PODER EXECUTIVO
MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE RORAIMA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES

RELATÓRIO DO PROJETO: PROCESSADOR TITANIC 2

ALUNOS:

Victor Barbosa Rocha - 2016000542
Francisco Nascimento -

Dezembro de 2018
Boa Vista/Roraima

PODER EXECUTIVO
MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE RORAIMA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES

RELATÓRIO DO PROJETO: PROCESSADOR TITANIC 2

Dezembro de 2018
Boa Vista/Roraima

Resumo

Este trabalho aborda o projeto e implementação de um processador MIPS de 16 bits em VHDL usando o programa Quartus como IDE e simulador.

Conteúdo

Lista de Figuras

Lista de Tabelas

1 Especificação

Nesta seção é apresentado o conjunto de itens para o desenvolvimento do processador TITANIC 2, bem como a descrição detalhada de cada etapa da construção do processador.

1.1 Plataforma de desenvolvimento

Para a implementação do processador TITANIC 2 foi utilizado a IDE Quartus Prime

Flow Status	Successful - Mon Aug 27 17:33:50 2012
Quartus II Version	9.0 Build 184 04/29/2009 SP 1 SJ Web Edition
Revision Name	Quantum
Top-level Entity Name	Quantum_Pro
Family	Cyclone III
Met timing requirements	N/A
Total logic elements	3,123 / 15,408 (20 %)
Total combinational functions	2,148 / 15,408 (14 %)
Dedicated logic registers	2,137 / 15,408 (14 %)
Total registers	2137
Total pins	188 / 347 (54 %)
Total virtual pins	0
Total memory bits	0 / 516,096 (0 %)
Embedded Multiplier 9-bit elements	1 / 112 (< 1 %)
Total PLLs	0 / 4 (0 %)
Device	EP3C16F484C6
Timing Models	Final

Figura 1 - Especificações no Quartus

1.2 Conjunto de instruções

O processador Titanic 2 possui 3 registradores: RG1, RG2, RG3. Assim como 12 formatos de instruções de 16 bits cada, Instruções do **tipo R e I**, seguem algumas considerações sobre as estruturas contidas nas instruções:

- **Opcode**: a operação básica a ser executada pelo processador, tradicionalmente chamado de código de operação;
- **RG1**: é o registrador de destino, de escrita;
- **RG2**: o primeiro registrador de leitura.
- **RG3**: o segundo registrador de leitura

Tipo de Instruções:

- Formato do tipo R: Este formatado aborda instruções aritméticas (Soma, Subtração...). Utilizarão quase todos os componentes do processador, exceto a memória de dados.

OPCODE	RG1	RG2	RG3
4 bits	4 bits	4 bits	4 bits
0-3	4-7	8-11	12-15

Formato do tipo I: Este formatado aborda instruções de Load (exceto *load Immediately*), Store, Branch equal e Branch not equal.

Formato para escrita de código na linguagem Titanic 2:

OPCODE	RG1	RG2	OFFSET
4 bits	4 bits	4 bits	4 bits
0-3	4-7	8-11	12-15

Visão geral das instruções do Processador Titanic 2:

O número de bits do campo **Opcode** das instruções é igual a quatro, sendo assim obtemos um total $(Bit(0e1)^{NumeroTodaldeBitsdoOpcode} \therefore 2^4 = 16)$ de 16 **Opcodes (0-15)** que são distribuídos entre as instruções, assim como é apresentado na Tabela 1.

Tabela 1 – Tabela que mostra a lista de Opcodes utilizadas pelo processador Titanic 2.

Opcode	Nome	Formato	Breve Descrição	Exemplo
0000	ADD	R	SOMA	$S1 = $S2 + $S3
0001	SUB	R	SUBTRAÇÃO	$S1 = $S2 - $S3
0010	SLT	R	SET LESS THAN	($S1 < $S2) $S1 = 1
0011	AND	R	AND	$S1 & $S2
0100	OR	R	OR	$S1 || $S2
0101	MUL	I	MULTIPLICAÇÃO	$S1 = $S2 x $S3
0110	LW	I	LOAD WORD	$S1 = i($S2)
0111	SW	I	STORE WORD	$S1 = i($sp)
1000	BEQ	I	BRANCH EQUAL	$S1 == $S2 (ADR)
1001	BNE	I	BRANCH NOT EQUAL	$S1 != $S2 (ADR)
1010	JUMP	J	JUMP	$ra

1.3 Descrição do Hardware

Nesta seção são descritos os componentes do hardware que compõem o processador Titanic 2, incluindo uma descrição de suas funcionalidades, valores de entrada e saída.

1.3.1 ALU ou ULA

Onde são feitaa as operações aritméticas.

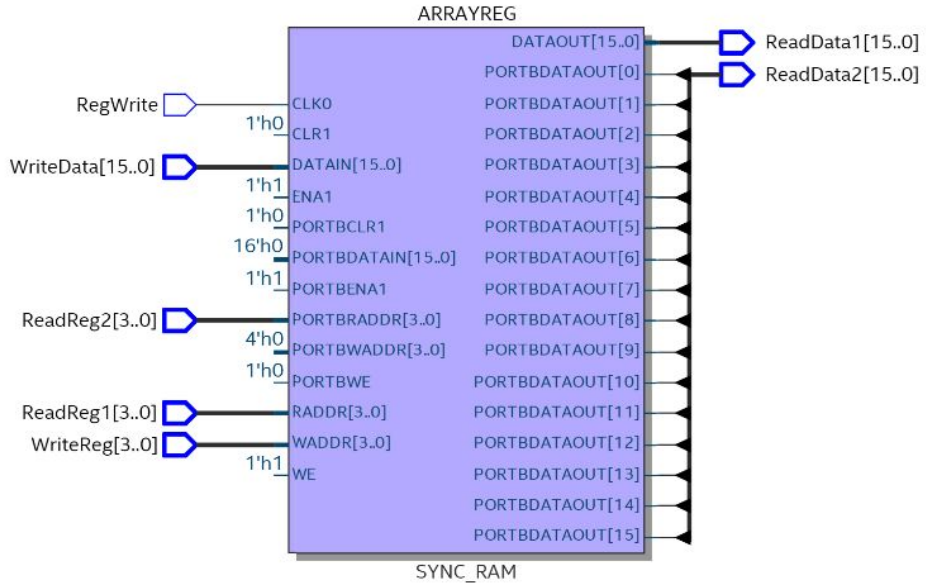
Figura 2 - Bloco simbólico do componente QALU gerado pelo Quartus

1.3.2 REGBANK16 (Banco de Registrador)

Representa o Banco de registrador e tem como componentes de entrada e saída:

```
ReadReg1      :      in      std_logic_vector(3 downto 0);
ReadReg2      :      in      std_logic_vector(3 downto 0);
WriteReg      :      in      std_logic_vector(3 downto 0);
WriteData:    in     std_logic_vector(15 downto 0);
RegWrite      :      in              std_logic;
ReadData1:    out    std_logic_vector(15 downto 0);
ReadData2:    out    std_logic_vector(15 downto 0)
```

- ReadReg1, ReadReg2 e WriteReg são os registradores RG2, RG3 e RG1 respectivamente.
- O dado a ser escrito está em WriteData.
- Se RegWrite for igual a 1, o conteúdo de WriteData é escrito em WriteReg.
- ReadData1 e ReadData2 recebem o valor de ReadReg1 e ReadReg2 para encaminhar para a ULA.



1.3.3 Controle

O componente Control tem como objetivo realizar o controle de todos os componentes do processador de acordo com o opcode ... Esse controle é feito através das flags de saída abaixo:

- **DesvioCondicional,**
- **MemParaReg**
- **read_or_write**
- **ulaFont**

Abaixo segue a tabela, onde é feita a associação entre os opcodes e as flags de controle:

Tabela 2 - Detalhes das flags de controle do processador.

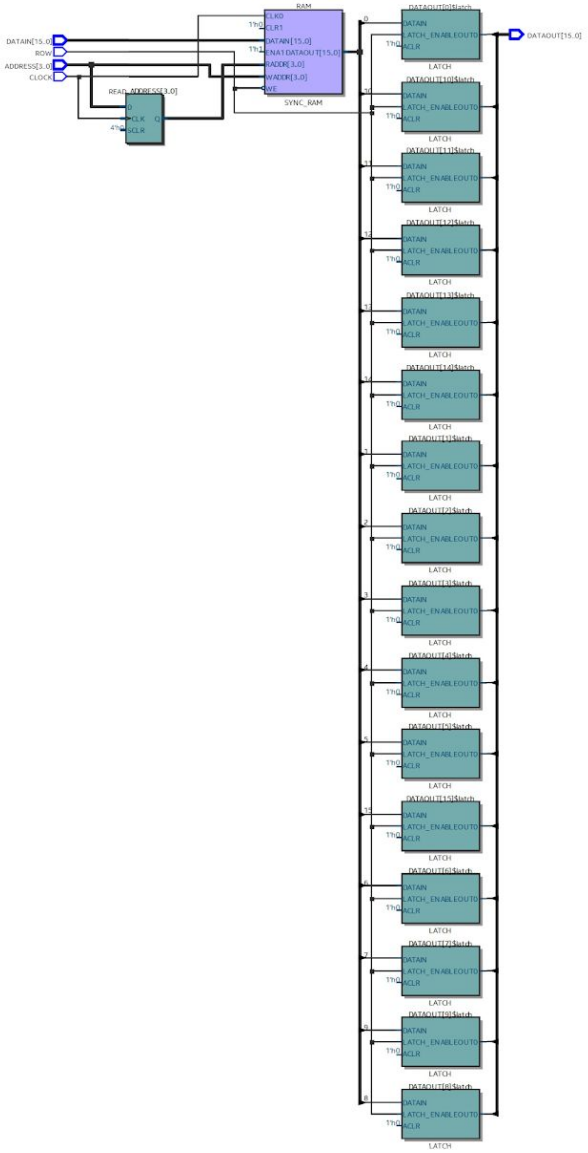
Comando	Desvio Condicional	Mem Para Reg	read or write	ula Font	ula opcode
ADD	0	0	1	0	0000
SUB	0	0	1	0	0001
SLT	0	0	1	0	0010
AND	0	0	1	0	0011
OR	0	0	1	0	0100
MUL	0	0	1	0	0101
LW	0	1	1	1	1001
SW	0	X	0	1	1010
BEQ	1	X	0	0	1011
BNE	1	X	0	0	1100

1.3.4 RAM16 (Memória de dados)

```
CLOCK   :       IN      STD_LOGIC;
ROW     :       IN      STD_LOGIC;
ADDRESS:        IN      STD_LOGIC_VECTOR (3 DOWNTO 0);  --ENDEREÇO PARA ROW
DATAIN: IN      STD_LOGIC_VECTOR (15 DOWNTO 0);         --DADO A SER ESCRITO
DATAOUT:        OUT     STD_LOGIC_VECTOR (15 DOWNTO 0)  --DADO A SER LIDO
```

- ROW vai indicar se vamos estar lendo ou escrevendo na memória de dados.
- ADDRESS é o endereço de 4 bits onde vamos acessar o array da RAM para ou escrever(store) os dados do DATAIN ou carregar (load) os dados desse endereço na saída DATAOUT.

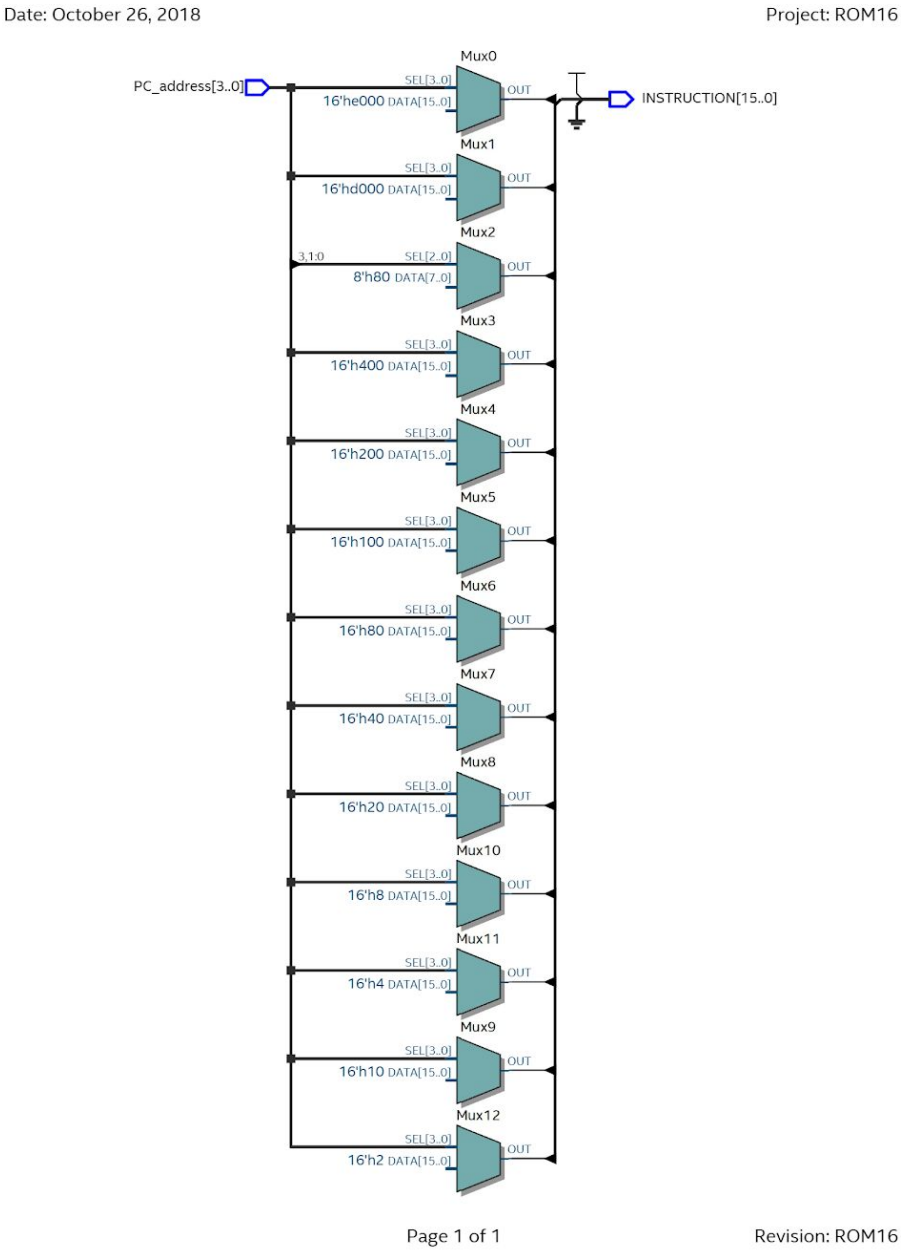
Date: October 26, 2018 Project: RAM16



1.3.5 ROM16 (Memória de Instruções)

```
PC_address:     IN STD_LOGIC_VECTOR(3 DOWNTO 0);
INSTRUCTION:    OUT STD_LOGIC_VECTOR(15 DOWNTO 0)
```

- O valor do Contador acessa um endereço de um array e vai incrementando nele para realizar as instruções.

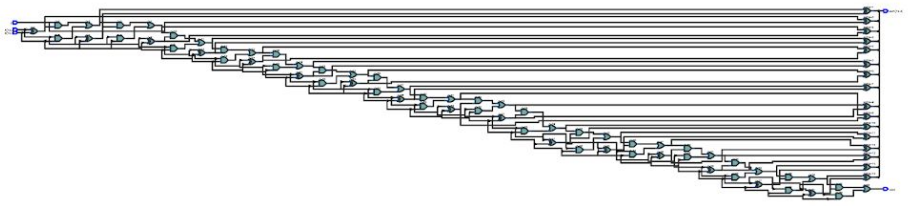


1.3.6 Somador

```
A, B    : in    std_ulogic_vector(15 downto 0);
cin     : in    std_ulogic;
cout    : out   std_ulogic;
sum     : out std_ulogic_vector(15 downto 0)
```

- A e B são os valores a serem somados, cin e cout são variáveis para representar Carry-In e Carry-Out respectivamente, sum recebe o valor da soma final.

```
soma(i) := A(i) xor B(i) xor C;
c := (A(i) and B(i)) or ((A(i) xor B(i)) and c);
end loop;
```



1.3.7 And

```
a, b : IN  std_ulogic_vector(15 downto 0);
x    : OUT std_ulogic_vector(15 downto 0)
```

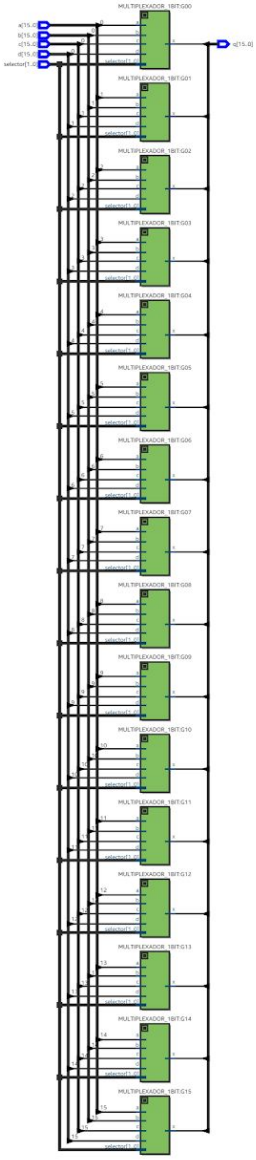
Compara dois bits seguindo a norma de 1 e 1 geram 1, do contrário o output é 0

2 Input AND gate		
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

1.3.8 Mux_4x1

```
a, b, c  : in  std_ulogic_vector(15 downto 0);
selector    : in  std_ulogic_vector(1 downto 0);
x           : out std_ulogic_vector(15 downto 0)
```

- a, b e c são as opções que podem ser atribuídas à saída x dependendo do valor do selector.



1.3.9 PC

```
entity PC is
port(
   A     : in  std_logic_vector(15 downto 0);
   Aout  : out std_logic_vector(15 downto 0)
);
end entity PC;
```

Para o contador era necessário criar um componente que guardasse um endereço que pudesse ser incrementado para mover entre os índices de um array.

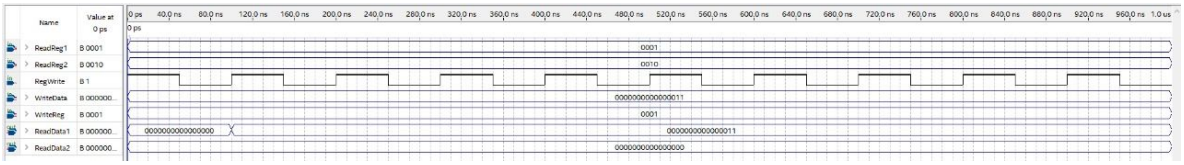
1.4 Datapath

É a conexão entre as unidades funcionais formando um único caminho de dados e acrescentando uma unidade de controle responsável pelo gerenciamento das ações que serão realizadas para diferentes classes de instruções.

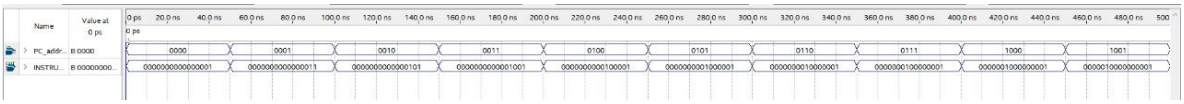
2 Simulações e Testes

Objetivando analisar e verificar o funcionamento do processador, efetuamos alguns testes analisando cada componente do processador em específico.

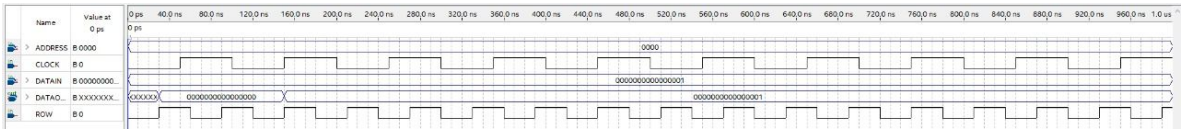
Banco de registrador



Memória de Instrução



Memória de dados



3 Considerações finais

Este trabalho apresentou o projeto e implementação do processador de 16 bits denominado de TITANIC 2, que embora não tenha conseguido ter todos os seus componentes interligados para funcionar de fato como um processador capaz de realizar algoritmos mais complexos, contribuiu bastante para o entendimento da grupo sobre como funciona a parte interna desse componente.