universidade de aveiro deti departamento de electrónica, telecomunicações e informática

Trabalho Prático 3

Ciruitos Combinatórios

Objetivos

- Introdução ao *DesignWorks* 5
- Análise de circuitos combinatórios
- Síntese com *gates* discretas
- Síntese com blocos elementares: descodificadores e *multiplexers*

Introdução

Os circuitos digitais dividem-se em *combinatórios* e *sequenciais*. Nos *combinatórios* o valor da saída só depende do valor atual das entradas. Nos *sequenciais* o valor da saída depende, para além do *estado atual* das entradas, também do *estado* em que o circuito se encontra. O *estado* atual do circuito depende do valor *passado* das entradas. Ou seja, o circuito sequencial possui *memória*.

Hoje em dia, os projetistas (*designers*) de circuitos digitais (*hardware*), para além de esboços em papel, usam ferramentas de *software* para validar os seus projetos com mais facilidade e rapidez. O *DesignWorks* (DW) é um programa popular deste género e irá ser usado nas próximas aulas. Permite criar (i.e., editar) e simular circuitos combinatórios e sequenciais. (O *software* DW5 encontra-se já instalado nos computadores das salas de aulas práticas).

Começamos por analisar um simples circuito do operador XOR com *gates*[1] discretas. Seguidamente, iremos implementar e simular vários circuitos:

a) um comparador de 2-bits
b) um descodificador para um *display* de 7-segmentos, com recurso a uma PROM
c) uma calculadora *bitwise* de funções Booleanas.

Como exercícios adicionais são ainda fornecidos outros exemplos de utilização de blocos lógicos elementares, como sejam o somador-de-1 bit, o descodificador e o *multiplexer,* na implementação de funções combinatórias de nível abstracional mais elevado.

[1] *gate* - Equivalente ao termo português 'porta lógica'.

Guião

1. Introdução ao *DesignWorks/LogicWorks*

Dada a limitação do tempo da aula prática (2 horas), recomenda-se a consulta prévia do tutorial *Tutorial_DW5.pdf* disponível no *site* da UC, para que a execução do guião proposto durante a aula decorra de forma mais célere.

Uma vez que o *DesignWorks5,* usado nas aulas, não é gratuito, o estudante pode ir adiantando o trabalho utilizando a versão gratuita do *LogicWorks* (variante do *DesignWorks*) que está disponível *online* no site do fabricante (existe um *link* para o *site* no *elearning*).

2. Circuitos com *gates* discretas

O circuito da Figura 1 representa uma implementação possível dum operador lógico.

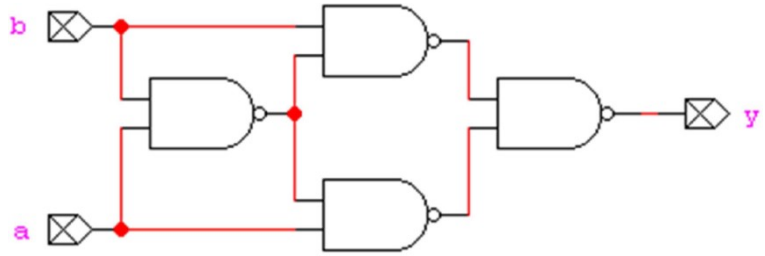


Figura 1 - Operador lógico com portas NAND

a) Determine analiticamente qual o operador em causa.

b) Desenhe o diagrama lógico da implementação direta (i.e., com *gates* AND, OR e NOT). Quais as vantagens e/ou desvantagens desta relativamente ao circuito da Figura 1?

3. Comparador de 2-bits[2]

Considere o circuito combinatório da Figura 2. Este ativa a saída (**F**) sempre que o valor A for maior do que o valor B. A e B são fornecidos através das entradas a_1, a_0, b_1 e b_0 (onde a_0 e b_0 são os bits menos significativos de A e B, respetivamente).

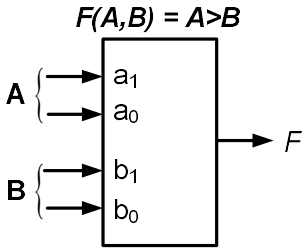


Figura 2 - Comparador de 2-bits

a) Determine a função lógica **F** na forma de soma-de-produtos (SOP), em função de a_1, a_0, b_1 e b_0.

b) Implemente no *DesignWorks* o circuito da função **F** e teste o valor da saída para todas as combinações das entradas.

[2] Consulte o Apêndice A.

3. Descodificador com PROM e simulação

a) Projete um descodificador de binário para um *display* de 7-segmentos, com a tabela de verdade da Figura 3 (depois de preencher as células em falta), usando uma PROM[3] com 4-bits de entrada e 8-bits de saída.

Seg		G	F	E	D	C	B	A	
Bin	B7	B6	B5	B4	B3	B2	B1	B0	Hex
0		0	1	1	1	1	1	1	3F
1		0	0	0	0	1	1	0	06
2		1	0	1	1	0	1	1	5B
3		1	0	0	1	1	1	1	4F
4		1	1	0	0	1	1	0	66
5		1	1	0	1	1	0	1	6D
6									
7									
8									
9		1	1	0	1	1	1	1	6F
A		1	1	1	0	1	1	1	77
b		1	1	1	1	1	0	0	7C
C									
d		1	0	1	1	1	1	0	5E
E		1	1	1	1	0	0	1	79
F		1	1	1	0	0	0	1	71

Figura 3 - Tabela de verdade do descodificador de 7-Seg

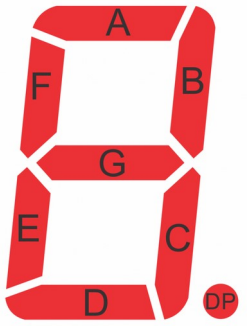


Figura 4 - *Display* de 7-Seg.

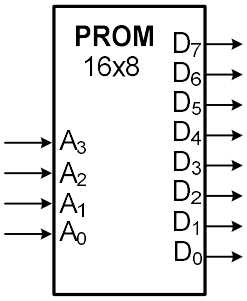


Figura 5 - *PROM 16 x 8-bit*

b) Teste no *DesignWorks* o circuito com a PROM e um display de 7-segmentos da Figura 6.

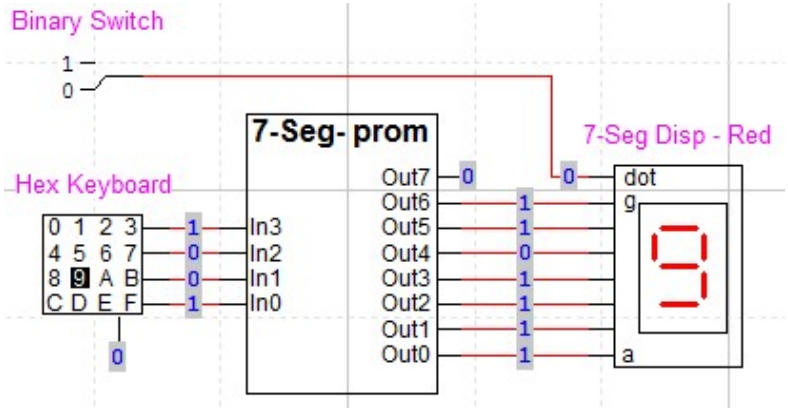


Figura 6 - Circuito de teste da PROM de descodificação de 7-Seg.

[3] No *DesignWorks,* para criar um dispositivo de simulação PROM use o *wizard* PROM/RAM/PLA. O *wizard* pode ser iniciado a partir do menu: 'Simulation-> PROM/RAM/PLA Wizard...'. Selecione o *device* PROM, defina 4 linhas de endereço e 8 bits de dados. Em seguida, introduza na janela de edição o conteúdo das posições de memória em hexadecimal separadas por espaço, e.g., "3F 06 ... 79 71" (sem as aspas).

4. Calculadora *Bitwise*

Pretende-se construir uma calculadora lógica (*bitwise*) com 2 entradas de dados (A e B) e 3 entradas de seleção de operação (C_2, C_1 e C_0), como indicado na Figura 7.

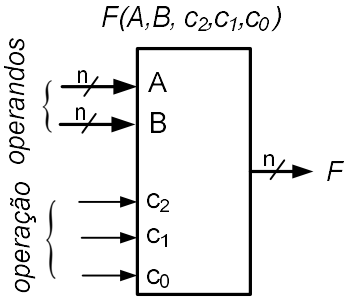


Figura 7 - Calculadora lógica

A saída do circuito (**F**) obedece à seguinte tabela de verdade:

C_2	C_1	C_0	*F*
0	0	0	1
0	0	1	A + B
0	1	0	$\overline{A \cdot B}$
0	1	1	$A \oplus B$
1	0	0	$\overline{A \oplus B}$
1	0	1	A . B
1	1	0	$\overline{A + B}$
1	1	1	0

Figura 8 - Calculadora: Tabela de verdade

Supondo que ambos os operandos A e B possuem uma largura de 2-bits, use o diagrama de blocos proposto na Figura 9 e simule o funcionamento das quatro operações indicadas.

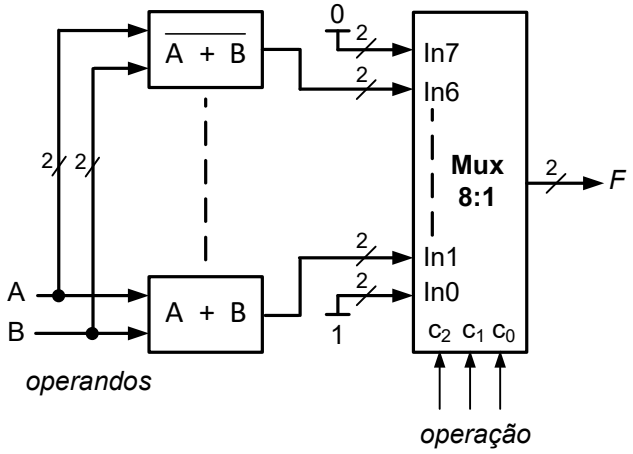


Figura 9 - Calculadora: Diagrama de blocos com 2-bits

Exercícios adicionais

5. Síntese com somadores de 1-bit

Projete um circuito somador construído a partir de somadores completos de 1 bit interligados em cascata.

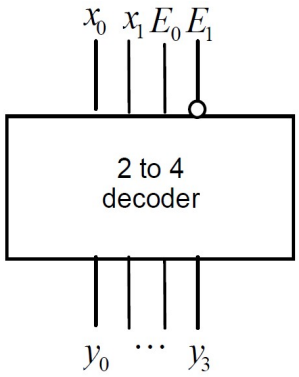
a) Implemente e simule o somador completo de 1 bit com *gates* elementares.

b) Usando o bloco elementar desenvolvido em **a**) construa e simule um circuito somador para palavras de 4 bits.

6. Síntese com descodificadores (*decoder*)

a) Projete e implemente com *gates* um descodificador de 2 entradas para 4 saídas. O circuito deverá ter 2 entradas adicionais de validação (*enable*) uma ativa a "1" e outra ativa a "0".

b) Construa, a partir do bloco "2 to 4 decoder" da alínea anterior, um descodificador de 4 entradas para 16 saídas, também com E_0 e E_1. Usando o programa *DesignWorks* implemente e simule o bloco "2 to 4 decoder" e o descodificador de 4 entradas para 16 saídas.



c) Considere a seguinte função Booleana não necessariamente mínima. Sugira uma implementação baseada em descodificadores de 4 entradas para 16 saídas e *gates* OR adicionais.

$$f(A,B,C,D) = \overline{A}.B.C + A.D + A.C$$

7. Síntese com *multiplexers*

Sugira implementações da função $f(A,B,C,D) = \sum m(0,3,5,7,11,12,13,15)$ baseadas em:

a) *Multiplexer* 16:1 e constantes 0 e 1

b) *Multiplexer* 8:1 e constantes 0 e 1

c) *Multiplexer* 4:1, constantes 0 e 1 e lógica elementar adicional.

Apendice A - Comparadores

Tabelas de verdade e equações lógicas, na forma soma-de-produtos (SOP), de comparadores de 1-bit e de 2-bits.

A.1 Comparador de 1-bit

A0	B0	L	E	G
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

Figura A.1 - Comparador de 1-bit

$A < B : L = \overline{A0}.B0$

$A = B : E = \overline{A0}.\overline{B0} + A0.B0 = \overline{A0 \oplus B0}$

$A > B : G = A0.\overline{B0}$

A.2 Comparador de 2-bits

A tabela da Figura A.2 está incompleta, devendo o aluno completar os campos a sobreado da coluna G, escrever e simplificar a respetiva equação lógica.

A1	A0	B1	B0	L	E	G
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1	0	
0	0	1	1	1	0	
0	1	0	0	0	0	
0	1	0	1	0	1	
0	1	1	0	1	0	
0	1	1	1	1	0	
1	0	0	0	0	0	
1	0	0	1	0	0	
1	0	1	0	0	1	
1	0	1	1	1	0	
1	1	0	0	0	0	
1	1	0	1	0	0	
1	1	1	0	0	0	1
1	1	1	1	0	1	0

Figura A.2 - Comparador de 2-bits

$A < B : L = \overline{A1}.B1 + \overline{A0}.B1.B0 + \overline{A1}.\overline{A0}.B0$

$A = B : E = \left(\overline{A0}.\overline{B0} + A0.B0\right)\left(\overline{A1}.\overline{B1} + A1.B1\right)$
$= \overline{A0 \oplus B0}\,.\overline{A1 \oplus B1}$

$A > B : G = ?$