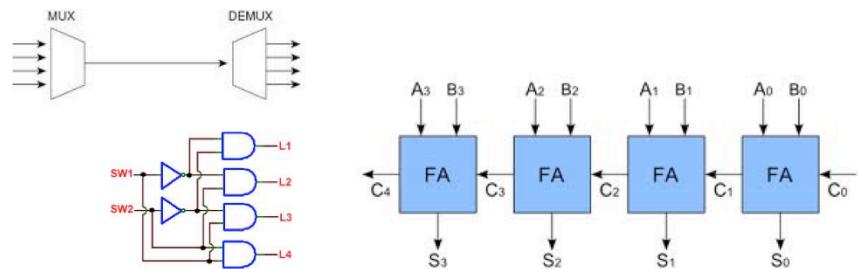




Introdução à Microeletrônica

Sistemas Digitais

Aula 3





Plano e bibliografia

Parte 3:

Dia	Segunda 08:20h	Segunda 10:10h
5 Outubro	Aula 1: Introdução, portas lógicas em CMOS	Aula 2: Álgebra de Boole
19 Outubro	Aula 3: Circuitos combinatórios	Intro VHDL e apresentação projeto
26 Outubro	Aula 4: Latches, Flip-Flops, registradores, contadores.	Aula 5: Máquinas de estado
9 Novembro	Laboratório 2 VHDL	Laboratório 3 VHDL
16 Novembro	Entrega projeto VHDL e revisão problemas	Prova Final

■ **Plano de aula 3:**

- ▶ Introdução aos circuitos combinatórios.
- ▶ Tempos de propagação e cálculo de atraso máximo em circuitos combinatórios.
- ▶ Descodificadores e multiplexadores.
- ▶ Somadores e subtraidores.
- ▶ Multiplicadores e divisores.

Nesta aula veremos o assunto relacionado à circuitos combinatórios.

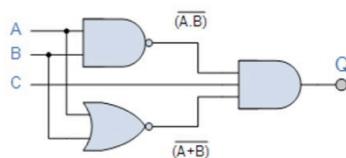


Noção de Circuito Combinatório

■ Circuito Combinatório:

- ▶ A saída é uma função que depende apenas da entrada actual;
- ▶ Definido através de:

- **Função Booleana** – Ex: $Q = (\bar{A} \cdot B) \cdot (\bar{A} + B) \cdot C$
- **Diagrama lógico**
- **Tabela de verdade**



C	B	A	Q
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Os circuitos combinacionais possuem o nível lógico na saída dependente apenas da combinação dos níveis lógicos presentes nas entradas. Um circuito combinacional não tem característica de memória, então sua saída depende apenas do valor atual de suas entradas.

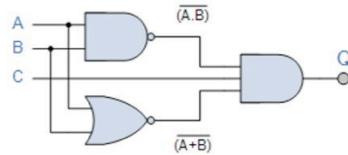
Este tipo de circuito pode ser definido por uma função booleana, diagrama lógico ou tabela da verdade.



Noção de Circuito Combinatório

■ Circuito Combinatório:

- A saída é uma função que depende apenas da entrada actual;



- Definido em contraste com a noção de **circuito sequencial**, em que a saída depende não só da entrada actual, mas também do valores anteriores dessa entrada...
i.e., circuitos sequenciais têm “efeito de memória”, enquanto que um circuito combinatório não.

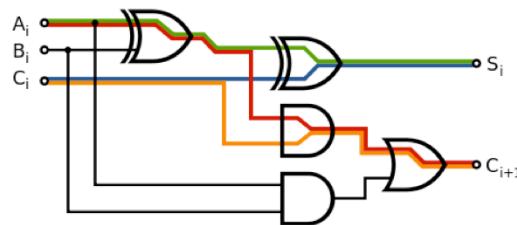
Os circuitos sequenciais podem ser associados à ‘sistemas de realimentação’



Noção de Circuito Combinatório

■ Circuito Combinatório:

- ▶ Até ao momento, tem-se assumido um modelo **ideal** dos circuitos lógicos, em que a saída muda *instantaneamente* face aos valores na entrada do circuito.
- ▶ Na realidade, todos os circuitos caracterizam-se por um certo **tempo de propagação**, entre as entradas e as saídas, e que depende no número e complexidade de portas lógicas envolvidas:



Nos Cis, esse tempo de propagação depende da quantidade de transistores fazendo chaveamento até que se tenha uma saída



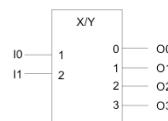
Descodificador

■ Descodificador (em inglês, *Decoder*)

- ▶ O descodificador binário é um circuito combinatório que permite, perante uma combinação de entradas, activar uma e só uma saída.

I1	I0	O0	O1	O2	O3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

DESCODIFICADOR 2:4



- ▶ No símbolo do componente, o índice dos sinais de entrada/saída permite identificar claramente as saídas e o “peso” de cada um dos sinais de entrada.

No decoder da figura, a combinação de duas entradas é responsável por ativar apenas 1 saída por vez (as outras permanecem em estado lógico 0)



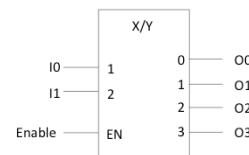
Descodificador

■ Descodificador com entrada de activação (*Enable*):

- A entrada de **ENABLE** permite, quando activa (neste caso, a “1”), que o descodificador funcione normalmente. Quando não activa, inibe o seu funcionamento fazendo com que todas as saídas fiquem inactivas (neste caso, todas a “0”).

EN	I1	I0	O0	O1	O2	O3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0

DESCODIFICADOR 2:4



Neste caso, a saída depende de 3 entradas



Descodificador

■ Descodificador: estrutura interna

- A figura representa a estrutura interna de um descodificador binário de 2 entradas.
- Cada saída representa uma das combinações possíveis das entradas

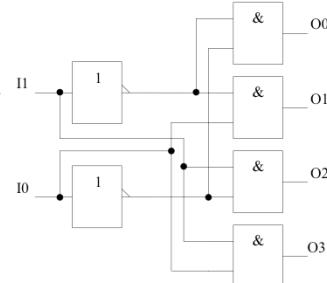
I1	I0	O0	O1	O2	O3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

$$O_0 = \overline{I}_1 \cdot \overline{I}_0$$

$$O_1 = I_1 \cdot \overline{I}_0$$

$$O_2 = \overline{I}_1 \cdot I_0$$

$$O_3 = I_1 \cdot I_0$$



Note que cada saída depende do produto lógico das duas entradas

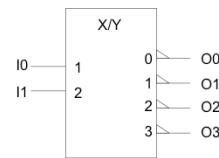


Descodificador

■ Descodificador com saídas activas a zero

- No símbolo do componente, o Δ na saída indica que esta é activa a “0”, i.e., a saída seleccionada tem um “0” e as outras têm um “1”. (funciona como se tivesse um inversor na saída)

I1	I0	O0	O1	O2	O3
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0



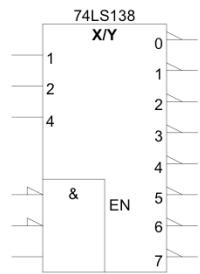
Vejamos.



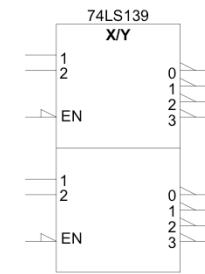
Descodificador

■ Descodificadores: exemplos de componentes

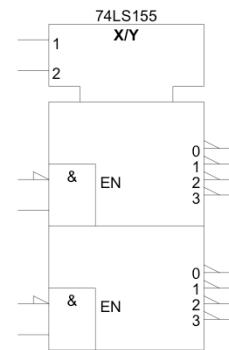
DESCODIFICADOR 3:8



DUAL DECODER 2:4



DUAL DECODER 2:4



Nos 3 exemplos os sinais de saída são activos a zero.

No 138 o Enable é um AND de 3 entradas, 2 delas negadas. No 139 o Enable é activo a zero. No 155 o Enable é um AND de 2 entradas, 1 delas negada.

Observa-se que o ENABLE é uma ‘função’, que pode ser gerado por mais de 1 sinal de entrada

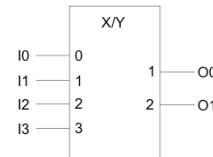


Codificador

■ Codificador (em inglês, *encoder*):

- ▶ O codificador binário é um circuito combinatório que indica qual das entradas possíveis é que está activa (neste caso, a “1”).

I3	I2	I1	I0	O1	O0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1



- ▶ Nesta versão simples, o codificador só considera 4 das 16 combinações possíveis de entrada.
- ▶ O circuito não distingue a situação de todas as entradas estarem a “0”.
- ▶ O circuito não distingue as situações em que estão a “1” mais do que uma entrada.

O *encoder* tem 2^n (ou menos) entradas para n saídas. As saídas geram o código binário correspondente ao valor de entrada

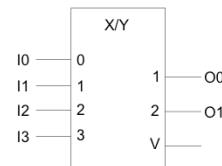


Codificador

■ Codificador de prioridade:

- As entradas deste codificador têm uma ordem de prioridades: em caso de mais de uma entrada activa (a “1”) é considerada a de maior prioridade.

I3	I2	I1	I0	O1	O0	V
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1



- A entrada I3 é a de maior prioridade, seguida da I2, da I1, e a I0 é a de menor prioridade.
- A saída V suplementar indica se existe pelo menos uma entrada activa (a “1”).

Observe que os estados lógicos X são indiferentes para o resultado desta tabela verdade.

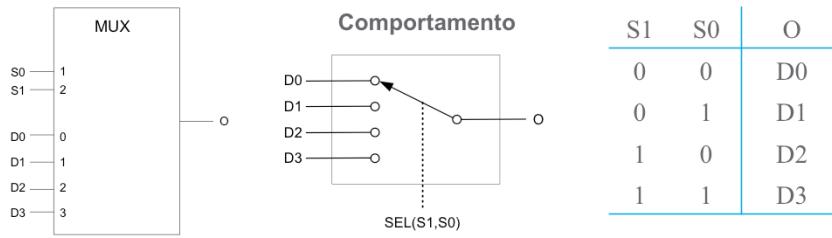


Multiplexer

■ Multiplexer:

- O multiplexer é um circuito combinatório que permite, através da especificação dos sinais de selecção, encaminhar uma das N entradas de dados para a saída.

Exemplo: multiplexer 4:1



- As entradas de selecção determinam a entrada de dados cujo valor é colocado na saída.

A selecção de uma linha de entrada específica é controlada por um conjunto de variáveis de entrada, chamadas entradas de selecção (S_1, S_0). Normalmente, existem 2^n linhas de entrada e n entradas de selecção cujas combinações determinam qual entrada é selecionada.

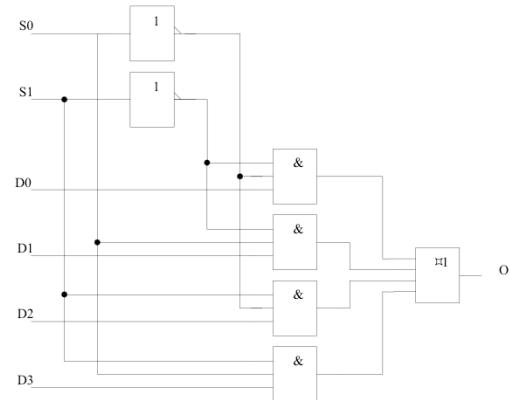


Multiplexer

■ Multiplexer: estrutura interna

S1	S0	O
0	0	D0
0	1	D1
1	0	D2
1	1	D3

$$O = D_0 \cdot \bar{S}_1 \cdot \bar{S}_0 + D_1 \cdot \bar{S}_1 \cdot S_0 + D_2 \cdot S_1 \cdot \bar{S}_0 + D_3 \cdot S_1 \cdot S_0$$



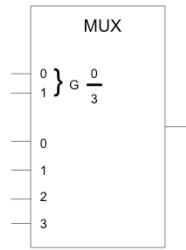
Aqui vemos a descrição da estrutura interna de um multiplexador.



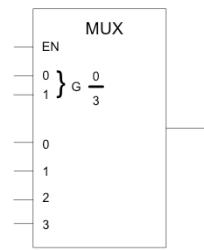
Multiplexer

■ Multiplexer: simbologia

MULTIPLEXER 4:1
simples



MULTIPLEXER 4:1
com enable



EN	S1	S0	O
1	0	0	D0
1	0	1	D1
1	1	0	D2
1	1	1	D3
0	X	X	0

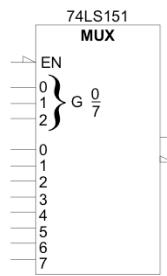
Já aqui temos um multiplexador com a adição do sinal “enable” e sua tabela da verdade.



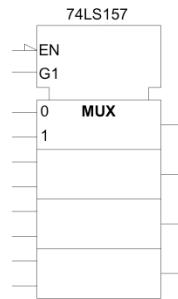
Multiplexer

■ Multiplexers: exemplos de componentes

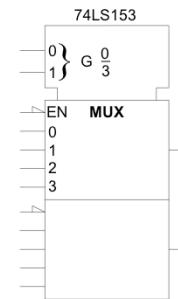
MUX 8:1



QUAD MUX 2:1



DUAL MUX 4:1



Nos 3 exemplos os sinais de Enable são activos a zero (a activação do funcionamento normal do componente acontece quando EN=0).

O 74151 tem uma saída suplementar que é a negação da outra.

Comparação de vários exemplo de multiplexadores.

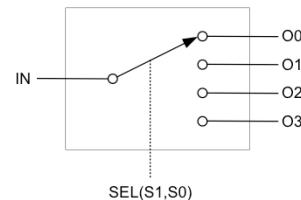
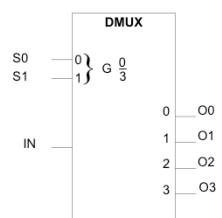


Demultiplexer

■ Demultiplexer:

- O demultiplexer é um circuito combinatório que permite, através da especificação dos sinais de selecção, encaminhar a entrada para uma das N saídas.

Exemplo: Demultiplexer 1:4



S1	S0	O0	O1	O2	O3
0	0	IN	0	0	0
0	1	0	IN	0	0
1	0	0	0	IN	0
1	1	0	0	0	IN

O inverso da seleção (mux) é a distribuição (demux). A informação recebida de uma única linha é transmitida para uma das $2n$ possíveis linhas de saída.



Demultiplexer

■ Demultiplexer: estrutura interna

DEMULTIPLEXER 1:4

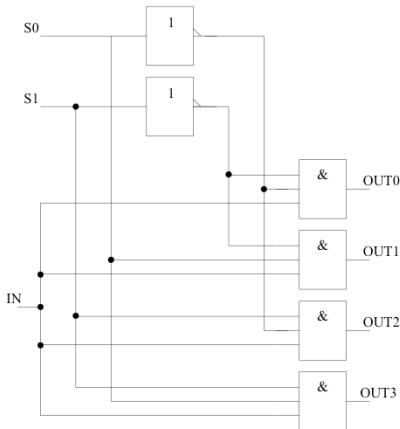
S1	S0	O0	O1	O2	O3
0	0	IN	0	0	0
0	1	0	IN	0	0
1	0	0	0	IN	0
1	1	0	0	0	IN

$$O_0 = IN \cdot \bar{S}_1 \cdot \bar{S}_0$$

$$O_1 = IN \cdot \bar{S}_1 \cdot S_0$$

$$O_2 = IN \cdot S_1 \cdot \bar{S}_0$$

$$O_3 = IN \cdot S_1 \cdot S_0$$



Demonstração da estrutura interna de um demultiplexador



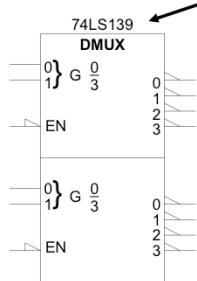
Demultiplexer vs Descodificador

■ Demultiplexeres e Descodificadores:

- Um **descodificador** com *enable* é equivalente a um **demultiplexer**, sendo as entradas de dados do primeiro as entradas de seleção do segundo e a entrada de *enable* do primeiro a entrada de dados do segundo.

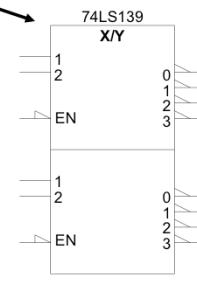
Nota: os 2 símbolos abaixo referem a mesma funcionalidade do circuito.

DUAL DMUX 1:4



74LS139

DECODER 2:4



Lembrando: o demux é um circuito que recebe informações em uma única linha e as transmite em uma das 2^n linhas de saída possíveis.

Nesse caso, a seleção de uma linha de saída específica é controlada pelos valores de bit de n linhas de seleção.



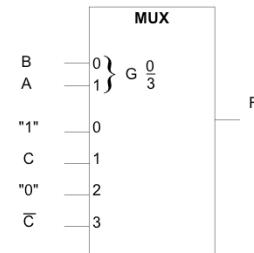
Aplicações

■ Multiplexers: aplicações (f. combinatórias)

- Exemplo de realização de funções combinatórias de 3 variáveis com MUX 4:1

$$F = \bar{A}\bar{B} + \bar{A}C + AB\bar{C}$$

A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0



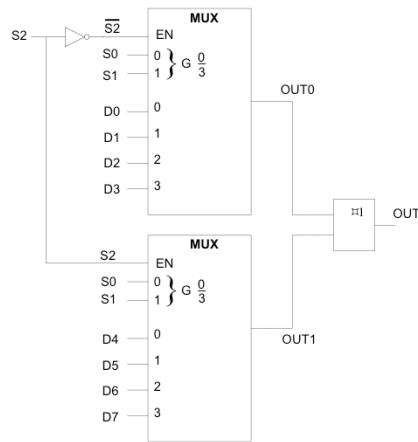
Observe que o mesmo processo poderia ser realizado utilizando um decoder com enable.



Aplicações

■ Multiplexers: aplicações (multiplexagem)

- Exemplo de realização de um MUX 8:1 tendo por base 2 MUXs 4:1



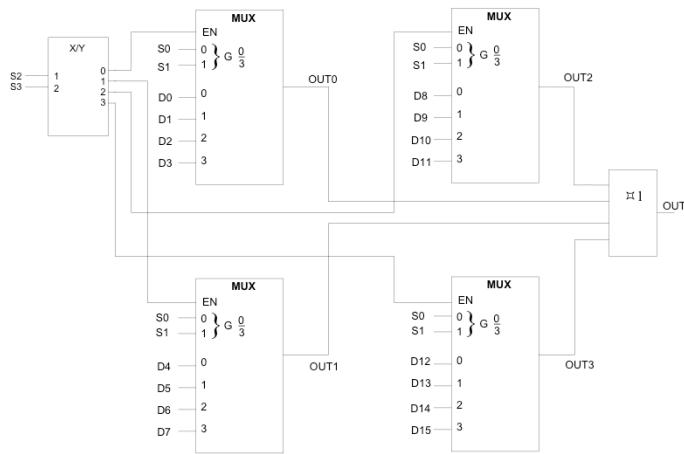
Os circuitos podem ser combinados para uma saída desejada. Para o mux 8:1, o enable tem valor lógico 1 alternado para os dois mux.



Aplicações

■ Multiplexers: aplicações (multiplexagem)

- Exemplo de realização de um MUX 16:1 tendo por base 4 MUXs 4:1



Prof. Héctor Pettenghi

Introdução à Microeletrônica

22

Vejamos o exemplo de circuito de mux 16:1.

PROBLEMAS

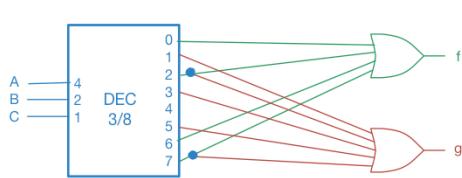
Problema 3.1. Implemente as funções $f(A,B,C)=m(0,2,6,7)$ e $g(A,B,C)=m(1,2,3,5,7)$ utilizando:

- a) Um descodificador 3:8 e o número mínimo de portas lógicas.
- b) Dois descodificadores 2:4 com *enable* e o número mínimo de portas lógicas adicionais.
- c) Multiplexadores com 3 entradas de seleção MUX(8:1).

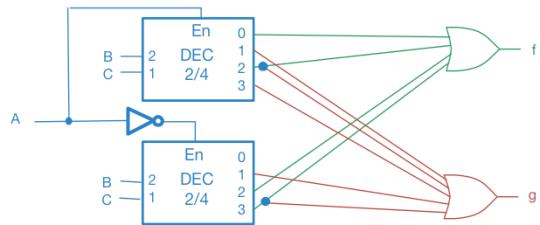
PROBLEMAS

Solução:

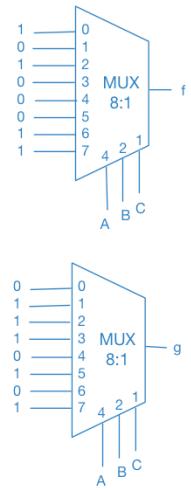
a)



b)



c)



PROBLEMAS

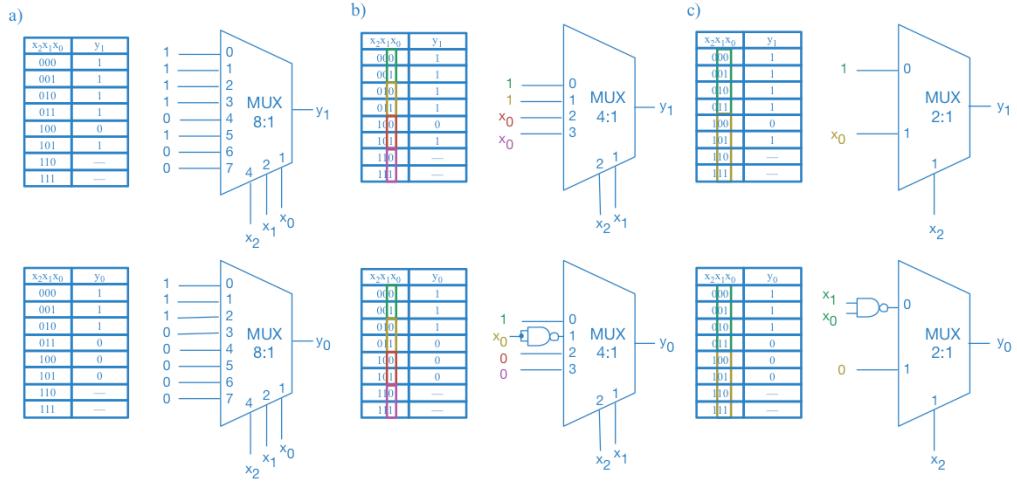
Problema 3.2. Faça o projeto do circuito com entrada de 3 bits, $X=\{x_2, x_1, x_0\}$, e saída de 2 bits, $Y=\{y_1, y_0\}$, que funciona de acordo com a tabela verdade apresentada utilizando:

- a) Um multiplexador com 3 entradas de seleção MUX(8:1).
- b) Um multiplexador com 2 entradas de seleção MUX(4:1)
e uma porta NAND de 2 entradas.
- c) Um multiplexador com 1 entrada de seleção MUX(2:1)
e uma porta NAND de 2 entradas.

x_2	x_1	x_0	y_1	y_0
0	0	0	1	1
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	0
1	0	1	1	0
1	1	0	X	X
1	1	1	X	X

PROBLEMAS

Solução:



PROBLEMAS

Problema 3.6. (Prova 2018.1) Considere a função $f(A,B,C) = \overline{A \oplus B \oplus C}$:

- Apresente a tabela de verdade correspondente.
- Utilizando apenas descodificadores com 2 entradas 2:4 com *enable* e um numero mínimo de portas NAND de 2 entradas (não pode usar portas inversoras), projete e implemente a função lógica $f(A,B,C)$.

Solução:

a)

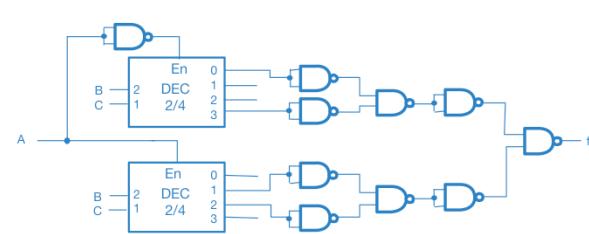
ABC	f
000	1
001	0
010	0
011	1
100	0
101	1
110	1
111	0

m_0

m_3

m_5

m_6





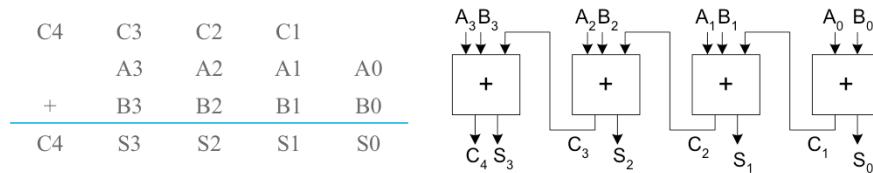
Somadores

■ Circuito para soma aritmética

- Exemplo: Somador de 2 números de 4 bits cada.

$$\begin{array}{r} & 0 & 1 & 1 & 0 & \leftarrow \text{Carry} \\ 7 & & 0 & 1 & 1 & 1 \\ + 2 & & + & 0 & 0 & 1 & 0 \\ \hline 9 & & & 1 & 0 & 0 & 1 \end{array}$$

- A estrutura mais simples resolve 1 bit de cada vez:



Lembrando: o sistema numérico utilizado internamente pelos computadores é o sistema de base 2 (1 ou 0), também conhecido como sistema binário.
É semelhante à adição decimal usual, (lembrando que em decimal, $1+1=2$ e 2 em binário é 10. Ou seja, $1+1$ é igual a 0 e vai 1 (para ser adicionado à próxima coluna). Este "vai 1" é chamado de carry).

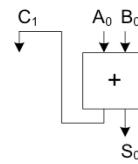
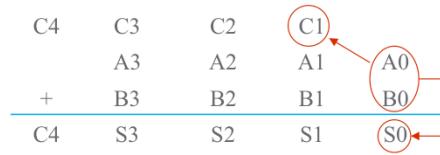
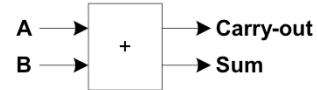
O circuito somador binário mais simples opera uma soma por vez.



Somadores

■ Circuito semi-somador

- ▶ O circuito semi-somador (em inglês, half-adder) soma 2 bits de entrada (sem transporte anterior) e produz 1 bit da soma e 1 bit de transporte.
- ▶ Corresponde p.ex. ao 1º passo do algoritmo de soma: soma os 2 bits de menor peso e obtém 1 bit S_0 da soma e o transporte C_1 para o passo seguinte.



O half-adder não leva em consideração os resultados das somas menos significativas (S_0 e C_1). O resultado é definido apenas pelos bits mais significativos.



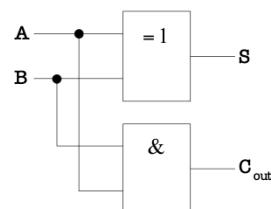
Somadores

■ Circuito semi-somador

A	B	C _{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$C_{out} = A \cdot B$$

$$S = A \oplus B$$



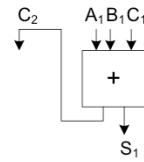
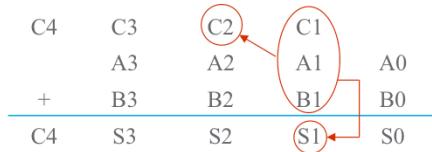
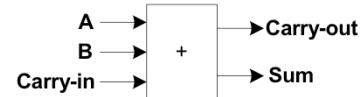
A e B são as entradas. 'Cout' é o "vai 1" , e expressa como uma AND entre A e B. 'S' é o bit menos significativo da soma e é representado pela XOR entre A e B.



Somadores

■ Circuito somador completo

- ▶ O circuito somador completo (em inglês, full-adder) soma 3 bits de entrada (incluindo o transporte anterior) e produz 1 bit da soma e 1 bit de transporte.
- ▶ P.ex. no 2º passo: soma 3 bits A1 e B1 e o transporte C1 do passo anterior, e obtém 1 bit S1 da soma e o transporte C2 para o passo seguinte.



Diferentemente do half-adder, este circuito considera todos os bits como resultado. O Carry-in é utilizado para somar números de 2 casas ou mais.



Somadores

■ Somador completo

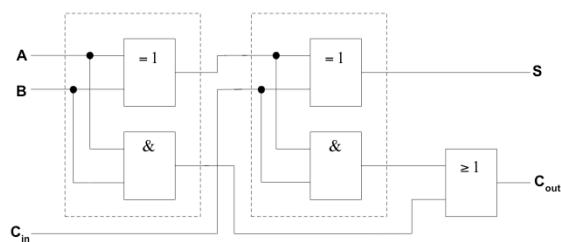
A	B	C _{in}	C _{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

A	B	00	01	11	10
C _{in}	0	0	1	0	1
1	1	1	0	1	0

$$\begin{aligned}S &= \bar{C}_{in} \cdot \bar{A} \cdot B + \bar{C}_{in} \cdot A \cdot \bar{B} \\&\quad + C_{in} \cdot \bar{A} \cdot \bar{B} + C_{in} \cdot A \cdot B \\&= A \oplus B \oplus C_{in}\end{aligned}$$

A	B	00	01	11	10
C _{in}	0	0	0	1	0
1	0	0	1	1	1

$$\begin{aligned}C_{out} &= A \cdot B + C_{in} \cdot A + C_{in} \cdot B \\&= A \cdot B + C_{in} \cdot (A \oplus B)\end{aligned}$$

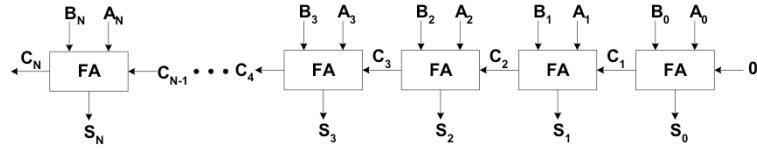


A e B são as entradas. Cin representa o bit da soma anterior (vai 1). Cout é o "vai 1" para o próximo bit e expressa uma operação (A AND B) or (A AND Cin) or (B AND Cin). S é o bit menos significativo da soma (A XOR B).



Somadores

■ Somador em cascata (ripple carry adder)



- A velocidade máxima de execução é limitada pela necessidade de propagar o "Carry" desde a soma do primeiro bit até à soma do bit mais significativo.
- No pior caso, o tempo de propagação do "Carry" será $N \times t_{PFA}$.

Exemplo:

$$\begin{array}{r} 0 | 0 | 0 | 0 \\ 0 \quad 0 \quad 0 \quad 0 \\ + 1 \quad 1 \quad 1 \quad 1 \\ \hline 1 \quad 1 \quad 1 \quad 1 \end{array} \quad \xrightarrow{\hspace{1cm}} \quad \begin{array}{r} 1 | 1 | 1 | 1 \\ 0 \quad 0 \quad 0 \quad 1 \\ + 1 \quad 1 \quad 1 \quad 1 \\ \hline 0 \quad 0 \quad 0 \quad 0 \end{array}$$

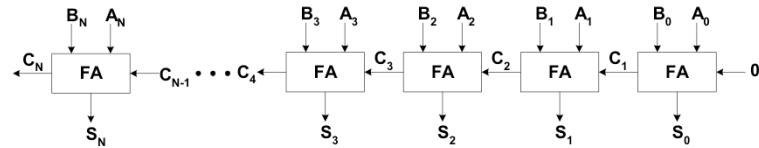
A_0 comuta de 0 para 1
 $A_i = 0, \forall_{i > 0}$
 $B_i = 1, \forall_i$

Este é um tipo de somador que utiliza em cascatas somadores completos, e enviando os carrys (vai 1) como entrada dos próximos.
A vantagem está na simplicidade e modularidade do circuito.



Somadores

■ Somador em cascata (ripple carry adder)



- ▶ O “Ripple Carry Adder” é o somador mais simples possível (que requer menos portas lógicas).
- ▶ Existem inúmeros circuitos alternativos para diversos compromissos velocidade/área.

Este tipo de somador é utilizado geralmente no processamento de sinais digitais

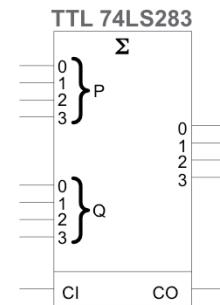


Somadores

■ Somador de 4 bits

► Somador de 4 bits completo:

- Soma:
 - 2 números de 4 bits cada
 - 1 bit de carry-in.
- Gera:
 - Resultado da soma, com 4 bits
 - 1 bit de carry-out.



Por exemplo. Um somador de 4 bits realizará a soma de dois números de 4 bits mais uma informação de carry-in e irá gerar um resultado com também 4 bits e uma informação de carry-out.

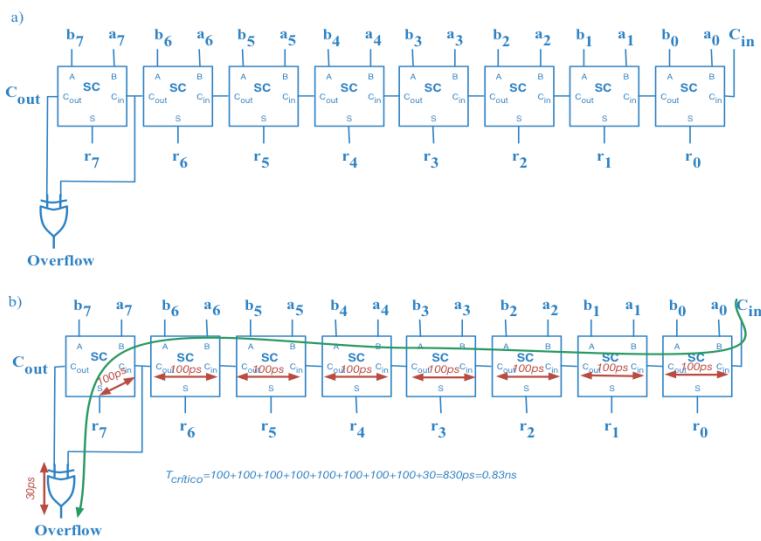
PROBLEMAS

Problema 3.7. Considere um circuito somador de 8-bits com *carry-in*, *carry-out* e *overflow*.

- a) Desenhe o diagrama interno do circuito somador de 8 bits.
- b) Considerando que o tempo máximo de propagação do *full-adder* é de 100ps e de portas lógicas de duas entradas 30ps, determine o tempo máximo de propagação do somador.

PROBLEMAS

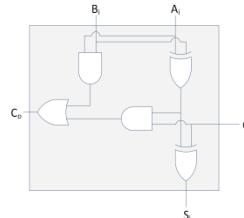
Solução:



PROBLEMAS

Problema 3.8. Considere o full-adder ilustrado na figura e os tempos de propagação da tabela.

- Calcule o tempo máximo de propagação dos sinais A_i, B_i e C_i para cada uma das saídas S_i e C_o . Indique qual o pior caso.
- Calcule o tempo máximo de propagação de um somador de 8 bits considerando que cada full-adder é implementado como se ilustra na figura.
- Proponha as alterações que achar convenientes ao full-adder de forma a minimizar o tempo de propagação. Indique qual o novo valor para o tempo de propagação do somador de 8 bits.



Porta lógica	Tempo de propagação
NOT	10ps
AND2	30ps
OR2	30ps
NAND2	20ps
NOR2	20ps
XOR2	50ps

PROBLEMAS

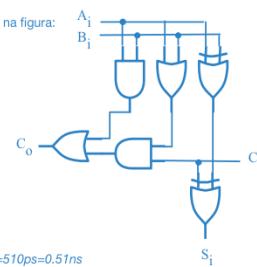
Solução:

a)

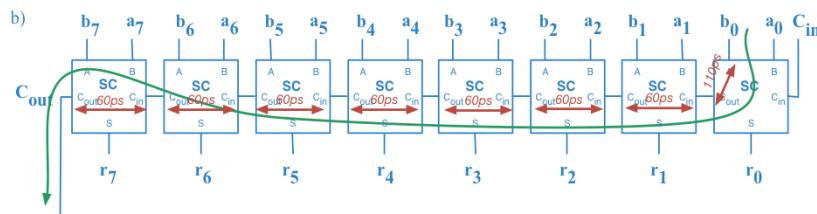
$A_i \rightarrow S_i$	$t_{propagação} = 100ps$
$B_i \rightarrow S_i$	$t_{propagação} = 100ps$
$A_i \rightarrow C_o$	$t_{propagação} = 110ps$
$B_i \rightarrow C_o$	$t_{propagação} = 110ps$
$C_i \rightarrow S_i$	$t_{propagação} = 50ps$
$C_i \rightarrow C_o$	$t_{propagação} = 60ps$

c) Uma versão do *full adder* está apresentada na figura:

$A_i \rightarrow S_i$	$t_{propagação} = 100ps$
$B_i \rightarrow S_i$	$t_{propagação} = 100ps$
$A_i \rightarrow C_o$	$t_{propagação} = 90ps$
$B_i \rightarrow C_o$	$t_{propagação} = 90ps$
$C_i \rightarrow S_i$	$t_{propagação} = 50ps$
$C_i \rightarrow C_o$	$t_{propagação} = 60ps$



$$T_{crítico\ somador} = 90 + 60 + 60 + 60 + 60 + 60 + 60 + 60 = 510ps = 0.51ns$$



$$T_{crítico} = 110 + 60 + 60 + 60 + 60 + 60 + 60 + 60 = 530ps = 0.53ns$$



■ Representação de números negativos

► Módulo + Sinal

- O bit mais significativo representa o sinal, e os restantes bits representam o seu valor absoluto.

Ex.: $-9 = 10001001$

- O valor zero tem duas representações...

Decimal	Módulo + Sinal
+7	0111
+6	0110
+5	0101
+4	0100
+3	0011
+2	0010
+1	0001
+0	0000
-0	1000
-1	1001
-2	1010
-3	1011
-4	1100
-5	1101
-6	1110
-7	1111
-8	-

?

Nos Slides anteriores vimos como realizar operações de soma com números inteiros positivos. Agora veremos como realizar cálculos com números inteiros negativos. Para isso precisamos de uma notação para valores negativos. Na aritmética comum, um número negativo é indicado por um sinal de menos e um número positivo por um sinal de mais. Por causa das limitações de hardware, os computadores devem representar tudo com 1s e 0s, incluindo o sinal de um número. Como consequência, podemos representar o sinal com um bit colocado na posição mais significativa de um número de n bits. A convenção é fazer o bit de sinal 0 para números positivos e 1 para números negativos.

Entretanto, aqui teremos 2 representações para o valor zero.



■ Representação de números negativos

► Complemento para 1

- O complemento para 1 de N, em n bits, é definido como $(2^n - 1) - N$.
- $2^n - 1$ é um número constituído por n 1's.
- Subtrair de 1 equivale a inverter o bit:
 $1 - 0 = 1$ e $1 - 1 = 0$.
- Portanto, complementar para 1 corresponde a inverter todos os bits ($0 \rightarrow 1$ e $1 \rightarrow 0$).
Ex.: $-9 = 11110110$
($= 11111111 - 00001001 = 25510 - 910$).
- **O valor zero tem duas representações...**

Decimal	Complemento para 1
+7	0111
+6	0110
+5	0101
+4	0100
+3	0011
+2	0010
+1	0001
+0	0000
-0	1111
-1	1110
-2	1101
-3	1100
-4	1011
-5	1010
-6	1001
-7	1000
-8	-

?

Para transformar um número binário positivo em uma representação negativa, podemos realizar o complemento de 1. Este consiste em inverter todos os bits do número positivo (aqueles bits que forem 0 se tornarão 1 e os que forem 1 se tornarão zero).

Entretanto aqui também teremos o valor zero com duas representações.



■ Representação de números negativos

► Complemento para 2

- O complemento para 2 de N, em n bits, é definido como $2^n - N$ para $N \neq 0$, e 0 para $N = 0$.
- Portanto, complementar para 2 corresponde a complementar para 1 e somar 1.

Ex.: $-9 = 11110111$

$$(= 100000000 - 00001001 = 25610 - 910).$$

- Na prática, o complemento para 2 pode ser formado do seguinte modo: mantém-se todos os 0's menos significativos e o primeiro 1, e invertem-se todos os outros bits mais significativos.

- **Uma única representação para o valor zero.**

Decimal	Complemento para 2
+7	0111
+6	0110
+5	0101
+4	0100
+3	0011
+2	0010
+1	0001
+0	0000
-0	-
-1	1111
-2	1110
-3	1101
-4	1100
-5	1011
-6	1010
-7	1001
-8	1000

?

Para solucionar o problema de duas representações para o zero, podemos realizar o complemento de 2. Este consiste em complementar para 1 e somar 1 ao resultado. Assim teremos apenas uma única representação para o valor zero.



Representação de números com sinal

■ Números binários com sinal

- ▶ As operações usando o sistema de sinal e valor são mais complicadas, devido à necessidade de gerir separadamente o sinal e o valor.
- ▶ Por isso, são normalmente utilizadas representações em complemento. A representação em **complemento para 2** é habitualmente preferida em sistemas digitais por ter uma única representação para o valor zero, e por as operações envolvidas serem mais simples.

Decimal	Complemento para 2	Complemento para 1	Módulo + Sinal
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
-0	-	1111	1000
-1	1111	1110	1001
-2	1110	1101	1010
-3	1101	1100	1011
-4	1100	1011	1100
-5	1011	1010	1101
-6	1010	1001	1110
-7	1001	1000	1111
-8	1000	-	-

?

Vejamos.



■ Extensão do sinal (complemento para 2)

- Representação de um número utilizando um determinado número de bits, através da adição/remoção de bits à esquerda iguais ao bit de sinal

Exemplos:

$$0100 = +4 \text{ (4 bits)} \rightarrow 00000100 = +4 \text{ (8 bits)}$$

$$1011 = -5 \text{ (4 bits)} \rightarrow 11111011 = -5 \text{ (8 bits)}$$

$$0010 = +2 \text{ (4 bits)} \rightarrow 010 = +2 \text{ (3 bits)}$$

$$1010 = -6 \text{ (4 bits)} \rightarrow ??? = -6 \text{ (3 bits)}$$

Para cálculos entre números que possuem números de bits diferentes, podemos usar a extensão de sinal. Por exemplo, suponha que um byte (8bits) 01101011, o qual representa o número 107, seja usado em um circuito que requer uma entrada de 16 bits. Uma maneira possível de produzir a entrada de 16 bits é extender com oito 0s à esquerda para produzir 0000000001101011.

Já para números negativos, o byte 10010101, que em complemento de 2 representa -107, estendido para 16 bits torna-se 111111110010101.

Vejamos os exemplos dos slides. Para realizar as operações indicadas podemos extender o sinal das representações para que esses possam ser alocados à entrada de um sistema corretamente, tendo em vista o número de bits deste determinado sistema.

Vejamos ainda que o número -6 não pode ser representado com apenas 3 bits em divergência com seu complementar positivo ("110"). O -6 apenas pode ser representado com 4 bits ou mais. Ou seja, transformando ("0110") em ("1010").



Representação de números com sinal

■ Soma aritmética de números com sinal usando complemento para 2

- A soma aritmética de dois números binários com sinal, representados em complemento para 2, é obtida pela simples adição dos dois números incluindo os bits de sinal. O último “carry out” não é considerado.

Exemplos:

$$\begin{array}{r} \boxed{0} \boxed{0} \boxed{0} \boxed{0} \\ 4 \quad 0 \ 1 \ 0 \ 0 \\ + 3 \\ \hline 7 \quad 0 \ 1 \ 1 \ 1 \end{array}$$

$$\begin{array}{r} \boxed{1} \boxed{1} \boxed{0} \boxed{0} \\ 4 \quad 0 \ 1 \ 0 \ 0 \\ + (-3) \quad + 1 \ 1 \ 0 \ 1 \\ \hline 1 \quad 0 \ 0 \ 0 \ 1 \end{array}$$

$$\begin{array}{r} \boxed{0} \boxed{0} \boxed{0} \boxed{0} \\ -4 \quad 1 \ 1 \ 0 \ 0 \\ + 3 \quad + 0 \ 0 \ 1 \ 1 \\ \hline -1 \quad 1 \ 1 \ 1 \ 1 \end{array}$$

$$\begin{array}{r} \boxed{1} \boxed{1} \boxed{0} \boxed{0} \\ -4 \quad 1 \ 1 \ 0 \ 0 \\ + (-3) \quad + 1 \ 1 \ 0 \ 1 \\ \hline -7 \quad 1 \ 0 \ 0 \ 1 \end{array}$$

Vejamos o exemplo de soma de números com sinal, usando complemento de 2.



Subtractores

■ Subtracção de números com sinal usando complemento para 2

- A subtracção de dois números binários com sinal, representados em complemento para 2, é obtida do seguinte modo:
- forma-se o complemento para 2 do subtraor
 - soma-se ao subtraendo.

Exemplo:

$$\begin{array}{r} 4 & 0 \ 1 \ 0 \ 0 \\ - 3 & - 0 \ 0 \ 1 \ 1 \\ \hline 1 & \end{array} \quad \rightarrow \quad \begin{array}{r} 4 & 0 \ 1 \ 0 \ 0 \\ + (-3) & + 1 \ 1 \ 0 \ 1 \\ \hline 1 & 0 \ 0 \ 0 \ 1 \end{array} \quad \rightarrow \quad \begin{array}{r} 0 \ 1 \ 0 \ 0 \\ + 1 \ 1 \ 0 \ 0 \\ \hline 0 \ 0 \ 0 \ 1 \end{array}$$

(através de complemento para 2) (através de complemento para 1)

Já aqui temos o exemplo de subtração para número com sinal usando o complemento de 2.

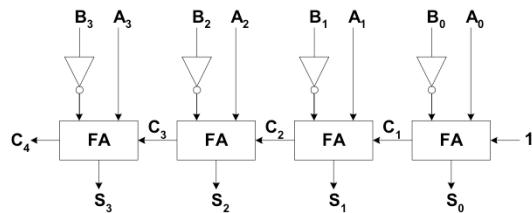


Subtractores

■ Subtração de números com sinal usando complemento para 2

► Complemento para 2 = (Complemento para 1) + 1

- A complementação para 1 é realizada invertendo todos os bits do subtractor.
- A adição de 1 é efectuada pondo o Carry inicial a 1.



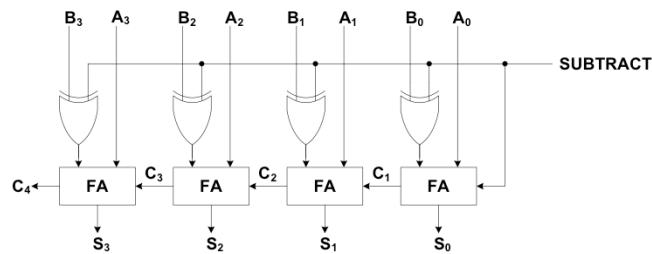
Este seria uma forma de realizar a subtração em hardware com complemento de 2. Veja que o círcuito realiza a inversão dos bits do operador B, por meio de portas NOT e há ainda a adição de +1 no carry in do primeiro Somador Completo (Full Adder).



Círcuito somador/subtractor

■ Círcuito somador/subtractor

- ▶ As operações de adição e subtração são habitualmente combinadas num único somador genérico, através da inclusão de 1 porta ou-exclusivo em cada Full-Adder.
- ▶ Quando o sinal de controlo SUBTRACT = 0, é realizada a adição $A + B$ (os operandos B_i não são invertidos e $C_0 = 0$).
- ▶ Quando o sinal de controlo SUBTRACT = 1, é realizada a subtração $A - B$ (os operandos B_i são invertidos e $C_0 = 1$).



Já aqui vemos o circuito genérico para soma e subtração. Este é controlado pelo sinal de controle (Aqui chamado de Subtract).

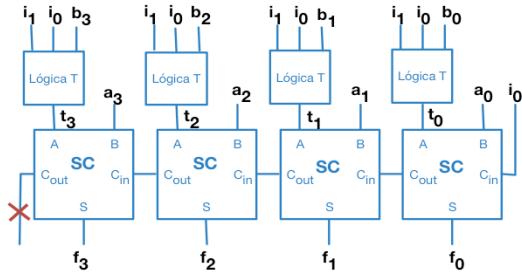
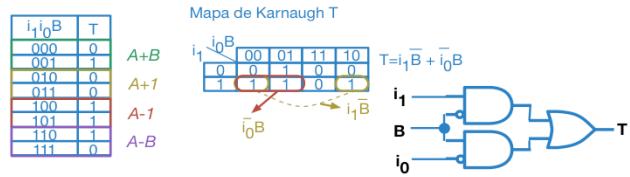
PROBLEMAS

Problema 3.9. Desenhe um circuito aritmético com duas entradas de seleção S_1 e S_0 que realize as seguintes operações aritméticas. Suponha A e B entradas de n -bits.

$S_1 S_0$	$C_{in}=0$	$C_{in}=1$
00	$F = A + B$	$F = A + B + 1$
01	$F = A$	$F = A + 1$
10	$F = \bar{B}$	$F = \bar{B} + 1$
11	$F = A + \bar{B}$	$F = A + \bar{B} + 1$

PROBLEMAS

Solução:

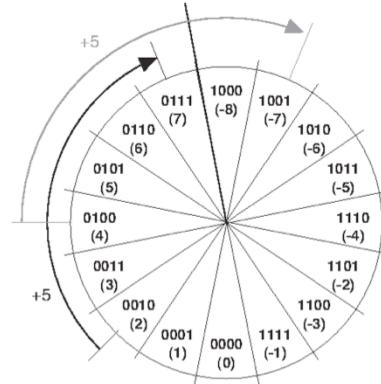




Excesso

■ Excesso (overflow)

$$\begin{array}{r} 0 \boxed{1} 0 0 \\ 4 \quad \quad 0 1 0 0 \\ + 5 \quad + 0 1 0 1 \\ \hline ??? \quad \quad 1 0 0 1 \end{array}$$



Em cada um dos exemplos anteriores de adição e subtração com complemento de 2, os números que foram adicionados consistiam em um bit de sinal e três bits de magnitude. As respostas também consistiam em um bit de sinal e três bits de magnitude. Qualquer carry para a posição do quinto bit foi desconsiderado.

Em todos esses casos considerados, a magnitude da resposta foi pequena o suficiente para caber em três bits. Entretanto, quando a magnitude do sinal requer um número maior de bits, dizemos que houve a ocorrência de overflow.



Excesso

■ Excesso (overflow)

- ▶ Para se obter um resultado correcto, na adição e na subtração, é necessário assegurar que o resultado tem um número de bits suficiente. Se somarmos dois números de N bits e o resultado ocupar $N+1$ bits diz-se que ocorreu um **overflow**.
- ▶ As unidades aritméticas digitais usam um número fixo de bits para armazenar os operandos e os resultados, sendo necessário detectar e sinalizar a ocorrência de um **overflow**.
 - Exemplo: um **overflow** pode ocorrer na adição se os dois operandos são ambos positivos ou se são ambos negativos.

Em resumo.



Excesso

■ Excesso (overflow)

- A condição de overflow pode ser detectada por inspecção dos dois bits de carry mais significativos.

Exemplo:

$$\text{Overflow} = \text{CarryOut}_{N-1} \oplus \text{CarryOut}_{N-2}$$

The diagram shows two 4-bit binary additions. In the first example, 4 + 5, the result is 1001. The two most significant carry bits (C₃ and C₂) are both 1, which is highlighted with green circles and a red box. In the second example, -4 + (-5), the result is 1100. The two most significant carry bits (C₃ and C₂) are both 1, also highlighted with green circles and a red box. Both cases result in overflow, indicated by the label "ovfl." under each result.

$$\begin{array}{r} 4 \\ + 5 \\ \hline \text{ovfl.} \end{array} \quad \begin{array}{r} -4 \\ + (-5) \\ \hline \text{ovfl.} \end{array}$$
$$\begin{array}{r} 0100 \\ + 0101 \\ \hline 1001 \end{array} \quad \begin{array}{r} 1100 \\ + 1011 \\ \hline 0111 \end{array}$$

As operações entre (+4)(+5) e (-4)(-5) resultam em overflow.



Excesso

- Qual a diferença entre os sinais de *carry* e *overflow*?

Representação	$C = C_{N-1} = 1$	$O = 1$
SEM sinal	Excedeu a capacidade de representação	Sem significado
COM sinal	Sem significado	Excedeu a capacidade de representação

Na tabela $C = \text{Carry}$ e $O = \text{Overflow}$



Outras Operações

■ Multiplicação

A		1	1	0	1	Multiplicando
B	x	1	0	1	0	Multiplicador
		0	0	0	0	Produto parcial
		1	1	0	1	Produto parcial
		0	0	0	0	Produto parcial
		1	1	0	1	Produto parcial
M	1	0	0	0	0	Resultado

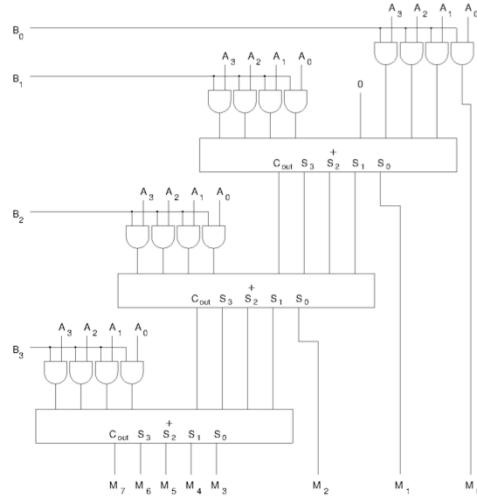
A multiplicação de números binários é feita da mesma maneira que a multiplicação de números decimais. O processo é realmente mais simples porque os dígitos do multiplicador são 0 ou 1 e, portanto, estamos sempre multiplicando por apenas 0 ou 1.

Vejamos que a multiplicação do primeiro zero do segundo multiplicador, pelo primeiro multiplicador, irá gerar o primeiro produto parcial, que será igual a “0000”. Assim continuamos multiplicando os bits do segundo multiplicador pelo primeiro e gerando produtos parciais. Finalmente, então, somamos os produtos parciais para chegar ao resultado final.



Outras Operações

■ Multiplicação (representação sem sinal)



Prof. Héctor Pettenghi

Introdução à Microeletrônica

56

Aqui temos o circuito de um multiplicador.

Cada célula contém uma porta AND de 2 entradas que forma um produto parcial e um somador completo (CSA) para adicionar o produto parcial à soma de cada etapa.

A primeira linha converte o primeiro produto parcial para entrada no CSA. Cada linha posterior usa o CSA para adicionar o produto parcial correspondente ao resultado redundante da linha anterior e gerar um novo resultado para a próxima linha. Os bits de saída menos significativos estão disponíveis como saídas de soma diretamente dos CSAs. Os bits de saída mais são transportados para a próxima etapa e apenas após toda propagação são convertidos na forma binária regular.



Outras Operações

■ Caso particular: multiplicação por uma potência inteira de 2

Exemplo:

$$\begin{aligned}6 \times 4 &= 24 \\ \Leftrightarrow (0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0) \times 4 &= 24 \\ \Leftrightarrow (0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0) \times 2^2 &= 24 \\ \Leftrightarrow 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 &= 24\end{aligned}$$

Ou seja: $000110 \times 4 = 0110\textcolor{blue}{00}$

Deslocamento à
esquerda de 2 posições

A multiplicação pela k potência de 2 (i.e. 2^k) corresponde a deslocar os bits do operando em k posições para a esquerda.

Ao realizar uma multiplicação por um número 2^k temos que esta multiplicação corresponde ao deslocamento do operando em k posições para a esquerda.



Outras Operações

■ Divisão

Dividendo	Divisor
1 0 0 1 0 0 1 1	0 1 0 1
- 0 1 0 1	
0 1 0 0 0	0 0 0 1 1 1 0 1
- 0 1 0 1	
0 0 1 1 0	Quociente
- 0 1 0 1	
0 0 0 1 1 1	
- 0 1 0 1	
0 0 1 0	Resto

O processo para dividir um número binário (o dividendo) por outro (o divisor) é o mesmo seguido para os números decimais, que geralmente chamamos de "divisão longa". O processo real é mais simples em binário porque quando estamos verificando quantas vezes o divisor "entra" no dividendo, existem apenas duas possibilidades, 0 ou 1.

No primeiro caso isolamos parte do dividendo referente à 1001 e subtraímos do divisor 0101. Isso equivale à divisão $9 \div 5$. O resultado do Quociente é "1" (Será o primeiro 1 da esquerda para a direita, pois estamos dividindo da esquerda para a direita) e o resto 4 (0100).

Como segunda etapa desce um zero do dividendo e retornamos à divisão. Agora 01000 dividido por 0101 ($8 \div 5$). Novamente quociente igual a 1 e resto igual a 0011 (3). Continuamos descendo mais um zero e teremos 00110 por 0101 ($6 \div 5$).

Resto será 1 e quociente será 1. Por fim descemos um bit para formar o número 00011. Entretanto este número é menor do que o divisor. Então colocamos um zero no quociente e descemos o último bit formando o número 000111 e dividimos por 0101 ($7 \div 5$). O resultado é 1 no quociente e o resto é 0010.

No total temos a divisão de 147 por 5 que é igual a 29 com resto 2.



Outras Operações

■ Divisão

- ▶ Não tem uma sequência fixa de operações elementares
- ▶ O número seleccionado de bits do dividendo em cada passo é variável
 - +
 - ▶ Operação pouco frequente, na maioria das aplicações
 - ▶ Operação complexa
 - ↓
 - ▶ Implementada tipicamente através de uma sequência de operações mais simples
 - ↓
 - Programa

Em resumo.



Outras Operações

■ Caso particular: divisão por uma potência inteira de 2

Exemplo:

$$\begin{aligned}36 \div 4 &= 9 \\ \Leftrightarrow (1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0) \div 4 &= 9 \\ \Leftrightarrow (1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0) \div 2^2 &= 9 \\ \Leftrightarrow (1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0) &= 9\end{aligned}$$

Ou seja: $100100 \div 4 = 1001 \cancel{0}$

Deslocamento à direita
de 2 posições

A divisão pela k potência de 2 (i.e. 2^k) corresponde a deslocar os bits do operando em k posições para a direita.

Ao realizar uma divisão por um número 2^k temos que esta divisão corresponde ao deslocamento do operando em k posições para a direita.

PROBLEMAS

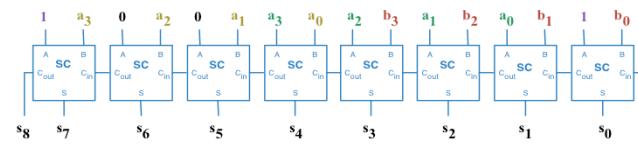
Problema 3.10. Deseja-se obter as seguintes operações aritméticas usando unicamente um somador de 8 bits com C_{in} - C_{out} :

- a) A operação $18 \times A + B + 130$ (A e B de 4-bits sem sinal).
- b) A operação $34 \times A + 1$ (A de 4-bits sem sinal).
- c) A operação $3 \times A$ (A de 4-bits com sinal).

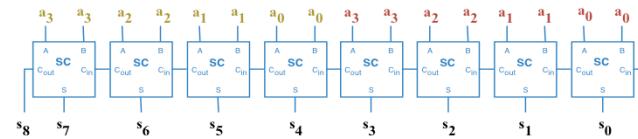
PROBLEMAS

Solução:

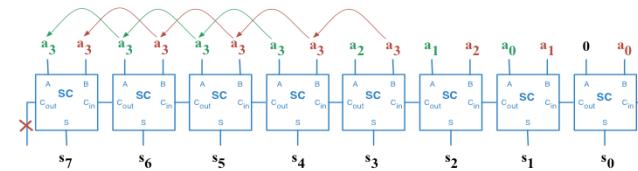
a) $18A + B + 130 = 16A + 2A + B + 130$



b) $34A + 1 = 16A + 16A + A + A + 1$



c) $3A = 2A + A$





Agradecimentos

Algumas páginas desta apresentação resultam da compilação de várias contribuições produzidas pelos professores:

- Guilherme Arroz, Horácio Neto, Nuno Horta, Nuno Roma, Pedro Tomás do IST Lisboa.