Guia tutorial das aulas práticas de Sistemas Digitais I

Engenharia de Sistemas e Informática - 2º Ano

versão 2001/2002

António Joaquim Esteves e João Miguel Fernandes

Dep. Informática, Universidade do Minho

Braga, Portugal

22 de Fevereiro de 2002

Introdução

Este documento funciona como referência para as aulas teórico-práticas de **Sistemas Digitais I** (LESI-2). Aquando da leccionação dum módulo, supõe-se que os alunos já leram o texto deste guia relativo a esse módulo.

O livro principal de suporte à disciplina é "John F. Wakerly, *Digital Design: Principles and Practices*, Prentice-Hall International, 2000, 3rd edition, ISBN 0-13-082599-9" e será daqui em diante referido por **DDPP3**. Muitos dos módulos constantes deste guia serão muito mais produtivos, caso o aluno leia previamente as partes do livro indicadas.

Além do livro referido, devem igualmente servir como material complementar de estudo os seguintes livros:

- M. Morris Mano, *Digital Design*, Prentice-Hall International, 1991, 2nd edition, ISBN 0-13-212994-9.
- Randy H. Katz, *Contemporary Logic Design*, The Benjamin/Cummings Publishing Company, 1994, ISBN 0-8053-2703-7.
- Daniel D. Gajski, *Principles of Digital Design*, Prentice-Hall, 1997, ISBN 0-13-301144-5.

Módulo 1
Implementação de sistemas digitais com componentes TTL

Objectivos

Pretende-se que o aluno se familiarize com várias operações Booleanas (AND, OR, NOT, XOR, etc.) e que tome conhecimento das características físicas e funcionais dos circuitos integrados TTL da família 74LSXXX;. É igualmente objectivo deste módulo que o aluno saiba concretizar um sistema digital, recorrendo a circuitos integrados dessa família.
Leituras: DDPP3, secção 3.11.

Sessão Laboratorial

1. Implementar uma unidade aritmética e lógica (ALU) simples, que apenas efectua a soma dos valores A e B de 2 bits e produz um resultado S de 3 bits, numa placa de testes (*breadboard*) e utilizando circuitos integrados TTL da família 74LSXXX. Na implementação deve usar apenas portas lógicas dos seguintes tipos: NAND, AND, OR, XOR e XNOR de 2 entradas.

2. Verificar o correcto funcionamento do circuito, aplicando todas as combinações possíveis aos sinais de entrada e observando se as saídas apresentam os valores esperados.

Figura 1: Diagrama da ALU de 2 bits a implementar.

Entrada B	Entrada A	Saída S	
		binário	decimal
B1 B0	A1 A0	S2 S1 S0	
0 0	0 0	0 0 0	0
0 0	0 1	0 0 1	1
0 0	1 0	0 1 0	2
0 0	1 1	0 1 1	3
0 1	0 0	0 0 1	1
0 1	0 1	0 1 0	2
0 1	1 0	0 1 1	3
0 1	1 1	1 0 0	4
1 0	0 0	0 1 0	2
1 0	0 1	0 1 1	3
1 0	1 0	1 0 0	4
1 0	1 1	1 0 1	5
1 1	0 0	0 1 1	3
1 1	0 1	1 0 0	4
1 1	1 0	1 0 1	5
1 1	1 1	1 1 0	6

Tabela 1: Tabela de verdade que define a funcionalidade da ALU de 2 bits.

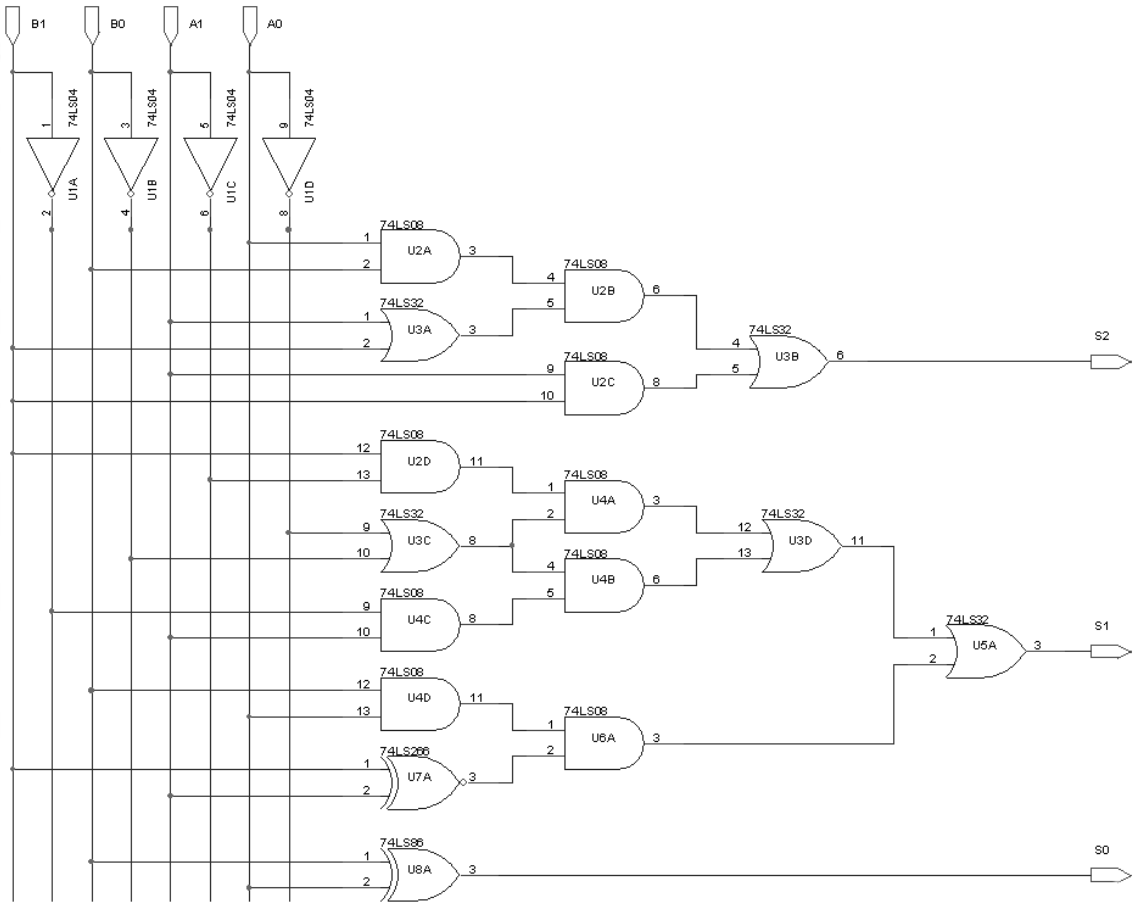


Figura 2: Esquemático da ALU de 2 bits.

Referência de alguns circuitos integrados TTL

Porta Lógica	Referência
Inversor (NOT)	74LS04
AND 2 entradas	74LS08
AND 3 entradas	74LS11
AND 4 entradas	74LS21
OR 2 entradas	74LS32
OR 4 entradas	74LS72
NAND 2 entradas	74LS00
NAND 3 entradas	74LS10
NAND 4 entradas	74LS20
NOR 2 entradas	74LS02
NOR 3 entradas	74LS27
XOR 2 entradas	74LS86
XNOR 2 entradas	74LS266
Flip-flop tipo D	74LS74

Características físicas de circuitos integrados TTL

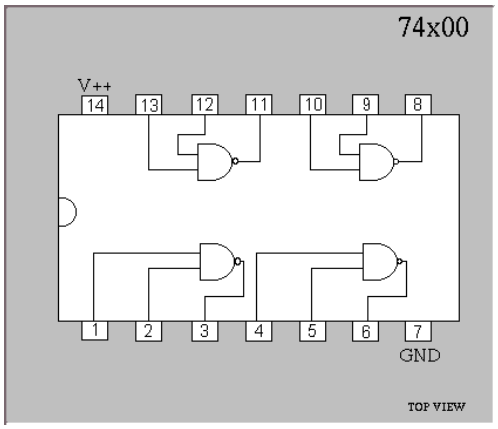


Figura 3: 7400: quatro portas NAND de 2 entradas

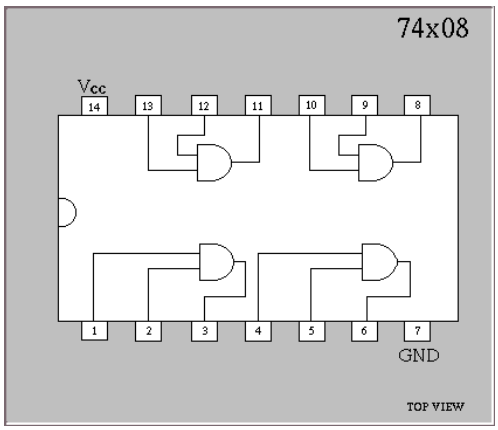


Figura 4: 7408: quatro portas AND de 2 entradas.

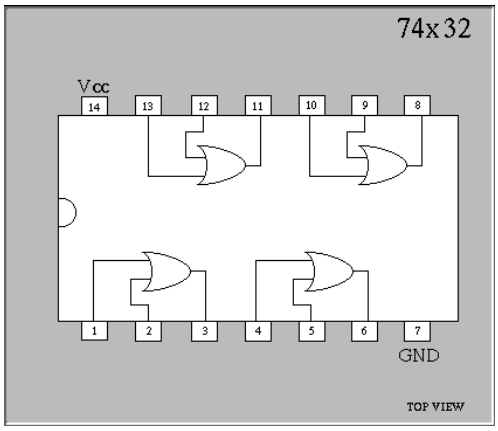


Figura 5: 7432: quatro portas OR de 2 entradas.

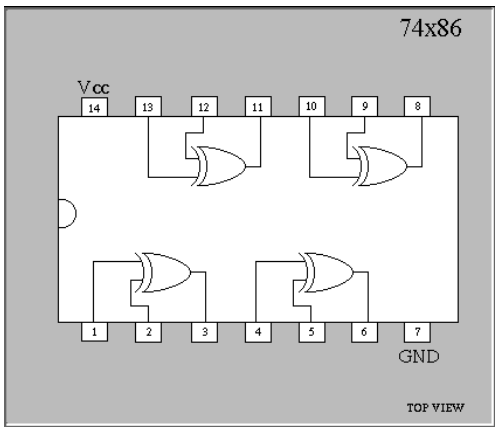


Figura 6: 7486: quatro portas XOR de 2 entradas.

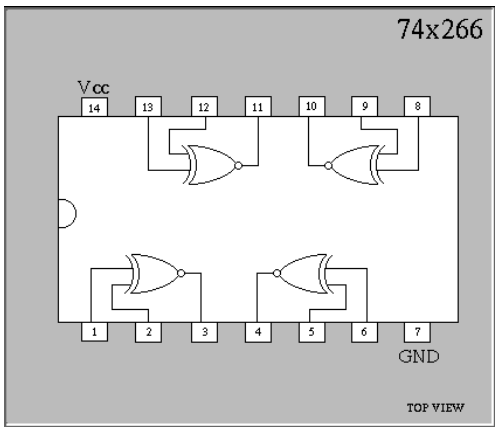


Figura 7: 74266: quatro portas XNOR de 2 entradas.

Figura 8: 7404: seis inversores.

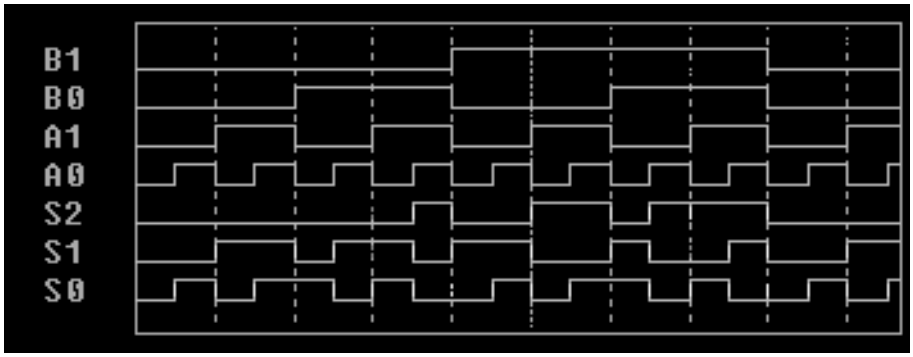


Figura 9: Resultado da simulação em OrCAD, a usar como cenário na verificação do funcionamento do circuito.

Módulo 2
Representação de sistemas digitais

Objectivos

Pretende-se que o aluno compreenda o relacionamento entre a representação por tabelas e por expressões booleanas. É igualmente objectivo deste módulo que o aluno saiba descrever problemas usando diversas representações (texto, tabelas, expressões algébricas, diagrama lógico) e que obtenha as expressões minimizadas pelo método exaustivo (algébrico).
Leituras: DDPP3, secções 4.1.6, 4.3.1 a 4.3.3.

Problemas

P1: Tabelas de verdade

Construa uma tabela de verdade que descreva o funcionamento de cada um dos seguintes circuitos:

1. Um dado motor deve ser activado, usando 3 interruptores, quando está ligado apenas um deles ou quando estão ligados simultaneamente os 3.

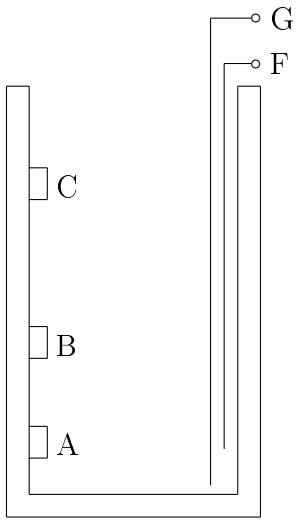
2. Numa unidade fabril, uma passadeira eléctrica é usada para recolher o material de 3 postos. Junto a cada posto está um botão que deve ser continuamente premido pelo funcionário do posto, assim que este tenha colocado material na passadeira. Esta só é activada quando os 3 botões estiverem a ser premidos. Contudo, com o intuito de avisar eventuais funcionários atrasados nas suas tarefas, quando 2 botões estiverem a ser premidos, uma luz deve ser activada, até o 3^{o}botão ser premido, altura em que a passadeira é posta em actividade.

3. Num determinado processo industrial, 3 sensores de nível condicionam o funcionamento de 2 bombas F e G, de acordo com as seguintes regras:

   - Ambas as bombas funcionam quando $\text{nível}_{\text{água}} \geq \text{nível}_{\text{C}}$.
   - A bomba F funciona e a G não, quando $\text{nível}_{\text{água}} \geq \text{nível}_{\text{B}}$ e $\text{nível}_{\text{água}} < \text{nível}_{\text{C}}$.

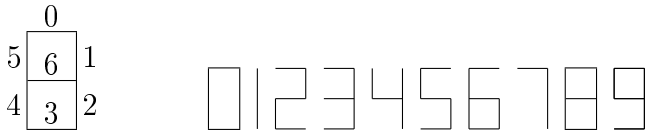
- A bomba G funciona e a F não, quando $\text{nível}_{\text{água}} \geq \text{nível}_{\text{A}}$ e $\text{nível}_{\text{água}} < \text{nível}_{\text{B}}$.
- Ambas as bombas não funcionam quando $\text{nível}_{\text{água}} < \text{nível}_{\text{A}}$.

Considere que quando os sensores detectam água, devolvem o valor lógico 1 e que as bombas funcionam com o valor lógico 1.



P2: Circuito conversor binário para 7 segmentos

Pretende-se construir um circuito que, dados 4 bits de entrada B3..B0, representando um número em binário de 0 a 9, produza as saídas C6..C0 correspondentes a um visor de 7 segmentos. Considerar o aspecto gráfico para os algarismos abaixo apresentado.



1. Desenhar o diagramas de blocos;
2. Construir a tabela com a funcionalidade do conversor;
3. Obter as expressões booleanas canónicas para as saídas **C0** e **C1**, por inspecção da tabela anterior;
4. Minimizar as expressões de **C1** e **C0**;
5. As combinações correspondentes aos números 10 a 15 não estão especificadas. Muitas vezes, há algumas combinações que não têm qualquer utilidade para o problema a resolver, i.e. não são definidas as saídas. Repetir a minimização das expressões de **C1** e **C0**, considerando que nos casos 10 a 15 as saídas são *don't care* (i.e. que podem tomar o valor '0' ou o valor '1').
6. Como trabalho para casa, minimizar as expressões de **C2** a **C6** do conversor.

P3: Simplificação de funções

Considere as seguintes funções booleanas:

- $P(A,B,C) = \overline{A}.\overline{B}.C + \overline{A}.B.\overline{C} + A.\overline{B}.\overline{C} + A.B.C$
- $Q(A,B,C) = \overline{A}.\overline{B}.\overline{C} + \overline{A}.\overline{B}.C + A.\overline{B}.\overline{C} + A.\overline{B}.C + A.B.\overline{C} + A.B.C$
- $R(A,B,C) = \sum_{A,B,C} m(1,2,3,6)$
- A função S(A,B,C) dada pela seguinte tabela

A	B	C	S
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

- $T(A,B,C,D) = \overline{A}.\overline{B}.\overline{C}.\overline{D} + \overline{A}.\overline{B}.C.\overline{D} + A.\overline{B}.\overline{C}.\overline{D} + A.\overline{B}.C.D + A.B.\overline{C}.\overline{D} + A.B.C.D$
- $U(A,B,C,D) = \sum_{A,B,C,D} m(3,4,6,9,12,13,15)$

1. Simplifique as funções usando as leis e os teoremas da Álgebra Booleana.
2. Relativamente à função P, desenhe os diagramas lógicos para a expressão na forma canónica e para a expressão na forma simplificada que obteve.

Módulo 3
Representações canónicas em SOP e POS

Objectivos

Pretende-se que o aluno aplique a Lei de De Morgan e o Teorema da Dualidade para mudar a forma das funções Booleanas de SOP para POS (e vice-versa). É igualmente objectivo deste módulo que o aluno saiba as vantagens das representações por literais e índices de SOP e POS e quais as diferenças entre a forma canónica e a forma minimizada.
Leituras: DDPP3, secção 4.1.6.

Matéria teórica

NOTA: O conteúdo desta secção não será dado na aula prática. Os alunos devem aproveitar o material aqui descrito, lendo-o antes da aula, para complementar com a matéria que foi dada nas aulas teóricas.

1. Leis de De Morgan
   - $\overline{A+B} = \overline{A}.\overline{B}$
   - $\overline{A.B} = \overline{A}+\overline{B}$
2. O dual duma expressão é obtido trocando os AND's por OR's e vice-versa, os 0's por 1's e vice-versa e deixando os literais intactos.
3. Qualquer afirmação verdadeira para uma expressão booleana, também é verdadeira para o seu dual. Se é provado um teorema para uma expressão, então o teorema para o seu dual é obtido como bónus. Por exemplo, a distributividade da conjunção em relação à disjunção ($A.(B+C) = (A.B)+(A.C)$) é verdade, logo o seu dual (a distributividade da disjunção em relação à conjunção, $A+(B.C) = (A+B).(A+C)$) também o é.
4. Formas canónicas em SOP e POS

- com literais: *e.g.* $F(A,B,C) = \overline{A}.B.C + A.\overline{B}.C$
- com índices: *e.g.* $F(A,B,C) = \sum m(3,5)$

5. Obtém-se um *mintermo* multiplicando as variáveis, porque se pretende que o termo seja 1 quando a função é 1. Consequentemente, a variável deve ser negada se tiver o valor 0 nessa linha.

6. Obtém-se um *maxtermo* adicionando as variáveis, porque se pretende que o termo seja 0 quando a função é 0. Consequentemente, a variável deve ser negada se tiver o valor 1 nessa linha.

A	B	C	mintermos	maxtermos
0	0	0	$m_0 = \overline{A}.\overline{B}.\overline{C}$	$M_0 = A + B + C$
0	0	1	$m_1 = \overline{A}.\overline{B}.C$	$M_1 = A + B + \overline{C}$
0	1	0	$m_2 = \overline{A}.B.\overline{C}$	$M_2 = A + \overline{B} + C$
0	1	1	$m_3 = \overline{A}.B.C$	$M_3 = A + \overline{B} + \overline{C}$
1	0	0	$m_4 = A.\overline{B}.\overline{C}$	$M_4 = \overline{A} + B + C$
1	0	1	$m_5 = A.\overline{B}.C$	$M_5 = \overline{A} + B + \overline{C}$
1	1	0	$m_6 = A.B.\overline{C}$	$M_6 = \overline{A} + \overline{B} + C$
1	1	1	$m_7 = A.B.C$	$M_7 = \overline{A} + \overline{B} + \overline{C}$

7. Regras para construção das formas canónicas:

   - Para construir a forma canónica usando mintermos (SOP), tem que se somar todos os mintermos para os quais a função é verdadeira. Notar que só quando as variáveis têm o padrão binário correspondente a um dos mintermos é que a função é 1.
   - Para construir a forma canónica usando maxtermos (POS), deve-se multiplicar todos os maxtermos para os quais a função é falsa.

8. Conversões entre formas canónicas:

   - **SOP para POS**: reescrever como produtório ($\prod$) dos maxtermos cujos índices não estavam presentes, pois agora, em vez dos 1's, está-se a agrupar os 0's.
   - **POS para SOP**: reescrever como somatório ($\sum$) dos mintermos cujos índices não estavam presentes, pois agora, em vez dos 0's, está-se a agrupar os 1's.
   - **Função complementar** Reescrever na mesma forma (SOP ou POS) utilizando os índices que não estavam presentes, já que $\overline{F} = 0$ para os casos em que $F = 1$ e vice-versa.

Problemas

- Negue as expressões que se seguem:

  a) $A + B.C$

  b) $(\overline{A} + B).\overline{C.D}$

c) $\overline{A}.B + \overline{A}.C.D + \overline{B}.D$

- Calcule o dual das funções que se seguem:

  a) $F = A + B$

  b) $\overline{A + B}$

  c) $H^D = (A + B).(\overline{C} + A).0$

  d) $I = (A.B + C.D).A.D.1$

  e) $A.\overline{B} = A.(\overline{A + B})$

- Escreva a seguinte função e a sua complementar nas formas SOP e POS:

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	1
1	1	0	1
1	1	1	0

- Considere a função $F_{(A,B,C,D)} = \sum m_{1,5,9,13,14,15}$

  a) Escreva-a na forma POS.

  b) Escreva a função complementar em SOP e POS.

  c) Escreva a função dual em SOP e POS.

Módulo 4
Mapas de Karnaugh

Objectivos

Pretende-se que o aluno saiba usar mapas de Karnaugh para simplificar funções Booleanas. É igualmente objectivo deste módulo que o aluno aplique o método de simplificação, na presença de condições não especificadas (don't care).
Leituras: DDPP3, secções 4.3.4 a 4.3.7.

Matéria teórica

NOTA: O conteúdo desta secção não será dado na aula prática. Os alunos devem aproveitar o material aqui descrito, lendo-o antes da aula, para complementar com a matéria que foi dada nas aulas teóricas.

- Teorema de Unidade: $A.B + A.\overline{B} = A.(B + \overline{B}) = A$;
- A essência da simplificação duma dada função consiste em, repetidamente, encontrar dois (min)termos, nos quais apenas uma variável muda o valor;
- Um mapa de Karnaugh para uma função booleana especifica valores da função para todas as combinações das variáveis de entrada (representação equivalente às tabelas);
- Num mapa de Karnaugh usam-se códigos de Gray para etiquetar as linhas e as colunas, de forma a que dois elementos vizinhos fiquem à distância de 1 (apenas uma variável muda de valor).
- O método (i.e. algoritmo) para obter uma expressão mínima (em SOP), usando mapas de Karnaugh que incluem apenas 0's e 1's, é o seguinte:

  1. Escolher um "1" do *on-set* (conjunto de todos os 1s da função). Marcar todos os grupos máximos (implicantes maiores) de 1s que incluam o "1" escolhido. Não esquecer as adjacências nas direcções horizontal e vertical. Os implicantes maiores (grupos de adjacência) contêm sempre um número de elementos igual

a uma potência inteira de 2 $(1, 2, 4, 8, \ldots)$.
Repetir este passo para cada elemento do *on-set* para obtenção de todos os implicantes maiores.

2. Visitar um elemento do *on-set*. Se ele está coberto por um único implicante maior então esse implicante é essencial e contribuirá com um termo para a soma final. Os 1's cobertos por esse implicante não necessitam de ser revisitados.
Repetir este passo até todos os implicantes essenciais estarem incluídos.
3. Se sobrarem 1's não cobertos por implicantes essenciais, seleccionar um número mínimo de maiores implicantes que os incluiam. Experimentar várias alternativas para encontrar aquela com o menor número de implicantes, sendo estes de dimensão tão grande quanto possível.

- O algoritmo para obter uma expressão mínima (em SOP), usando mapas de Karnaugh que incluem 0's, 1's e X's (*don't cares*), é o seguinte:

1. Escolher um "1". Marcar todos os grupos máximos de 1's e X's que incluiam o "1" escolhido. Não esquecer as adjacências nas direcções horizontal e vertical. Os grupos contêm sempre um número de elementos igual a uma potência inteira de 2.
Repetir este passo para todos os 1's de modo a obter todos os maiores implicantes.
2. Visitar um elemento do *on-set*. Se ele está coberto por um único maior implicante então trata-se de um implicante essencial e contribuirá com um termo para a soma final. Os 1's cobertos por implicantes não necessitam de ser revisitados.
Repetir este passo até todos os implicantes essenciais estarem incluídos.
3. Se sobrarem 1's não cobertos por implicantes essenciais, seleccionar um número mínimo de maiores implicantes que os incluiam. Experimentar várias alternativas para encontrar aquela com menos implicantes.

Problemas

Usando mapas de Karnaugh, simplificar as seguintes funções:

- As saídas C1 e C0 do conversor binário para 7 segmentos (ver problema P2 do módulo).
- $C_{out}(A, B, C_{in}) = \sum m(3, 5, 6, 7)$
- $F(A, B, C) = \sum m(0, 4, 5, 7)$
- $G(A, B, C, D) = \sum m(0, 4, 5, 6, 7, 9, 10, 11, 13, 14)$
- $H(W, X, Y, Z) = \sum m(1, 7, 11, 13) + \sum d(0, 5, 10, 15)$
- $L(V, W, X, Y, Z) = \sum m(0, 1, 2, 9, 13, 16, 18, 24, 25) + \sum d(8, 10, 17, 19)$

Módulo 5
Introdução à linguagem VHDL

Objectivos

Pretende-se que o aluno descreva, na linguagem VHDL, circuitos combinatórios simples. É igualmente objectivo deste módulo que o aluno tenha um primeiro contacto com um ambiente de simulação (CAD electrónico), a fim de avaliar se a especificação do sistema está de acordo com o esperado.
Leituras: DDPP3, secção 4.7.

Problemas

P1: Circuitos combinatórios

Descrever em VHDL, os seguintes circuitos:

1. semi-somador de 1 bit (fig. 10).
2. somador de 1 bit completo (fig. 11).
3. multiplexador 4:1 da figura 12).

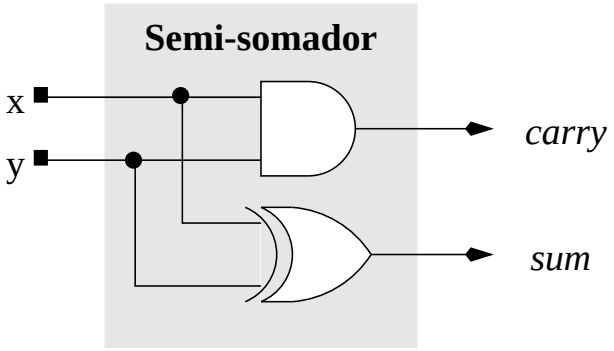


Figura 10: Diagrama do semi-somador.

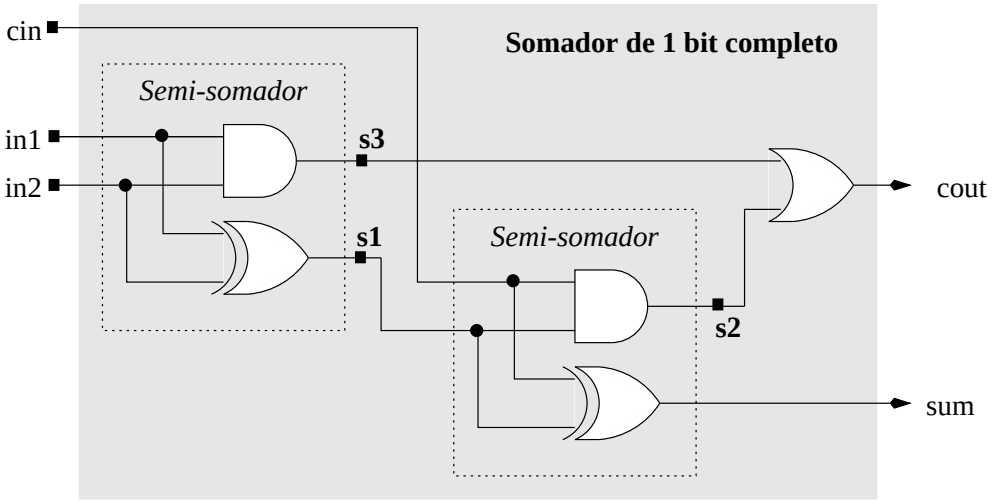


Figura 11: Diagrama do somador de 1 bit completo.

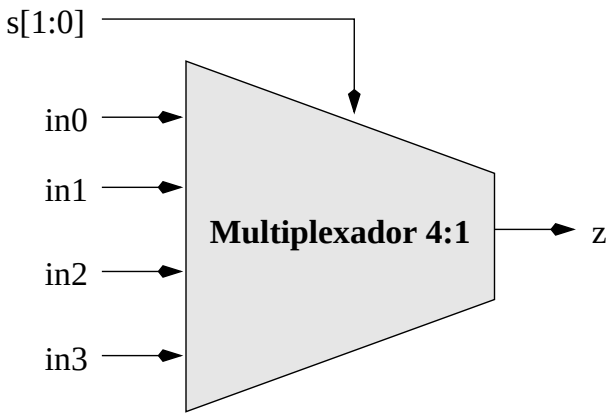


Figura 12: Diagrama do multiplexador 4:1.

P2: ALU de 1 bit

1. Descrever em VHDL a ALU (*Arithmetic and Logic Unit*) de 1 bit ilustrada na figura 13, a qual executa as operações E-lógico, OU-lógico e soma.

   O resultado gerado pela ALU é determinado pela entrada ***opcode***. Para descrever a ALU deve recorrer-se aos componentes *somador de 1 bit completo* e *multiplexador 4:1* (revisto para o caso de apenas 3 entradas) obtidos nos exercícios anteriores.

2. Simular a descrição obtida para a ALU através dum *testbench*, validando o funcionamento da ALU em cada uma das suas operações.

3. Alterar a ALU de 1 bit de modo a incluir as operações OU-lógico exclusivo, subtracção e complemento.

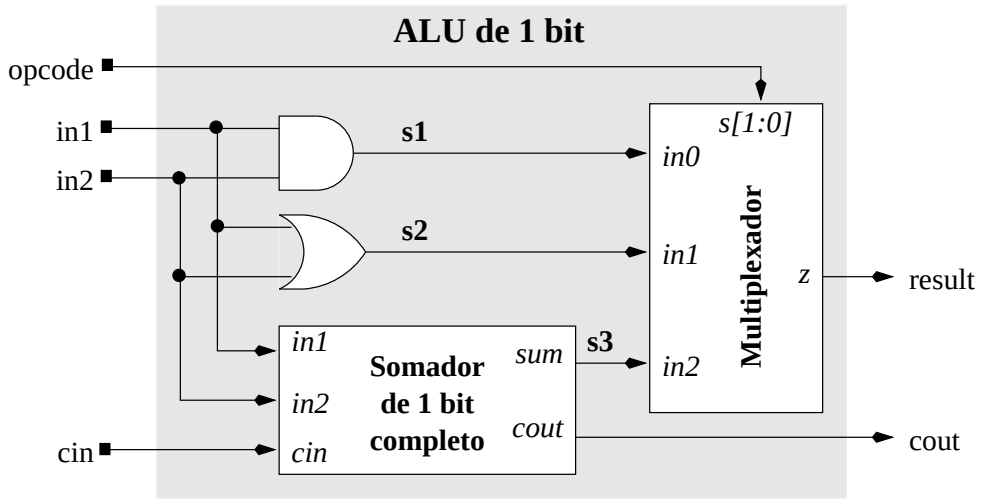


Figura 13: Diagrama da ALU de 1 bit.

Módulo 6
Descrição e simulação em VHDL: ALU do MIPS

Objectivos

Pretende-se que o aluno descreva, na linguagem VHDL, circuitos combinatórios relativamente complexos, usando, para esse efeito, alguns mecanismos da linguagem para tratar essa complexidade. Para cumprir esse objectivo, o aluno deve descrever em VHDL a ALU de 32 bits do MIPS e proceder, de seguida, à sua simulação.
Leituras: DDPP3, secção 5.10.

Sessão Laboratorial

1. Descrever em VHDL a ALU de 32 bits do MIPS. Começar com uma ALU de 1 bit e expandi-la para 32 bits. A ALU efectua as operações AND (e lógico), OR (ou lógico), ADD (soma), SUB (subtracção) e SLT *set less than* e dá apoio às instruções de salto baseadas no facto de 2 registos serem iguais ou diferentes.

2. Simular a descrição da ALU através dum *testbench*, validando o funcionamento da ALU em cada uma das 5 operações e aplicando nos operandos um conjunto relevante de valores.

O aluno deve ainda ler a secção seguinte, que descreve duma forma incremental a ALU de 32 bits do MIPS. Adicionalmente, aconselha-se a consulta dos seguintes livros, que poderão auxiliar o aluno na resolução desta sessão laboratorial:

- David A. Patterson e John L. Hennessy, "*Computer Organization and Design: the Hardware/Software Interface*", pág. 230-241, Morgan Kaufmann Publishers, 2ª edição, 1998. ISBN 1-55860-491-X.

- Sudhakar Yalamanchili, "*Introductory VHDL: from Simulation to Synthesis*", Prentice-Hall, 2001. ISBN 0-13-080982-9.

Descrição Incremental da ALU do MIPS

Para explicar a arquitectura da ALU do MIPS, utiliza-se um processo evolutivo que parte duma ALU simples de 1 bit que é refinada em várias etapas até se obter uma ALU de 32 bits baseada numa ALU de 1 bit mais elaborada que a original. Este processo é descrito em seis etapas.

1. A ALU de 1 bit inicial executa apenas as operações mais fáceis de implementar em *hardware*, ou seja, as operações lógicas OR e AND (figura 14). De acordo com o valor do sinal ***operation***, o multiplexador 2:1 selecciona qual o resultado a colocar na saída da ALU: o E-lógico ou o OU-lógico das entradas de 1 bit ***a*** e ***b***.

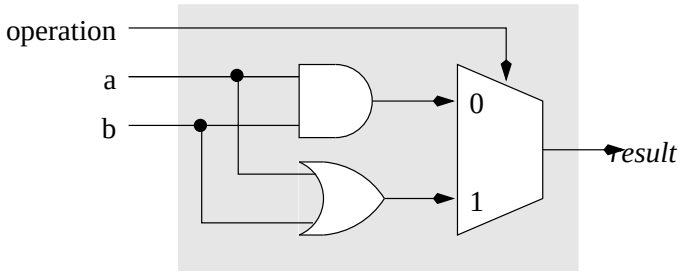


Figura 14: ALU de 1 bit que executa as operações AND e OR.

2. Ao incluir a operação de soma na ALU de 1 bit, implementada por um somador de 1 bit completo como o que se ilustra na figura 11, obtém-se a ALU da figura 15, que coincide com a que se obteve no exercício 4 do módulo nº 5. Como mostra a figura 11, ao aumentar de 2 para 3 o número de operações distintas efectuadas pela ALU, o número de entradas do multiplexador aumenta na mesma proporção. Adicionalmente, é necessário incluir os sinais *cin* e *cout*.

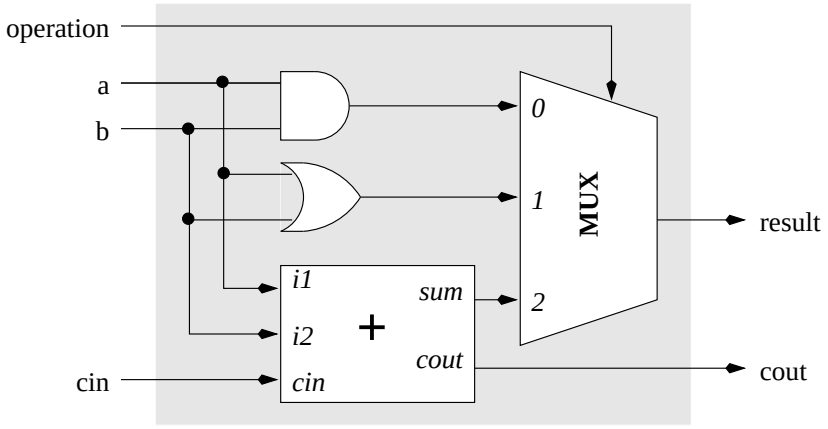


Figura 15: ALU de 1 bit que executa as operações AND, OR e soma.

3. Constrói-se uma ALU de 32 bits que executa as operações AND, OR e soma, através da ligação de 32 ALUs como a que se obteve na etapa anterior. Na ALU de 32 bits da figura 16, o sinal de transporte na saída da ALU de ordem i ($\boldsymbol{cout_i}$) é ligado ao sinal de transporte na entrada da ALU de ordem $i+1$ ($\boldsymbol{cin_{i+1}}$).

4. Altera-se a ALU de 32 bits de modo a poder efectuar subtracções de valores representados em complemento para 2. A estratégia a utilizar na implementação da

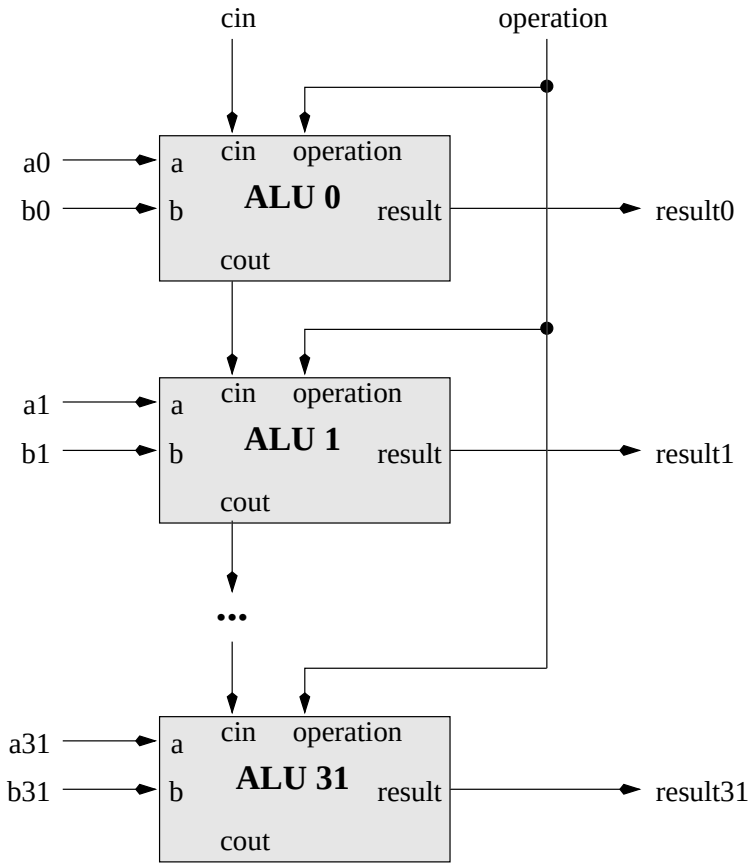


Figura 16: ALU de 32 bits que executa as operações AND, OR e soma.

subtracção entre a e b resulta de $a - b = a + \overline{b} + 1$, em que $\overline{b}$ é o complemento para 1 de b. O complemento para 1 dum número obtém-se com uma simples inversão dos bits deste número. Deste modo, para a ALU poder efectuar a soma e a subtracção de a com b, inclui-se, na ALU de 1 bit, um multiplexador 2:1 que selecciona o segundo operando da ALU entre b e $\overline{b}$. Para concretizar a subtracção na forma $a + \overline{b} + 1$ falta somar "1" ao resultado da soma entre a e $\overline{b}$. O termo "1" é introduzido, na ALU de 32 bits, através do entrada *cin* da ALU de 1 bit menos significativa (*ALU* 0), uma vez que esta é uma entrada do seu somador. A ALU de 1 bit, alterada de modo a executar as operações AND, OR, soma e subtracção, é apresentada na figura 17.

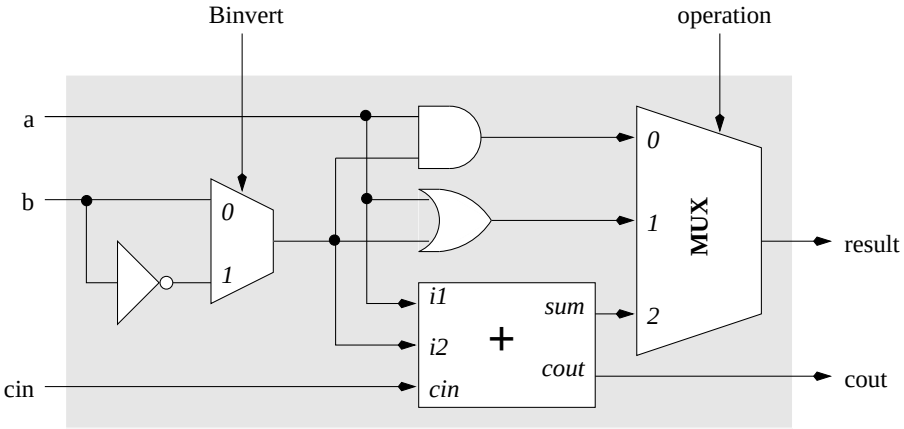


Figura 17: ALU de 1 bit que executa as operações AND, OR, soma e subtracção.

5. A ALU de 32 bits obtida na fase anterior é alterada de modo a poder executar a instrução *set less than* (SLT) do MIPS. A instrução SLT produz o resultado 00..01 se o primeiro operando for menor que o segundo e 00..00 nos outros casos. Para a ALU de 1 bit suportar a instrução SLT, é necessário introduzir mais uma entrada (4ª) no multiplexador que selecciona o resultado da ALU, obtendo-se assim a ALU da figura 18 (i). Esta entrada, designada por ***less*** na figura 18, é ligada a '0' nas ALUs que processam os bits 31 a 0. Para obter o sinal *less* na ALU menos significativa, é preciso atender a que ele será '1' quando o resultado da subtracção entre o primeiro e o segundo operandos da instrução for negativo e será '0' se o resultado da subtracção for positivo. Assim, a entrada *less* da *ALU* 0 coincide com o bit de sinal do resultado da subtracção necessária à instrução SLT, ou seja, a entrada *less* da *ALU* 0 coincide com a saída *sum* do somador da ALU mais significativa (*ALU* 31). Deste modo, a *ALU* 31 tem que ser diferente das outras, com o objectivo de disponibilizar para o seu exterior a saída *sum* do somador: saída ***set*** na figura 18 (ii).

   Com as duas versões da ALU de 1 bit incluídas na figura 18, pode construir-se a ALU de 32 bits do MIPS (figura 19).

6. Para concluir a ALU do MIPS, falta apenas incluir suporte para os saltos, em que a decisão de saltar resulta do facto de os conteúdos de 2 registos (operandos da ALU) serem ou não iguais. A comparação dos registos é implementada com uma subtracção na ALU, em que o resultado da subtracção é nulo quando os 2 operandos são iguais e é diferente de zero quando os 2 operandos são diferentes: $a - b = 0 \Leftrightarrow a = b$. Para verificar se o resultado da subtracção é ou não nulo, utiliza-se um OU-lógico negado entre os 32 bits de resultado da ALU (figura 20).

   Para efectuar a operação de subtracção (utilizada nas instruções de subtracção, *set less than* e saltos), os sinais ***cin*** e ***Binvert*** têm que ser '1', enquanto que para executar as operações AND, OR e soma exige-se que este sinais sejam '0'. Portanto, os sinais ***cin*** e ***Binvert*** são sempre iguais, podendo por isso ser substituídos por um único sinal: ***Bnegate*** na figura 20. Combinando a entrada ***Bnegate*** (bit 2) com a entrada ***operation*** (bits 1:0) obtém-se a entrada única ***operation*** com 3 bits, que indica a operação a executar pela ALU (tabela 2). A ALU do MIPS é representada pelo símbolo da figura 21.

Sinal *operation*	**Operação da ALU**
000	AND
001	OR
010	soma
110	subtracção
111	SLT

Tabela 2: Correspondência entre o valor aplicado no sinal de controlo *operation* e a operação executada pela ALU do MIPS.

Em **conclusão**, a ALU a implementar é ilustrada na figura 20 e as ALUs de 1 bit com que ela é construída encontram-se na figura 18.

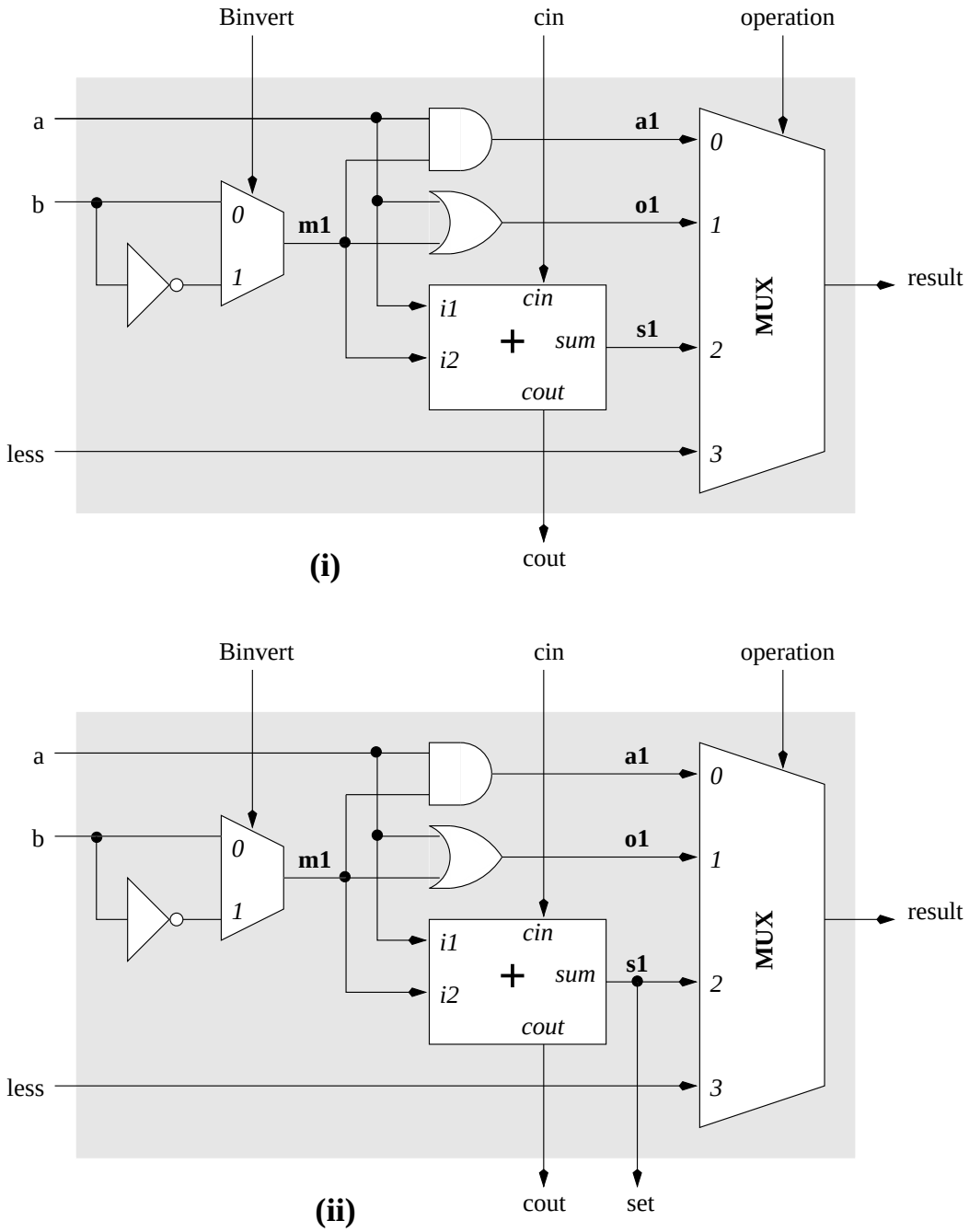


Figura 18: ALU de 1 bit do MIPS, que permite executar as operações AND, OR, soma, subtracção e *set less than*: (i) ALU para os bits 0 a 30 e (ii) ALU para o bit mais significativo (bit 31).

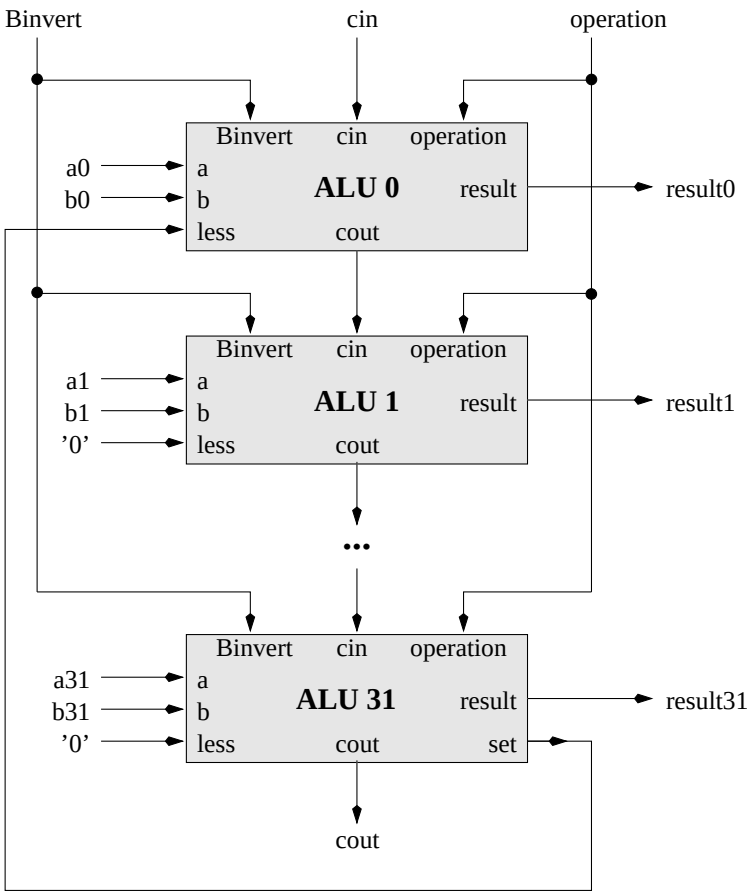


Figura 19: ALU de 32 bits do MIPS, que permite executar as operações AND, OR, soma, subtracção e *set less than*.

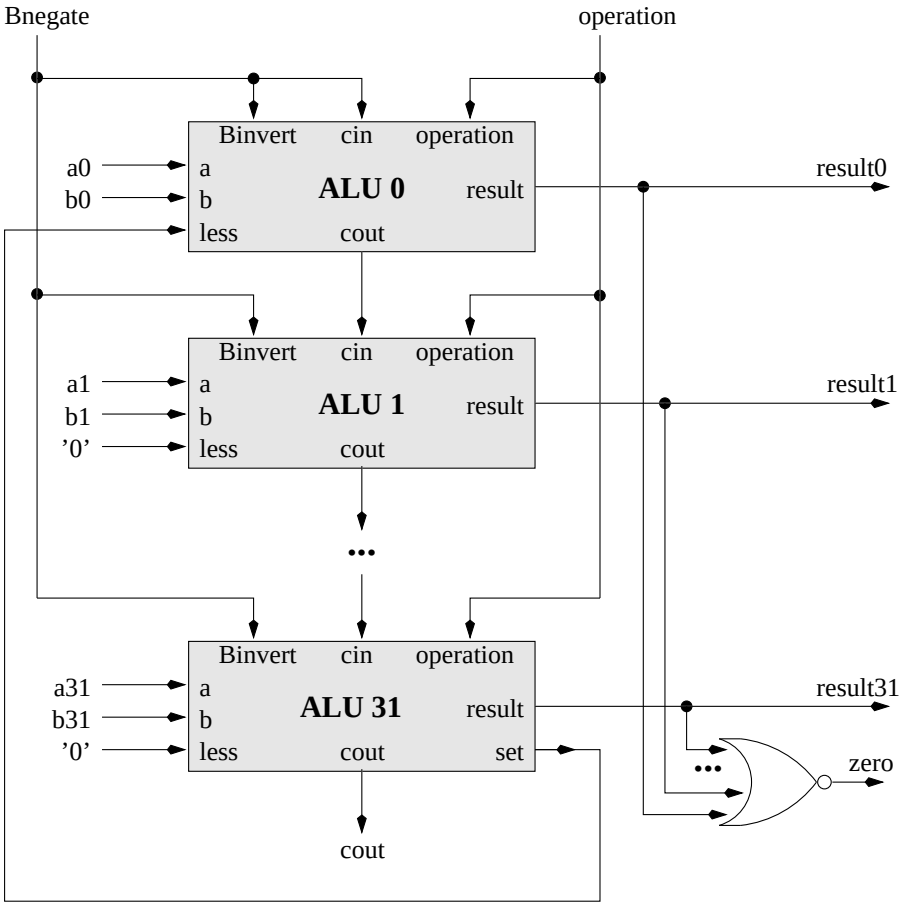


Figura 20: ALU final do MIPS, que permite executar as operações AND, OR, soma, subtracção e *set less than*.

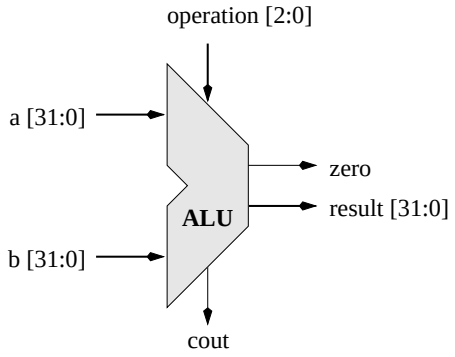


Figura 21: Símbolo para a ALU do MIPS.

Módulo 7
Sistemas sequenciais

Objectivos

Pretende-se que o aluno transforme uma descrição em língua natural do funcionamento dum circuito sequencial na respectiva máquina de estados e que consiga sintetizar manualmente circuitos sequenciais descritos por máquinas de estados. É igualmente objectivo deste módulo que o aluno saiba descrever circuitos sequenciais em VHDL para posteriormente proceder à sua simulação e à sua síntese.
Leituras: DDPP3, secções 7.4 e 7.5.

Problemas

P1: Máquina de refrescos

Pretende-se conceber o controlador duma máquina de venda de refrescos. Cada refresco custa EUR 0.30. A máquina aceita moedas de EUR 0.10 e 0.20 escudos e vende dois tipos de refrescos. Não há devolução de troco, i.e., se o cliente introduzir moedas cujo montante supera o valor do refresco, a máquina vende o refresco mas não entrega ao cliente o devido troco.

Entradas:

- M10 - moeda de EUR 0.10;
- M20 - moeda de EUR 0.20;
- R1 - cliente premiu o botão correspondente ao refresco 1;
- R2 - cliente premiu o botão correspondente ao refresco 2.

Saídas:

- P1 - fornecer refresco 1.

- P2 - fornecer refresco 2.

1. Desenhe o diagrama de blocos;
2. Desenhe o diagrama de estados, assumindo que se trata de uma máquina de Moore;
3. Reestude o problema e desenhe um novo diagrama de estados, considerando que se trata de uma máquina de Mealy;
4. Construa a tabela de estados para a máquina de Mealy;
5. Atribua uma combinação binária a cada estado;
6. Construa a tabela de verdade;
7. Construa a tabela de excitação para *flip-flops* D;
8. Utilizando mapas de Karnaugh, obtenha as expressões minimizadas para as saídas dos dois blocos de lógica combinatória que compõem a máquina de estados. Nota: procure eliminar entradas que não afectem as saídas.

Altere a solução obtida considerando as seguintes hipóteses:

1. A máquina deve agradecer, por exemplo acendendo uma luz, sempre que forem introduzidas moedas depois da quantia de EUR 0.30 ter sido atingida se ainda não foi escolhido um refresco;
2. Em alternativa ao ponto anterior, a máquina pode fechar a ranhura por onde são introduzidas as moedas, quando a quantia de EUR 0.30 for atingida. Outra hipótese, nessas circunstâncias, é fazer as moedas introduzidas, depois de atingido o preço dos refrescos, ser expelidas por uma ranhura apropriada.
3. Adicionalmente aos pontos anteriores, a máquina deve ter a possibilidade de dar troco, quando for caso disso. Pode ainda existir um botão que permite interromper a qualquer instante a intenção de compra dum refresco, devolvendo todas as moedas introduzidas até então.
4. Outra funcionalidade, que pode ser obrigatória por lei, é a máquina dar um recibo, quando o cliente o solicitar.

Fazendo uso dum ambiente de simulação em VHDL:

- Descrever a máquina de venda de refrescos em VHDL;
- Obter um *testbench* em VHDL para este sistema;
- Efectuar a simulação de modo a verificar se a máquina se comporta como é suposto.

[P2]: Máquina de lavar

Pretende-se conceber o controlador de uma máquina de lavar. A máquina inicia a sua operação quando uma moeda é introduzida. Em seguida, sequencia as seguintes fases: *soak, wash, rinse, spin*. Há um botão de "double-wash" que, quando activado, causa uma segunda sequência de *wash* e *rinse*. Há também um temporizador, podendo assumir-se que cada fase demora o mesmo tempo.

O temporizador começa a funcionar logo que a moeda é depositada e activa um sinal T no fim do período de tempo; em seguida, efectua o seu próprio *reset* e começa novamente. Se a tampa da máquina for levantada durante o ciclo de *spin*, a máquina pára até a tampa ser fechada. Pode-se supor que o temporizador fica suspenso quando a tampa é levantada.

- Identifique as entradas e saídas deste controlador (diag. blocos)
- Desenhe o diagrama de estados correspondente.
- Construa a tabela de estados
- Atribua uma combinação binária a cada estado;
- Construa a tabela de verdade;
- Construa a tabela de excitação para *flip-flops* D

[P3]: Semáforos

Duas ruas de sentido duplo tem a sua intersecção controlada pela seguinte sinalização luminosa. Na direcção Este-Oeste, as luzes seguem a sequência (verde - amarelo - vermelho). As luzes no sentido Sul efectuam o mesmo tipo de sequência e estão no Vermelho quando as luzes Este-Oeste estão no Verde ou Amarelo e vice-versa. Contudo, as luzes no sentido Norte incluem também uma seta verde para virar à esquerda. Estas luzes efectuam a seguinte sequência: (vermelho - seta verde - seta amarela - verde - amarelo - vermelho). Considere ainda os seguintes aspectos adicionais:

1. Quando a luz da seta está verde ou amarela, as luzes nas outras três direcções estão vermelhas.
2. As temporizações para as luzes no sentido Norte são as seguintes: Vermelho:60s; Seta Verde:20s; Seta Amarela:10s; Verde:45s; Amarelo:15s
3. As temporizações para as outras luzes pode ser derivadas das especificações 1 e 2.

Presuma que tem disponíveis os temporizadores necessários, e que estes podem ser carregados com as constantes apropriadas (em segundos) e que activam uma saída quando a contagem decrescente atinge o valor zero.

- Construa um diagrama temporal que ilustre o comportamento das luzes, mostrando no eixo dos XX' a temporização e no eixo dos YY' as luzes para os 4 sentidos (Este, Oeste, Norte e Sul).

- Identifique as entradas e saídas deste controlador (diag. blocos).
- Desenhe o diagrama de estados correspondente, indicando explicitamente todos os sinais de controlo (entradas e saídas) necessários para implementar o semáforo.
- Construa a tabela de estados.
- Atribua uma combinação binária a cada estado;
- Construa a tabela de verdade;
- Construa a tabela de excitação para *flip-flops* D.