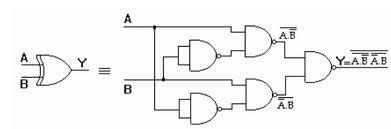
ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA VẬT LÝ-VẬT LÝ KỸ THUẬT



BÁO CÁO



Giáo Viên: ThS.Nguyễn Anh Huy

Sinh Viên : Hà Tiến Tự

Nguyễn Toàn Thắng

Nguyễn Ngọc Hà

# 

I. NỘI DUNG

- Khái niệm cơ bản về hệ thống số thập phân, nhị phân, bát phân, và thập luc phân.

- Chuyển đổi số giữa các hệ thống với nhau.

-

-

II. KHÁI NIỆM.

- Trong bài chúng ta chỉ xét đến các số nguyên dương.

1. Hệ thập phân.

- Hệ thâp phân (hệ đếm cơ số 10) là hệ đếm dùng số 10 làm cơ số.

- Một số bao gồm các chữ số có 10 giá trị cụ thể từ 0 đến 9, và mỗi chữ số đại diện cho một bội của cơ số 10.

- Để phân biệt với hệ thống các số khác, chúng ta thường viết số nhị phân như sau: (4520)10 nhưng thông thường chúng ta hay viết 4520 và hiểu rằng nó là số thập phân.

- Tổng quát:

Số thập phân: D = dn-1­dn-2…d1d0

Giá trị của D: V(D) = dn-1x10n-1 + dn-2x10n-2 + … + d1x101 + d0x100

=

Ví dụ: Giá trị của 7532 là: V = 7x103 + 5x102 + 3x101 + 2x100

2. Hệ nhị phân.

- Hệ nhị phân (hệ đếm cơ số 2) là hệ đếm dùng hai ký tự 0 và 1 để biểu đạt 1 giá trị số nào đó.

- Hệ nhị phân thường được dùng rộng rãi trong các mạch kỹ thuật số. Hai ký tự 0 và 1 thường được gọi là các bit.

- Để phân biệt với hệ thống các số khác, chúng ta thường viết số nhị phân như sau: (1011)2 với 1011 là số nhị phân

- Tổng quát:

Số nhị phân: B = bn-1­bn-2…b1b0

Giá trị của B: V(B) = bn-1x2n-1 + bn-2x2n-2 + … + b1x21 + b0x20

=

Ví dụ: Số nhị phân: (1101)2 biễu diễn giá trị là:

V = 1x23 + 1x22 + 0x21 + 1x20

= (13)10

3. Hệ bát phân.

- Hệ bát phân (hệ đếm cơ số 8) là hệ đếm dùng số 8 làm cơ số.

- Một số bao gồm các chữ số có 8 giá trị cụ thể từ 0 đến 7, và mỗi chữ số đại diện cho một bội của cơ số 8.

- Để phân biệt với hệ thống các số khác, chúng ta thường viết số bát phân như sau: (4571)8 với 4571 là số bát phân

- Tổng quát:

Số bát phân: O = On-1­On-2…O1O0

Giá trị của D: V(O) = On-1x8n-1 + On-2x8n-2 + … + O1x81 + O0x80

=

Ví dụ: Giá trị của (204)8 là: V = 2x82 + 0x81 + 4x80

= (68)10

4. Hệ thập lục phân.

- Hệ thập lục phân (hệ đếm cơ số 16) là hệ đếm dùng số 16 làm cơ số.

- Một số bao gồm các chữ số có 16 giá trị, giống như số thập phân, từ 0 đến 9 và các con số tiếp theo 10, 11, 12, 13, 14, 15 được biễu thị bằng các chữ cái A, B, C, D, E và F.

- Để phân biệt với hệ thống các số khác, chúng ta thường viết thêm chữ H vào sau số thập lục phân như sau: A74FH với A74F là số thập lục phân.

- Tổng quát:

Số thập lục phân: H = hn-1­hn-2…h1h0

Giá trị của H: V(H) = hn-1x16n-1 + hn-2x16n-2 + … + h1x161 + h0x160

=

Ví dụ: Số thập lục phân: A74FH biễu diễn giá trị là:

V = Ax163 + 7x162 + 4x161 + Fx160

= 10x162 + 7x162 + 4x161 + 15x160

= (42831)10

III. Chuyển đổi giữa các hệ thống.

1. Chuyển đổi giữa hệ thống thập phân và nhị phân.

- Để chuyển số từ thập phân sang nhị phân, ta lấy số thập phân chia cho 2, lúc này sẽ xảy ra 2 trường hợp là dư 0 hoặc 1, ta sẽ lấy số dư này là số đầu tiên của số nhị phân. Rồi tiếp tục lấy thương chia cho 2, và điền số dư vào bên phải số dư trước. Tiếp tục chia cho đến khi thương bằng 0 thì dừng.  
Ví dụ: Chuyển đổi số 567 sang số nhị phân, ta làm như sau:

dư   
 567 ÷ 2 = 283 1

283 ÷ 2 = 141 1

141 ÷ 2 = 70 1

70 ÷ 2 = 35 0

35 ÷ 2 = 17 1

17 ÷ 2 = 8 1

8 ÷ 2 = 4 0

4 ÷ 2 = 2 0

2 ÷ 2 = 1 0

1 ÷ 2 = 0 1

Vậy số 567 chuyển sang nhị bằng là : (1000110111)2

- Để chuyển từ nhị phân sang thập phân, ta lấy bit thấp nhất (bên trái) nhân với 20 rồi tiếp tục lấy bit tiếp theo nhân với 21, tiếp tục như vậy đến khi lấy hết các bit của số nhị phân.

Ví dụ: chuyển (1000110111)2 sang số thập phân, ta làm như sau:

(1000110111)2 = 1x20 + 1x21 + 1x22 + 0x23 + 1x24 + 1x25 + 0x26 + 0x27 +0x28 +1x29

= (567)10

2. Chuyển đổi giữa hệ thống nhị phân và bát phân.

- Để chuyển từ số bát phân sang nhị phân, ta lấy chữ số cuối bên trái của số bát phân quy đổi ra 3 bit của số nhị phân. Tiếp tục lấy chữ số tiếp theo bên trái của số bát phân lại quy đổi ra 3 bit và viết bên phải của 3 bit nhị phân ở trước. Tiếp tục lấy và quy đổi đến khi lấy hết các chứ số của số bát phân.  
Ví dụ: chuyển (725)8 thành số nhị phân, ta làm như sau:

Quy đổi : 5 101

2 010 + = 010101

7 111 + = 111010101

Vậy số nhị phân thu được là (111010101)2

- Để chuyển từ số nhị phân sang bát phân, ta lấy 3 bít thấp nhất của số nhị phân quy đổi ra số thập phân, số này sẽ là chữ số đầu tiên của số bát phân. Tiếp tục lấy 3 bít tiếp theo bên trái của số nhị phân quy đổi ra rồi đặt bên phải của số bát phân trước, cứ tiếp tục như vậy khi lấy hết các bit của số nhị phân. Trường họp lần lấy cuối cùng chỉ còn 2 bit (hoặc 1) thì ta viết them số 0 vào trước 2 bit đó để cho đủ 3 bit rồi quy đổi.

Ví dụ: chuyển (110101001)2 thành số bát phân, ta làm như sau:

Quy đổi:

Vậy số ta thu được số bát phân là (651)8

3. Chuyển đổi giữa hệ thống nhị phân và thập lục phân.

- Để chuyển từ số nhị phân sang thập lục phân, ta lấy 4 bít thấp nhất của số nhị phân quy đổi ra số thập phân, số này sẽ là chữ số đầu tiên của số thập lục phân. Tiếp tục lấy 4 bít tiếp theo bên trái của số nhị phân quy đổi ra rồi đặt bên phải của số thập lục phân trước, cứ tiếp tục như vậy khi lấy hết các bit của số nhị phân. Trường họp lần lấy cuối cùng chỉ còn 3 bit (hoặc 2) thì ta viết thêm số các 0 vào trước 2 bit đó để cho đủ 4 bit rồi quy đổi.

Ví dụ: chuyển (1011001010100111)2 thành số thập lục phân, ta làm như sau:

Quy đổi:

Vậy ta thu được số thập lục phân là B2A7H.

- Để chuyển từ số thập lục phân sang nhị phân, ta lấy chữ số cuối bên trái của số thập lục phân quy đổi ra 4 bit của số nhị phân. Tiếp tục lấy chữ số tiếp theo bên trái của số thập lục phân lại quy đổi ra 4 bit và viết bên phải của 4 bit nhị phân ở trước. Tiếp tục lấy và quy đổi đến khi lấy hết các chứ số của số thập lục phân.

Ví dụ: chuyển từ 9F3H thành số nhị phân, ta làm như sau:

Quy đổi:

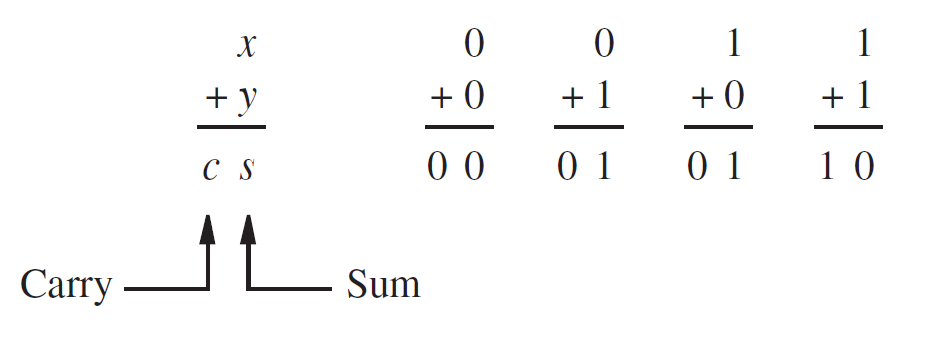
3 0011

F 1111 + = 11110011  
 9 1001 + = 100111110011

Vậy ta thu được số nhị phân là (100111110101)2

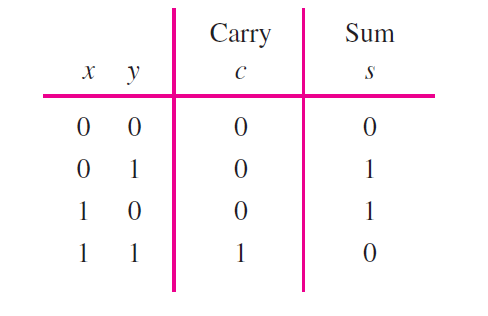
IV. PHÉP CỘNG CỦA CÁC SỐ KHÔNG DẤU

- Phép cộng của số nhị phận tương tự như phép cộng của số thập phân, tuy nhiên ở đây chỉ thực hiện với các giá trị 0 và 1. Kết quả của phép cộng gồm có 2 bit, bit bên phải được gọi là sum, kí hiệu s. Bit bên trái được gọi là carry, kí hiệu c. Khi cộng 2 bit với nhau thì sẽ có 4 trường hợp xảy ra.



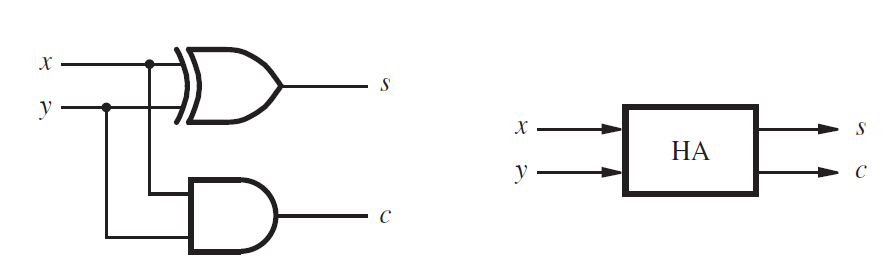
Hình 1. Bốn trường hợp xảy ra

- Bảng chân trị của phép cộng trên như hình 2:



Hình 2. Bảng chân trị

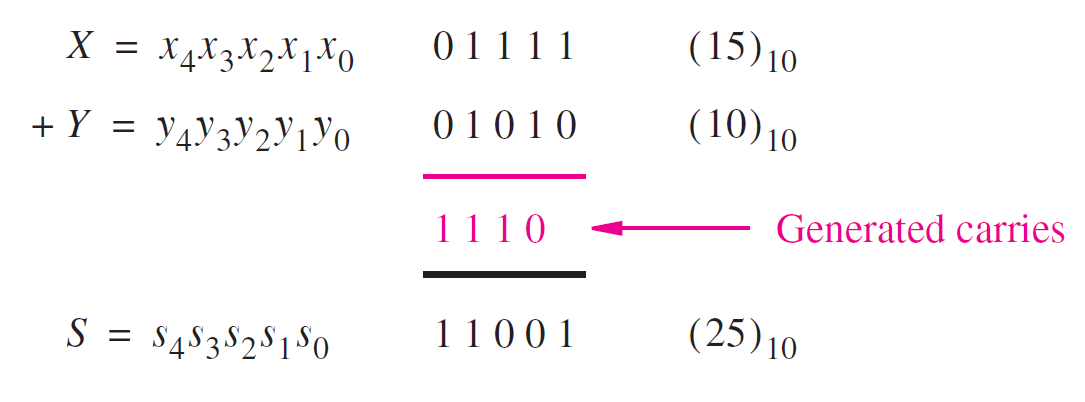
- Theo như bảng chân trị, bit s chính là hàm XOR còn bit c là hàm AND với 2 đầu vào là x và y.



Hình 3. Mạch logic

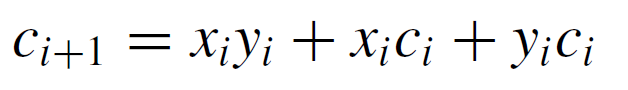
- Mạch thực hiện phép cộng chỉ có 2 bit, được gọi là half-adder.

- Trường hợp số nhị phân có nhiều bit, mỗi bit ở vị trí i, phép cộng có thể bao gồm carry-in (số nhớ) của bit ở vị trí i-1.

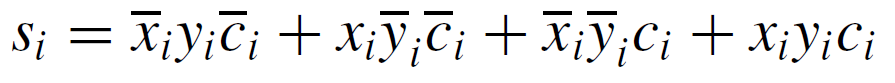


Hình 4. Ví dụ

- Đối với bit ở vị trí 0 ( tính từ phải qua trái), phép cộng giống như ví dụ ở hình 1. Đối với những bit khác ở vị trí i ( i khác 0), phép cộng có sự tham gia của 3 bit xi, yi và carry-in ci. Các hàm tổng và carry-out của các giá trị xi, yi, ci thì được xác định trong bản chân trị ở hình 5. Giá trị carry-out (ci+1) bằng 1 nếu tổng của xi, yi và ci bằng 2 hoặc 3. Hàm carry-out được biễu diễn như sau:



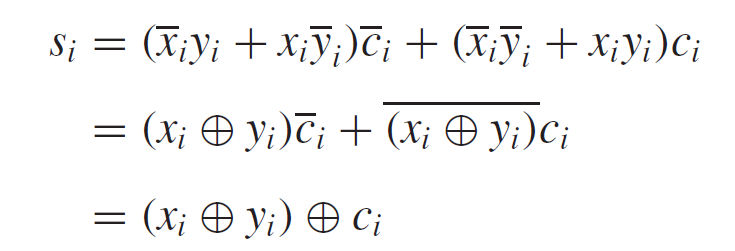
* Hàm biễu diễn si:



**SỬ DỤNG CỔNG XOR:**

- Hàm XOR của 2 biến x1, x2 được định nghĩa: x1 x2 = x2 +x1 .

- Chúng ta có thể biễu diễn lại phép cộng 2 số nhị phân mà chỉ dùng hàm XOR, biến đổi như sau:

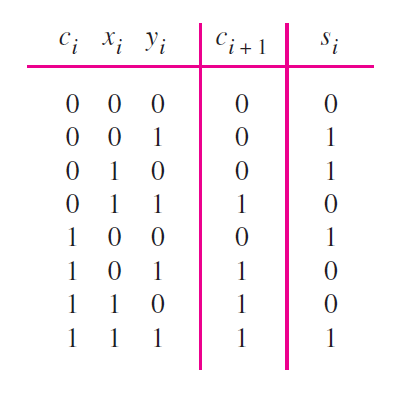


- Có thể viết lại như sau:

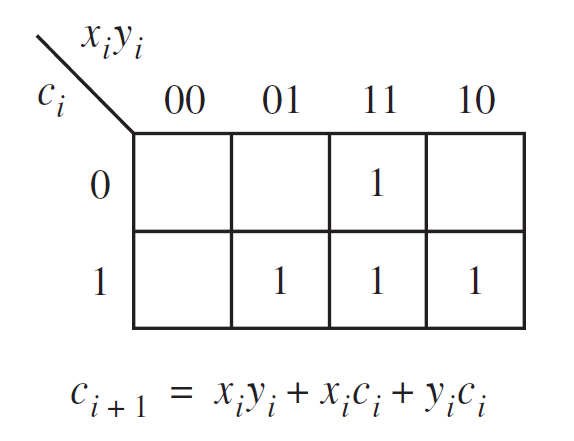
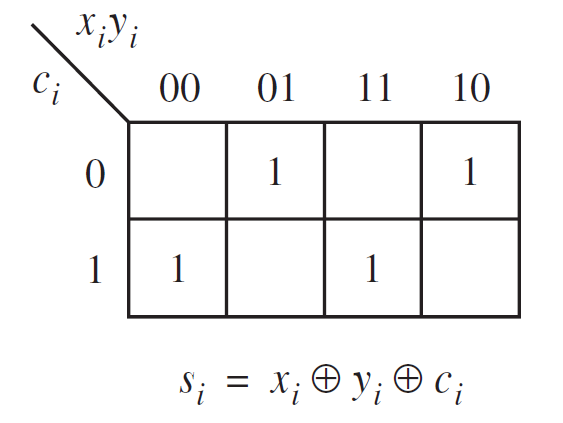


- Vì vậy, chúng ta có thể biễu diễn si bằng cổng XOR với 3 đầu vào.

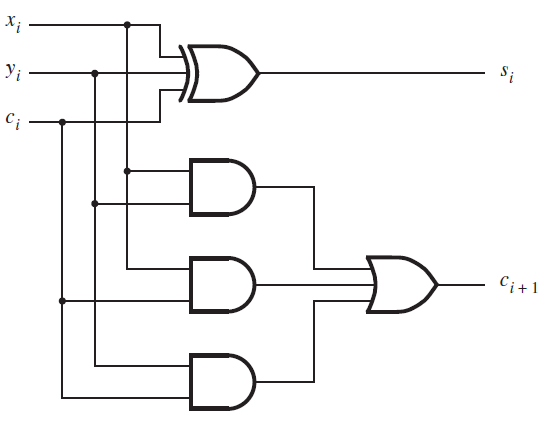
- Cổng XOR hoạt động như là ngõ ra của 2 tổng các ngõ vào. Ngõ ra sẽ có giá trị bằng 1 nếu các ngõ vào có số bit có giá trị 1 là lẻ, và ngược lại ngõ ra sẽ bằng 0. Mạch logic biểu diễn bảng chân trị hình 5a được xem như là full-adder (hình 5c).



Hình 5a. Bảng chân trị



Hình 5b. Bìa Karnaugh



Hình 5c. Mạch full-adder

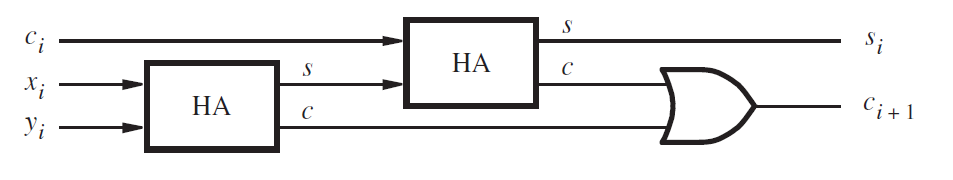
- Cổng XOR với hai đầu vào có thể dùng làm tín hiệu điều khiển để xác định giá trị ngõ ra là đảo hay không đảo. Từ định nghĩa của hàm XOR: x y = y +x , chọn x là ngõ vào điều khiển. Nếu x = 0, ngõ ra sẽ bằng giá trị của y, nếu x = 1, ngõ ra sẽ bằng .

- Bên cạnh cổng XOR, hàm đảo của nó được gọi là XNOR, ký hiệu là:

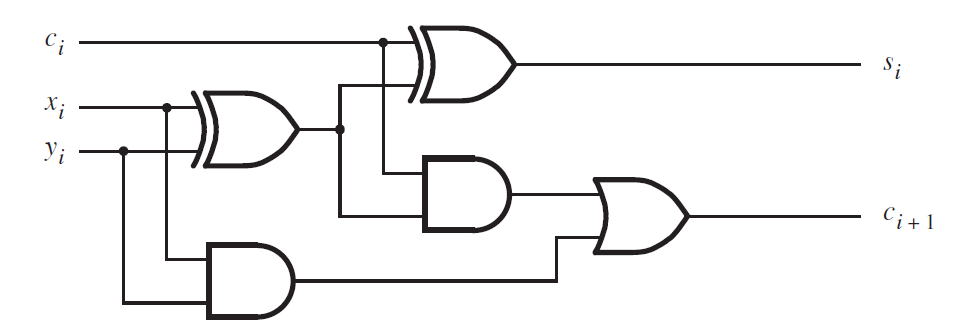
= , được sử dụng phổ biến. Ngõ ra của XNOR có giá trị là 1 khi 2 đầu vào có giá trị giống nhau ( 0 hoặc 1).

**1. Phân tích Full-Adder**

- Mạch full-adder được ghép từ 2 bộ half-adder



Hinh 6a. Sơ đồ khối mạch full-adder



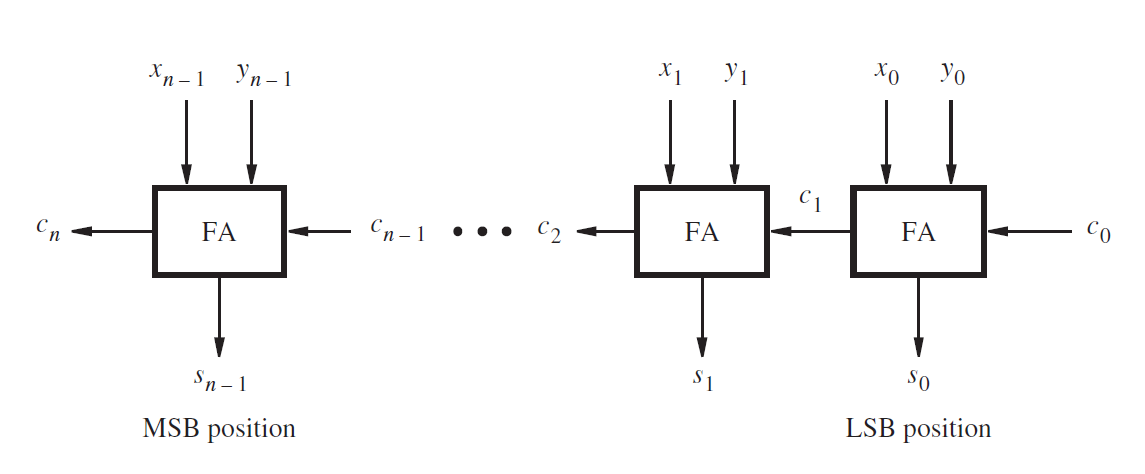
Hinh 6b. Biểu đồ chi tiết

**2. Ripple-Carry Adder**

- Phép cộng được thực hiện từ phải qua trái, từ bit thấp nhất (LSB) đến bit cao nhất (MSB).

- Khi các các toán hạng X, Y được đưa vào mạch cộng, cần có một khoảng thời gian để nhận được giá trị ra sum S. Mỗi mạch full-adder cũng đều có một thời gian trễ nhất định trước khi cho ra si và ci+1, thời gian trễ này được ký hiệu là . Carry-out từ phép cộng đầu tiên (c1) được tạo ra mất khoảng thời gian là . Carry-out từ phép cộng thứ hai (c2) được tạo ra mất 2. Như vậy tín hiệu c­n-1 sẽ được tạo ra sau thời gian trễ là (n-1), có nghĩa là phép cộng được hoàn thành sao thời gian trễ là n. Các tín hiệu carry ra theo dạng “ripple” trong suốt các giai đoạn full-adder. Vì vậy, mạch ở hình dưới (hình 7) được gọi là Ripple-Carry Adder.

- Từ đầu, chúng ta thực hiện phép cộng chỉ với các số nguyên dương, vì vậy phép cộng ở vị trí đầu tiên không bao gồm carry-in c0. Trong hình 7, có sự xuất hiện c0 vì thế bộ Ripp-Carry Adder này có thể sử dụng cho các số nguyên âm, phần này sẽ được trình bày ở phần V.



Hình 7. Bộ n-bit Ripple-Carry Adder

**3. Thiết kế mẫu**

* Yêu cầu: Có 1 số nhị phân 8-bit: A = a7a6a5a4a3a2a1a0

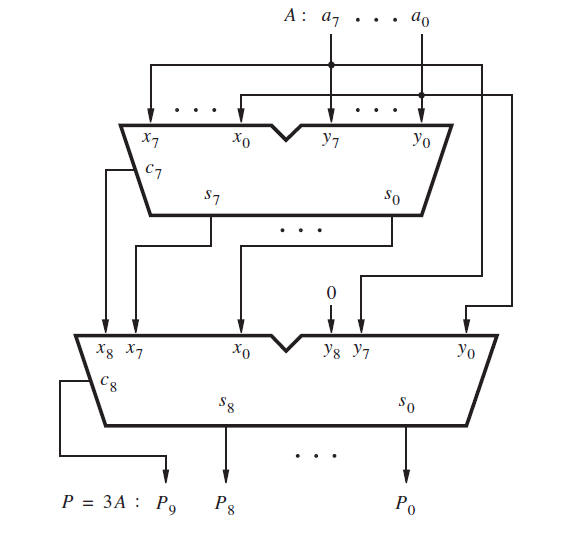
P = p9 p8 p7 p6 p5 p4 p3 p2 p1 p0 và P = 3A

Biểu diễn P thông qua A.

* Các bước:

- Bước 1: ta dùng bộ ripple-carry thực hiên phép cộng: A+A = 2A, lúc này 2A sẽ có 9bit ( 8 bit ra với bit carry c7).

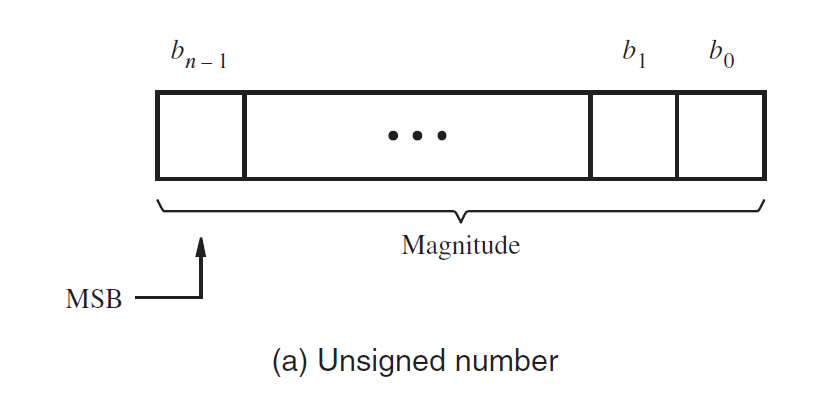
- Bước 2: ta dùng ripple-carry thực hiện phép cộng: 2A+A= 3A=P.Trong lúc thực hiện phép cộng, số A chỉ có 8 bit nên ta thêm số 0 vao bit thứ 9 để A thành 9bit. Lúc này P được biễu diễn bằng 10bit đúng yêu cầu.

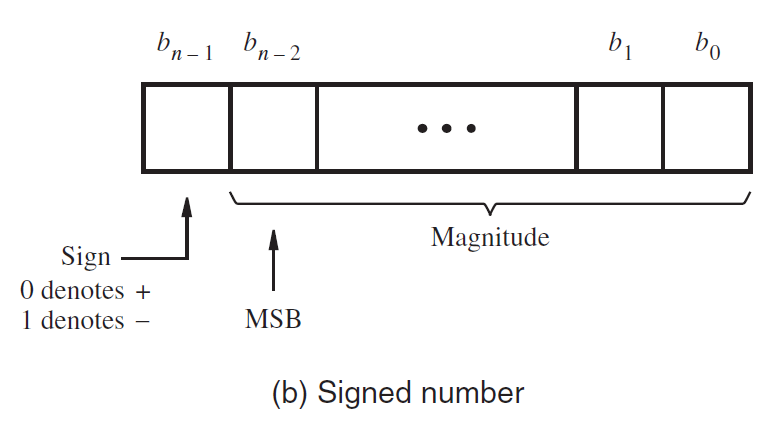


Hình 8. Hình minh họa

V. SỐ CÓ DẤU

- Khi biểu diễn số nhị phân, bit ngoài cùng bên trái (bit dấu) sẽ cho biết số đó là số âm hoặc dương, các bit còn lại đại diện cho độ lớn. Nếu bit dấu có giá trị là 0 thì số nhị phân đó là số dương, ngược lại nếu là 1 thì đó là số âm.





Hình 9. Biểu diễn số nguyên

1. **Số âm**

- Số âm có thể được biểu diễn bằng 3 cách: sign-and-magnitude,

1’s complement, and 2’s complement.

* **sign-and-magnitude**

- Cách biểu diễn này rất đơn giản. Chúng ta biểu diễn bit dấu trước các bit độ lớn. Ví dụ biểu diễn số nhị phân 4 bit như sau:

+ 5 = và - 5 =

* **1’s complement**

- Một số âm n-bit, K, được biểu diễn bằng cách: K = (2n-1) – P

Với P là số dương tương đương của K.

- Ví dụ: n = 4, thì K = (24 – 1) – P

= (15)10 – P

= (1111)2 – P

Giả sử muốn biểu diễn số -5, ta có:

-5 = 1111 – 0101 = 1010

Vậy -5 được biểu diễn là: 1010

* **2’s complement**

- Một số âm n-bit, K, được biểu diễn bằng cách: K = 2n – P

Với P là số dương tương đương của K.

- Ví dụ: n = 4, thì K = 24 – P

= (16)10 – P

= (10000)2 – P

Giả sử muốn biểu diễn -5, ta có:

-5 = 10000 – 0101 = 1011

Vậy -5 được biểu diễn là: 1011

- Ứng với trường hợp **1’s complement,** ta có: K1 = (2n-1) – P

- Ứng với trường hợp **2’s complement,** ta có: K2 = 2n – P

K2 = K1 + 1

Ta thấy để biểu diễn K2 bằng cách thêm 1 vào K1, ví dụ:

-5 với K­1 là: 1010 -5 với K2 là: K1 + 1 = 1010 + 1 = 1011

- Có một mẹo nhỏ để tìm ra K2 nhanh chóng bằng cách áp dụng quy luật sau:

Cho B = bn-1 bn-2 …b1b0, giả sử K2 của B là: K­2 = kn-1 kn-2 …k1k0

K2 được xác định bằng cách lấy tất cả các bit bằng 0 của B từ phải qua trái và bit đầu tiên bằng 1. Sau đó các bit còn lại lấy bù. Ví dụ:

B = 0110 k0 = b0 = 0; k1 = b1 = 1;

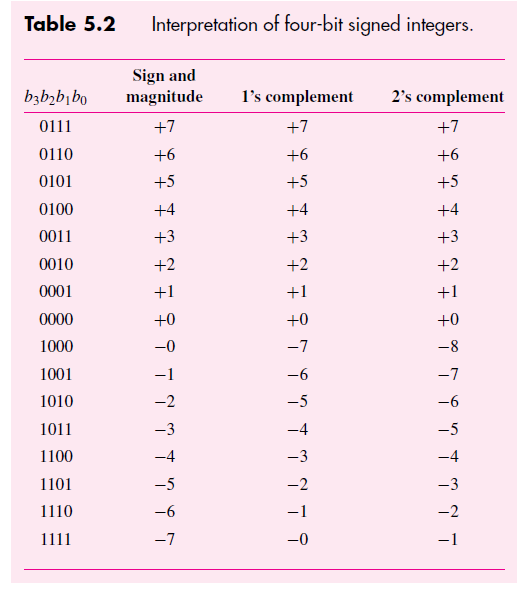
k2 = 2 = 0; k3 = 3 = 1.

K2  = 1010

- Số nhị phân n-bit B = bn-1 bn-2 …b1b0 có giá trị:



- Số âm lớn nhất, 100…00, có giá trị -2n-1. Số dương lớn nhất, 011…11, có giá trị 2n-1 - 1.



**2. Phép cộng và phép trừ**

- Trong các cách biểu diễn số nhị phân thì chúng ta có các phép cộng và trừ khác nhau.

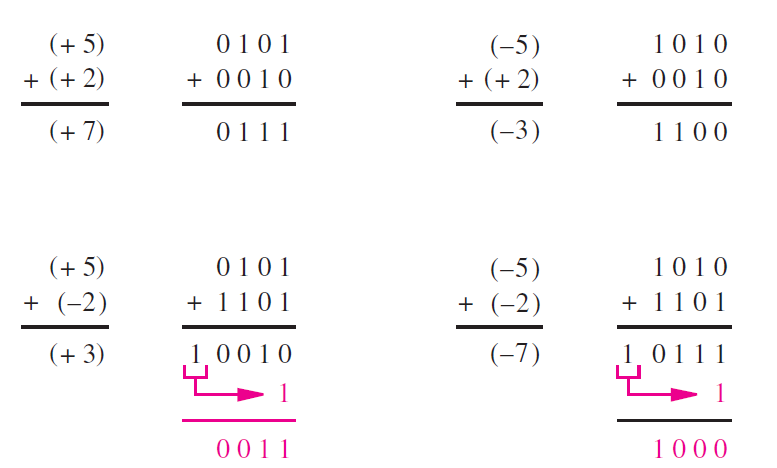
* **sign-and-magnitude Addition**

- Đối với cộng 2 số có cùng dấu, thì ta chỉ việc cộng 2 độ lớn của 2 số lại với nhau và sau đó chúng ta đặt bit dấu ở bên trái kết quả.

- Đối với cộng 2 số khác dấu, ta phải xác định xem số nào có giá trị lớn hơn rồi đem trừ cho số còn lại. Sau đó lấy bit dấu của số lớn đặt vào bên trái kết quả.

* **1’s Complement Addition**

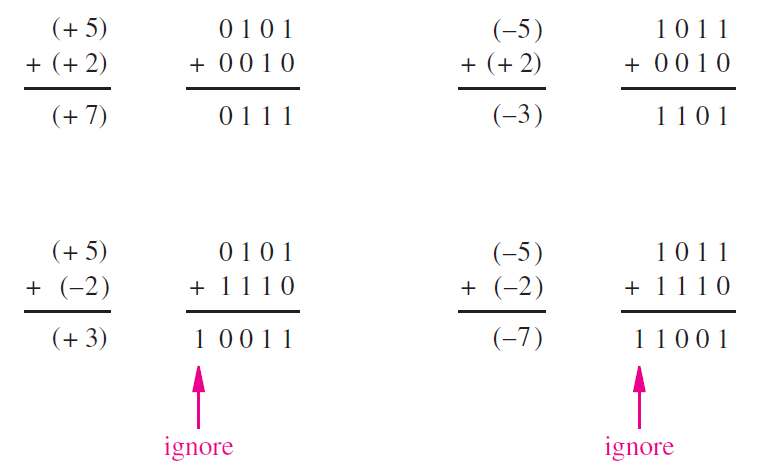
-Việc cộng tương tự như trường hợp **sign-and-magnitude Addition.** Trường hợp carry-out ở phép cộng ở vị trí bit cuối cùng bằng 1, thì ta lấy carry-out này cộng vào bit thấp nhất của kết quả (LSB).



Hình 10. Ví dụ các phép cộng của 1’s Complement

* **2’s Complement Addition**

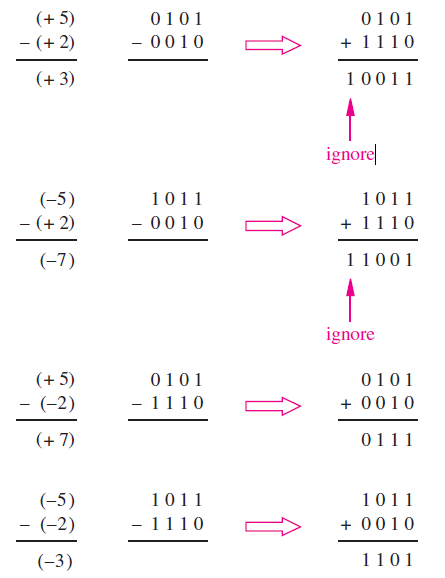
- Việc cộng tương tự như trường hợp **sign-and-magnitude Addition.** Đối với những trường hợp mà phép cộng ở vị trí bit cuối cùng có carry-out bằng 1, thì chúng ta sẽ bỏ qua giá trị này.



Hình 11. Ví dụ các phép cộng của 2’s Complement

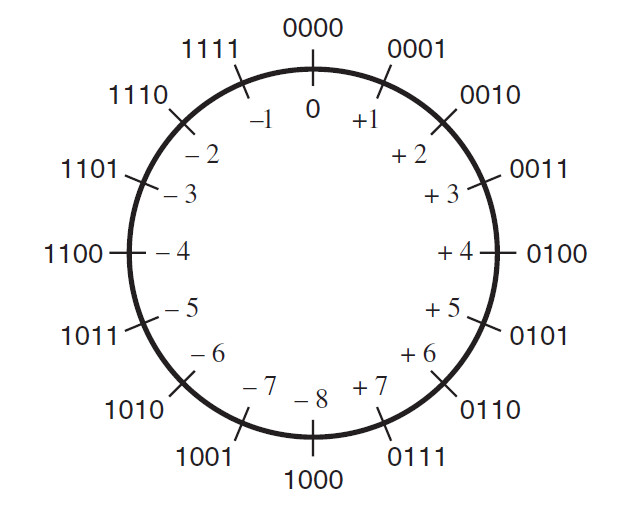
* **2’s Complement Subtraction**

**-** Để thực hiện phép trừ, chúng ta sẽ tìm số biểu diễn của **2’s Complement** của số bị trừ và lấy số trừ trừ đi cho số vừa tìm được.



Hình 12. Ví dụ các phép trừ của 2’s Complement

* Đối với tất cả các mô hình 4bit, chúng ta có thể đặt chúng trên một hình tròn modulo-16 như sau:



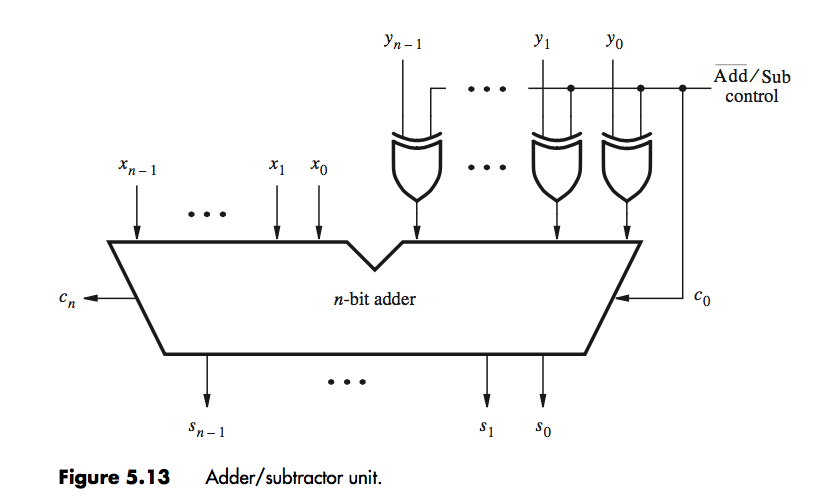
Hình 13. Mô hình 4bit của 2’s complement

- Đối với phép cộng, chúng ta sẽ đi theo hướng chiều kim đồng hồ. Ví dụ (-5) +2 được xác định bằng cách bắt đầu từ 1011(= -5) và di chuyển theo chiều kim đồng hồ 2 bước,chúng ta sẽ thu được kết quả 1101(= -3).

- Đối với phép trừ, chúng ta sẽ đi theo hướng ngược chiều kim đồng hồ. Ví dụ (-5) – (+2) được xác định bằng cách bắt đầu từ 1011(= -5) và di chuyển ngược chiều kim đồng hồ 2 bước, chúng ta sẽ thu được kết quả 1001(= -7).

**3. Các đơn vị cộng và trừ.**

- Sự khác nhau duy nhất của phép cộng và phép trừ là phép trừ cần thiết phải sử dụng dạng 2’s Complement của một toán hạng.Cho X và Y là hai toán hạng của phép trừ và Y là số bị trừ trong phép trừ. Trong phần 5.3.1 chúng ta đã biết rằng số dạng 2’s Complement có thể thu được bằng cách thêm 1 vào dạng 1’s Complement của số đó. Thêm 1 vào vị trí bit LSB có thể thực hiện bằng cách cho bit carry-in c0 lên 1. Dạng 1’s Complement có thể được thực hiện bằng cách bổ sung cho mỗi bit của nó. Điều đó có thể được thực hiện bằng cổng NOT. Nhưng chúng ta cần một mach linh hoạt hơn, nơi chúng ta có thể sử dụng giá trị của Y cho phép cộng hoặc các dạng complement của nó cho phép trừ.

 - Trong phần 5.2 chúng ta biết rằng cổng XOR 2 ngõ vào có thể được sử dụng để chọn giữa đảo hay không đảo của giá trị một ngõ vào, dưới sụ điều khiển của ngõ vào còn lại. Ý tưởng này có thể được sử dụng để thiết kế các đơn vị bộ cộng và trừ như sau. Giả sử rằng có một tín hiệu điều khiển để chọn phép cộng hay phép trừ được thực hiện. Tín hiệu đó gọi là Add/Sub. Giá trị là 0 cho phép cộng và 1 cho phép trừ. Bây giờ để mỗi bit của Y kết nối vào một đầu vào của cổng XOR. Các đầu ra của cổng XOR bằng Y nếu Add/Sub bằng 0 và bằng đảo của Y nếu Add/Sub bằng 1. Dẫn tới mach điện dưới đây:

- Sự kết hợp các đơn vị cộng và trừ là một ví dụ tốt về một khái niệm quan trọng trong việc thiết kế các mạch logic. Nó hữu ích cho việc thiết kê các mạch có được sự linh hoạt và khai thác hết phần chung của một mạch cho nhiều các tác vụ khác nhau. Phương pháp này làm giảm số lượng các cổng cần thiết và giảm sự phức tạp của các mạch điện đi đáng kể.

4. Radix-Complement Schemes.

- Ý tưởng để thực hiện phép trừ bằng cách sử dụng dạng 2’s Complement của số bị trừ không bị giới hạn bởi số nhị phân. Chúng ta nhìn lại về 2’s Complement trong hệ thống số thập phân. Hãy xem xét các phép trừ của số thập phân có hai chữ số. Tính toán kết quả của phép tính 74 – 33 rất dễ dàng bởi vì mỗi chữ số của số bị trừ nhỏ chữ sô tương ứng của số trừ. Nhưng tính toán 74 – 36 thì không dễ bởi vì phải vay mượn thêm chữ số của sô thấp nhất của số trừ, việc tính toán trở nên phức tạp hơn.

- Giả sử chúng ta tính toán như sau:

74 – 36 = 74 + 100 – 100 – 36 = 74 + (100 – 36) + 100.

Hai phép trừ cần thực hiện, để ý rằng 100 – 36 vẫn còn phải vay mượn. Cần lưu ý rằng 100 – 36 = 99 + 1 – 36, chúng ta sẽ tránh được sự vay mượn.

74 – 36 = 74 + (99 – 36) + 1 – 100.

- Các phép trừ trong ngoặc không cần sự vay mượn. Phép tính (99 – 36) tương tự như bổ sung số bị trừ để tìm dạng 1’s Complement của nó. Giống việc trừ từng bit đi 1. Sử dụng số thập phân, chúng ta tìm ra dạng 9’s Complement của số bị trừ bằng cách trừ từng chữ số của nó cho 9. Trong hình 5.13, chúng ta bổ sung bit carry-in thêm 1 để thành dạng 2’s Complement của Y. Trong ví dụ về số thập phân của chúng ta (99 – 36) + 1 = 64. Ở đây 64 là dạng 10’s Complement của số 36. Số thập phân có n chữ số, N, dạng 10’s Complement của nó, được định nghĩa là K10 = 10n – N. Dạng 9’s Complement được định nghĩa K9 = (10n – 1) – N.

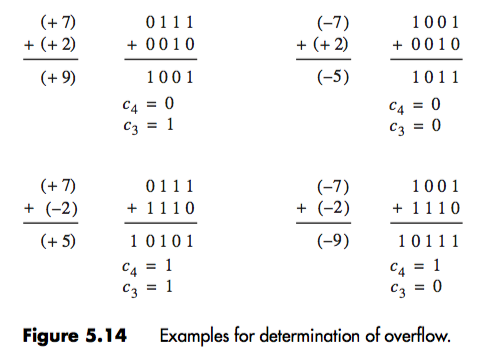
- Do đó, phép trừ (74 – 36) có thể được thực hiện bằng cách lấy 10’s Complement của 36, được như sau:

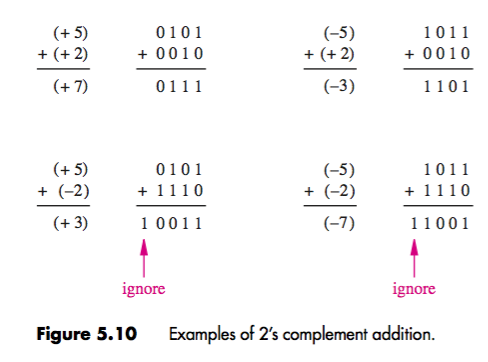
74 – 36 = 74 + 64 – 100 = 38

- Các phép trừ 138 – 100 là tầm thường vì nó có nghĩa là bỏ đi số đầu tiên trong 138. Điều này giống như việc bỏ qua các carry-out từ mạch điện trong hình 5.13.

**5. Sự tràn số.**

- Kết quả của việc công hoặc trừ phù hợp với các bit quan trọng được sử dụng để đại diện cho một con số. Nếu n bit được sử dụng để đại diện cho số có dấu, kết quả phải nằm trong phạm vi -2n-1 đến 2n-1 – 1. Nếu kết quả không nằm trong phạm vi này, số đã bị tràn. Để đảm bảo hoạt động đúng đắn của một mạch số học, điều quan trọng là phát hiện được khi nào việc tràn số xảy ra.

 - Hình 5.14 trình bày bốn trường hợp số 2’s Complement với độ lớn của 7 và 2 được thêm vào. Bởi vì chúng ta sử dụng số bốn bit, có 3 bit quan trọng, bit b2-0. Khi các số có dấu ngược lại, thì không có hiện tượng tràn số. Nhưng nếu cả hai số cùng dấu, thì độ lớn của kết quả là 9, không được đại diện bởi ba số có ý nghĩa, do đó, tràn số xảy ra.

 - Cách để phát hiện được khi nào xảy ra tràn số là bit carry-out từ vị trí MSB, gọi là c3 trong hình, và từ vị trí bit dấu, gọi là c4. Hình 5.14 cho thấy tràn số xảy ra khi những carry-out này có giá trị khác nhau. Và sẽ có giá trị chính xác khi hai carry-out này có cung giá trị. Thật vậy, điều đó đúng cho cả phép cộng và phép trừ trường hợp 2’s Complement. Hãy xem xét ví dụ sau, các con số đủ nhỏ để tràn số không xảy ra trong những trường hợp này.

- Trong hai ví dụ phía trên, có một carry-out của 0 từ cả hai dấu và vị trí MSB. Trong hai ví dụ dưới, có một carry-out của 1 từ cả hai vị trí. Vì thế, trong các ví dụ ở hai hình 5.14 và 5.10, sự tràn số được phát hiện bởi:

Overflow = *c*3*c*4 + *c*3*c*4 = *c*3 ⊕ *c*4

Và cho n bit, ta có: Overflow = *c*n - 1 ⊕ *c*n

- Vì thế, mach điện trong hình 5.13 có thể bao gồm thêm kiểm tra sự tràn số bằng cách bổ sung vào them một cổng XOR.