

Trabalho 2 de OAC

ULA e Gerador de Imediatos



Alunos:

Gabriel Matheus - 17/0103498

Guilherme Braga - 17/0162290

Victor Castor - 17/0115127

Professor: Marcelo Grandi Mandelli

Entrega: 10/10/2019

1- Marco Teórico: ULA e Gerador de Imediatos

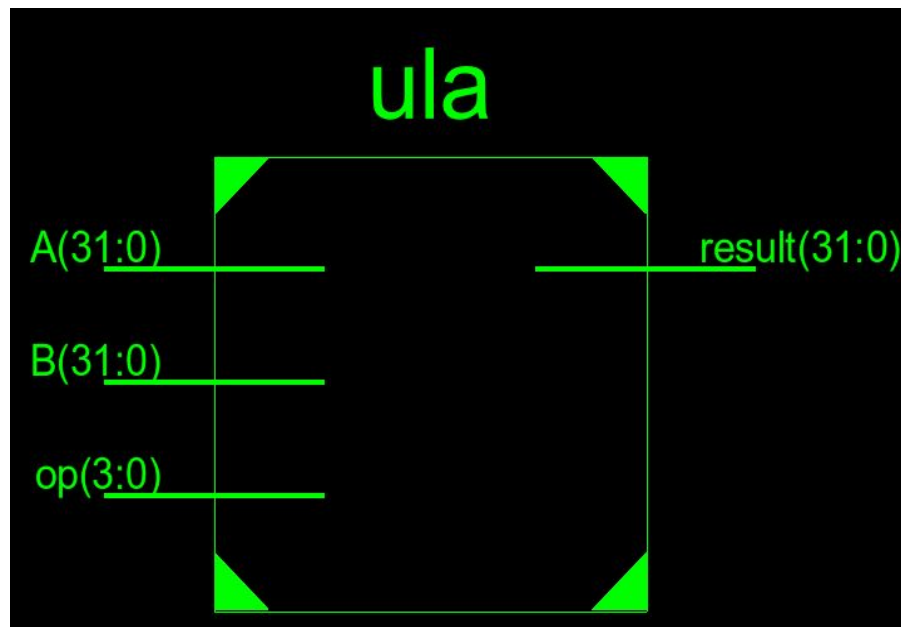
1.1- ULA

A unidade lógica e aritmética (proposta pelo matemático John von Neumann), em inglês Arithmetic Logic Unit (ALU), é um circuito digital que realiza um conjunto predefinido de operações lógicas, dado um input. É considerada a mais simples unidade de processamento central. A ULA aqui proposta segue a seguinte estrutura, utilizando funções prontas da linguagem padrão do VHDL:

Operação	Significado	Op code
ADD	$A + B$	0000
SUB	$A - B$	0001
SLL	A deslocado B bits para a esquerda	0010
SLT	$A < B$? (com sinal)	0011
SLTU	$A < B$? (sem sinal)	0100
XOR	$A \text{ XOR } B$	0101
SRL	A deslocado B bits para a direita sem sinal	0110

SRA	A deslocado B bits para a direita com sinal	0111
OR	A OR B	1000
AND	A AND B	1001
SEQ	A == B?	1010
SNE	A != B?	1011
SGE	A >= B? (com sinal)	1100
SGEU	A >= B? (sem sinal)	1101

Após a execução do código, teremos um módulo com esta estrutura:



1.2 - Gerador de Imediatos

RISC-V é um conjunto de instruções (ISA) aberto e baseado em RISC ("Reduced Instruction Set Computing"). Para este trabalho iremos considerar a geração de instruções de 32 bits, onde, dependendo da instrução, será necessária a extração de uma constante, o imediato. Ou seja, este módulo em VHDL gerará o imediato para alguns tipos de instruções.

Após a execução em VHDL, teremos um módulo com esta estrutura:



2- Procedimento

2.1- ULA

O código da ULA se concentra em duas partes: a definição de sinais e a designação destes sinais e de operações “inline”. Para operações de comparação, o retorno será uma informação do tipo booleana. A partir desta informação, designamos para a resposta um vetor composto unicamente de zeros (para indicar o sinal negativo) ou um vetor com o 1 em sua posição inicial (para indicar o sinal positivo).

Por sua importância, destacamos o seguinte trecho de código. Este serve para efetivamente designar os sinais desejados caso o opcode seja correspondido:

```
result <= std_logic_vector(signed(A) + signed(B))
std_logic_vector(signed(A) - signed(B))
std_logic_vector(shift_left(unsigned(A), to_integer(unsigned(B))))
sinal_A_menor_com_sinal
sinal_A_menor_sem_sinal
std_logic_vector(unsigned(A) xor unsigned(B))
std_logic_vector(shift_right(unsigned(A), to_integer(unsigned(B))))
std_logic_vector(shift_right(signed(A), to_integer(unsigned(B))))
std_logic_vector(A or B)
std_logic_vector(A and B)
sinal_A_igual_B
sinal_A_dif_B
sinal_A_maiorigual_B_com_sinal
sinal_A_maiorigual_B_sem_sinal
sinal_default;
end ula;
```

```
when op="0000" else
when op="0001" else
when op="0010" else
when op="0011" else
when op="0100" else
when op="0101" else
when op="0110" else
when op="0111" else
when op="1000" else
when op="1001" else
when op="1010" else
when op="1011" else
when op="1100" else
when op="1101" else
```

2.2- Gerador de Imediatos

Para o gerador de immediatos, separamos a resposta necessária em pré-determinada ordem para poder definir o sinal correto, nos valendo dos recursos de concatenação do VHDL. Sinais auxiliares foram criados para a fácil interpretação do código.

Por sua importância, destacamos o seguinte trecho de código. Este serve para efetivamente designar os sinais desejados caso o opcode seja correspondido:

```

imm32 <= (aux & inst(31 downto 20))
      (aux & inst(31 downto 20))
      (aux & inst(31 downto 20))
      (aux & inst(31 downto 25) & inst(11 downto 7))
      (aux & inst(7) & inst(30 downto 25) & inst(11 downto 8) & '0')
      (inst(31 downto 12) & "000000000000")
      (inst(31 downto 12) & "000000000000")
      (aux_j & inst(19 downto 12) & inst(20) & inst(30 downto 21) & '0')
      default;

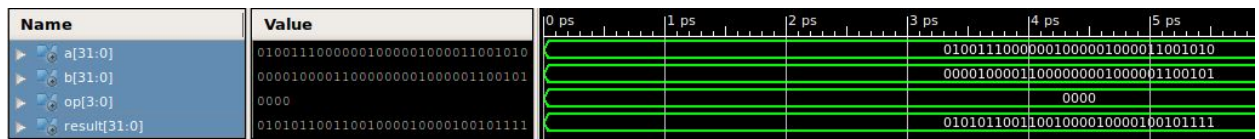
when opcode = "0000011" else
when opcode = "0010011" else
when opcode = "1100111" else
when opcode = "0100011" else
when opcode = "1100011" else
when opcode = "0010111" else
when opcode = "0101111" else
when opcode = "1101111" else

```

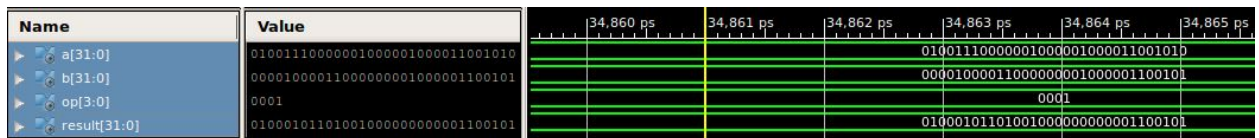
3- Resultados

3.1- Para a ULA:

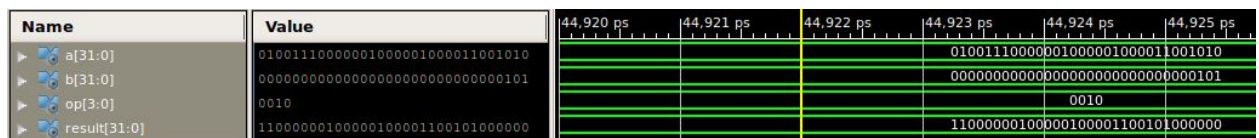
- A+B



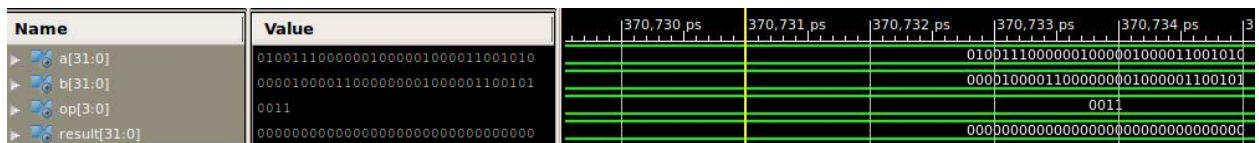
- A-B



- SLL (A deslocado à esquerda "B" vezes)



- SLT (A<B com sinal)



- SLTU (A<B sem sinal)

Name	Value	385.370 ps	385.371 ps	385.372 ps	385.373 ps	385.374 ps	385.375 ps
a[31:0]	01001110000001000001000011001010				01001110000001000001000011001010		
b[31:0]	00001000011000000001000001100101				00001000011000000001000001100101		
op[3:0]	0100				0100		
result[31:0]	00000000000000000000000000000000				00000000000000000000000000000000		

- A XOR B

Name	Value	402.440 ps	402.441 ps	402.442 ps	402.443 ps	402.444 ps	402.445 ps
a[31:0]	01001110000001000001000011001010				01001110000001000001000011001010		
b[31:0]	00001000011000000001000001100101				00001000011000000001000001100101		
op[3:0]	0101				0101		
result[31:0]	01000110011001000000000010101111				01000110011001000000000010101111		

- SRL (A deslocado à direita “B” vezes sem sinal)

Name	Value	424.390 ps	424.391 ps	424.392 ps	424.393 ps	424.394 ps	424.395 ps
a[31:0]	01001110000001000001000011001010				01001110000001000001000011001010		
b[31:0]	00000000000000000000000000000101				00000000000000000000000000000101		
op[3:0]	0110				0110		
result[31:0]	00000010011100000010000010000110				00000010011100000010000010000110		

- SRA (A deslocado à direita “B” vezes com sinal)

Name	Value	742.240 ps	742.241 ps	742.242 ps	742.243 ps	742.244 ps	742.245 ps
a[31:0]	01001110000001000001000011001010				01001110000001000001000011001010		
b[31:0]	00000000000000000000000000000000				00000000000000000000000000000000		
op[3:0]	0111				0111		
result[31:0]	00000010011100000010000010000110				00000010011100000010000010000110		

- A OR B

Name	Value	471.950 ps	471.951 ps	471.952 ps	471.953 ps	471.954 ps	471.955 ps
a[31:0]	01001110000001000001000011001010				01001110000001000001000011001010		
b[31:0]	00001000011000000001000001100101				00001000011000000001000001100101		
op[3:0]	1000				1000		
result[31:0]	010011100110010000001000011101111				010011100110010000001000011101111		

- A AND B

Name	Value	492.680 ps	492.681 ps	492.682 ps	492.683 ps	492.684 ps	492.685 ps
a[31:0]	01001110000001000001000011001010				01001110000001000001000011001010		
b[31:0]	00001000011000000001000001100101				00001000011000000001000001100101		
op[3:0]	1001				1001		
result[31:0]	00001000000000000001000001000000				00001000000000000001000001000000		

- SEQ (comparação A == B)

Name	Value	503,660 ps	503,661 ps	503,662 ps	503,663 ps	503,664 ps	503,665 ps
a[31:0]	01001110000001000001000011001010				01001110000001000001000011001010		
b[31:0]	01001110000001000001000011001010				01001110000001000001000011001010		
op[3:0]	1010				1010		
result[31:0]	00000000000000000000000000000001				00000000000000000000000000000001		

- SNE (comparação A != B)

Name	Value	529,270 ps	529,271 ps	529,272 ps	529,273 ps	529,274 ps	529,275 ps
a[31:0]	01001110000001000001000011001010				01001110000001000001000011001010		
b[31:0]	000010000110000000001000001100101				000010000110000000001000001100101		
op[3:0]	1011				1011		
result[31:0]	00000000000000000000000000000001				00000000000000000000000000000001		

- SGE (A >= B com sinal)

Name	Value	546,340 ps	546,341 ps	546,342 ps	546,343 ps	546,344 ps	546,345 ps
a[31:0]	01001110000001000001000011001010				01001110000001000001000011001010		
b[31:0]	000010000110000000001000001100101				000010000110000000001000001100101		
op[3:0]	1100				1100		
result[31:0]	00000000000000000000000000000001				00000000000000000000000000000001		

- SGEU (A >= B sem sinal)

Name	Value	568,290 ps	568,291 ps	568,292 ps	568,293 ps	568,294 ps	568,295 ps
a[31:0]	01001110000001000001000011001010				01001110000001000001000011001010		
b[31:0]	000010000110000000001000001100101				000010000110000000001000001100101		
op[3:0]	1101				1101		
result[31:0]	00000000000000000000000000000001				00000000000000000000000000000001		

- SINAL DEFAULT

Name	Value	590,240 ps	590,241 ps	590,242 ps	590,243 ps	590,244 ps	590,245 ps
a[31:0]	01001110000001000001000011001010				01001110000001000001000011001010		
b[31:0]	000010000110000000001000001100101				000010000110000000001000001100101		
op[3:0]	1111				1111		
result[31:0]	11111111111111111111111111111111				11111111111111111111111111111111		

3.2 - Para o gerador de imediatos:

- Instrução tipo R

RISC-V: add t0, zero, zero

Instrução: 0x000002B3

Imediato: 0x00000000

- **Instrução tipo I**

- Instrução tipo S

- Instrução tipo B

- Instrução tipo U

Name	Value	90,000 ps	90,001 ps	90,002 ps	90,003 ps	90,004 ps	90,005 ps
inst[31:0]	0000000000000000000010010000110111					0000000000000000000010010000110111	
imm32[31:0]	0000000000000000000010000000000000					0000000000000000000010000000000000	

- Instrução tipo J

RISC-V: jal rot

Instrução: 0x00C000EF

Imediato: 12

Name	Value	103,330 ps	103,331 ps	103,332 ps	103,333 ps	103,334 ps	103,335 ps
▶ Inst[31:0]	00000000110000000000000011101111					00000000110000000000000011101111	
▶ imm32[31:0]	00000000000000000000000000001100					00000000000000000000000000001100	

4- Link para o trabalho no Github:

https://github.com/therealguib545/OAC_Trabalho2