AULA 8 – ARITMÉTICA DIGITAL: OPERAÇÕES E CIRCUITOS

Os computadores e as calculadoras digitais realizam várias operações aritméticas sobre números representados no formato binário. O tema da aritmética digital pode ser muito complexo se desejarmos entender os diversos métodos de computação e a teoria que os envolve.
A abordagem que iremos fazer está concentrada nos princípios básicos necessários para entender como as máquinas digitais realizam as operações aritméticas básicas.
Primeiro estudaremos como as várias operações aritméticas são realizadas sobre números binários usando "lápis e papel" e, em seguida, estudaremos os circuitos lógicos atuais que realizam essas operações em um sistema digital.

ADIÇÃO BINÁRIA:

A adição de dois números binários é realizada da mesma forma que a adição de números decimais:

```
  3     7     6  → LSD ( dígito menos significativo )
+
  5     5     2
  9     2     8
```

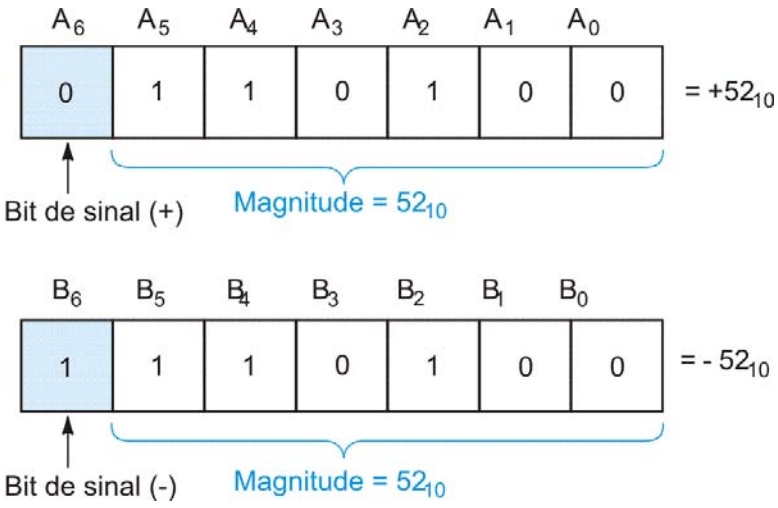
A soma sempre começa pelo dígito menos significativo e assim sucessivamente da esquerda para direita, sempre que a soma exceder 9 temos um "vai um" (carry), 6+2=8; 7+5=12 (resulta 2 e vai 1 para próxima casa mais significativa); 3+5=8 mais 1 herdado da casa anterior igual a 9.
Na soma binária podemos ter quatro situações:
0+0=0; 0+1=1; 1+1=10 (resultado é zero e vai 1); 1+1+1=11 (resultado é 1 e vai 1)

```
(1)   (1)
      1     1     1 ( 7 )
    +
      1     1     0 ( 6 )
1     1     0     1 ( 13 )
```

Sempre será efetuada a operação de adição de dois números por vez, quando mais de dois números forem somados, soma-se os dois primeiros números e o resultado é somado ao terceiro e assim por diante.
A adição é a operação mais importante nos sistemas digitais, iremos verificar isso nas próximas operações.

REPRESENTAÇÃO DOS NÚMEROS COM SINAL:

Podemos representar o sinal de um número binário reservando um bit para designar se ele é positivo ou negativo, em geral é utilizado o bit mais significativo para isso. Se este bit estiver em "0" o número é positivo, se estiver em "1" é negativo, conforme o exemplo abaixo podemos verificar o número convertido de binário para decimal +52 ou -52, este sistema é conhecido como sinal-magnitude:



Onde temos o sinal e depois a magnitude, ou seja o seu valor que no caso do nosso exemplo pode variar entre 000000 a 111111, em decimal seria de 0 a 63.
Pela complexidade na implementação deste sistema, é utilizado um outro sistema, chamado sistema de complemento que iremos discutir a seguir.

COMPLEMENTO DE 1:

O complemento de 1 é muito simples basta inverter o valor de cada bit, ou seja, substituindo o “0” por “1” os bits em “1” por “0”, vejamos o exemplo abaixo:

001111 → 110000

COMPLEMENTO DE 2:

O complemento de 2 é o resultado da soma do complemento de 1 de um número binário com 1, vejamos o exemplo abaixo:

101101	(número binário de 45)
010010	(complemento de 1)
+ 1	(adiciona-se 1)
010011	(complemento de 2 do número original)

Podemos ainda representar os sinais no sistema de complemento de 2 da seguinte forma:

Se o número é positivo representamos ele na forma original direta e a esquerda do bit mais significativo colocamos o bit de sinal “0” (se trata de número positivo), já se for negativo devemos representa-lo na forma do complemento de 2 e inserir a esquerda do bit mais significativo o bit de sinal “1” denotando o número negativo. Vaja abaixo outros exemplos:

a) +13 = 0 (bit de sinal) 1101 (Magnitude 13) => 01101
b) -9 = 1 (bit de sinal) 1001 → 0110 (complemento de 1) + 1 = 0111 (complemento de 2) => 10111
c) +3 = 0 (bit de sinal) 0011 (Magnitude 3) => 00011
d) -8 = 1 (bit de sinal) 1000 → 0111 + 1 = 1000 => 11000

NEGAÇÃO:

A operação de negação consiste em tornar um número positivo em negativo ou vice-versa, vejamos abaixo:
O número +9 em binário seria 0 (bit de sinal) 1001 (Magnitude 9) que resulta em 01001, fazendo a sua negação teremos -9 que em binário seria 1 (sinal negativo) 0111 (complemento de 2) que resulta em 10111 se negarmos novamente chegamos a origem +9, primeiro devemos trocar o bit de sinal de 1 para 0 e depois aplicar o complemento de 2 para a magnitude (0111):

0111 → 1000 + 1 = 1001 o que resultaria 01001 que é +9.

ADIÇÃO NO SISTEMA DE COMPLEMENTO DE 2:

Analisaremos agora como as operações de adição e subtração são realizadas em máquinas digitais que utilizam a representação do complemento de 2 para números negativos.

Na primeira situação temos a soma de dois números positivos, vejamos o exemplo abaixo:

+ 8 → 0 1000 (1ª parcela)
+ 3 → 0 0011 (2ª parcela)
+11→ 0 1011

Em outra situação podemos ter a soma de um número positivo por outro menor negativo:

+ 9 → 0 1001 (1ª parcela)
- 4 → 1 1100 (2ª parcela)
+5→ 10 0101

O carry gerado depois (à esquerda) do bit de sinal sempre será desprezado, logo o resultado é +5 (0 0101).

Temos também a soma de dois números onde o número negativo é maior que o número positivo:

- 9 → 1 0111 (1ª parcela)
+4 → 0 0100 (2ª parcela)
- 5 → 1 1011

Temos ainda a situação de adição de dois números negativos:

- 9 → 1 0111 (1ª parcela)
- 4 → 1 1100 (2ª parcela)
-13→11 0011

Novamente desprezamos o carry gerado depois do bit de sinal, como se trata de um sinal negativo temos o resultado na forma de complemento de 2 (0011) que resulta em 1101 que é 13.
Logo podemos perceber que a subtração no sistema de complemento de 2 é rigorosamente igual a uma soma de dois números de sinais diferentes, ou seja, teremos como resultado a diferença entre estes dois números (veja o exemplo +9 + -4).

ESTOURO (Overflow) NO RESULTADO:

Em todos os nossos exemplos estamos trabalhando com 1 bit de sinal e 4 bits de magnitude, logo existe um limite para a representação, nas situações anteriores este limite não foi excedido, entretanto se tivermos uma situação como a que segue abaixo podemos enfrentar problemas:

+ 9 → 0 1001 (1ª parcela)
+ 8 → 0 1000 (2ª parcela)
+17→ 1 0001

Podemos verificar claramente que nem o sinal, nem a magnitude estão corretos, isto se deve a magnitude resultante ser 17 e necessitar de 5 bits para sua representação. Outra condição indesejada ocorreria se subtraíssemos -8 de +9, ou seja, +9-(-8)= +17, também seria necessário uma quantidade de bits maior para representação dessa magnitude. Todo computador possui um circuito de overflow para detectar esta condição.

MULTIPLICAÇÃO DE NÚMEROS BINÁRIOS:

Esta operação é realizada rigorosamente da mesma forma que em número decimais, ainda mais fácil porque o produto será sempre por zero ou um, vejamos o exemplo abaixo:

1001 → Multiplicando = $9_{(10)}$
x1011 → Multiplicador = $11_{(10)}$
1001
1001+
0000++
1001+++
1100011 → Produto Final = $99_{(10)}$

Na multiplicação no sistema de complemento de dois números se ambos forem positivos estão na forma binária direta e portanto é idêntico a forma descrita anteriormente, se ambos forem negativos deverão estar na forma do complemento de 2, aplicamos o complemento de 2 novamente para atingir a forma binária e o sistema é o mesmo, já quando tivermos um número positivo e outro negativo, aplicamos o complemento de 2 no número negativo, efetuamos o produto e o resultado por ser negativo nos obriga a aplicar o complemento de 2 no produto final.

ARITMÉTICA HEXADECIMAL:

Os números são representados em hexadecimal em programação Assembly (programação de baixo nível, linguagem de máquina). O processo de soma é o mesmo que em decimal só que agora o maior dígito é F ao invés de 9, então devemos sempre efetuar a soma de cada dígito, sendo que de A a F devemos considerar mentalmente de 10 a 15, após efetuada a soma devemos verificar, se o resultado é menor ou igual a 15, basta anotar o resultado, se for maior ou igual a 16 devemos subtrair 16 transportar um "carry" para a casa imediatamente mais significativa e o resultado da subtração anotar:

$58_{(16)}$	$58_{(16)}$	$3AF_{(16)}$
$+24_{(16)}$	$+4B_{(16)}$	$+23C_{(16)}$
$7C_{(16)}$	$A3_{(16)}$	$5EB_{(16)}$

No primeiro exemplo é direto 8+4=12 (C em hexa) e 5+2=7, já no segundo exemplo temos "carry" 8+B=19 (uma vez que B=11), então 19 é maior que 16, logo devemos anotar a diferença e "levar 1", daí a próxima soma fica 5+4+1=10 que em hexa é igual a A. No último exemplo temos F+C que é igual a 15+12=27 que é maior que 16, anotamos a diferença 11 (que em hexa é B) na próxima soma temos A+3+1=14 (que em hexa é E) por último a soma simples de 3+2=5, resultado final $5EB_{(16)}$.

A subtração em hexadecimal também é realizada em complemento de 2, até porque a forma hexadecimal é forma distinta de representação dos próprios binários. Então basta obter o complemento de 2 de um hexadecimal e depois soma-lo a outro número, existem duas formas para fazermos isto:
A primeira delas mais intuitiva, podemos transformar os números hexadecimais em binários e depois aplicarmos o complemento de 2 ou ainda subtrair cada dígito de F e depois somar 1, esta forma é mais rápido.

$73A_{(16)}$ → $011100111010_{(2)}$ →Aplicando complemento de 1 → 100011000101 + 1 = $100011000110_{(2)}$ → $8C6_{(16)}$, portanto o complemento de 2 de $73A_{(16)}$ é $8C6_{(16)}$.

$FFF_{(16)} - 73A_{(16)} = 8C5_{(16)} + 1 = 8C6_{(16)}$.

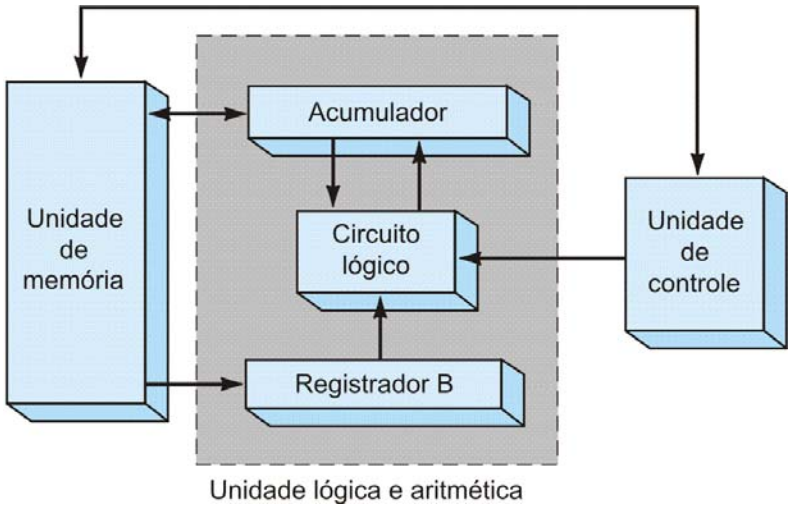
CIRCUITOS ARITMÉTICOS:

Nos circuitos lógicos combinacionais podemos projetar circuitos aritméticos que estão presentes na ULA (Unidade Lógica e Aritmética) de qualquer processador dentro de um computador ou de uma calculadora.

Muito embora estes circuitos comercialmente estão prontos em circuitos integrados, iremos exercitar a construção de um somador e subtrator para fins didáticos.

UNIDADE LÓGICA E ARITMÉTICA:

Todas as operações lógicas e aritméticas são realizadas na ULA, abaixo vemos o diagrama em blocos da ULA, onde temos pelo menos dois registradores, um acumulador e outro registrador B onde são realizadas as operações.



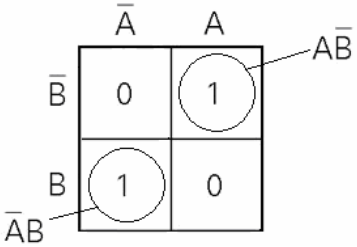
A unidade de controle recebe as instruções provenientes da unidade de memória especificando o número armazenado em uma determinada posição (endereço) da memória que será somado ao número armazenado no registrador acumulador, o número a ser somado é transferido da memória para o registrador B, os números do registrador acumulador e o registrador B são somados no circuito lógico, o resultado da soma é então enviado ao acumulador para ser armazenado, o novo número no acumulador pode ser mantido nele de forma que um outro número pode ser somado a ele, se o processo aritmético foi finalizado o número é armazenado na memória.

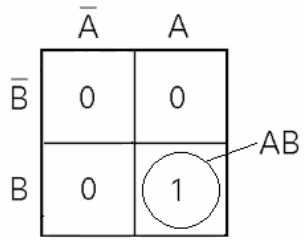
O nome do registrador acumulador mostra justamente a função dele que é a de acumular o resultado intermediário até chegar ao final.

MEIO SOMADOR:

Conhecendo as possibilidades da soma de 2 números binários de um algarismo, vamos montar a tabela verdade:

A	B	S	Cy
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

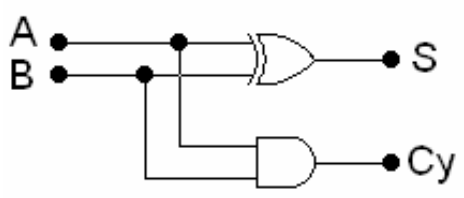




$$S = A\bar{B} + \bar{A}B$$
$$S = A \oplus B$$

$$Cy = AB$$

O meio somador possibilita a soma de números binários apenas de 1 algarismo, se desejarmos efetuar a soma de números binários de mais algarismos necessitamos de um somador completo.

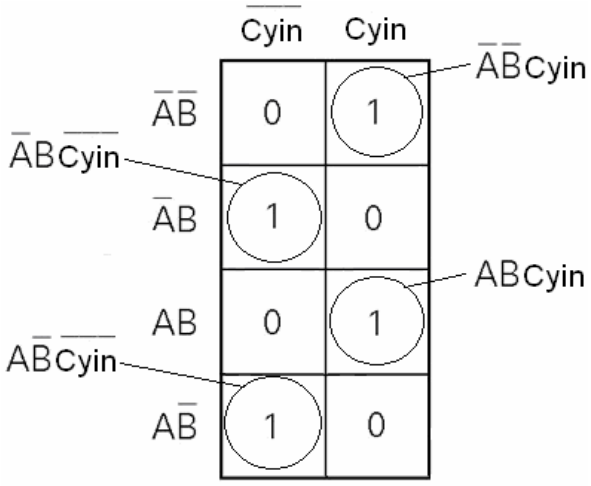


SOMADOR COMPLETO:

Este circuito prevê o envio do “Carry” (estouro) para a coluna mais significativa, abaixo segue a tabela verdade de um somador completo:

A	B	Cyin	S	Cyout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Resolvendo S:



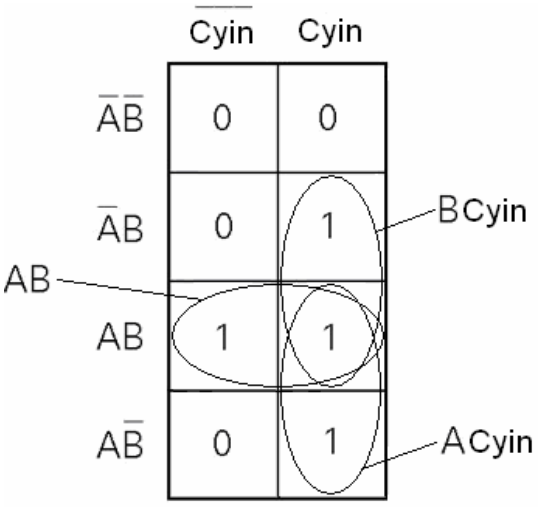
$$S = \overline{A}\overline{B}Cyin + \overline{A}B\overline{Cyin} + ABCyin + A\overline{B}\overline{Cyin}$$

$$S = Cyin(\overline{A}\overline{B} + AB) + \overline{Cyin}(\overline{A}B + A\overline{B})$$

$$S = Cyin(\overline{A \oplus B}) + \overline{Cyin}(A \oplus B)$$

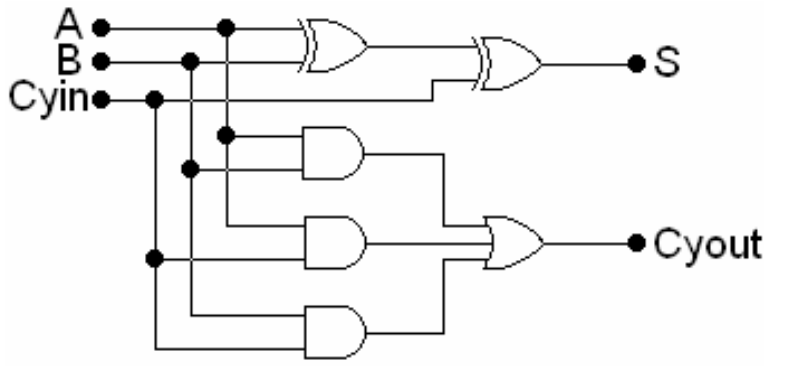
$$S = A \oplus B \oplus Cyin$$

Resolvendo Cyout:

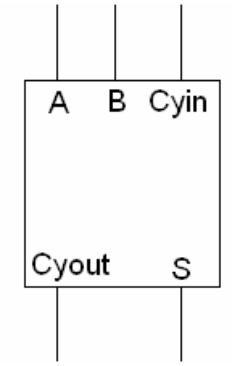


Cyout = ACyin+BCyin+AB

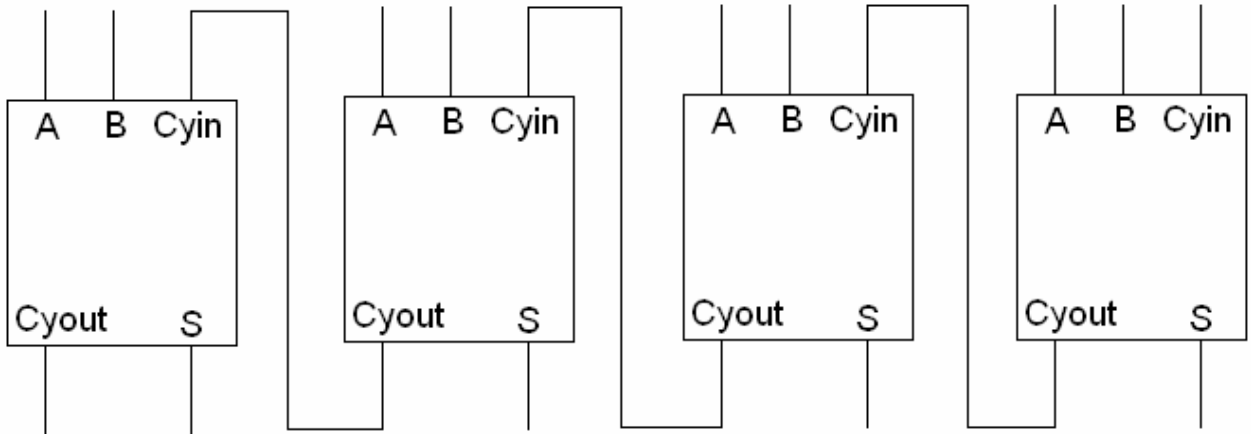
SOMADOR COMPLETO:



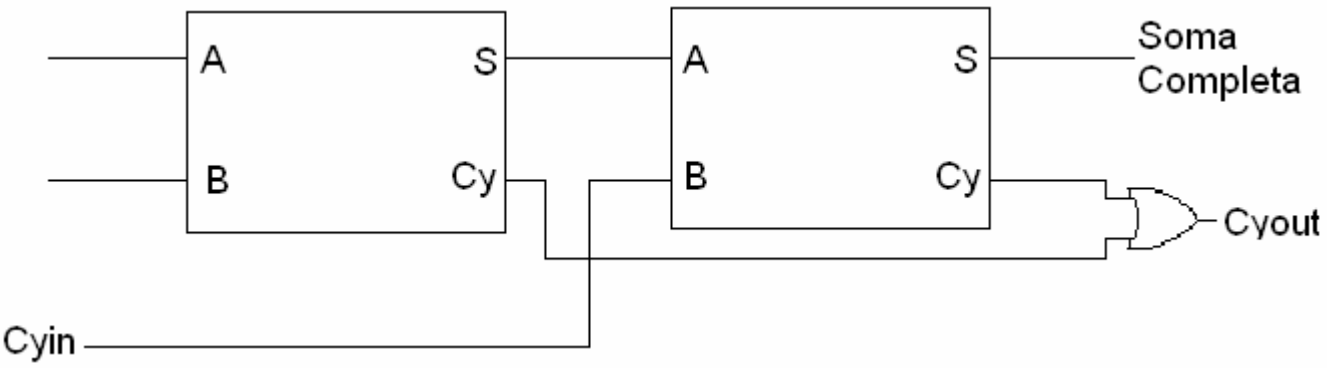
Podemos representar o circuito acima genericamente como a figura abaixo:



Portanto se quisermos somar dois números binários de 4 algarismos cada teremos 4 circuitos de somador completo:



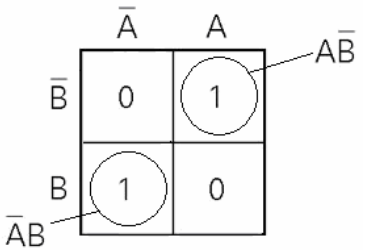
Podemos ainda conectar 2 meio-somadores para fazer um somador completo como é mostrado abaixo:



MEIO SUBTRATOR:

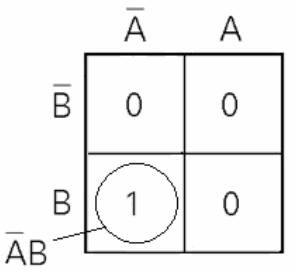
0 – 0 = 0
0 – 1 = 1 (empresta 1)
1 – 0 = 1
1 – 1 = 0

A	B	S	Cy
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



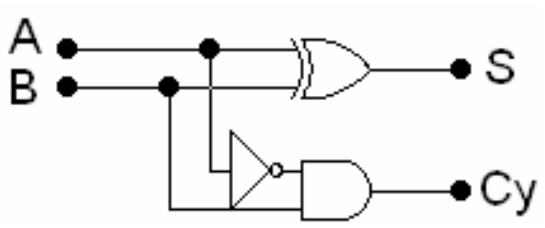
$S = \overline{A}B + A\overline{B}$

$S = A \oplus B$



$Cy = \overline{A}B$

CIRCUITO MEIO SUBTRATOR:

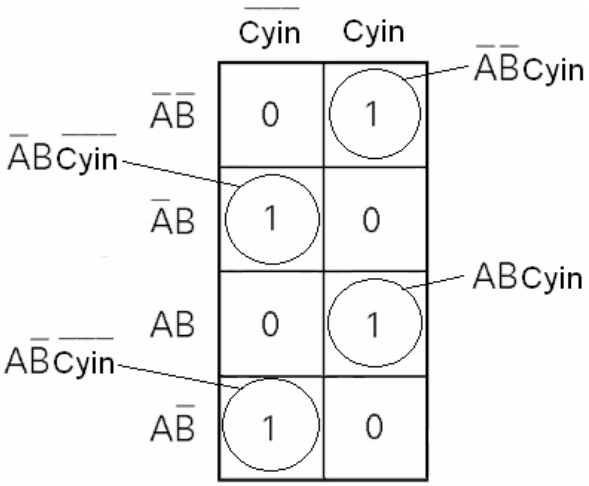


SUBTRATOR COMPLETO:

Assim como no somador completo, o subtrator completo é capaz de tratar Cyin e Cyout, vejamos a tabela verdade:

A	B	Cyin	S	Cyout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Resolvendo S:



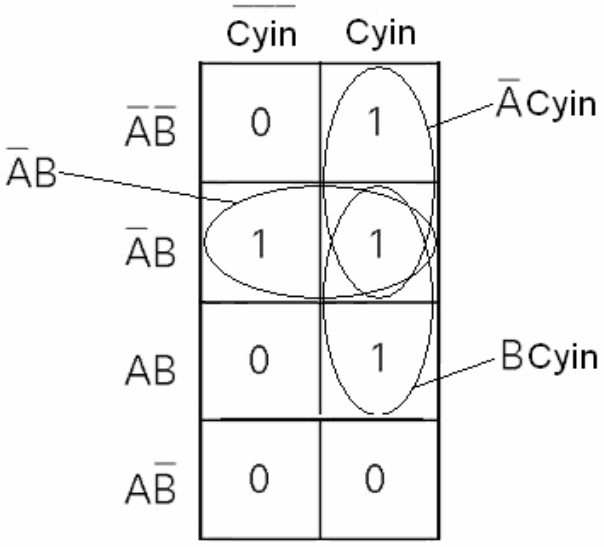
$S = \overline{A}\overline{B}Cyin + \overline{A}B\overline{Cyin} + ABCyin + A\overline{B}\overline{Cyin}$

$S = Cyin(\overline{A}\overline{B} + AB) + \overline{Cyin}(\overline{A}B + A\overline{B})$

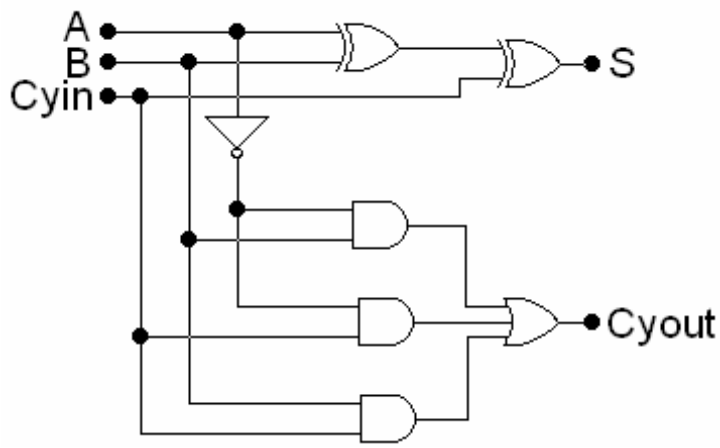
$S = Cyin(\overline{A \oplus B}) + \overline{Cyin}(A \oplus B)$

$S = A \oplus B \oplus Cyin$

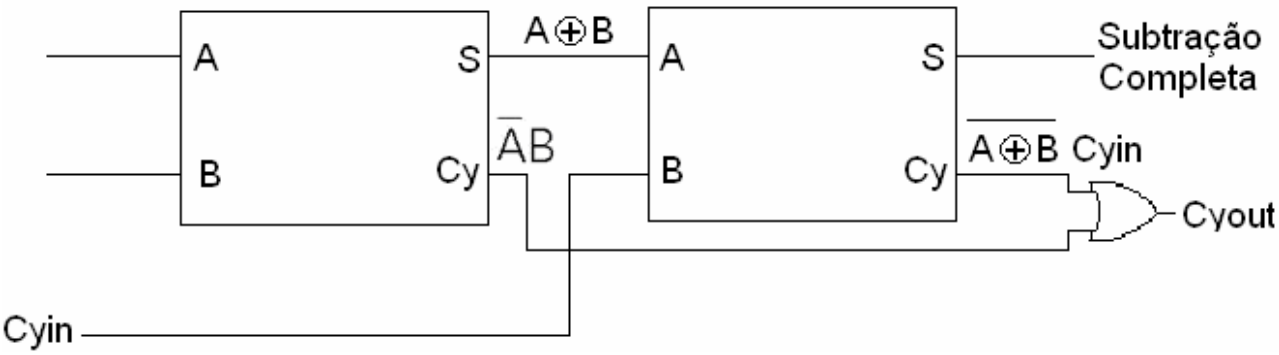
Resolvendo o Carry Out:



CIRCUITO_SUBTRATOR_COMPLETO:



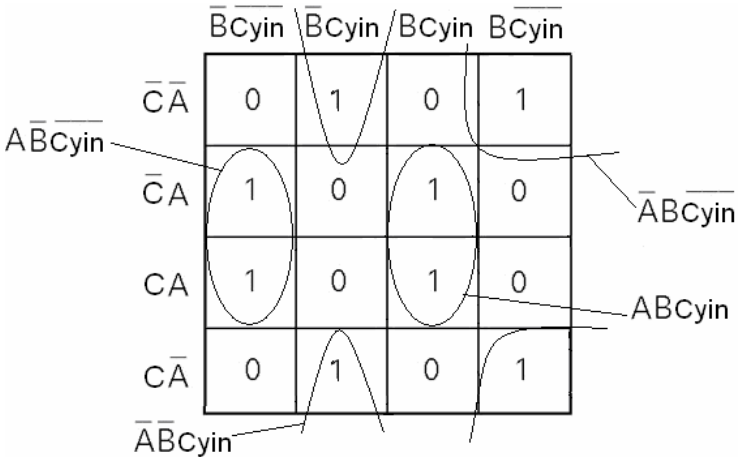
Da mesma forma podemos associar vários subtratores completos para fazer a subtração entre dois números binários de vários algarismos, necessitamos de um subtrator completo para cada algarismo.
Também podemos ter um subtrator completo a partir de dois meio subtratores:



SOMADOR/SUBTRATOR COMPLETO:

Com um sinal de controle podemos ter no mesmo circuito um somador completo e um subtrator completo, dependendo do nível do sinal de controle.
Chamaremos o sinal de controle de C, quando C = 0 o circuito efetua soma completa, quando C = 1 a subtração completa será realizada, vejamos a tabela verdade:

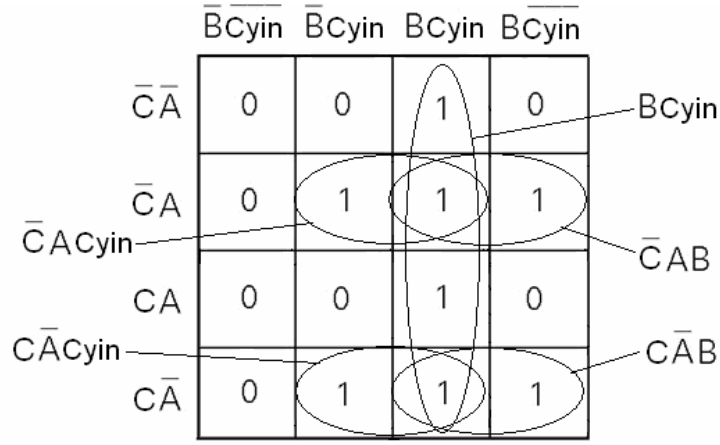
C	A	B	Cyin	S	Cyout
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	0	1
0	1	0	0	1	0
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	1	1	1
1	0	1	0	1	1
1	0	1	1	0	1
1	1	0	0	1	0
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	1	1



RESOLVENDO S:

$$S = \overline{A}\overline{B}Cyin + \overline{A}B\overline{Cyin} + ABCyin + A\overline{B}\overline{Cyin}$$
$$S = Cyin(\overline{A}\overline{B} + AB) + \overline{Cyin}(\overline{A}B + A\overline{B})$$
$$S = Cyin(\overline{A \oplus B}) + \overline{Cyin}(A \oplus B)$$
$$S = A \oplus B \oplus Cyin$$

RESOLVENDO CARRY OUT:



REESCREVENDO A EQUAÇÃO:

$$Cyout = BCyin + \overline{C}AB + \overline{C}ACyin + C\overline{A}B + C\overline{A}Cyin$$
$$Cyout = BCyin + B(\overline{C}A + C\overline{A}) + Cyin(\overline{C}A + C\overline{A})$$
$$Cyout = BCyin + B(C \oplus A) + Cyin(C \oplus A)$$
$$Cyout = BCyin + (C \oplus A)(B + Cyin)$$

O circuito para o somador/subtrator fica: