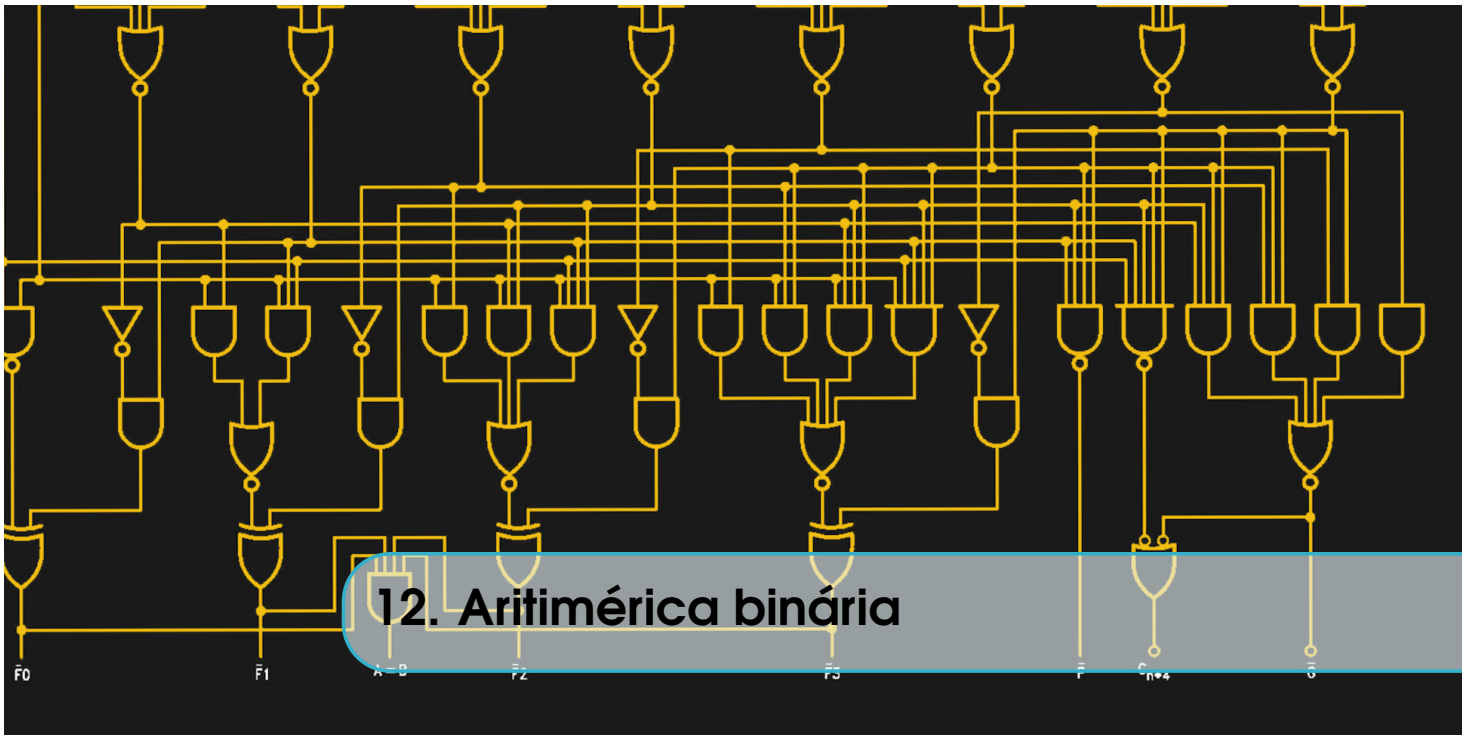

12. Aritimérica binária

12.1 Somadores elementares

12.1.1 Meio Somador

Tabela 12.1: Tabela verdade do meio somador

x	y	p	c_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$p = x \oplus y \tag{12.1}$$

$$c_{out} = x \cdot y \tag{12.2}$$

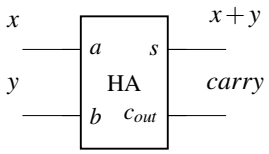

Figura 12.1: Diagrama em blocos do meio somador

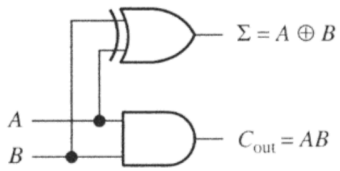

Figura 12.2: Implementação do meio somador com portas lógicas. (reprodução de: https://ufsj.edu.br/portal2-repositorio/File/nepomuceno/ca/06b-ED_C6.pdf)

12.1.2 Somado Completo

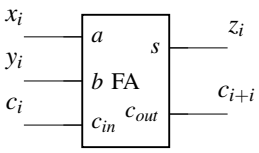

Figura 12.3: Diagrama em blocos do somador completo.

Tabela 12.2: Tabela verdade do somador completo

x_i	y_i	c_i	z	c_{i+1}
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

$$z_i = x_i \oplus y_i \oplus c_i \tag{12.3}$$

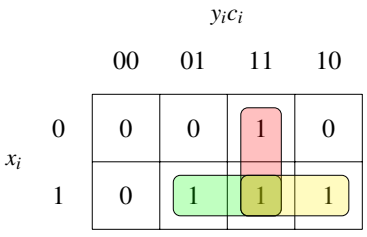

Figura 12.4: Mapa-K para a saída de $c_{out} = c_{i+1}$ do somador completo

A expressão tirada do mapa para c_{i+1} é:

$$c_{i+1} = x_i \cdot c_i + x_i \cdot y_i + y_i \cdot c_i \tag{12.4}$$

No entanto, olhando a tabela verdade, se ambos os operandos x_i e y_i forem iguais a 1 ou a soma deles for igual a 1 e c_i igual a 1 o resultado do c_{out} também é 1. Escrevendo algebricamente a frase anterior vem:

$$c_{i+1} = x_i \cdot y_i + (x_i \oplus y_i) \cdot c_i \tag{12.5}$$

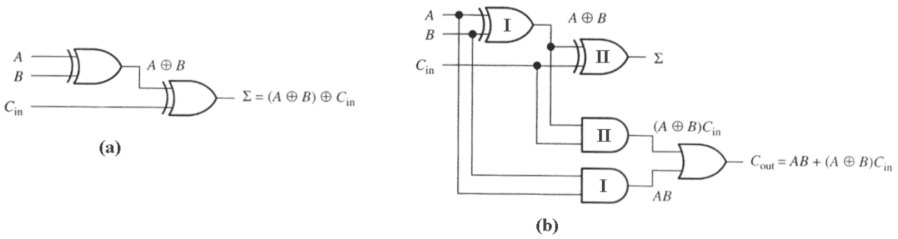

Figura 12.5: Implementação do somador completo com portas lógicas. (reprodução de: https://ufsj.edu.br/portal2-repositorio/File/nepomuceno/ca/06b-ED_C6.pdf)

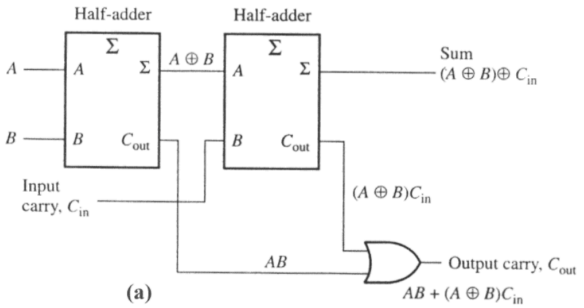

Figura 12.6: Implementação do somador completo usando meios somadores. (reprodução de: https://ufsj.edu.br/portal2-repositorio/File/nepomuceno/ca/06b-ED_C6.pdf)

12.1.3 somador de transporte propagado

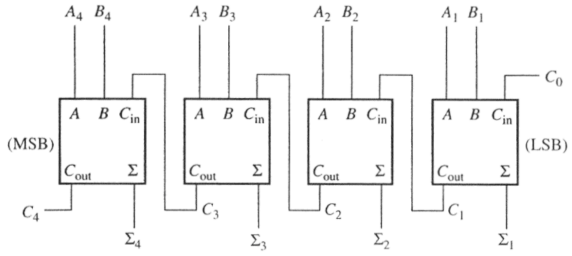

Figura 12.7: Implementação do somador completo usando meios somadores. (reprodução de: https://ufsj.edu.br/portal2-repositorio/File/nepomuceno/ca/06b-ED_C6.pdf)

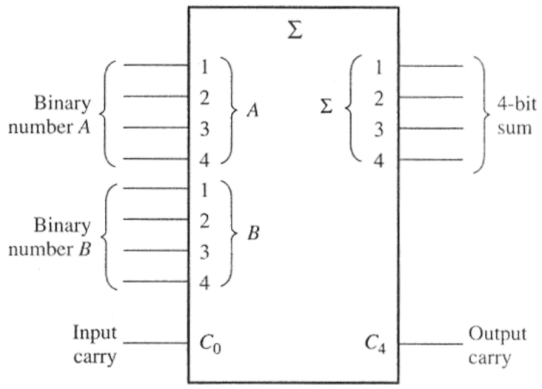

Figura 12.8: Implementação do somador completo usando meios somadores. (reprodução de: https://ufsj.edu.br/portal2-repositorio/File/nepomuceno/ca/06b-ED_C6.pdf)

12.2 Soma/subtração de inteiros com sinais

Exemplo na base 10 Como fazer a operação 20 - 6 apenas com somas?

Sabe-se que:

$$20 \mod 100 = 20 \tag{12.6}$$

e

$$-6 \mod 100 = 94 \tag{12.7}$$

então,

$$20 - 6 \mod 100 = 20 + 94 \mod 100 = 14. \tag{12.8}$$

Tabela 12.3: Representação de inteiros positivos e negativos na notação complementar na base 10.

Número	Representa
50	0/-50
51	-49
⋮	⋮
97	-3
98	-2
99	-1
00	0
01	1
02	2
03	3
	⋮
48	48
49	49

12.2.1 Sistema binário sinal magnitude

Um número inteiro de n bits é representado por:

$$x_{n-1}x_{n-2}\ldots x_1x_1x_0, \tag{12.9}$$

em que o bit x_{n-1} representa o sinal e o número $x_{n-2}\ldots x_1x_1x_0$ representa a magnitude da quantidade que se quer representar. Normalmente o sinal é tal que:

$$x_{n-1} = \begin{cases} 0, & \text{representa um número positivo} \\ 1, & \text{representa um número negativo} \end{cases} \tag{12.10}$$

12.2.2 Sistema complemento a dois

O sinal é o número são reapresentados em um único bloco.

- Mapeamento 1

$$x_r = x \mod 2^n \tag{12.11}$$

- Mapeamento 2

$$x_r = \sum_{i=0}^{n-1} x_i 2^i \tag{12.12}$$

$$x_r = \begin{cases} x, & \text{se } x \geq 0 \\ 2^n - |x|, & \text{se } x < 0 \end{cases} \tag{12.13}$$

no caso de $x < 0$ vem:

$$\begin{aligned} x_r &= 2^n - |x| \\ &= 2^n - (-x) \\ &= 2^n + x \end{aligned} \tag{12.14}$$

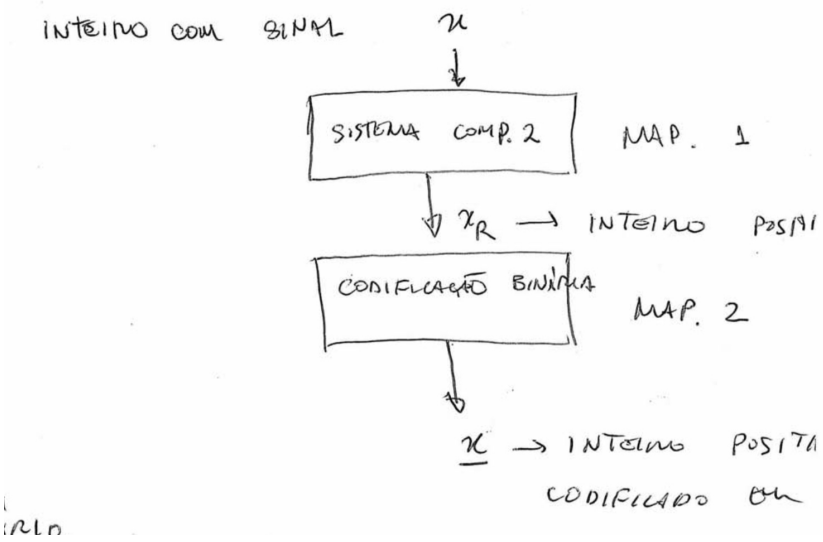

Figura 12.9: Mapeamento no sistema complementar

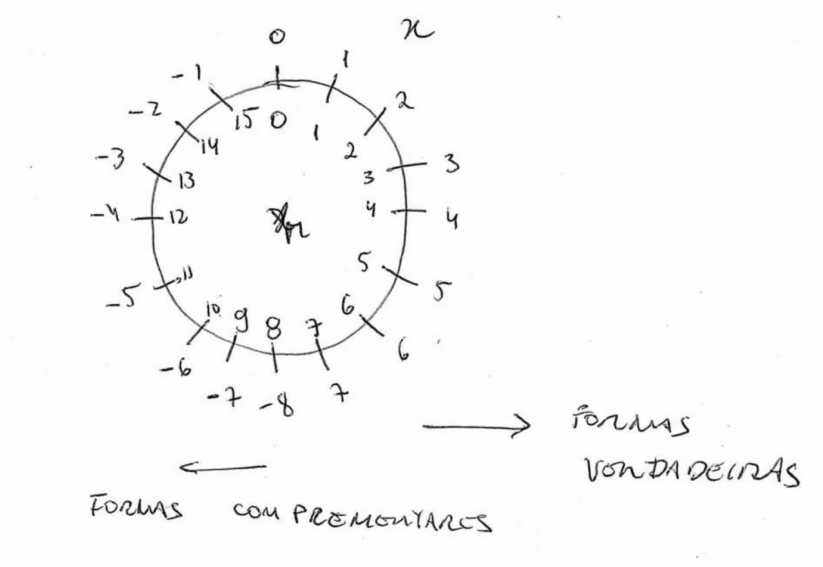

Figura 12.10: Mapeamento no sistema complementar com 4 bits.

- Formas Verdadeiras - Nenhuma transformação entre x e x_r, corresponde aos inteiros positivos.
- Formas Complementares - A representação é obtida a partir da função mod, corresponde aos inteiros negativos.

■ **Exemplo 12.1** Sistema complementar de 3 bits. ■

Como obter x a partir de x_r?

Tabela 12.4: Representação de inteiros positivos e negativos na notação complementar na base 2.

x	x_r	$\underline{\mathrm{x}}$
0	0	$00\ldots00$
1	1	$00\ldots01$
2	2	$00\ldots10$
$\vdots$	$\vdots$	$\vdots$
$2^{n-1}-1$	$2^{n-1}-1$	$01\ldots11$
-2^{n-1}	2^{n-1}	$10\ldots00$
$-(2^{n-1}-1)$	$2^{n-1}+1$	$10\ldots01$
$\vdots$	$\vdots$	$\vdots$
-2	2^n-2	$11\ldots10$
-1	2^n-1	$11\ldots11$

Tabela 12.5: Exemplo para o sistema complementar de 3 bits.

x	x_r	$\underline{x}$
-4	4	100
-3	5	101
-2	6	110
-1	7	111
0	0	000
1	1	001
2	2	010
3	3	011

Como visto:

$$x_r = \sum_{i=0}^{n-1} x_i 2^i \tag{12.15}$$

Separando o bit mais significativo vem:

$$x_r = x_{n-1} + \sum_{i=0}^{n-2} x_i 2^i \tag{12.16}$$

Há dois casos possíveis: $x = x_r$ e $x \neq x_r$.

- Caso 1 $x = x_r$

$$x = x_r = 0 \cdot 2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i \tag{12.17}$$

- Caso 2 $x \neq x_r$

Aqui $x = x_r - 2^n$ como $x_r \geq 2^{n-1}$ o bit $x_{n-1} = 1$. Logo:

$$\begin{aligned} x &= x_r - 2^n \\ &= \left(1 \cdot 2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i\right) - 2^n \\ &= \left(1x2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i\right) - 2 \cdot 2^{n-1} \\ &= 1 \cdot 2^{n-1} - 2 \cdot 2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i \\ &= -1 \cdot 2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i \end{aligned} \tag{12.18}$$

Combinando ambos os casos é sempre possível escrever:

$$x = -x_{n-1} \cdot 2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i \tag{12.19}$$

12.2.3 complemento a dois e módulo de mudança de sinal

A soma pode ser feita implementando um bloco somador da seguinte forma:

$$\underline{z} = \text{ADD}(\underline{y}, \underline{x}) \tag{12.20}$$

Esse bloco faz a soma entre os vetores binários $\underline{x}$ e $\underline{y}$. Para fazer a subtração é necessário implementar um bloco de mudanças sinal $CS()$ e fazer:

$$\underline{z} = \text{ADD}(\underline{y}, \text{CS}(\underline{x})) \tag{12.21}$$

Como implementar esse bloco $CS()$?

Como encontrar a representação de $-x$ na notação complementar? Pela definição da função mod se $x > 0$ para se encontrar o equivalente complementar negativo de x com n bits se faz $2^n - x$:

$$\begin{array}{ccccc} 1 & 0 & 0 & \ldots & 0 \\ - & x_{n-1} & x_{n-2} & \ldots & x_0 \end{array}$$

Estratégia é subtrair o número $100\ldots00$ de uma unidade e depois somar 1:

$$\begin{array}{ccccc} & 1 & 1 & \ldots & 1 \\ - & x_{n-1} & x_{n-2} & \ldots & x_0 \end{array}$$

Resolvendo:

$$\begin{array}{cccc} & 1 & 1 & \ldots & 1 \\ - & x_{n-1} & x_{n-2} & \ldots & x_0 \\ \hline - & \overline{x_{n-1}} & \overline{x_{n-2}} & \ldots & \overline{x_0} \\ + & 0 & 0 & \ldots & 1 \\ \hline \end{array}$$

12.2.4 Overflow

Condições para ocorrência do overflow:

- Operando de sinais opostos - Nunca ocorre overflow
- Operando de sinais iguais - Ocorre overflow se o resultado da operação for de sinal diferente do sinal dos operandos.

A condição de overflow para a soma $\underline{z} = \underline{x} + \underline{y}$ em que os vetores binários são vetores de *n bits* é:

$$Ovr = \overline{x_{n-1}} \cdot \overline{y_{n-1}} \cdot z_{n-1} + x_{n-1} \cdot y_{n-1} \cdot \overline{z_{n-1}} \tag{12.22}$$