Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

DCC819 - Arquitetura de Computadores

Relatório I - Unidade Lógica Aritmética

Guilherme Batista Santos
Iuri Silva Castro
João Mateus de Freitas Veneroso
Ricardo Pagoto Marinho

Belo Horizonte - MG
16 de outubro de 2017

1 Introdução

A Unidade Lógica Aritmética, ou *ULA*, é um circuito digital responsável por realizar operações lógicas e aritméticas no microprocessador, como por exemplo, executando operações de adição, subtração, deslocamentos lógicos, operações lógicas *and* e *or*. A Figura 1 abaixo mostra a forma geral de uma ULA.

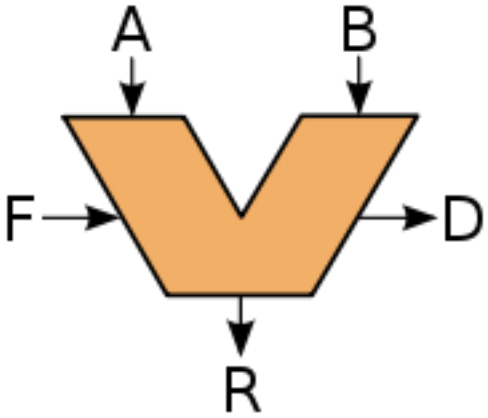

Figura 1: Unidade Lógica Aritmética.

A unidade possui duas entradas e uma saída de dados, A, B e R respectivamente, além de entradas e saídas de sinais de controle, F e D. As entradas A e B são conhecidas como *operandos*, valores em que a unidade executará a operação designada. O resultado da operação será dado em R. Os sinais entrada de controle F indicam a unidade, por exemplo, qual operação deverá ser feita, e os sinais de saída D são utilizados, por exemplo, para indicar o estado do resultado da operação como indicação de valor zero, negativo ou se houve *overflow* na operação.

Propõe-se, para esse trabalho, implementar uma Unidade Lógica Aritmética capaz de realizar operações específicas. Será utilizada a linguagem de descrição de hardware *Verilog HDL* e a plataforma *Quartus II* para a implementação. O simulador *ModelSim* será utilizado para auxiliar na verificação e testes da implementação. Testes físicos serão feitos no módulo de prototipação Altera DE2.

Futuramente, a ULA descrita neste trabalho será integrada como parte de um microprocessador.

2 Descrição

Deseja-se que o microprocessador seja capaz de executar as instruções conforme a tabela 1. As instruções são de 16-bits e trabalham com registradores de 16-bits e imediatos absolutos de 4-bits. São definidas como um código de operação de 4-bits, código do registrador de destino $s4 de 4-bits, código do registrador $s3 ou o imediato absoluto de 4-bits e o código do registrador operando $s2 de 4-bits.

Observa-se que há instruções que trabalham com operandos registro-registro e operandos registro-imediato. Tais operações não são distinguidas pela ULA, indicando apenas se um dos operandos virá do banco de registradores ou da própria instrução. A unidade precisa ser capaz de apenas executar as operações de adição, subtração, comparação, E, OU e OU-EXCLUSIVO, deixando a parte de decisão de operandos para a unidade de decodificação.

Observa-se também que para a instrução de subtração, em que a posição dos operandos altera o resultado, a subtração entre registros e a subtração entre registro-imediato, tem a forma $s4 =

Código	Instrução	Operação	Descrição
0	ADD $s4,$s3,$s2	$s4 = $s2 + $s3	Adição entre registros
1	SUB $s4,$s3,$s2	$s4 = $s3 - $s2	Subtração entre registros
2	SLTI $s4,imm,$s2	$s2 > imm ? $s4 = 1 : $s4 = 0	Comparação entre registro e imediato
3	AND $s4,$s3,$s2	$s4 = $s2 and $s3	*AND* lógico com dois registros
4	OR $s4,$s3,$s2	$s4 = $s2 or $s3	*OR* lógico com dois registros
5	XOR $s4,$s3,$s2	$s4 = $s2 xor $s3	*XOR* lógico com dois registros
6	ANDI $s4,imm,$s2	$s4 = $s2 and imm	*AND* lógico com um registro e um imediato
7	ORI $s4,imm,$s2	$s4 = $s2 or imm	*OR* lógico com um registro e um imediato
8	XORI $s4,imm,$s2	$s4 = $s2 xor imm	*XOR* lógico com um registro e um imediato
9	ADDI $s4,imm,$s2	$s4 = $s2 + imm	Adição entre registro e um imediato
10	SUBI $s4,imm,$s2	$s4 = $s2 - imm	Subtração entre registro e um imediato

Tabela 1: Descrição das instruções requisitadas.

$s3 - $s2 e $s4 = $s2 - imm. Para tornar a ULA genérica e diminuir a complexidade, optou-se pela modificação da instrução de subtração entre registros, ficando da forma $s4 = $s2 - $s3.

Sinais de indicação de características do resultado da operação são desejáveis, assim opta-se por um registro de indicadores de *zero*, *negativo* e *overflow*, conforme a tabela 2 abaixo, sendo esse registro atualizado a cada operação da unidade.

Índice	Sinal
0	*Overflow*
1	Negativo
2	Zero

Tabela 2. Sinais de saída da ULA.

Definido as instruções, parte-se para a implementação.

3 Implementação

3.1 ULA

Esta seção fala da implementação da ULA em *Verilog*. Nela, mostramos como a ULA foi implementada além de decisões de projeto tomadas.

O módulo desenvolvido possui 5 entradas e duas saídas: *OpA*, *OpB*, *Op*, *RST*, *CLK*, *Res* e *FlagReg* respectivamente.

As entradas *OpA* e *OpB* representam as entradas *A* e *B* da Figura 1. A entrada *Op* indica qual operação a ALU vai fazer (Add, Sub, etc.) e faz parte da entrada *F* na Figura 1. As saídas *Res* e *FlagReg* indicam, respectivamente o resultado da operação e o sinal de saída da ALU, na Figura 1, as saídas *R* e *D* respectivamente.

Na implementação, as entradas *OpA* e *OpB* são de *16 bits*, que, de acordo com a especificação do trabalho, é o tamanho dos registradores. Aqui, apenas o *OpB* pode ser um imediato. Neste caso, o imediato possui apenas *4 bits*, sendo necessário estender mais *12 bits*. Mais a frente será mostrado como e onde essa operação é feita. A entrada *Op* indica a operação a ser realizada e possui *4 bits*, ou seja, a ALU implementada possui um máximo de 16 operações diferentes.

As entradas *RST* e *CLK* possuem a função de reiniciar os valores de entrada, caso necessário, e dar o *clock* da máquina respectivamente. Assim, a cada subida de *clock*, uma operação é realizada.

A saída *Res* possui *16 bits*, assim como as entradas *OpA* e *OpB* já que é o resultado da operação e a saída *FlagReg* possui *3 bits*, um para cada sinal de saída.

Decidimos fazer uma ULA genérica, ou seja, não diferenciamos instruções para imediatos, deixando para outra unidade o trabalho de identificar se a instrução utilizada é com um imediato ou não. Assim, criou-se uma ULA com as operações mostradas na Tabela 3.

Código	Operação
0000	Adição
0001	Subtração
0010	Comparação
0011	*AND* lógico
0100	*OR* lógico
0101	*XOR* lógico

Tabela 3. Descrição das operações implementadas.

3.2 Banco de registradores

Para carregar e armazenar valores dos registradores, um banco de registradores foi necessário. O banco de registradores implementado consiste em 16 registradores de 16 bits cada. No início das simulações e dos testes, o banco inteiro foi inicializado com o valor 0, ou seja, todos os registradores, inicialmente, possuem o valor 0.

O módulo do banco possui como entrada os endereços de dois registradores de entrada, *AddrRegA* e *AddrRegB*, o endereço de um registrador para escrita, *AddrWriteReg*, o dado de escrita, *data*, e três sinais:

- WEN -> *Write Enable*
- RST -> *Reset*
- CLK -> *Clock*

O primeiro sinal, WEN, indica se o banco será lido ou escrito. Ele será escrito, caso WEN=1 e lido caso WEN=0. Se quisermos escrever no banco, então o endereço do registrador para escrita é utilizado e o dado é escrito naquele registrador. Se quisermos ler do banco, então utilizamos as duas saídas do banco de registradores, que são os valores que serão utilizados como entrada da ULA. A saída *RegA* se conecta à entrada *OpA* do módulo da ULA, enquanto a saída *RegB*, se conecta à entrada *OpB* do módulo da ULA. Lembrando que essa entrada da ULA pode ser um imediato, a decisão de pegar a saída do banco de registradores ou o imediato será discutida mais a frente.

3.3 Código

Para isso foi especificado que a ULA deve possuir entradas para dois operandos de 16 bits e um entrada para o código da operação a ser executada, que possui 4 bits. Para este trabalho, utilizamos apenas 16 registradores, ou seja, apenas 4 bits são necessários para fazer o endereçamento no banco de registradores.

Ou seja, a ULA implementada deve ser capaz de realizar 11 operações diferentes. Observe que é possível, além de fazer operações com registradores, realizar operações com imediatos, *i.e.*, números absolutos. Esses números devem possuir 4 bits de largura.

3.4 Prototipação

A FPGA utilizada para a prototipação possui 4 botões para que possamos inserir dados e realizar as operações, 16 *switches* para informar o valor dos dados inseridos e 8 *displays* que mostram o valor dos dados. Cada *switch* possui dois estados: *cima* e *baixo*. Quando um *switch* está no estado *cima*, ele possui valor 1, e quando está no estado *baixo*, ele possui valor 0. Desta forma, cada *switch* se comporta com 1 bit do dado.

Quando o botão de nome *KEY0* for apertado, os *displays HEX7* e *HEX6* mostram o valor no conjunto de *switches SW8* a *SW11* (4 bits) e os *displays HEX5* e *HEX4* mostram o valor no conjunto de *switches SW4* a *SW7* (4 bits). Desta forma, é possível visualizar os valores passados para a placa.

Se apertarmos o botão *KEY3*, os *switches* serão interpretados como uma instrução completa, ou seja, com código da operação e operandos de entrada e saída, sendo que o *switch SW0* é o menos significativo enquanto o *SW15* o mais significativo. Assim, cada instrução possui 16 bits, como especificado no documento. O conjunto de *switches SW0* a *SW3* indica a entrada *A* e do *switch SW4* ao *SW7*, a entrada *B* da Figura 1. Lembrando que a entrada *B* pode ser um imediato. Já o conjunto de *switches SW8* a *SW11* indica a saída do resultado (saída R na Figura 1), enquanto os *switches SW12* a *SW15* indicam a operação a ser realizada, *i.e.*, o código da operação.

A Figura 2 mostra a divisão na placa.

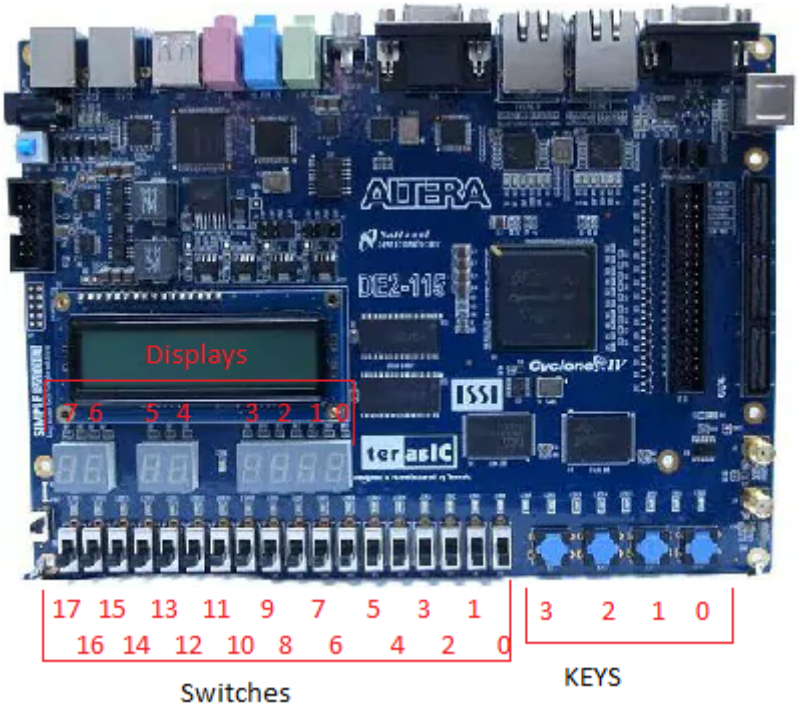

Figura 2. Módulo Altera DE2.

4 Experimentos

Alguns experimentos foram realizados com o intuito de testar as funcionalidades da Unida Lógica Aritmética implementada em Verilog HDL. Observe que a ULA não distingue operações com registradores e imediatos, uma vez que os operandos A e B são tratados pelo módulo *Decoder* e transformados em sua representação númerica de 16 bits antes de serem encaminhados à ULA. Durante o

Instrução	Operando A	Operando B	Resultado	O	N	Z
ADD	0011 1111 1111 1111	0100 0000 0000 0000	0111 1111 1111 1111	0	0	0
ADD	0100 0000 0000 0000	0100 0000 0000 0000	1000 0000 0000 0000	1	1	0
ADD	0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000	0	0	1
ADD	0010 0000 0000 0000	1100 0000 0000 0000	1110 0000 0000 0000	0	1	0
SUB	0100 0000 0000 0000	0011 1111 1111 1111	0000 0000 0000 0001	0	0	0
SUB	0011 1111 1111 1111	0100 0000 0000 0000	1111 1111 1111 1111	0	1	0
SLT	0100 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0001	0	0	0
SLT	0000 0000 0000 0000	0100 0000 0000 0000	0000 0000 0000 0000	0	0	1
AND	0111 1111 1111 1111	0011 1111 1111 1111	0011 1111 1111 1111	0	0	0
OR	0000 0000 1111 1111	1111 1111 0000 0000	1111 1111 1111 1111	0	1	0
XOR	0101 0101 0101 0101	0010 1010 1010 1010	0111 1111 1111 1111	0	0	0

Tabela 4. Experimentos.

estágio de decodificação, os valores dos registradores enderaçados pela instrução são lidos do banco de registradores e um *padding* de 12 bits é concatenado ao valor dos operandos imediatos. Dessa forma, a ULA precisa executar apenas seis tipos de instruções: ADD, SUB, SLT, AND, OR e XOR.

A tabela 4 descreve os resultados de onze instruções e o valor das *flags* da ULA após o término de cada operação. As colunas O, N e Z se referem respectivamente às flags *Overflow*, *Negative* e *Zero*. Os experimentos também estão descritos no formato de um *Test Bench* implementado no arquivo "*ULA.v*".

Referências

[1] John L. Hennessy I. and David A. Patterson. *Computer Architecture: a Quantitative Approach (Fifth Edition)*. 2012.