và bộ giải ghép kênh đều có một hoặc nhiều bit chọn (selector bit), đóng vai trò quyết định đầu vào hoặc đầu ra nào sẽ được chọn.

Tương tự như bộ mã hóa (encoder) và bộ giải mã (decoder), bộ ghép kênh và bộ giải ghép kênh có thể được phân tích thành các thành phần mạch điện nhưng thường được biểu diễn dưới dạng các vi mạch (chip) để đơn giản hóa hình ảnh trực quan. Trong bài thực hành này, chúng ta sẽ phân tích bộ ghép kênh và bộ phân kênh.

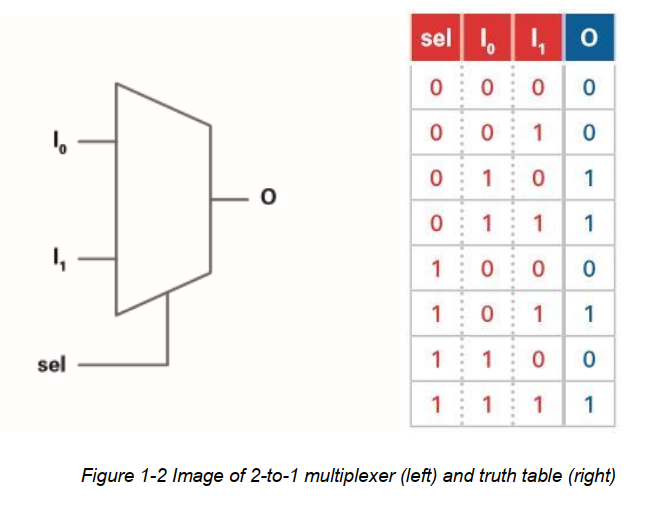
Trong lab7:

* Xem xét những điểm tương đồng và khác biệt giữa bộ mã hóa và bộ ghép kênh.
* Khảo sát chức năng của một bộ ghép kênh 2-chọn-1 cơ bản bằng cách sử dụng các cổng logic.
* Quan sát hành vi của quá trình ghép kênh xung nhịp bằng cách sử dụng dao động ký (oscilloscope).
  1. Theory and Background

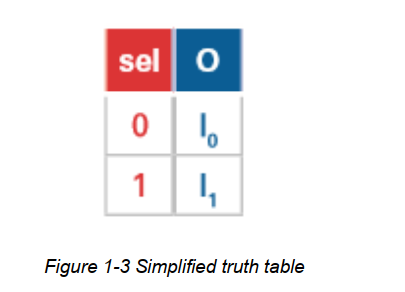
Bộ ghép kênh

* Bộ ghép kênh, viết tắt là MUX, là một mạch logic tổ hợp có nhiều đầu vào dữ liệu, một hoặc nhiều đầu vào chọn và một đầu ra.
* Nó truyền dữ liệu từ một trong các đầu vào đến đầu ra, tùy thuộc vào tín hiệu chọn.
* Với sự hỗ trợ của mạch logic này, nhiều tín hiệu có thể sử dụng chung một đầu ra dữ liệu.
* Bộ ghép kênh có đầu vào và s đường chọn, giúp xác định đầu vào nào sẽ được truyền tới đầu ra.
* Bộ ghép kênh là một trong những mạch tổ hợp được sử dụng rộng rãi nhất, với các ứng dụng bao gồm:
  + Định tuyến dữ liệu
  + Trình tự hoạt động
  + Chuyển đổi từ song song sang nối tiếp
  + Tạo dạng sóng

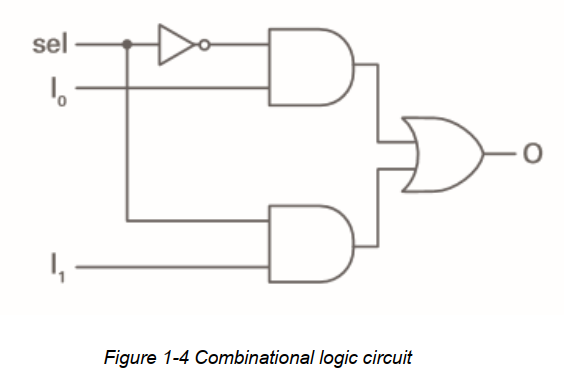
Mạch đơn giản nhất là bộ ghép kênh 2-chọn-1, với ký hiệu đồ họa được trình bày trong hình bên trái. Chức năng của nó được mô tả bởi bảng chân lý đi kèm. Bộ ghép kênh dưới đây chỉ có độ rộng 1 bit vì đường bit được kết nối với một đầu ra bit duy nhất.



Bảng chân lý:



Mạch logic:



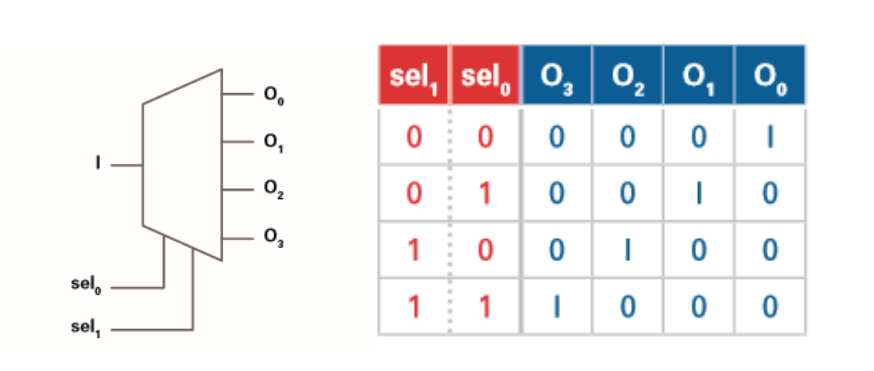
**Clock multiplexing**là một kỹ thuật được sử dụng để vận hành cùng một chức năng logic ở các tần số xung nhịp khác nhau từ các nguồn (đầu vào) khác nhau.

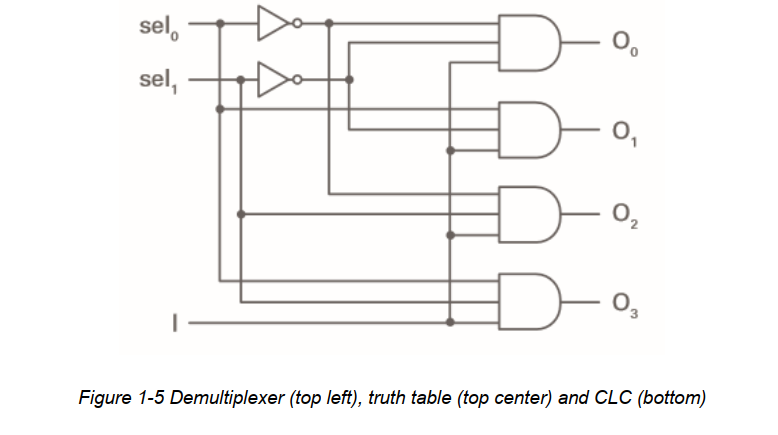
* Các mạch logic được chuyển đổi bởi tín hiệu chọn thường xuyên trong khi mạch đang hoạt động.
* Quá trình chuyển đổi này không an toàn và có thể dẫn đến lỗi glitch, xảy ra khi một tín hiệu đang giảm xuống trong khi tín hiệu khác đang tăng lên.
* Có thể triển khai các bộ chuyển mạch an toàn để loại bỏ lỗi glitch.

Bộ phân kênh (Demultiplexers)

Bộ giải đa hợp (DEMUX) có chức năng ngược lại với bộ ghép kênh.

* Nó truyền giá trị của một đầu vào dữ liệu duy nhất đến nhiều đầu ra dữ liệu khác nhau dựa trên tín hiệu chọn.
* Thông thường, bộ giải đa hợp có s đầu vào chọn và đầu ra.





* Vì bộ giải đa hợp nhận một đầu vào và kết nối nó với nhiều đầu ra, một số ứng dụng của nó bao gồm truyền thông (giao tiếp hai chiều thường sử dụng cả bộ ghép kênh và bộ giải đa hợp) và chuyển đổi tín hiệu từ nối tiếp sang song song.
* Ký hiệu đồ họa cho bộ giải đa hợp 1-to-4 được hiển thị bên dưới (bên trái), cùng với bảng chân lý tương ứng 1-to-4 DEMUX (ở giữa) và sơ đồ mạch logic (CLC - bên phải).

Ques1 : Viết hàm Boolean dạng tổng tích (Sum-of-Products) cho bộ ghép kênh 2-to-1:  
Bộ ghép kênh 2-to-1 có hai đầu vào dữ liệu , và một tín hiệu chọn . Hàm Boolean của đầu ra Y là:

Trong đó:

* là phủ định của tín hiệu chọn.
* Khi , đầu ra sẽ là .
* Khi , đầu ra sẽ là .

Ques2: Viết hàm Boolean dạng tổng tích (Sum-of-Products) cho bộ giải đa hợp 1-to-4:  
Bộ giải đa hợp 1-to-4 có một đầu vào dữ liệu D, hai đầu vào chọn , ​ và bốn đầu ra ​. Các hàm Boolean của các đầu ra là:

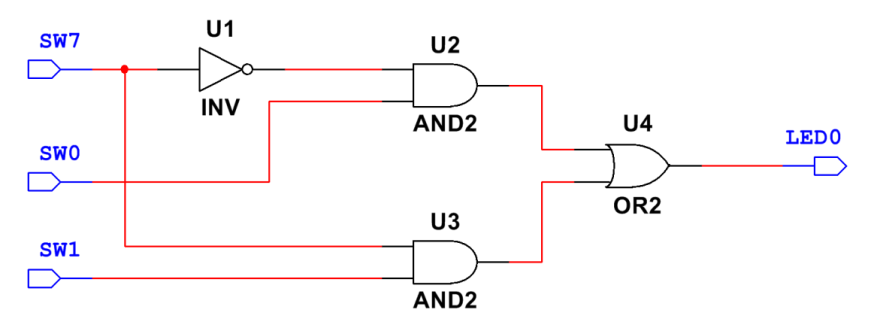
Trong đó:

* là các tín hiệu chọn.
* Chỉ một trong bốn đầu ra sẽ nhận giá trị của tại một thời điểm, dựa trên giá trị của

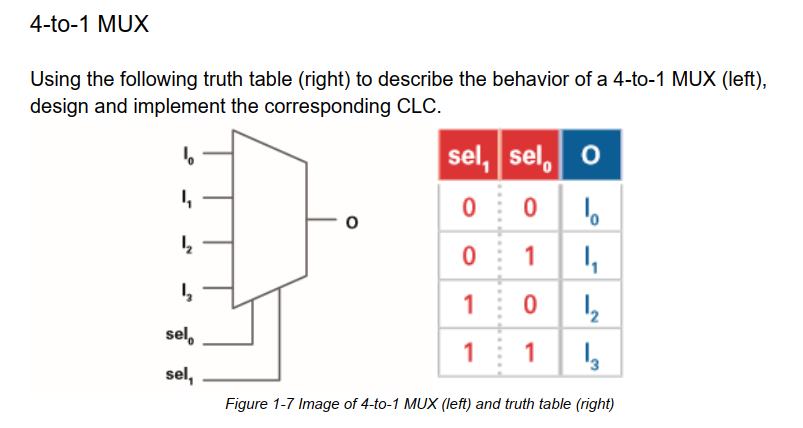
Ques3 Chức năng của tín hiệu chọn (Sel) trong bộ ghép kênh và bộ giải đa hợp là gì?  
Tín hiệu chọn (Sel) có vai trò quyết định đầu vào hoặc đầu ra nào sẽ được kích hoạt:

* Trong bộ ghép kênh (MUX): Sel xác định đầu vào nào sẽ được truyền đến đầu ra.
* Trong bộ giải đa hợp (DEMUX): Sel xác định đầu ra nào sẽ nhận tín hiệu từ đầu vào duy nhất.

Nói cách khác, tín hiệu chọn điều khiển luồng dữ liệu, giúp chuyển đổi và định tuyến tín hiệu trong mạch số.

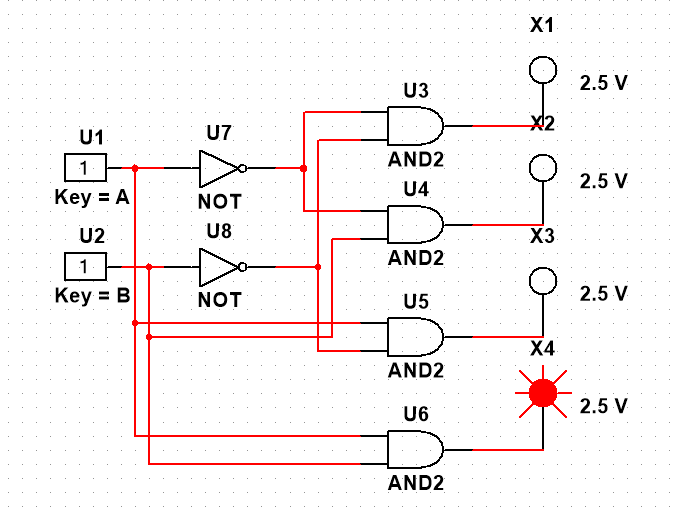


* 1. Implement: Multiplexers Using Logic Gates

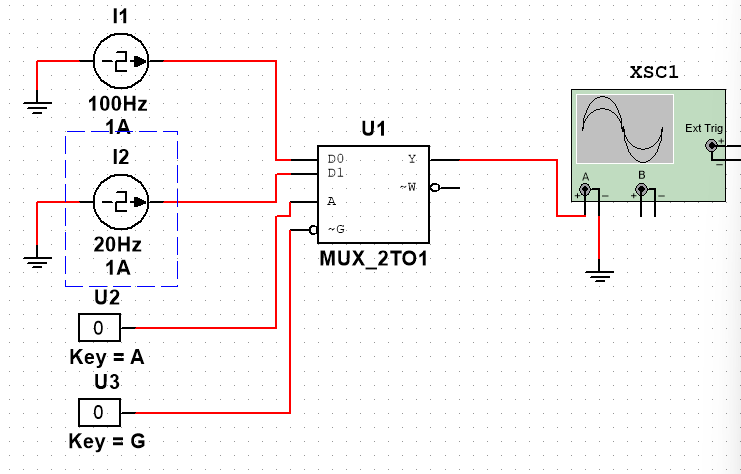


Biểu thức logic:

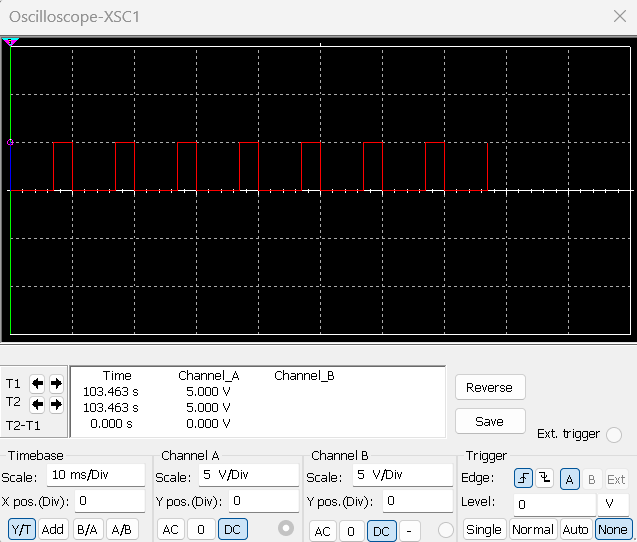
Mạch nguyên lý:



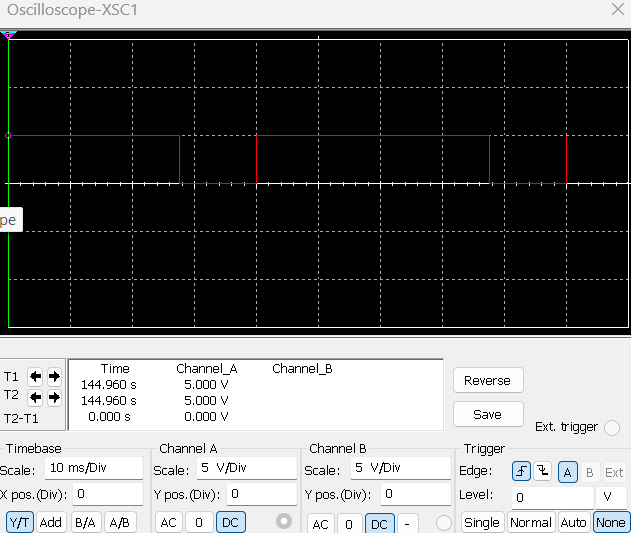
* 1. Implement: Clock Multiplexing



Ques4: khi A = 0 đầu ra Y là nguồn từ D0, có tần số 100Hz, độ rộng xung là 30%

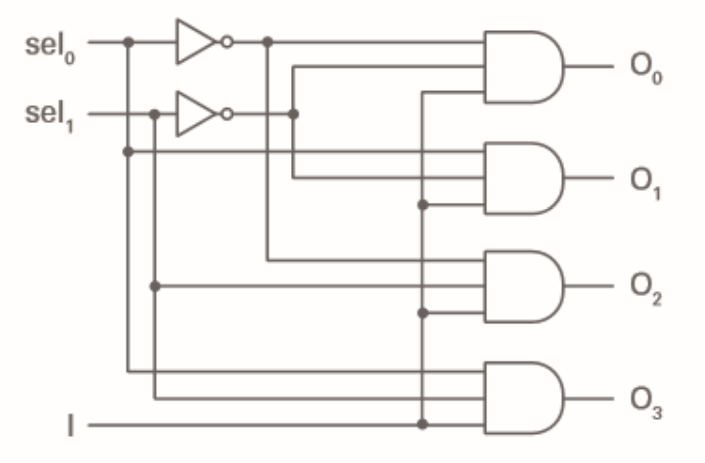


Ques5: Khi A = 1, đầu ra Y là clock từ D1, tần số 20Hz, độ rộng xung 75%

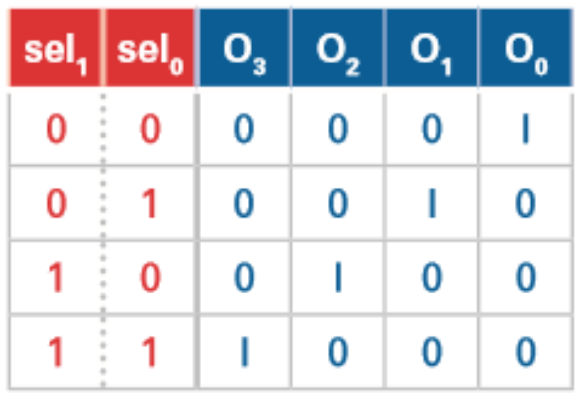


* 1. Implement: Demultiplexer

Mạch phân kênh (Demultiplexer) được cho bởi ví dụ dưới đây:



Truth table:



Ques6: Mở rộng ra phân kênh 1 ngõ vào 8 ngõ ra.

Sử dụng 3 chân điều khiển , , ()

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |
| 0 | 0 | 0 |  |  |  |  |  |  |  | I |
| 0 | 0 | 1 |  |  |  |  |  |  | I |  |
| 0 | 1 | 0 |  |  |  |  |  | I |  |  |
| 0 | 1 | 1 |  |  |  |  | I |  |  |  |
| 1 | 0 | 0 |  |  |  | I |  |  |  |  |
| 1 | 0 | 1 |  |  | I |  |  |  |  |  |
| 1 | 1 | 0 |  | I |  |  |  |  |  |  |
| 1 | 1 | 1 | I |  |  |  |  |  |  |  |

* 1. Conclusion

Ques7: Đầu ra thay đổi tùy theo giá trị chọn

Khi giá trị của chân chọn (selector) thay đổi, mẫu tín hiệu hiển thị trên dao động ký cũng thay đổi theo.

* Nếu selector = 0: tín hiệu từ đầu vào D0 sẽ được truyền đến đầu ra, dao động ký sẽ hiển thị mẫu của D0.
* Nếu selector = 1: tín hiệu từ đầu vào D1 sẽ xuất hiện ở đầu ra, và dao động ký sẽ phản ánh mẫu của D1.

Ques8: Một số ưu điểm và nhược điểm của việc sử dụng bo Digital Electronics Board so với mô phỏng mạch

Ưu điểm:

* Thao tác trực tiếp với linh kiện thật giúp hiểu rõ nguyên lý hoạt động.
* Có thể kiểm tra ảnh hưởng của nhiễu, trễ thời gian và các yếu tố vật lý thực tế.
* Trải nghiệm giống thực tế hơn, hữu ích cho thực hành kỹ thuật.

Nhược điểm:

* Dễ mắc lỗi kết nối phần cứng.
* Cần thời gian nhiều hơn để lắp ráp và kiểm tra.
* Nếu linh kiện bị hỏng thì việc sửa chữa và thay thế mất thời gian.

Ques9: Multiplexer hữu ích trong các ứng dụng cần lựa chọn tín hiệu đầu vào:

Dưới đây là phần trả lời cho các câu hỏi từ **1-10 đến 1-14** về **demultiplexer (giải kênh), multiplexer (hợp kênh)** và các khái niệm liên quan trong điện tử số:

Ques10. Demultiplexer hữu ích trong các hệ thống cần chuyển tín hiệu từ một đầu vào sang nhiều đầu ra riêng biệt.

Ques11. Một multiplexer luôn chỉ có một đầu ra duy nhất, vì nó chọn một trong nhiều đầu vào để truyền ra ngoài.

Ques14: Một demux 1-4 có 4 đầu ra, và để chọn 1 trong 4 đầu ra thì cần 2 bit chọn (selector), vì 2² = 4.

**Lab 8: Comparators**

Trong Lab này, sẽ học cách thực hiện các phép toán <, >, và = trong một mạch logic tổ hợp. Một bộ so sánh có thể so sánh hai số nhị phân hai bit.

* 1. Theory and Background

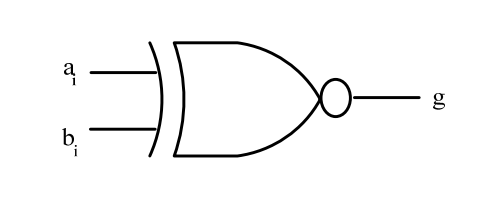
So sánh hai số là một phép toán nhằm xác định xem một số lớn hơn, nhỏ hơn hay bằng số còn lại.  
Một bộ so sánh là một mạch tổ hợp dùng để so sánh hai số A và B và xác định độ lớn tương đối giữa chúng. Kết quả của phép so sánh được biểu diễn bởi ba biến nhị phân ()

Bộ so sánh 1 bit

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Từ bảng trạng thái có: với

Mạch logic của đầu ra so sánh bằng nhau 1 bit:



Ques1: Xây dựng bộ so sánh 4 bit

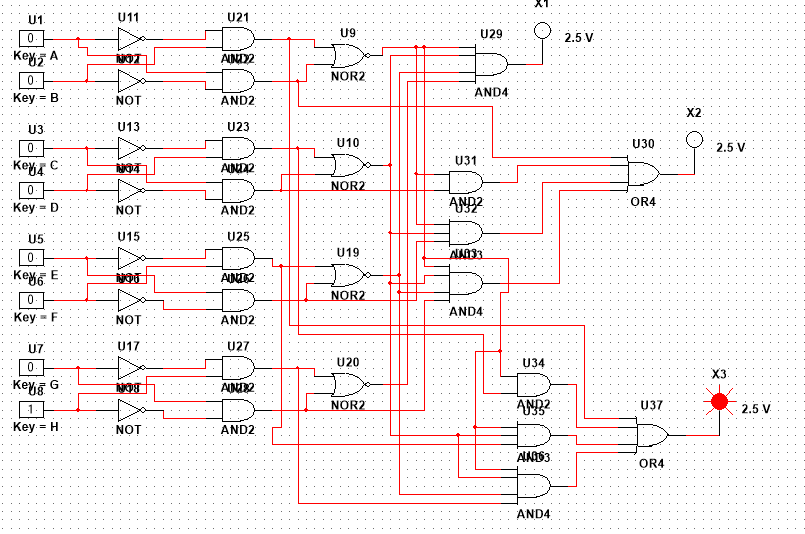
Giải sử so sánh 2 số nhị phân 4 bit và

Hai số bằng nhau khi: với

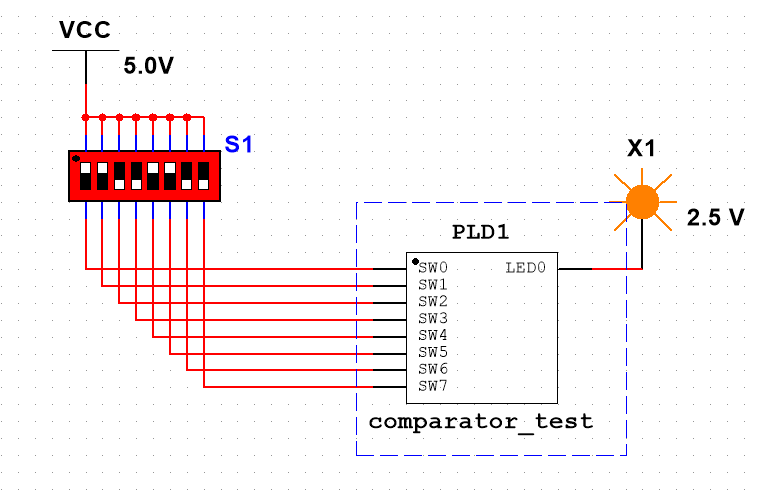
Khi số với

Để xác định A lớn hơn hay nhỏ hơn B thì sẽ so sánh từng bit từ bit có trọng số lớn nhất (MSB). Nếu hai bit bằng nhau, sẽ so sánh đến hai bit có trọng số thấp hơn. Lặp lại như vậy đến khi gặp 2 bit không bằng nhau nếu xác định thì và thì

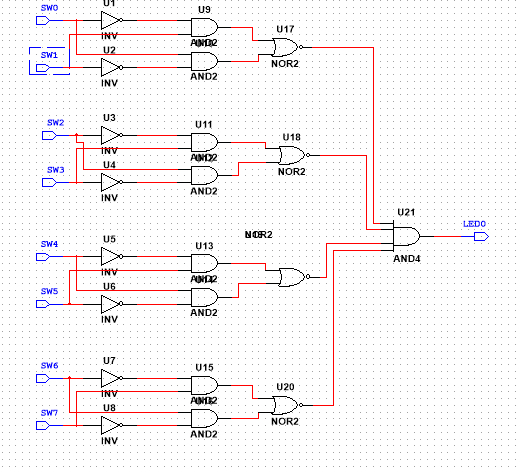
Mạch nguyên lý:



* 1. Implement: Building a 4-bit Parallel Comparator



PLD1 được mô tả kĩ ở sơ đồ dưới đây:



* 1. Conclusion

Ques2: Sử dụng sub circuit để chia nhỏ các mạch lớn phức tạp thành các phần nhỏ hơn để dễ quản lý.

Ques3: Giống và khác nhau giữa sub circuit và chip component trong multism

*Giống nhau:*

Cả sub circuit và chip component đều hoạt động như một khối chức năng riêng biệt trong sơ đồ mạch.

Đều có thể được kết nối với các linh kiện khác thông qua các chân đầu vào/ra.

Mục đích chính là đơn giản hóa thiết kế, giúp sơ đồ mạch rõ ràng và dễ quản lý hơn.

*Khác nhau:*

| Tiêu chí | Sub Circuit | Chip Component |
| --- | --- | --- |
| Nguồn gốc | Do người dùng tạo ra từ các linh kiện cơ bản. | Là linh kiện có sẵn trong thư viện Multisim. |
| Cấu trúc bên trong | Có thể xem, chỉnh sửa và tùy biến. | Không thể xem hoặc sửa đổi bên trong. |
| Tính linh hoạt | Linh hoạt, có thể tùy biến logic bên trong theo nhu cầu. | Cố định chức năng (ví dụ IC 7408 là AND). |
| Mục đích sử dụng | Gom nhóm mạch phức tạp thành khối đơn giản hơn. | Dùng để mô phỏng các IC thương mại thực tế. |

Ques5: Bộ so sánh là các mạch logic tổ hợp.

Ques6: Trong bộ so sánh độ lớn một bit, đầu ra A = B tương ứng với cổng logic → Cổng XNOR

Ques7: Bộ so sánh độ lớn bắt đầu so sánh từ: → Bit có trọng số cao nhất

Ques8: Tổ hợp cổng logic có thể sử dụng để tạo một bộ so sánh 4-bit → 4 cổng XNOR và 1 cổng AND

Ques9: Mạch con trong thiết kế PLD:

* Là một mạch cấp dưới so với mạch chính
* Là cách để đơn giản hóa mạch phức tạp
* Mở rộng một phần trong mạch lớn hơn

**Lab 9: Latches and Sequential Logic Circuits**

Các bài thí nghiệm trước làm việc với các mạch logic tổ hợp. Một mạch logic tổ hợp là một mạch mà tất cả các đầu ra có thể được xác định từ các đầu vào hiện tại. Trong Lab9, các mạch logic tuần tự thông qua việc ứng dụng các chốt (latch). Một mạch logic tuần tự là một mạch mà đầu ra phụ thuộc vào các đầu vào trước đó ngoài các đầu vào hiện tại. Chúng ta cũng sẽ xem xét cách tín hiệu xung nhịp có thể được triển khai vào một mạch, dựa trên những gì đã được đề cập trước đó trong Bài thí nghiệm 7: Bộ ghép kênh (Multiplexers) và Bộ giải ghép kênh (Demultiplexers).

Trong Lab9 sẽ:

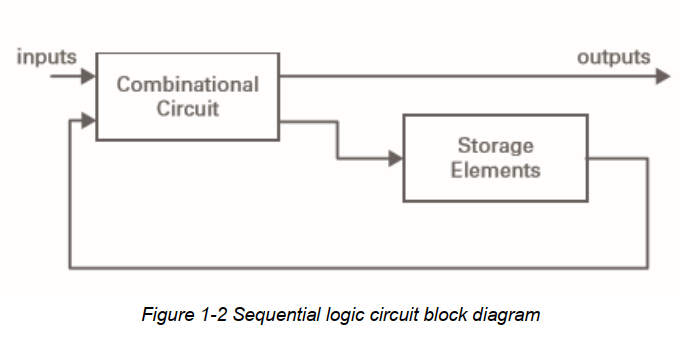
* Hiểu sự khác biệt giữa mạch tuần tự đồng bộ và mạch tuần tự không đồng bộ.
* Kiểm tra và so sánh các mạch triger D bằng cách sử dụng cả cổng logic và chốt.
* Xác nhận bảng đặc tính của triger SR có cổng điều khiển.
* Quan sát và trình bày sự khác biệt giữa trigrer D và triger SR.

1.1 Theory and Background

Mạch tuần tự (Sequential Circuit)

* là mạch mà đầu ra không chỉ phụ thuộc vào tổ hợp hiện tại của các đầu vào mà còn phụ thuộc vào trạng thái trước đó của mạch.
* Khối xây dựng cơ bản của mạch tuần tự là mạch hai trạng thái ổn định (bistable), một loại mạch có hai trạng thái ổn định.
* Trạng thái của một mạch tuần tự được biểu diễn bằng một tập hợp các bit, gọi là biến trạng thái (state variables), chứa tất cả thông tin về quá khứ cần thiết để giải thích hành vi tương lai của mạch.
* Đầu ra của các mạch tuần tự là một hàm của cả đầu vào hiện tại và lịch sử của các đầu vào đó.

Sơ đồ khối của một mạch logic tuần tự sẽ được trình bày bên dưới.



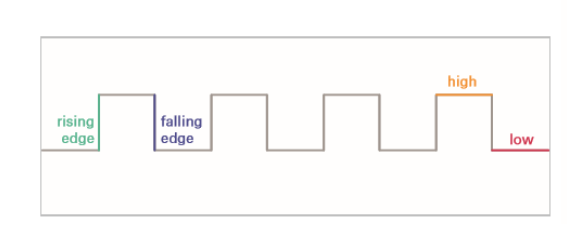
Có hai loại mạch tuần tự chính, được phân loại dựa trên thời gian của tín hiệu:

* Mạch tuần tự đồng bộ (Synchronous sequential circuits) – hành vi của chúng có thể được xác định khi biết tín hiệu tại các thời điểm cụ thể.
* Mạch tuần tự không đồng bộ (Asynchronous sequential circuits) – hành vi của chúng có thể được xác định khi biết tín hiệu đầu vào tại bất kỳ thời điểm nào và thứ tự mà chúng thay đổi.

Thí nghiệm này trình bày về mạch tuần tự đồng bộ, sử dụng tín hiệu xung nhịp để đồng bộ hóa. Trong suốt thí nghiệm này, thuật ngữ "mạch tuần tự" sẽ được dùng để chỉ các mạch tuần tự đồng bộ.

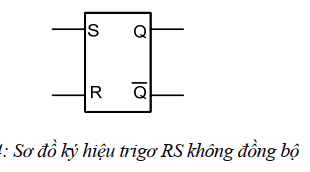
Tín hiệu xung nhịp (Clock Signals)

* Tín hiệu xung nhịp là một chuỗi xung hình chữ nhật. Nó có độ rộng xung chính xác và khoảng thời gian chính xác giữa các xung, được gọi là chu kỳ xung nhịp (clock cycle time).
* Những tín hiệu này được sử dụng trong các mạch tuần tự để đồng bộ hóa hoạt động bên trong mạch và đồng bộ hóa quá trình cập nhật các giá trị đã lưu trữ.
* Hầu hết các mạch tuần tự thay đổi trạng thái tại một trong hai cạnh của xung nhịp (cạnh lên hoặc cạnh xuống), được gọi là kích hoạt theo cạnh (edge-triggered).
* Thời điểm xảy ra sự kiện rất quan trọng khi làm việc với các mạch logic tuần tự. Vì vậy, bên cạnh bảng sự thật (truth table), ta cũng cần biết thứ tự xảy ra của các sự kiện, tức là biểu đồ thời gian logic (logic timing diagram).
* Bảng biểu diễn hoạt động của mạch tuần tự thường được gọi là bảng đặc trưng (characteristic table) thay vì bảng sự thật, vì nó không thể hiện một mạch tổ hợp.
* Phần lớn các hệ thống số là các mạch đồng bộ, giúp thiết kế và khắc phục sự cố dễ dàng hơn, vì chúng chỉ thay đổi đầu ra tại các thời điểm cụ thể. Tuy nhiên, chúng vẫn có chứa một số mạch không đồng bộ.
* Một ví dụ về chuỗi xung nhịp được hiển thị bên dưới (bên trái), cùng với một biểu đồ thời gian cho cổng AND (bên phải).

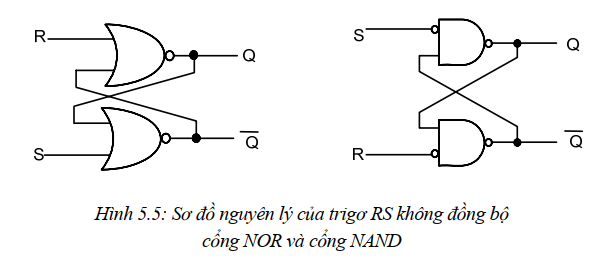


Triger RS

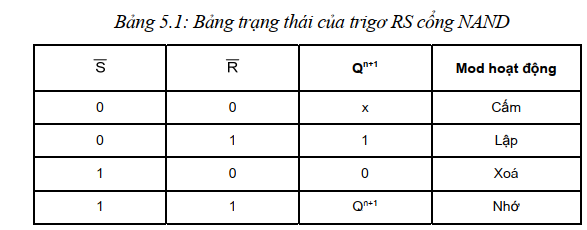
Có hai chân điều khiển S (set), R (reset)



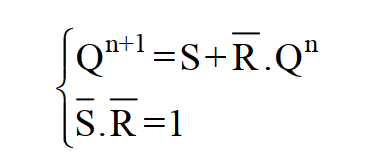
Sơ đồ nguyên lý của triger RS không đồng bộ cổng NOR và cổng NAND



Bảng trạng thái:

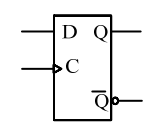


Công thức:

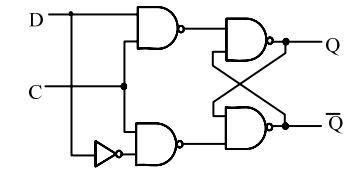


Triger D

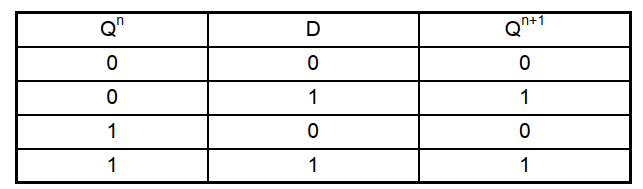
Kí hiệu:



Sơ đồ nguyên lý:



Bảng trạng thái:



Công thức:

Ques1: Mạch tuần tự khác so với mạch logic tổ hợp:

* Mạch logic tổ hợp chỉ phụ thuộc vào đầu vào hiện tại để xác định đầu ra.
* Mạch tuần tự không chỉ phụ thuộc vào đầu vào hiện tại mà còn phụ thuộc vào trạng thái trước đó (lịch sử hoạt động).
* Mạch tuần tự sử dụng các phần tử nhớ (flip-flop, latch) để lưu trữ trạng thái.
* Mạch tổ hợp có thể biểu diễn bằng bảng chân lý, trong khi mạch tuần tự cần biểu đồ trạng thái để mô tả hành vi của nó.

Ques2:. Sự khác biệt giữa mạch tuần tự đồng bộ và không đồng bộ:

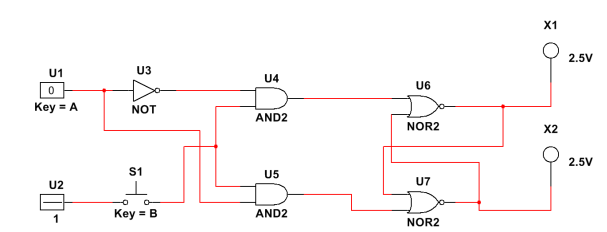
Mạch tuần tự đồng bộ:

* Hoạt động dựa trên tín hiệu xung nhịp (clock).
* Trạng thái của mạch chỉ thay đổi tại các cạnh của xung nhịp (cạnh lên hoặc cạnh xuống).
* Ít bị ảnh hưởng bởi sự thay đổi bất thường của tín hiệu đầu vào.

Mạch tuần tự không đồng bộ:

* Không phụ thuộc vào tín hiệu xung nhịp.
* Trạng thái của mạch thay đổi ngay khi đầu vào thay đổi.
* Có thể gặp vấn đề về nhiễu và điều kiện đua (race condition).

1.2 Implement: Building a Gated D Latch Circuit using Logic Gates



Ques4: Set U1 ở logic cao (1), không có hiệu tượng gì xảy ra, (Ques5) ấn nút S1 đầu ra đèn x1 sáng, x2 tối, (Ques6) Set U1 ở logic thấp (0), không có hiện tượng gì xảy ra, (Ques7) ấn nút S1 đầu ra đền x1 tối đèn x2 sáng

Ques9: Giải thích hoạt động của triger D

Khi có một cạnh xung nhịp (cạnh lên hoặc cạnh xuống tùy theo thiết kế), Flip-Flop D sẽ lật giá trị của D tại thời điểm đó và truyền nó ra đầu ra Q

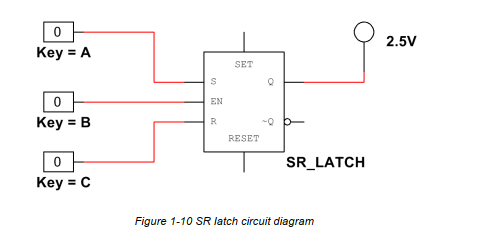
Nếu không có cạnh xung nhịp, giá trị Q sẽ giữ nguyên, không bị thay đổi dù Dcó thay đổi.



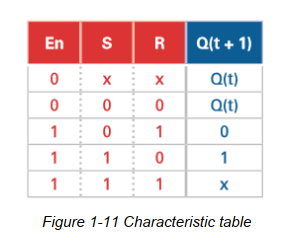
Ques12: Mạch hoạt động giống với mạch ở phía trên, đều là trigger D

1.3 Exercise: Verifying the Gated SR Latch Characteristic Table

Sơ đồ:



Bảng chức năng:



Ques13: Khi đầu vào Enable = 0 thì không có hiện tượng gì ở đầu ra, Ques14 Khi E =1, S =1 thì đầu ra Q = 1 đèn sáng; Khi S = 0, R =1 đầu ra Q =0 đèn tắt.

1.4 Conclusion

Ques20: Những khác biệt chính giữa triger D và triger SR là gì:

* Triger D chỉ có một đầu vào dữ liệu (D), giúp loại bỏ trạng thái không xác định như triger SR (khi S = 1, R = 1).
* SR latch có hai đầu vào riêng biệt Set (S) và Reset (R), cho phép người dùng chủ động thiết lập hoặc xóa trạng thái.

Ques21:

* Triger D dùng khi cần lưu trữ một bit dữ liệu ổn định, chẳng hạn như trong thanh ghi hoặc bộ nhớ.
* Triger SR dùng trong các mạch điều khiển, nơi cần một trạng thái bật/tắt đơn giản dựa trên tín hiệu điều khiển riêng biệt (Set và Reset).

Ques22: Sự khác biệt giữa đầu vào xung clock (Clk) và đầu vào Enable (En) là gì?

* Clock (Clk) được sử dụng trong Flip-Flop, hoạt động dựa trên cạnh xung clock (rising hoặc falling edge). Điều này giúp đồng bộ hóa mạch logic với một tín hiệu thời gian cụ thể.
* Enable (En) chỉ đơn giản là một tín hiệu điều khiển, quyết định khi nào Latch có thể thay đổi trạng thái theo đầu vào của nó. Nó không phụ thuộc vào cạnh xung mà chỉ kiểm tra mức logic của tín hiệu.

**Lab 10: Flip-Flops**

Trong Lab9, nói về khái niệm mạch logic tuần tự bằng cách nghiên cứu các latches và đã tìm hiểu các triger D và SR. Trong lab10, sẽ tiếp tục tìm hiểu về mạch logic tuần tự bằng cách tập trung vào Flip-Flop.

Flip-Flop khác với Latch ở chỗ chúng kích hoạt theo cạnh xung (edge-triggered), trong khi Latch nhạy mức (level-sensitive). Flip-Flop có thể được tạo ra từ Latch. Cả Flip-Flop và Latch đều là những khối cơ bản trong các mạch logic tuần tự.

Trong Lab10 sẽ:

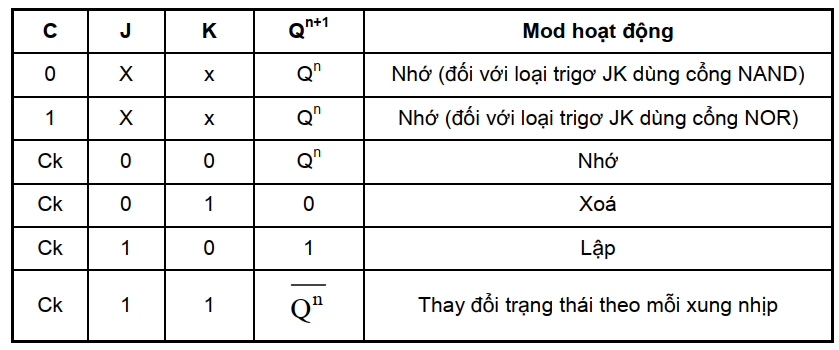
* Tìm hiểu triger JK và T.
* Tìm hiểu các biến thể của Flip-Flop một cách chi tiết hơn (ví dụ: mối quan hệ Master-Slave).
* Tính linh hoạt của Flip-Flop D (DFF) và ứng dụng của nó trong các mạch điện khác.

1. Theory and Background

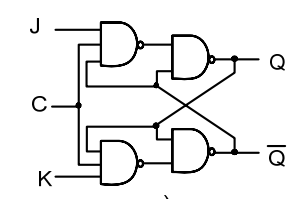
Triger JK

Có hai đầu vào J và K, ưu điểm hơn RS là không tồn tại tổ hợp cấm bằng các đường hồi tiếp từ Q về R và từ Q’ về S; Triger JK là triger đồng bộ.

Bảng trạng thái của triger JK đồng bộ:



Sơ đồ nguyên lý của triger JK:



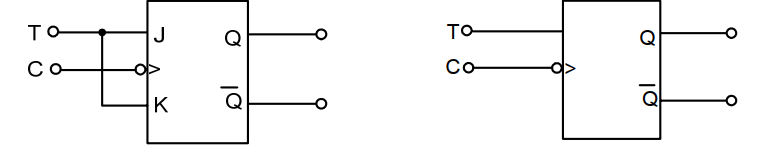
Công thức:



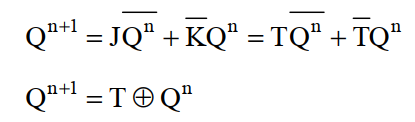
Triger T

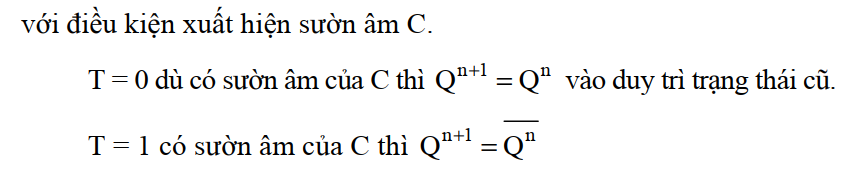
Triger T là mạch có chức năng duy trì và đổi trạng thái tùy thuốc vào tín hiệu đầu vào T trong điều kiện định thời của C.

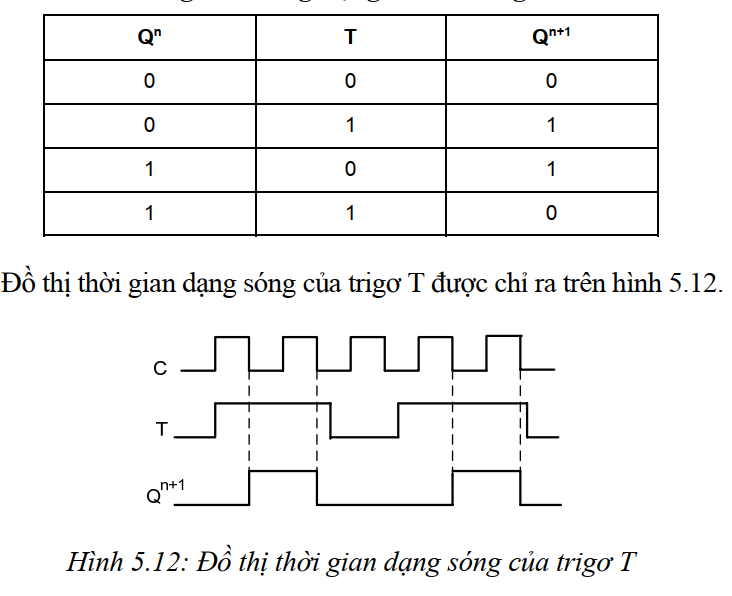
Kí hiệu:



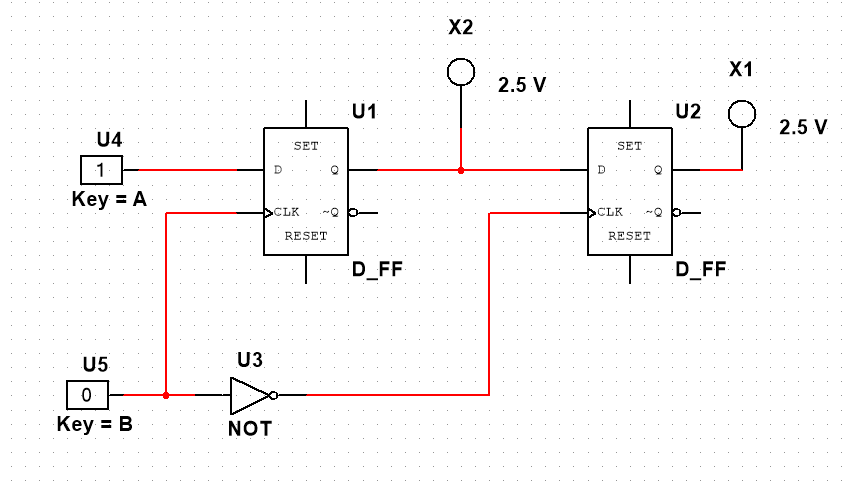
Nếu cho ta có triger từ triger , phương trình đặc trưng:







1. Simulate: Building a Master-Slave D Flip-Flop



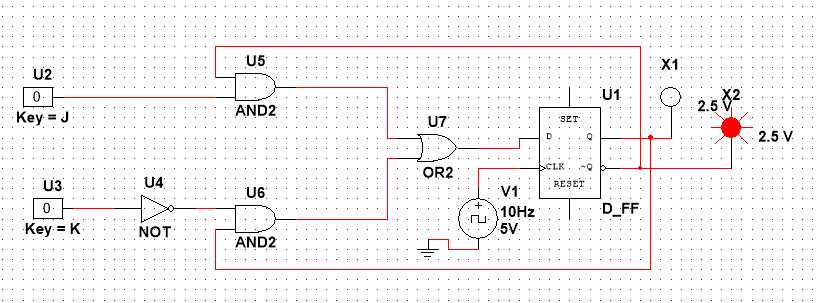
Ques3: Bắt đầu với A =1 probe x1 tối, Ques4: khi thay đổi B =1 probe x1 không thay đổi, probe x2 sáng, Ques5: khi thay đổi B =0, probe x1 sáng, probe x2 sáng. Ques6: thay đổi A =0, probe x1 không thay đổi, probe x2 tối. Ques7: Khi B thay đổi probe x1 sẽ có cùng trạng thái với A khi B chuyển từ 1 về 0. Ques8: mạch này gọi là kích cạnh âm.

1. Simulate: Building a JK Flip Flop Using a DFF and Logic Gates

Có biểu thức của D-FF là:

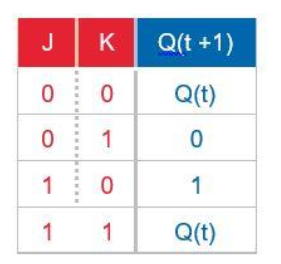
Biểu thức của JK-FF là:

Kết hợp hai biểu thức trên có:



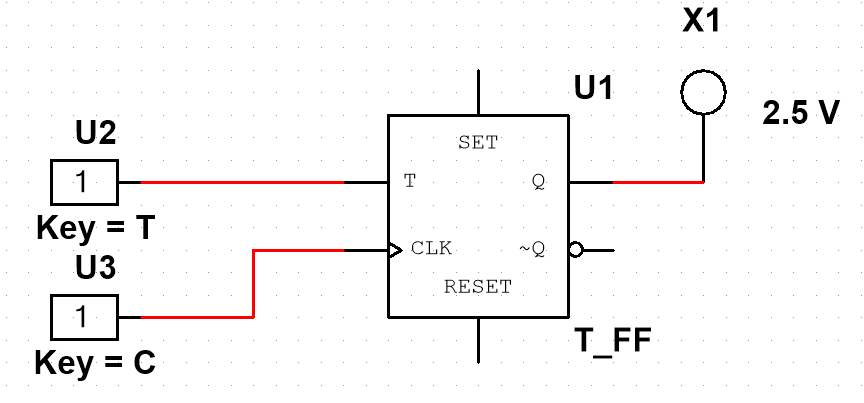
Ques9: Khi và thì khi đó xung tiếp theo đầu ra ; Khi và khi đó, với xung tiếp theo đầu ra ; Khi thì đầu ra Q sẽ thay đổi theo clock; Khi thì đầu ra sẽ không đổi trạng thái (được giữ nguyên).

Kết quả trên được mô tả bởi bảng trạng thái sau:



1. Simulate: Building a T Flip-Flop

Có công thức của T-FF là:

**

Khi , clock không làm thay đổi trạng thái đầu ra của T-FF, Khi clock làm thay đổi đầu ra của T-FF.

1. Conclusion

Ques14. việc mô phỏng cho phép kiểm tra và xác minh thiết kế mạch trước khi thực hiện thực tế, giúp phát hiện lỗi sớm, tiết kiệm thời gian và chi phí. Nếu cả cổng logic và chip đều hoạt động giống nhau trong mô phỏng, ta có thể yên tâm rằng kết quả thực tế cũng sẽ chính xác.

Ques15. flip-flop được kích hoạt tại cạnh xuống của xung đồng hồ (từ mức cao xuống mức thấp), nên chỉ khi có sự thay đổi từ 1 → 0 ở tín hiệu clock thì flip-flop mới thay đổi đầu ra.

Ques16. Khi nào bạn có thể sử dụng T-FF   
Flip-flop loại T thường được sử dụng trong bộ đếm nhị phân vì đảo trạng thái mỗi lần nhận xung, phù hợp để tạo các xung nhịp chia đôi, hoặc các bộ đếm tuần tự.

**Lab 11: Counters**

1.1 Theory and Background

**Bộ đếm** là các mạch số tuần tự được sử dụng để đếm số xung đầu vào. Bộ đếm trải qua một chuỗi trạng thái xác định trước khi có xung nhịp.

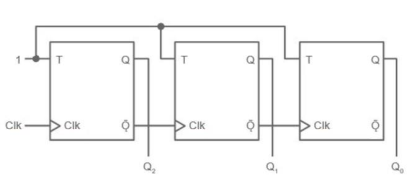
Bộ đếm được chia thành hai nhóm:

1. **Bộ đếm không đồng bộ** (*ripple counter*): đầu ra của một flip-flop được kết nối với đầu vào xung nhịp của flip-flop tiếp theo.
2. **Bộ đếm đồng bộ** (*synchronous counter*), trong đó tín hiệu xung nhịp được kết nối trực tiếp đến đầu vào xung nhịp của tất cả các flip-flop.

**Bộ đếm không đồng bộ**

Các bộ đếm này được gọi như vậy vì các flip-flop không thay đổi trạng thái đồng bộ với tín hiệu xung nhịp được áp dụng, chỉ có flip-flop đầu tiên phản hồi với các xung nhịp.

Hình dưới đây trình bày mạch đơn giản nhất cho bộ đếm lên ba bit, được xây dựng bằng các flip-flop loại T.



* Đầu vào T của mỗi flip-flop trong ba flip-flop được kết nối với mức 1, vì vậy chúng sẽ đảo trạng thái mỗi khi đầu vào xung nhịp thay đổi từ **0** sang **1**.
* Các loại bộ đếm này cũng được gọi là **bộ đếm ripple** (*ripple counters*) vì hành vi của chúng tương tự như quá trình lan truyền trong một bộ cộng ripple-carry.
* Chúng cũng có thể được triển khai bằng flip-flop loại **D** hoặc **J-K**.

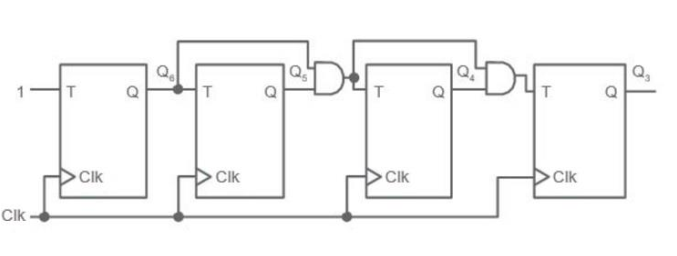
Trong bộ đếm, tín hiệu đầu ra của flip-flop cuối cùng sẽ có tần số bằng tần số của xung nhịp đầu vào chia cho số mô-đun của bộ đếm. Nhược điểm chính của bộ đếm không đồng bộ đến từ độ trễ lan truyền trong các flip-flop, khiến các mạch loại này chỉ hữu ích trong các ứng dụng tần số thấp.

**Bộ đếm đồng bộ**

Bộ đếm đồng bộ có tín hiệu xung nhịp được cấp trực tiếp vào đầu vào xung nhịp của tất cả các flip-flop tạo nên nó. Tất cả các flip-flop được kích hoạt đồng thời (song song). Chúng khắc phục vấn đề độ trễ tích lũy trong bộ đếm ripple bằng cách áp dụng xung nhịp cho tất cả các flip-flop. Quyết định có đảo đầu ra của một flip-flop hay không được xác định bởi giá trị của các đầu vào dữ liệu, chẳng hạn như **T**, **J** và **K**, tại thời điểm xung nhịp.

Trong trường hợp bộ đếm nhị phân, flip-flop tương ứng với bit ít quan trọng nhất sẽ được đảo trạng thái với mỗi xung nhịp. Bất kỳ flip-flop nào khác sẽ đảo trạng thái khi tất cả các flip-flop ở các bậc có trọng số thấp hơn đều ở mức 1. Bộ đếm đồng bộ có thể cập nhật trạng thái của nó ở cạnh lên hoặc cạnh xuống của tín hiệu xung nhịp và có thể được xây dựng bằng các flip-flop loại J-K, T, D kết hợp với cổng XOR.

* Loại bỏ độ trễ tích lũy trong bộ đếm ripple và có thể hoạt động ở tần số cao, nhưng phải đánh đổi bằng mức tiêu thụ điện năng cao hơn và mạch phức tạp hơn.
* Bộ đếm dưới đây là một bộ đếm đồng bộ với kích hoạt nối tiếp, trong đó các tín hiệu kích hoạt đảo trạng thái được tính toán tuần tự.



Enable and Clear Inputs

* Đầu vào **Enable** ngăn chặn việc đếm bất kể trạng thái tín hiệu xung nhịp.
* Đặt lại không đồng bộ (asynchronous resetting) của bộ đếm cũng có thể được yêu cầu trong một số trường hợp.
* Mức thấp tại đầu vào **Clear** đặt tất cả các flip-flop trong mạch ở mức thấp, bất kể đầu vào.

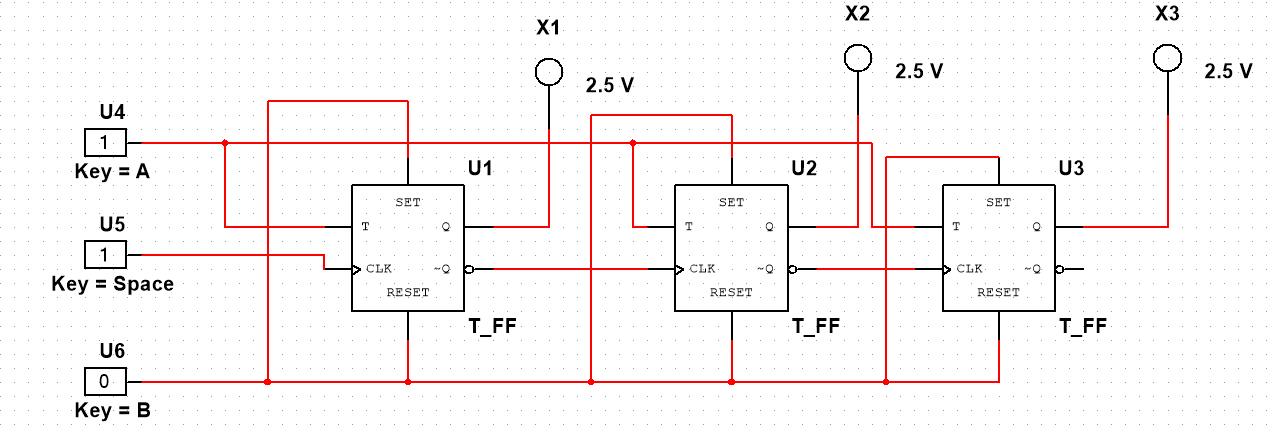
Bộ đếm tăng đồng bộ bốn bit với các đầu vào Enable và Clear:



Ques1: Đầu vào Enable và Clear ảnh hưởng đến bộ đếm:

Enable quyết định liệu bộ đếm có thể đếm hay không, kiểm soát việc bộ đếm có hoạt động hay không; Clear dùng để reset bộ đếm về giá trị 0 (hoặc một giá trị đã định trước), ghi đè lên tất cả các đầu vào khác và ngừng quá trình đếm cho đến khi có hành động tiếp theo.

1.2 Simulate: Building a Three-bit Binary Ripple Counter



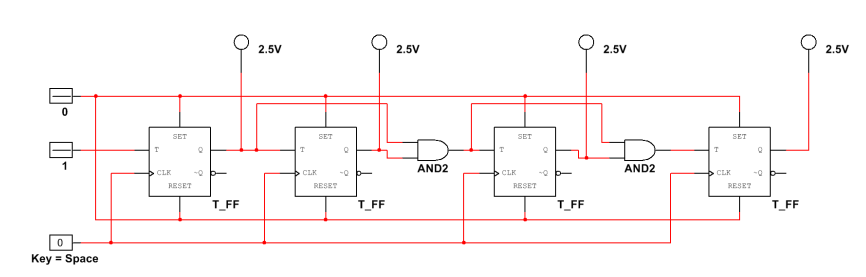
Ques2: chức năng của các digital constant là để thay đổi trạng thái logic ở các cổng của các trigger và các cổng Clear, Set của trigger,

Ques3: Khi có xung clock chuyển trạng thái từ mức thấp sang mức cao, bộ đếm (counter) sẽ tăng một đơn vị.

Ques4: Khi có xung clock đầu ra của trigger sẽ thay đổi theo phương trình:

Ques5: Khi có xung clock, bộ đếm (Counter) đếm tăng dần từ khi tràn thì đếm lại từ đầu

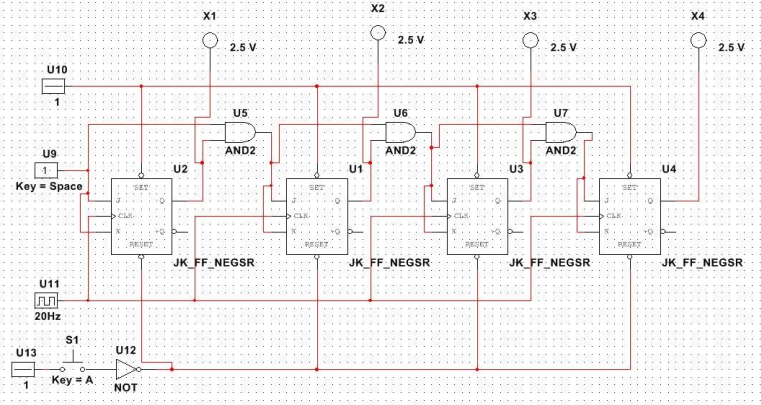
1.3 Simulate: Building a Four-Bit Up-Counter



Ques6: Thay đổi giá trị đầu vào khi , , và có xung clock thì mạch đếm tăng dần; Ques7: chuyển từ 0 lên 1 thì mạch đếm tăng dần; Ques8: Mạch đếm từ

Ques9: Hai mạch khác nhau ở cách cấp xung clock, mạch ở 1.2 là ripple counter đâu ra của phần từ FF phía trước là xung clock của phần từ FF phía sau; Mạch ở 1.3 xung clock được cấp đồng thời cho các phần tử FF.

* 1. Simulate: Building a Four-Bit Up-Counter with Enable and Clear



Ques10: thay đổi thì mạch đếm từ ; Ques11: Khi ấn button bộ đếm bị xóa.

1.5 Conclusion

Ques12. Sau khi quan sát bộ đếm lên, dự đoán bộ đếm xuống sẽ hoạt động ngược lại với bộ đếm lên. Thay vì tăng giá trị đếm sau mỗi xung nhịp, nó sẽ giảm giá trị. Ví dụ, nếu bộ đếm lên đi , thì bộ đếm xuống sẽ đi từ.

Ques13. Bộ đếm giống latch hay bộ cộng hơn? Giải thích.

Bộ đếm giống latch hơn vì nó lưu trữ trạng thái và thay đổi trạng thái đó theo từng xung nhịp. Trong khi bộ cộng thực hiện phép toán tại một thời điểm, bộ đếm duy trì giá trị đếm qua thời gian giống như latch duy trì dữ liệu cho đến khi có tín hiệu điều khiển mới.

Ques14. Yếu tố nào quyết định bộ đếm có thể đếm cao đến mức nào? Cần làm gì để tăng giới hạn đếm của nó?

Số bit của bộ đếm quyết định nó có thể đếm cao đến mức nào. Với n bit, bộ đếm có thể đếm từ 0 đến . Để tăng giới hạn đếm, ta cần tăng số bit trong thiết kế của bộ đếm.

**Lab 12: Finite State Machines**

Trong Lab12, phân tích một cách tiếp cận mới để mô hình hóa: FSM - Finite State Machine. FSM thể hiện các trạng thái khác nhau mà một hệ thống có thể có và sự kết hợp của các đầu vào cần thiết để đạt được những trạng thái đó. Bạn sẽ khám phá các loại hệ thống đèn giao thông khác nhau và cách FSM có thể được sử dụng để mô hình hóa chúng.

* 1. Theory and Background

*Finite State Machines*

FSM - Finite State Machine là một mô hình trừu tượng được dùng để mô tả hành vi của một đối tượng.

Mỗi vòng tròn đại diện cho một trạng thái trong hệ thống, trong khi mỗi mũi tên biểu thị sự chuyển trạng thái sang một trạng thái khác. Vì số lượng vòng tròn là hữu hạn, nên chỉ có một số lượng hữu hạn các trạng thái có thể xảy ra. FSM chỉ có thể ở một trạng thái duy nhất tại một thời điểm. Trạng thái này được gọi là trạng thái hiện tại. Nhãn trên mũi tên là các sự kiện kích hoạt sự chuyển trạng thái.

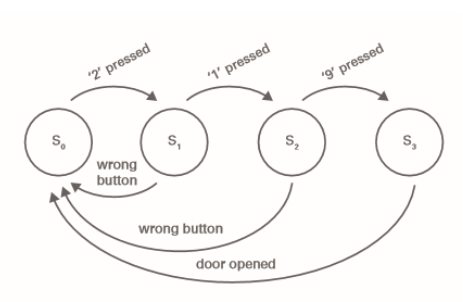
Finite State Machine được sử dụng trong:

* Máy bán hàng tự động: phân phối sản phẩm khi đã nạp đúng số tiền.
* Thang máy: thả người xuống các tầng trên rồi quay trở lại.
* Đèn giao thông: thay đổi màu đèn theo chu kỳ định sẵn.

Ví dụ FSM đơn giản cho một hệ thống mở khóa khi nhấn đúng chuỗi nút, ví dụ này đại diện cho cánh cửa đang bị khóa:

Trạng thái 0 (S0) đại diện cho trạng thái khi khóa đang bật.

* S0 chuyển sang S1 sau khi nút ‘2’ được nhấn.
* S1 có hai lựa chọn cho trạng thái tiếp theo – S2 nếu nút ‘1’ được nhấn, nếu không sẽ trở lại S0.
* S3 là trạng thái khi khóa bị tắt và chỉ bật lại khi cửa đã được mở.



Cách mà một đèn giao thông có thể được tạo ra:



Timer: tạo ra các khoảng thời gian cố định để điều khiển chu kỳ hoạt động của đèn giao thông (ví dụ: đèn đỏ sáng trong 30 giây, đèn xanh trong 60 giây). Counter (Bộ đếm) đếm số lần tín hiệu từ Timer được gửi đến và quyết định khi nào chuyển đổi trạng thái đèn (từ đỏ sang xanh, từ xanh sang vàng). LED là tín hiệu (đỏ, vàng, xanh) và hiển thị trạng thái đèn giao thông tương ứng theo tín hiệu từ Counter.

Bảng trạng thái cho mạch đèn tín hiệu này:

Dưới đây là bảng đặc trưng (characteristic table) cho máy trạng thái gồm 16 trạng thái, với các đầu vào A, B, C, D và đầu ra là đèn Green, Yellow, Red đúng theo yêu cầu:

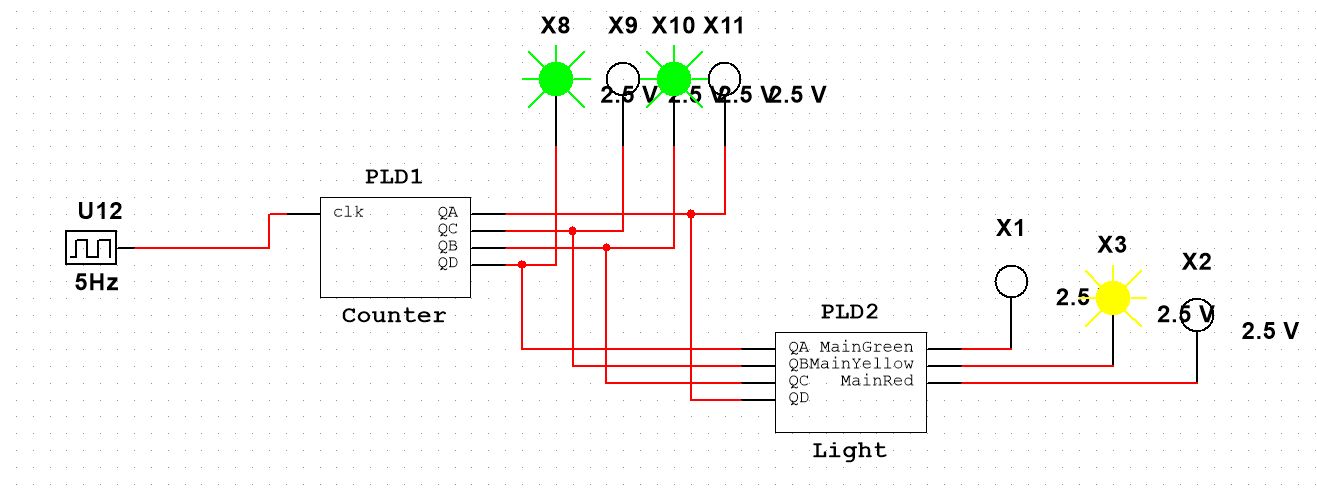
| **A** | **B** | **C** | **D** | **Green** | **Yellow** | **Red** |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 1 |

**Tổng cộng:** Green: 8 trạng thái; Yellow: 3 trạng thái; Red: 5 trạng thái

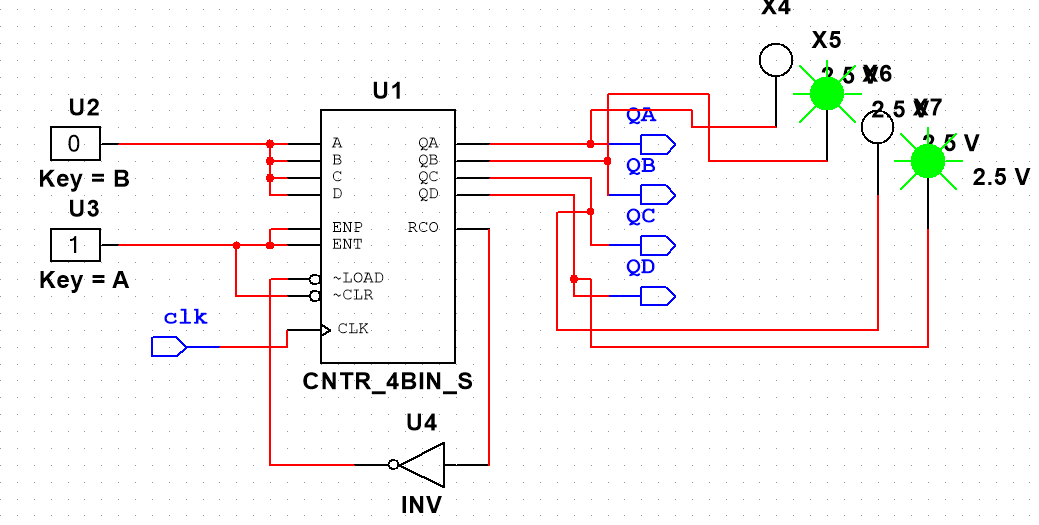
Dựa vào bảng trạng thái trên có:

* 1. Exercise: Building a Signal Traffic Light in PLD Design

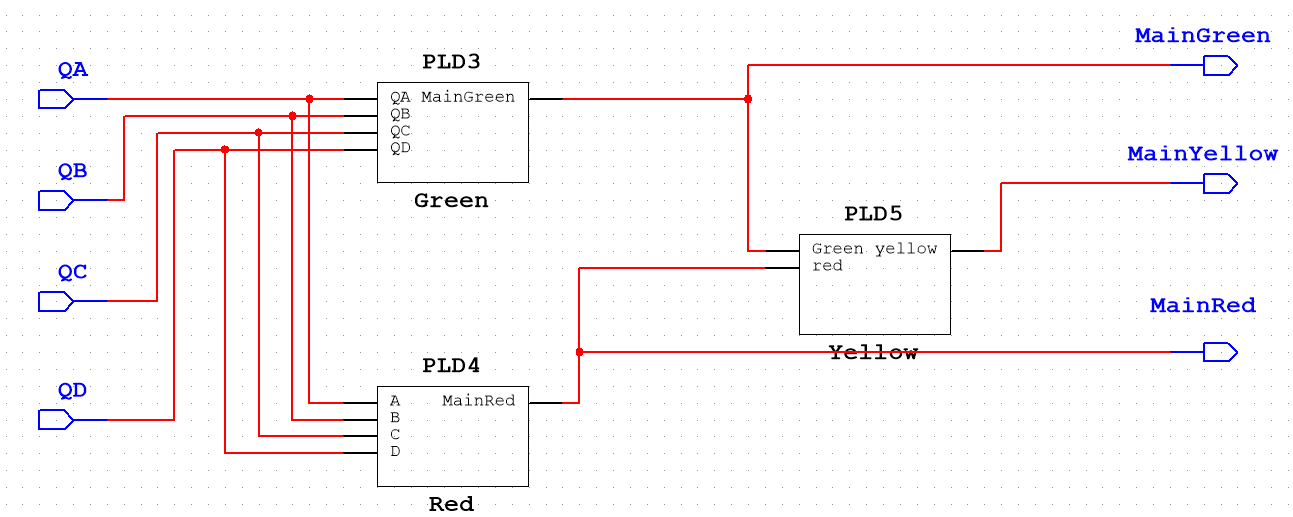
Mạch Signal Traffic Light



Mạch counter:



Mạch Light:



Các mạch con:

|  |  |  |
| --- | --- | --- |
| Green | Red | Yellow |

* 1. Simulate: Testing a Single Traffic Light in PLD Design

Đèn tín hiệu chạy theo chu trình:

Ques2: Thời gian chạy hết một chu kỳ là khi counter chạy từ tổng cộng có 16 trạng thái, counter được chạy với clock tức là mỗi trạng thái kéo dài . Như vậy, một chu kỳ của đèn tín hiệu là .

Ques3: Có 8 trạng thái của counter làm đèn sáng: . Như vậy, thời gian sáng của đèn xanh là

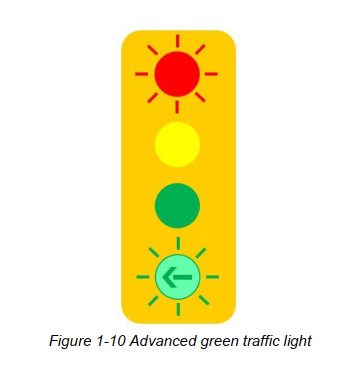
Ques4: Số trạng thái đèn xanh sáng là 8/16 trạng thái. THời gian đèn xanh sáng là 1.6/3.2

* 1. Implement: Using a Function Generator

(Thực nghiệm trên phần cứng)

* 1. Exercise: Two Way Traffic Lights

Đôi khi, đèn giao thông có đèn thứ tư cho phép người lái xe rẽ trái trong khi dòng xe đối diện dừng lại và thường bật trong một khoảng thời gian ngắn trong khi đèn đỏ cũng đang bật.

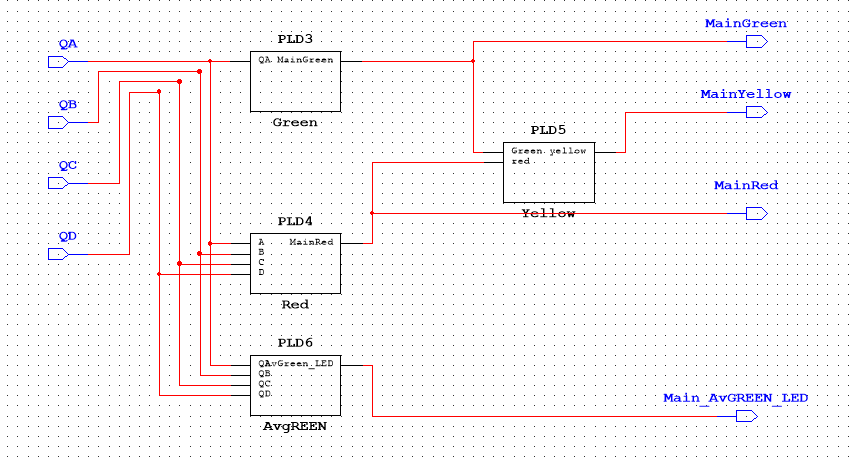


Dựa vào bảng trạng thái từ phần 1.1 và đổi đèn đỏ thành xanh, và đèn xanh thành đèn đỏ:

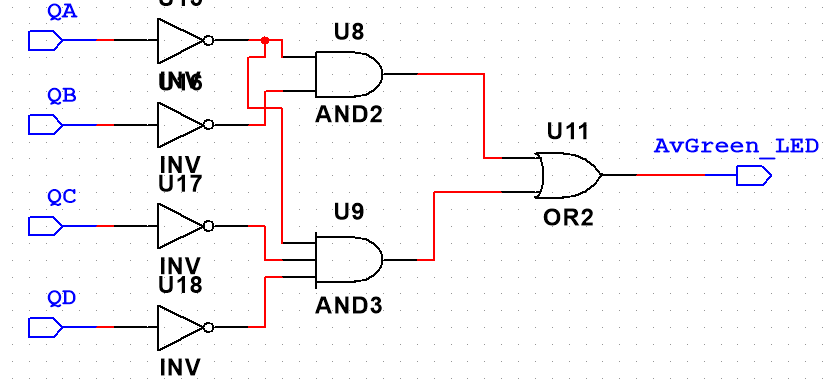
Đèn phụ sẽ sáng trong một khoảng thời gian trong khi đèn đỏ vẫn sáng. Giả sử đèn phụ sáng cùng với đèn đỏ từ

Khi đó: Biểu thức logic của đèn phụ sẽ là:

Mạch logic mới:



Mạch đèn AvGreen\_LED là:



* 1. Conclusion

Ques5: Tín hiệu clock điều khiển các đèn bằng cách tạo ra các xung thời gian đều đặn, từ đó điều khiển quá trình chuyển trạng thái của FSM. Mỗi khi có xung clock, hệ thống chuyển sang trạng thái tiếp theo, làm thay đổi đèn (đỏ, vàng, xanh) theo trình tự đã lập trình sẵn.

Ques6: Việc chia mạch thành các PLD (Programmable Logic Devices) giúp thiết kế trở nên gọn gàng, dễ bảo trì, và dễ mở rộng, có thể tái sử dụng các khối logic và lập trình linh hoạt mà không cần phải thay đổi phần cứng vật lý.

Ques7: Để tạo ra nhiều trạng thái hơn, bạn cần thêm nhiều bit cho thanh ghi trạng thái (n-state register), hoặc mở rộng logic trạng thái trong FSM (ví dụ: từ 2 bit → 3 bit cho phép tăng số trạng thái từ 4 → 8).

Ques8: FSM chỉ có một số lượng trạng thái nhất định.

**Lab 13: Shift Registers**

Trong Lab11: tìm hiểu cách các flip-flop có thể được triển khai để tạo ra bộ đếm. Trong Lab này, bạn sẽ khám phá cách các flip-flop có thể được sắp xếp để tạo ra thanh ghi dịch. Thanh ghi dịch (Shift Registers) là một nhóm các flip-flop được sắp xếp sao cho các bit được lưu trữ được dịch sang trái hoặc sang phải, có hiệu quả nhân hoặc chia đầu vào tương ứng.

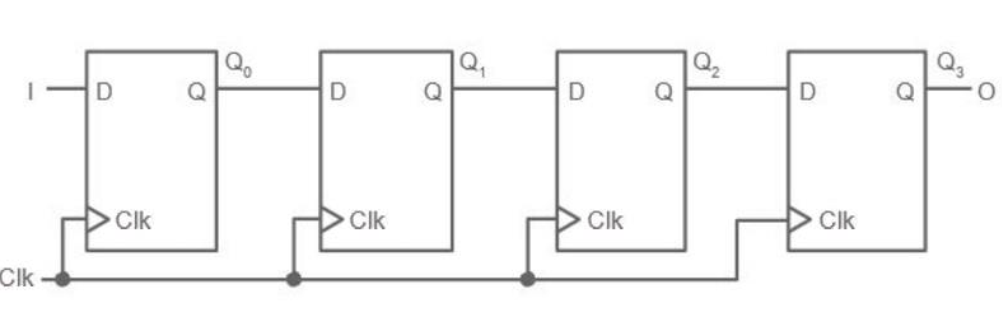
* 1. Theory and Background

Thanh ghi dịch (Shift Registers)

Shift registers là các mạch dùng để lưu trữ và di chuyển dữ liệu, được tạo thành từ một nhóm các flip-flop. Nhóm các flip-flop này được sắp xếp sao cho dữ liệu lưu trữ sẽ được dịch chuyển từ flip-flop này sang flip-flop kế tiếp sau mỗi xung nhịp (clock pulse) theo một hướng đã chọn. Chúng được sử dụng trong việc chuyển đổi nối tiếp/song song, truyền dữ liệu nối tiếp, các chức năng số học và các phần tử trễ (delay elements).

Phép nhân nhị phân của một số với 2 được thực hiện bằng cách dịch các bit của nó sang trái và chèn một bit 0 vào vị trí bit ít quan trọng nhất. Bằng cách dịch tất cả các bit của một số nhị phân sang phải một vị trí, nó sẽ được chia cho 2.

Hình bên dưới trình bày thanh ghi dịch đơn giản nhất, được tạo ra bằng cách nối đầu ra của mỗi flip-flop vào đầu vào của flip-flop nằm bên phải nó.



Nó dịch các bit dữ liệu của nó sang phải một vị trí, là một thanh ghi dịch một chiều.

Dữ liệu được nạp vào thanh ghi dịch theo kiểu nối tiếp thông qua đầu vào , và dữ liệu của từng flip-flop sẽ được chuyển đến flip-flop kế tiếp (với flip-flop cuối cùng thì chuyển đến đầu ra O) tại cạnh lên (rising edge) của tín hiệu đồng hồ (clock). là đầu vào nối tiếp, trong khi là đầu ra nối tiếp của thanh ghi dịch.

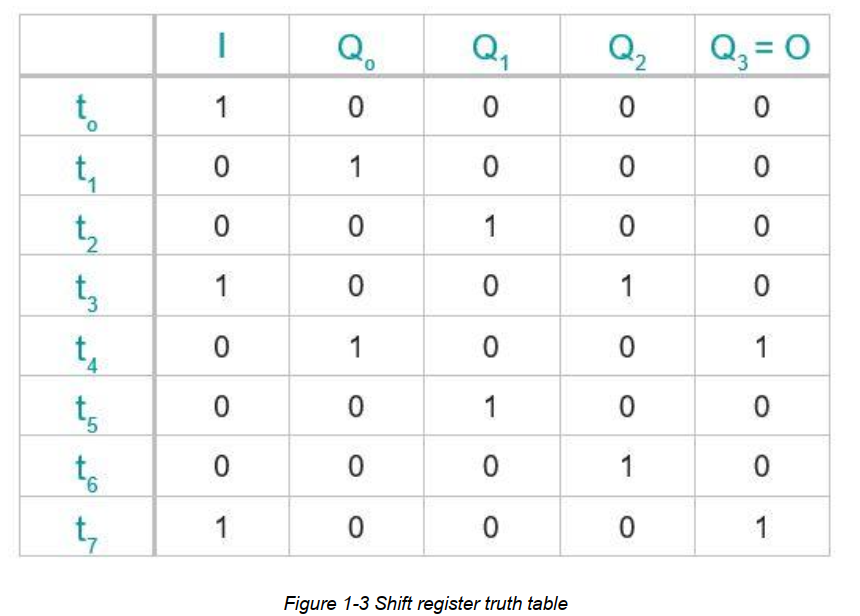
Các loại thanh ghi dịch (Types of Shift Registers)

Có bốn loại thanh ghi dịch:

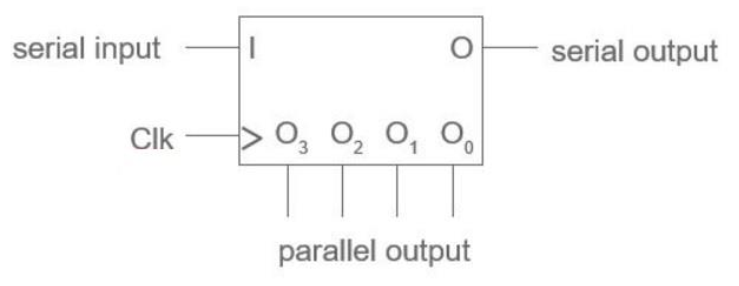
* Serial-in/Parallel-out (SIPO): Dữ liệu nối tiếp được nạp vào thanh ghi từng bit một và có thể truy xuất tại đầu ra dưới dạng song song cùng lúc.
* Serial-in/Serial-out (SISO): Dữ liệu được nạp vào và xuất ra khỏi thanh ghi dịch từng bit một.
* Parallel-in/Serial-out (PISO): Dữ liệu được nạp vào đồng thời (song song), nhưng được dịch ra từng bit một theo kiểu nối tiếp.
* Parallel-in/Parallel-out (PIPO): Dữ liệu được nạp vào đồng thời và được dịch ra đồng thời.

Các thanh ghi dịch chỉ có thể được xây dựng bằng cách sử dụng mạch kích hoạt theo cạnh (edge-triggered), không phải bằng các chốt (latch) nhạy mức (level-sensitive). Khi sử dụng các mạch nhạy mức, giá trị của đầu vào có thể lan truyền qua nhiều phần tử trong thời gian tín hiệu clock ở mức 1.

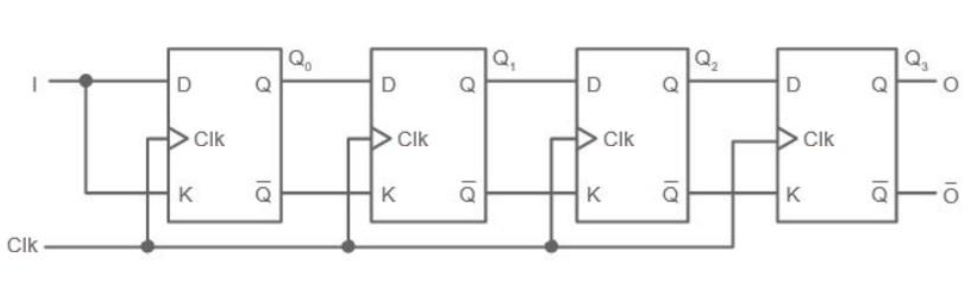
Hình dưới đây minh họa cách mà dữ liệu lưu trữ trong thanh ghi dịch thay đổi phụ thuộc vào đầu vào , trong một chuỗi gồm tám chu kỳ xung nhịp, và giả sử rằng trạng thái ban đầu của tất cả các flip-flop là .



Hình dưới đây hiển thị ký của một thanh ghi dịch bốn bit:



Mạch ghi dịch được built sử dụng JK-FF:



Ques1. Những điểm khác biệt chính về cấu trúc và chức năng giữa thanh ghi dịch một chiều (unidirectional) và hai chiều (bidirectional):

Cấu trúc:

* Thanh ghi dịch một chiều chỉ cho phép dịch dữ liệu theo một hướng duy nhất (ví dụ: từ trái sang phải hoặc ngược lại).
* Thanh ghi dịch hai chiều (reversible) được thiết kế để có thể dịch dữ liệu theo cả hai hướng, nhờ có thêm mạch điều khiển hướng dịch và các cổng logic phụ trợ.

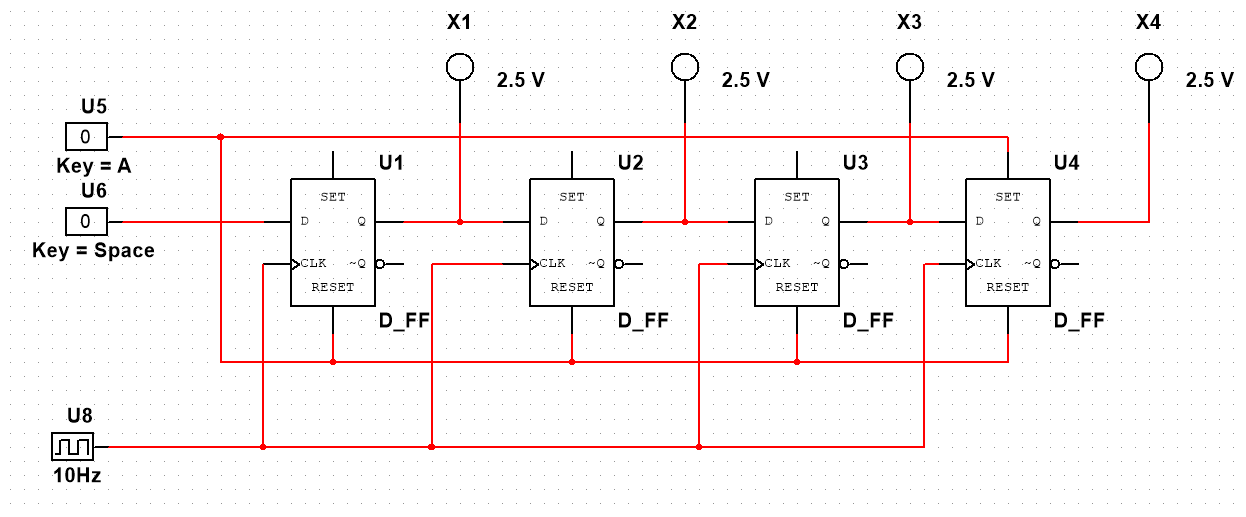
Chức năng:

* Thanh ghi một chiều phù hợp cho các ứng dụng đơn giản, như truyền dữ liệu nối tiếp hoặc chuyển đổi giữa nối tiếp và song song theo một hướng.
* Thanh ghi hai chiều linh hoạt hơn, cho phép thao tác dữ liệu theo cả hai hướng, hữu ích trong các phép toán số học, xử lý tín hiệu hoặc khi cần truyền dữ liệu qua lại.

Điều khiển:

* Loại một chiều chỉ cần tín hiệu đồng hồ (clock) để kích hoạt dịch bit.
* Loại hai chiều cần thêm tín hiệu điều khiển hướng (direction control signal) để xác định hướng dịch.
  1. Exercise: Building and Testing a Four-Bit Shift Register

Mạch ghi dịch 4 bit:



Ques2: Thay đổi đầu vào của D-FF đầu từ đầu ra dữ liệu sẽ được dịch lần lượt từ trái sang phải

Ques3: Thời gian dịch sang D-FF bên cạch là do tần số clock đang là

Ques4: Tăng tần số clock lên thời gian dịch sang D-FF bên cạch là

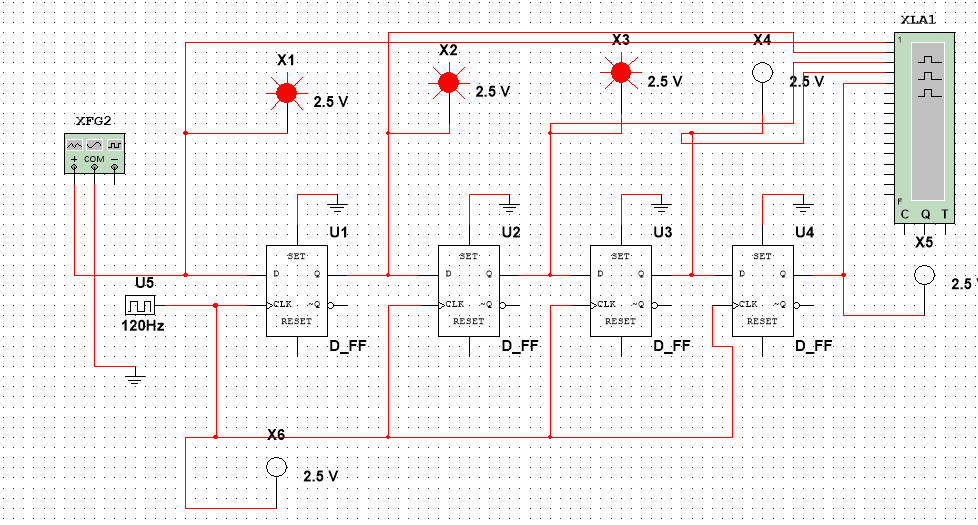
Ques5: Giảm tần số clock còn thời gian dịch sang đầu ra kế tiếp là

Ques6: Tốc độ dịch của mạch phụ thuộc vào tần số xung clock, khi có xung clock dữ liệu ở đầu vào D của mỗi D\_FF sẽ được dịch sang đầu ra Q, các D\_FF này được nối tiếp nhau nên data sẽ được dịch lần lượt từ trái sang phải.

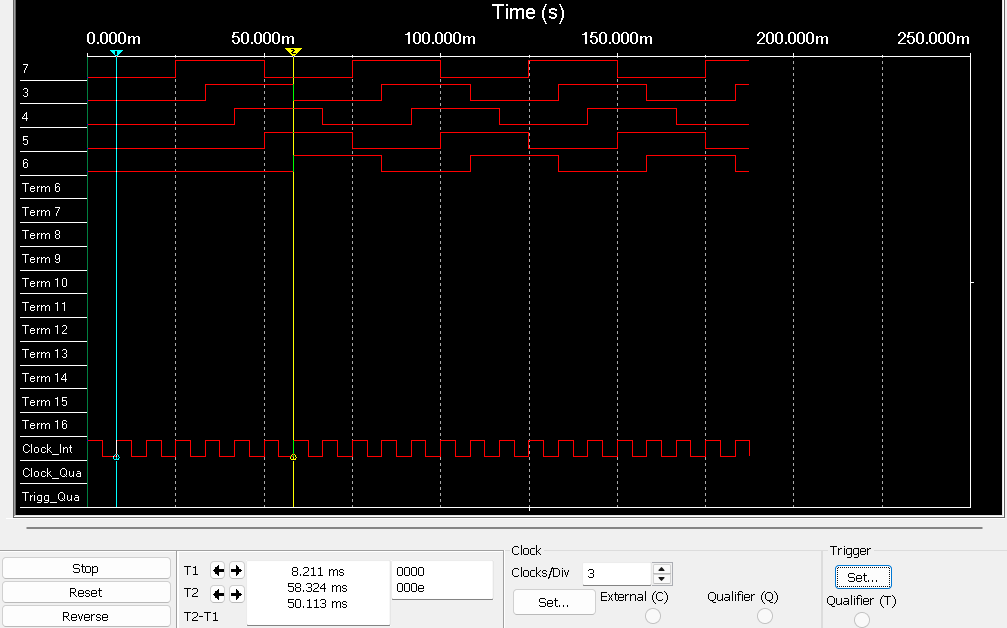
Ques7: Mạch trên là mạch ghi dịch vào nối tiếp

* 1. Exercise: Building and Testing a Serial-In/Parallel-Out Shift Register

Mạch ghi dịch vào nối tiếp – ra song song:



Kết quả tín hiệu đầu ra trên các probe và clock



Ques8: Phân tích Dạng Sóng: Clock tín hiệu clock chính, là xung nhịp điều khiển mạch. tín hiệu clock có tần số cố định và dạng sóng vuông. Term 3, 4, 5, 6: Đây là các tín hiệu đầu ra của các flip-flop trong mạch ghi dịch. Sự dịch chuyển: Các tín hiệu này dịch chuyển theo thời gian dữ liệu đang được dịch qua các flip-flop. Có độ trễ nhỏ giữa các tín hiệu, thời gian cần thiết để tín hiệu clock truyền qua mỗi flip-flop và làm thay đổi đầu ra của nó. Các flip-flop đang hoạt động theo cạnh lên của clock.

* 1. Conclusion

Ques9. T flip-flop không có khả năng lưu trữ và dịch dữ liệu theo chuỗi như yêu cầu của thanh ghi dịch, thanh ghi dịch cần flip-flop có đầu vào dữ liệu cụ thể (như D hoặc JK) để truyền dữ liệu từ flip-flop này sang flip-flop khác.

Ques10. Một universal shift register (thanh ghi dịch đa năng) có khả năng thực hiện tất cả các chế độ hoạt động:

* Dịch dữ liệu sang trái và sang phải (bidirectional shift),
* Nạp dữ liệu theo kiểu nối tiếp hoặc song song,
* Xuất dữ liệu theo kiểu nối tiếp hoặc song song.

Vì vậy, chức năng chính của nó là kết hợp tất cả bốn loại thanh ghi dịch: SIPO, SISO, PISO, PIPO, và nó còn có thể dịch theo cả hai hướng.

Ques14. Khi tần số của xung clock ở đầu vào thay đổi thời gian cần để dữ liệu dịch giữa các flip-flop thay đổi, tần số clock càng cao thì dữ liệu sẽ dịch nhanh hơn, và ngược lại.

**Lab 14: Semiconductor Memory**

**Lab 15: Digital Dice**

**Lab 16: Digital Clock – Application Lab #1**

-

**Lab 17: Electronic Safe – Application Lab #2**

**Lab 18: Digital Communications – Application Lab #3**