Módulo 5
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: Wakerly, secção 4.7.

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 12). 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***.

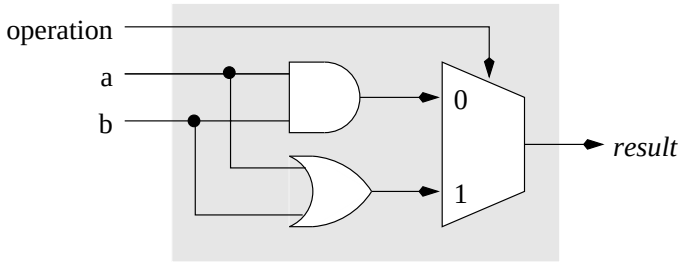

Figura 12: ALU de 1 bit que executa as operações AND e OR.

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 9, obtém-se a ALU da figura 13, que coincide com a que se obteve no exercício 4 do módulo nº 5. Como mostra a figura 9, 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*.

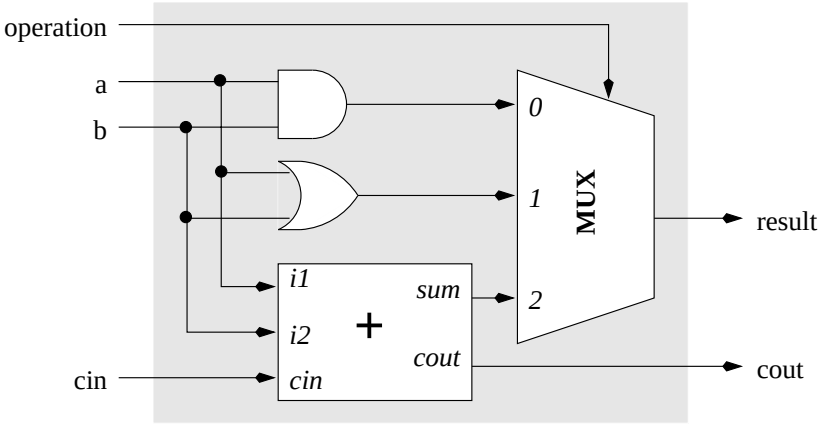

Figura 13: ALU de 1 bit que executa as operações AND, OR e soma.

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 14, 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

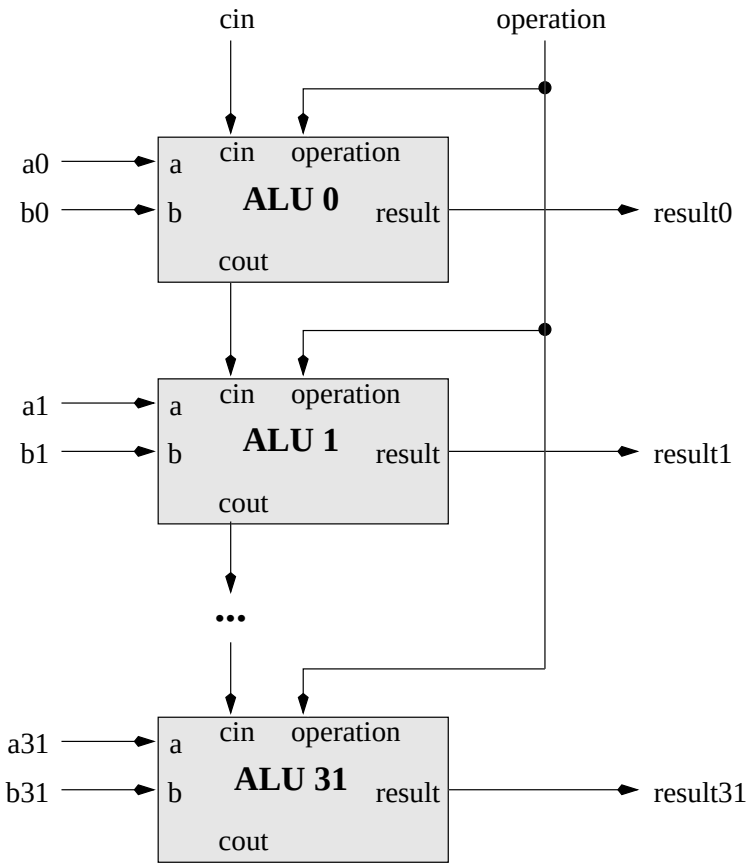

Figura 14: ALU de 32 bits que executa as operações AND, OR e soma.

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 15.

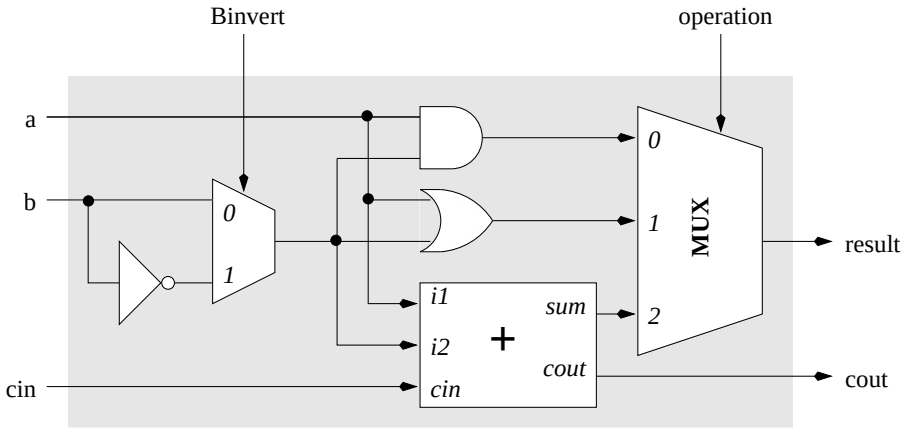

Figura 15: ALU de 1 bit que executa as operações AND, OR, soma e subtracção.

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 16 (i). Esta entrada, designada por ***less*** na figura 16, 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 16 (ii).

   Com as duas versões da ALU de 1 bit incluídas na figura 16, pode construir-se a ALU de 32 bits do MIPS (figura 17).

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 18).

   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 estes sinais sejam '0'. Portanto, os sinais ***cin*** e ***Binvert*** são sempre iguais, podendo por isso ser substituídos por um único sinal: ***Bnegate*** na figura 18. 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 1). A ALU do MIPS é representada pelo símbolo da figura 19.

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

Tabela 1: 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 18 e as ALUs de 1 bit com que ela é construída encontram-se na figura 16.

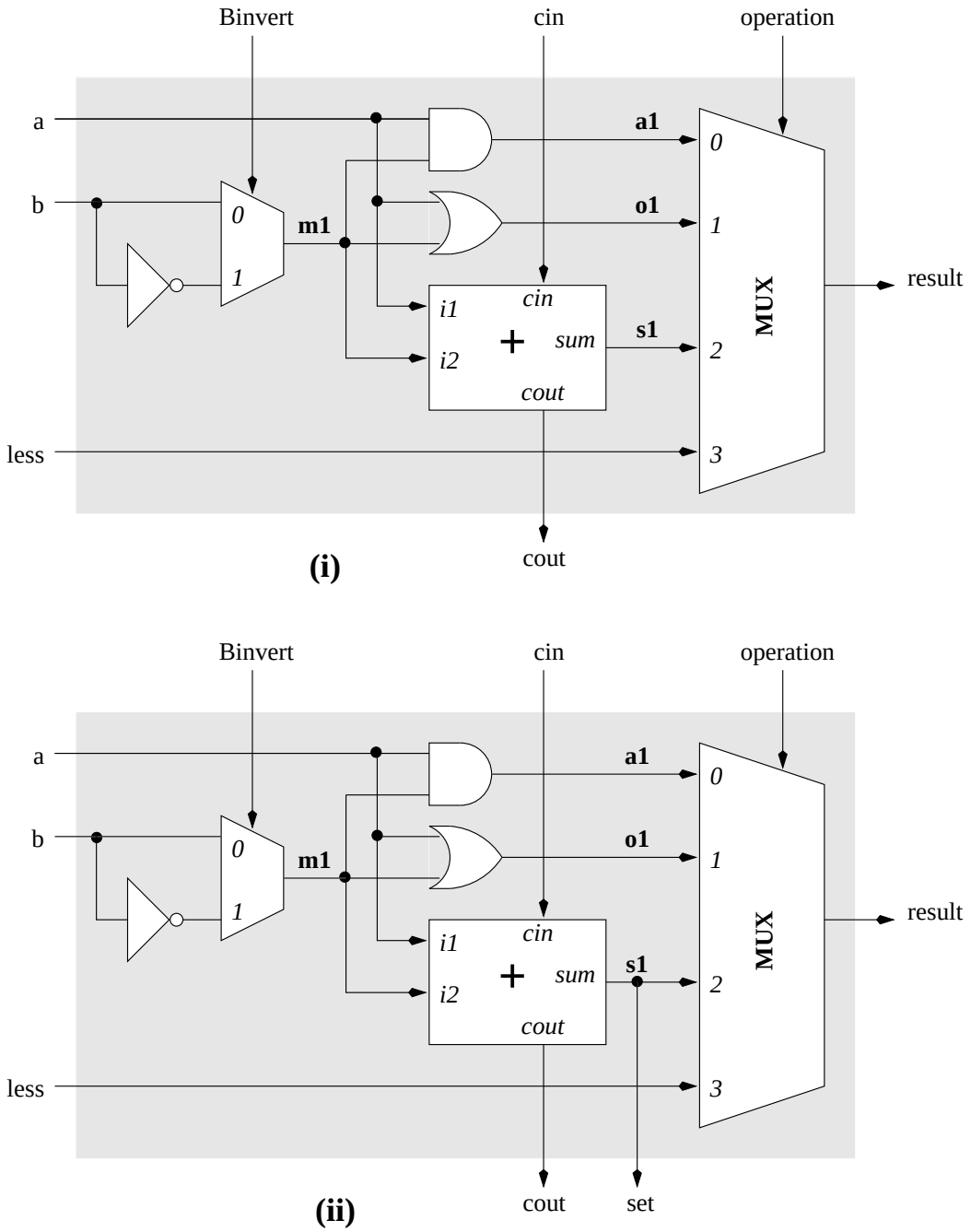

Figura 16: ALU de 1 bit do MIPS, que permite executar as operações AND, OR, soma, subtracção e *set less than*: (i) ALU para os bits 0 a 30 e (ii) ALU para o bit mais significativo (bit 31).

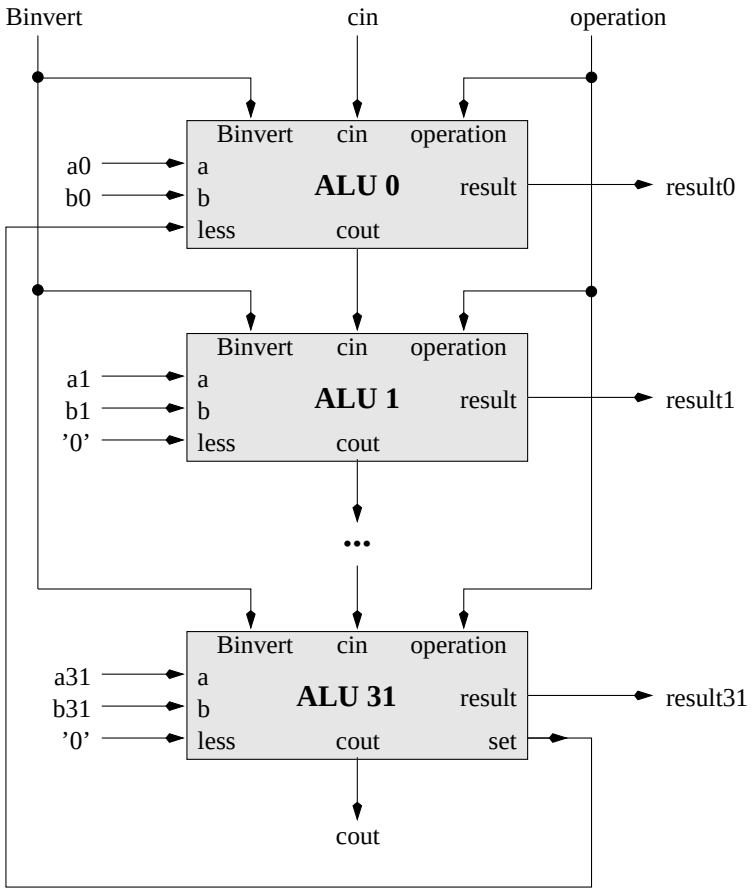

Figura 17: ALU de 32 bits do MIPS, que permite executar as operações AND, OR, soma, subtracção e *set less than*.

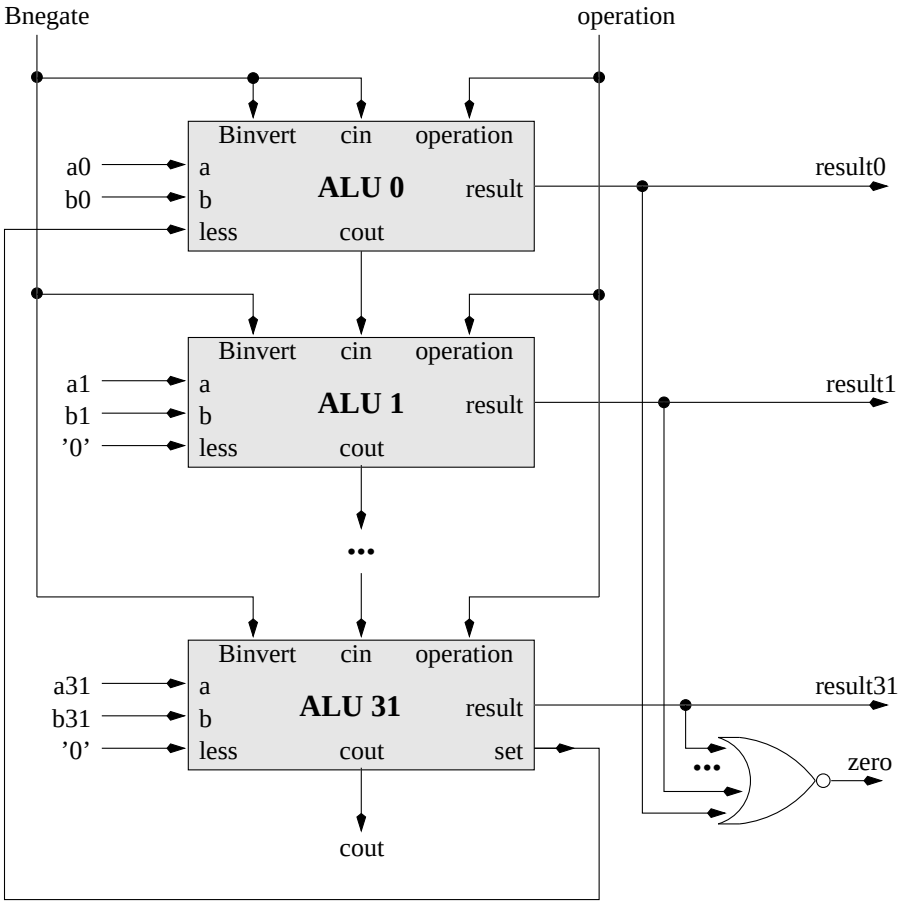

Figura 18: ALU final do MIPS, que permite executar as operações AND, OR, soma, subtracção e *set less than*.

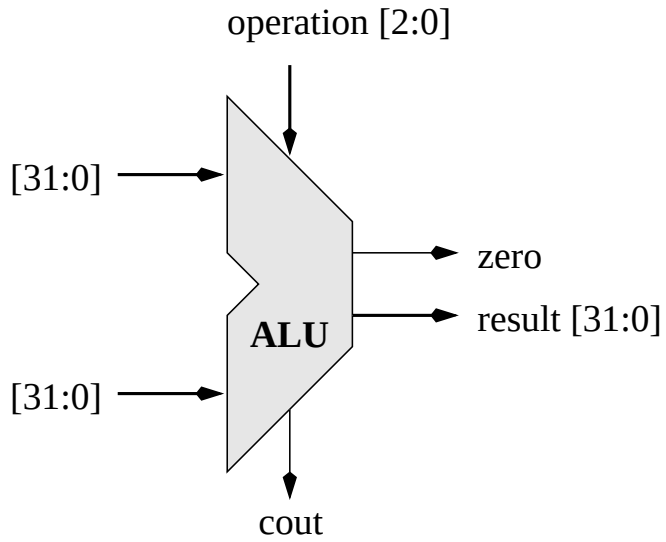

Figura 19: Símbolo para a ALU do MIPS.