****

**PODER EXECUTIVO**

**MINISTÉRIO DA EDUCAÇÃO**

**UNIVERSIDADE FEDERAL DE RORAIMA**

**DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO**

**ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES**

**RELATÓRIO DO PROJETO: PROCESSADOR Falc-On MM**

**ALUNOS:**

**João Pedro Oliveira Silva - 1201524403**

**Talles Bezerra de Assunção - 1201524410**

**Janeiro 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 Falc-On MM**

**Janeiro de 2018**

**Boa Vista/Roraima**

**RESUMO**

O presente documento relata alguns pontos importantes do planejamento, implementação e testes do processador Falc-On MM. O mesmo foi desenvolvido de forma de trabalho final da disciplina de Arquitetura e Organização de Computadores (DCC301) na Universidade Federal de Roraima no semestre 2017.2. O processador em questão pode fazer 13 operações possíveis. Cada componente do processador em questão foi desenvolvido na linguagem de descrição de hardware VDHL. E os testes executados foram feitos utilizando-se das waveforms geradas que simulam o comportamento do hardware descrito.

**Conteúdo**

[1 Especificação 7](#_Toc504634108)

[1.1 Plataforma de desenvolvimento 7](#_Toc504634109)

[1.2 Conjunto de instruções 8](#_Toc504634110)

[1.2.1 Instruções do tipo R: 8](#_Toc504634111)

[1.2.2 Instruções do tipo J: 8](#_Toc504634112)

[1.2.3 Instruções do tipo I: 9](#_Toc504634113)

[1.2.4 Visão geral das instruções do Processador Falc-On MM: 9](#_Toc504634114)

[1.3 Descrição do Hardware 10](#_Toc504634115)

[1.3.1 PC 10](#_Toc504634116)

[1.3.2 Memória de instruções (ROM) 10](#_Toc504634117)

[1.3.3 Unidade de controle 11](#_Toc504634118)

[1.3.4 Banco de registradores 12](#_Toc504634119)

[1.3.5 ULA 12](#_Toc504634120)

[1.3.6 Memória de dados (RAM) 13](#_Toc504634121)

[1.3.7 Extensores de sinais 13](#_Toc504634122)

[1.3.8 Multiplexadores 14](#_Toc504634123)

[1.3.9 Somador 15](#_Toc504634124)

[1.4 Datapath 15](#_Toc504634125)

[2 Simulações e Testes 17](#_Toc504634126)

[3 Considerações finais 18](#_Toc504634127)

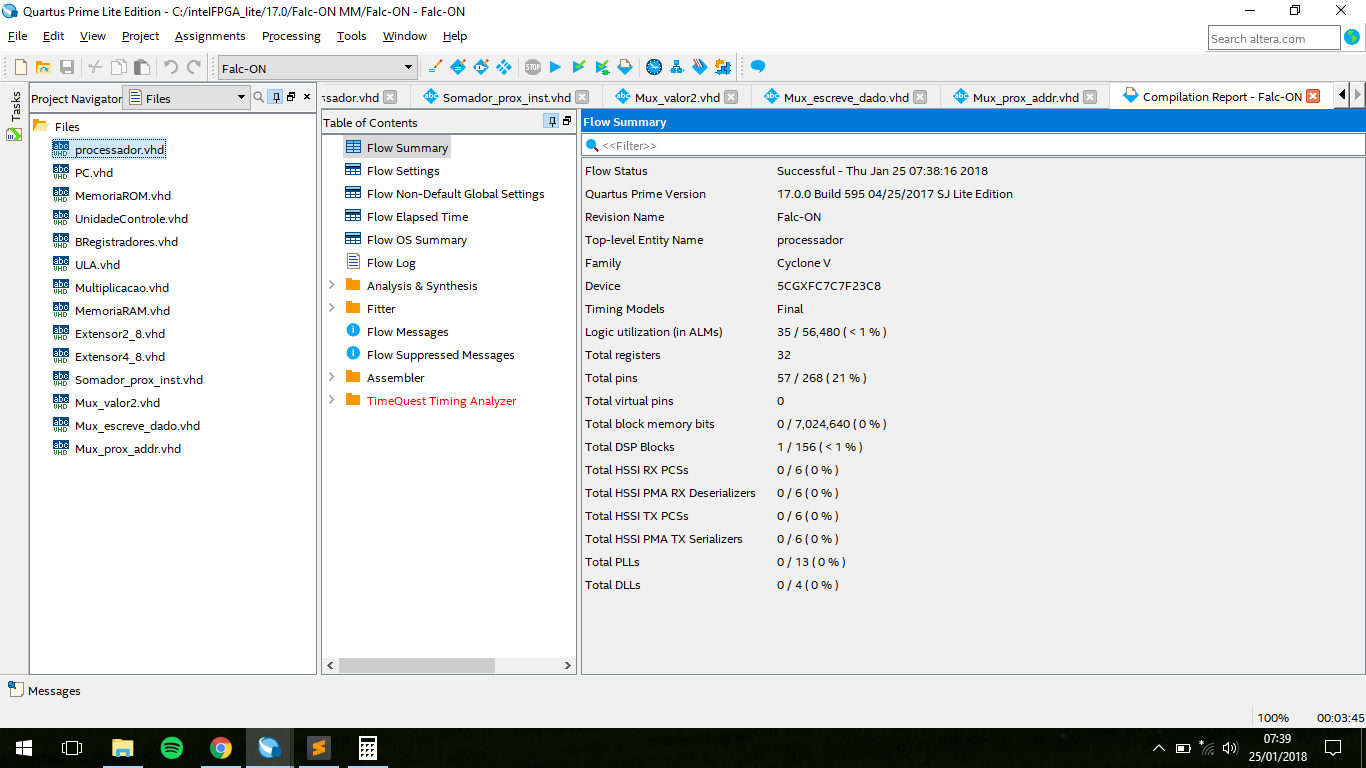
# Especificação

O Falc-On MM foi desenvolvido para a disciplina de Arquitetura e Organização de Computadores da Universidade Federal de Roraima no semestre 2017.2. O Processador é multiciclo, ou seja, executa cada tipo de instrução com diferentes ciclos de clock.

## Plataforma de desenvolvimento

Para a implementação do processador Falc-On MM foi utilizado a IDE Quartus Prime Lite Edition versão 17.0 fazendo uma simulação do dispositivo 5CGXFC7C7F23C8 da família Cyclone V.

Toda a descrição do hardware foi feita utilizando a linguagem VHDL que é usada para facilitar o design de circuitos digitais. A figura 1 mostra as especificações do projeto.



Especificações no Quartus

## Conjunto de instruções

O processador Falc-On MM possui 4 registradores: S0, ... e SN. Assim como 13 formatos de instruções de 8 bits cada, Instruções do **tipo R, tipo I e tipo J**, seguem algumas considerações sobre as estruturas contidas nas instruções:

* **Opcode**: a operação básica a ser executada pelo processador, tradicionalmente chamado de código de operação;
* **Reg1**: o registrador contendo o primeiro operando fonte e adicionalmente para alguns tipos de instruções (ex. instruções do tipo R) é o registrador de destino;
* **Reg2**: o registrador contendo o segundo operando fonte;

### Instruções do tipo R:

Abrange instruções de carregamento e gravação de dados na memória primária e instruções baseadas em operações aritméticas. A divisão de bits está descrita na tabela 1 abaixo.

Table 1 - Formato de intruções do tipo R

|  |  |  |
| --- | --- | --- |
| Opcode | Registrador 1 | Registrador 2 |
| 7-4 | 3-2 | 1-0 |
| 4 bits | 2 bits | 2 bits |

### Instruções do tipo J:

Abrange instruções onde é necessário fazer um salto condicional ou não entre os endereços, como um *if* ou um *goto*, por exemplo. A divisão de bits está descrita na tabela 2 abaixo.

Tabela 2 - Formato de instruções do tipo J

|  |  |
| --- | --- |
| Opcode | Endereço |
| 7-4 | 3-0 |
| 4 bis | 4 bits |

### Instruções do tipo I:

Abrange instruções de gravação de dados na memória primária e operações imediatas. A divisão de bits está descrita na tabela 3 abaixo.

Tabela 3 - Formato de instruções do tipo I

|  |  |  |
| --- | --- | --- |
| Opcode | Registrador 1 | Registrador 2 |
| 7-4 | 3-2 | 1-0 |
| 4 bits | 2 bits | 2 bits |

### Visão geral das instruções do Processador Falc-On MM:

O número de bits do campo **Opcode** das instruções é igual a quatro, sendo assim obtemos um total () de 16 **Opcodes (0-15)** que são distribuídos entre as instruções, assim como é apresentado na tabela 4.

Tabela 4 – Tabela que mostra a lista de Opcodes utilizadas pelo processador Falc-On MM.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Opcode | Nome | Formato | Breve Descrição | Exemplo |
| 0000 | ADD | R | Soma | Add $r1, $r2 -> $r1 = $r1 + $r2; |
| 0001 | SUB | R | Subtração | Sub $r1, $r2 -> $r1 = $r1 - $r2; |
| 0010 | ADDI | I | Soma imediata | Addi $r1, XX -> $r1 = $r1 + 000000XX; |
| 0011 | MULT | R | Multiplicação | Mult $r1, $r2 -> $r1 = $r1 \* $r2; |
| 0100 | AND | R | AND lógico | And $r1, $r2 -> $r1 = $r1 and $r2; |
| 0101 | OR | R | OR lógico | Or $r1, $r2 -> $r1 = $r1 or $r2; |
| 0110 | NOT | R | NOT lógico | Not $r1, $r2 -> $r1 = not $r1, $r2 é ignorado; |
| 0111 | XOR | R | XOR lógico | Xor $r1, $r2 -> $r1 = $r1 xor $r2; |
| 1000 | LW | I | Load word | Lw $r1, $r2 -> $r1 = 0($r2); |
| 1001 | SW | I | Store word | Sw $r1, $r2 -> 0($r2) = $r1; |
| 1010 | LI | I | Load imediato | Li $r1, XX -> $r1 = 000000XX; |
| 1011 | JMP | J | Jump | Jmp XXXX -> salto para XXXX; |
| 1100 | JR | J | Jump registrador | Jr $r1, $r2 -> salto para o valor em $r1, $r2 é ignorado; |

## Descrição do Hardware

Nesta seção são descritos os componentes do hardware que compõem o processador Falc-On MM, incluindo uma descrição de suas funcionalidades, valores de entrada e saída. Segue a descrição de cada um.

### PC

O Contador do programa (PC) a cada ciclo de clock, apenas recebe um endereço e repassa para a Memória ROM. Ele conta com 2 pinos de entrada, o inputPC de 8 bits e o clock, e 1 pino de saída de 8 bits.

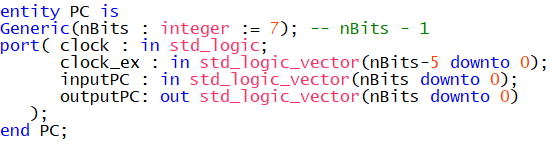
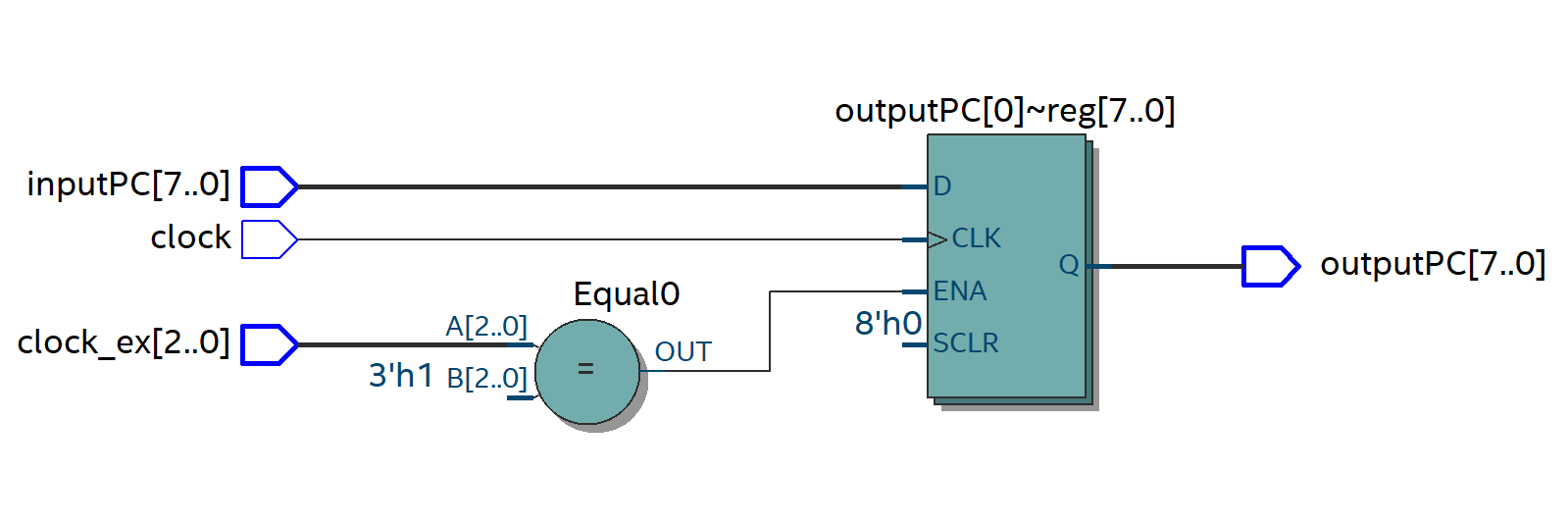


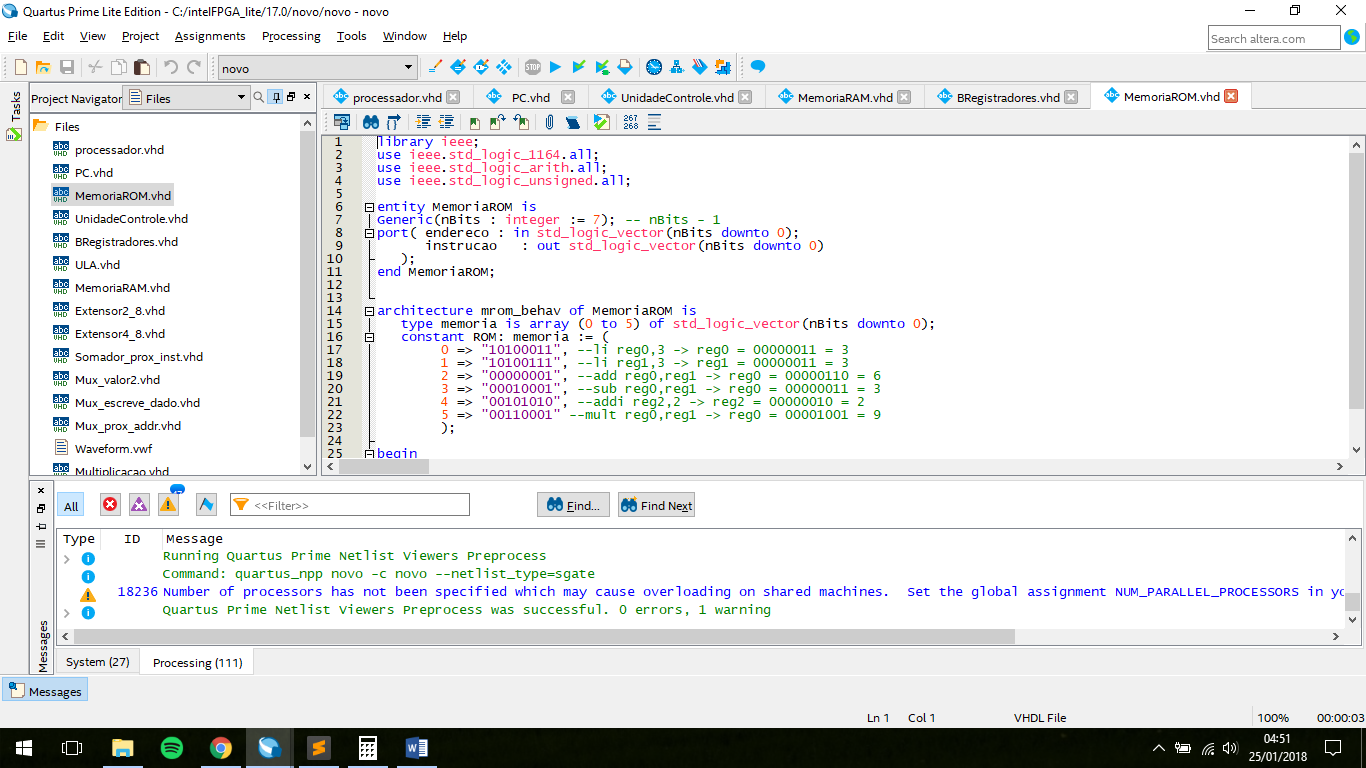
Figure 1 - Declaração do contador do programa

* **clock** – Sinal de clock do processador
* **inputPC** – Valor de entrada
* **outputPC** – Valor de saída

 Figure 2 - Visualização do PC

### Memória de instruções (ROM)

A memória de instruções foi implementada com uma memória de somente leitura, ou seja, suas informações são gravadas pelo fabricante uma única vez e após isso não podem ser alteradas ou apagadas, somente acessadas.

 Figure 3 - Declaração da memória de instruções

* **endereco** – Endereço do qual se desejar ler
* **instrucao** – Instrução armazenado na memória no endereço **endereco**

### Unidade de controle

A Unidade de Controle é responsável por identificar que tipo de instrução será executada enviar as flags de controle para o Banco de Registradores, ULA, RAM e os multiplexadores de acordo com o tipo de opcode que recebe do PC. Possui apenas o opcode com pino de entrada e 7 pinos de saída como as flags de controle.

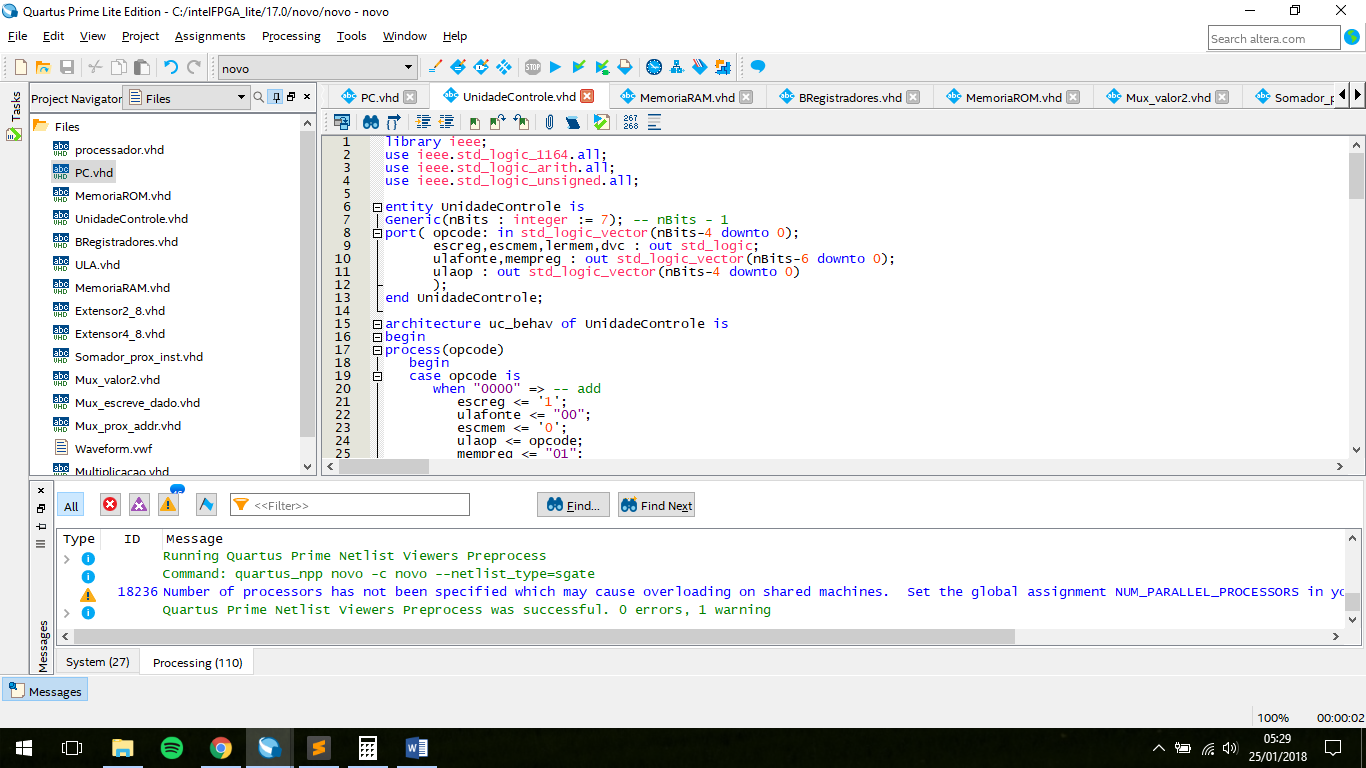


Figure 4 - Declaração da Unidade de Controle

* **opcode** – 4 primeiros bits da instrução que define o tipo de instrução que será executada
* **escreg** – Flag para escrever um valor no Banco de Registradores
* **escmem e lermem** – Flags para a Memória RAM de escrita e leitura de valores
* **dvc** – Flag para o multiplexador de desvio para selecionar o endereço da próxima instrução
* **ulafonte** – Flag para o multiplexador selecionar o segundo valor que será utilizado pela ULA
* **mempreg** – Flag utilizado o multiplexador que seleciona o valor que será escrito no Banco de Registradores
* **ulaop** – Flag para selecionar o tipo de operação na ULA

### Banco de registradores

O Banco de Registradores do Falc-On MM conta com um vetor com 4 registradores de 8 bits cada um, como foi citado na sessão **1.2 Conjunto de Instruções**. Foi declarado com 5 pinos de entrada e 6 pinos de saída.

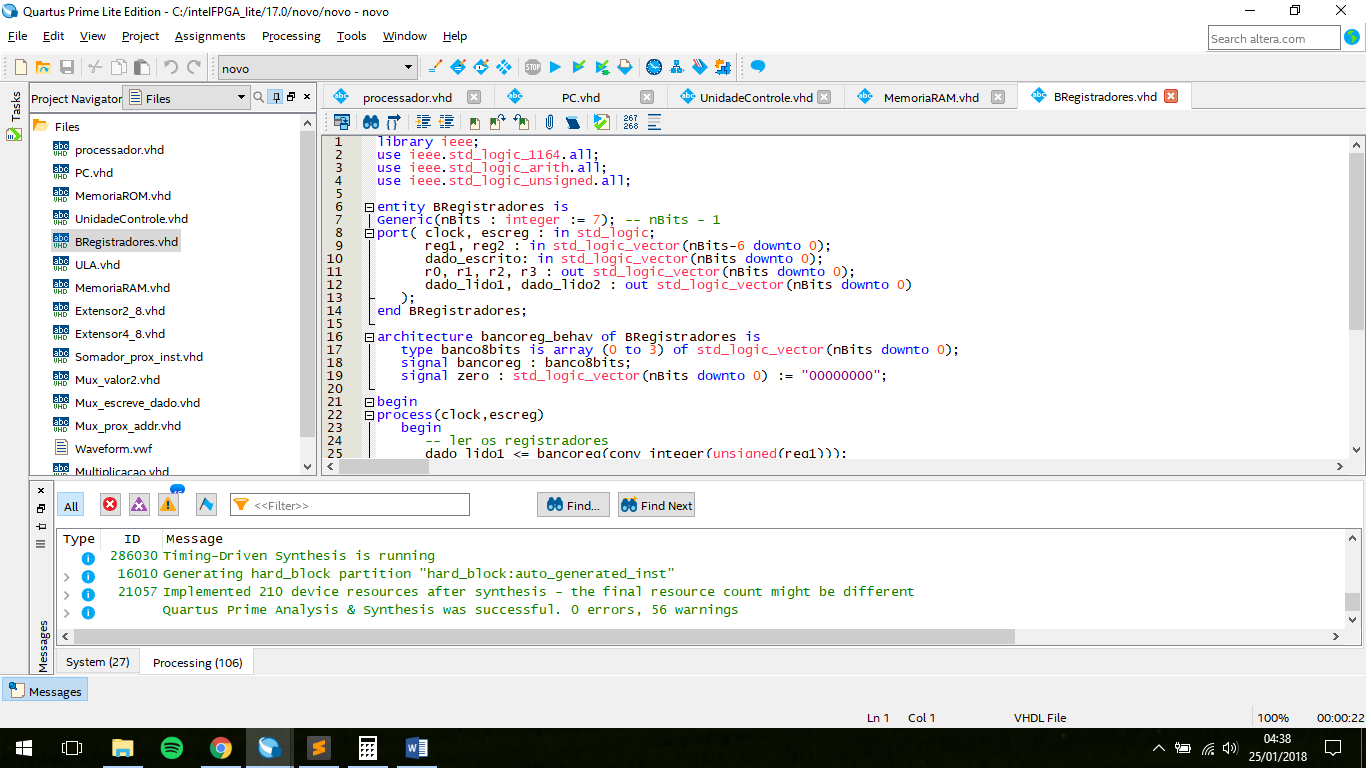


Figure 5 - Declaração do banco de registradores

* **clock** – Sinal de clock do processador.
* **escreg**– Quando ‘1’, escreve no endereço do registrador 1
* **reg1 e reg2** – Endereço de 2 bits dos registradores passados na instrução
* **dado\_escrito** – Valor que será escrito no registador 1
* **r0,...,r3** – Saídas para a visualização dos valores nos registradores
* **dado\_lido1 e dado\_lido2** – Dado lido dos endereços de registradores passados na instrução

### ULA

A Unidade Lógica e Aritmética (ULA) desenvolvida conta com 13 operações aritméticas, todas listadas na tabela 4.

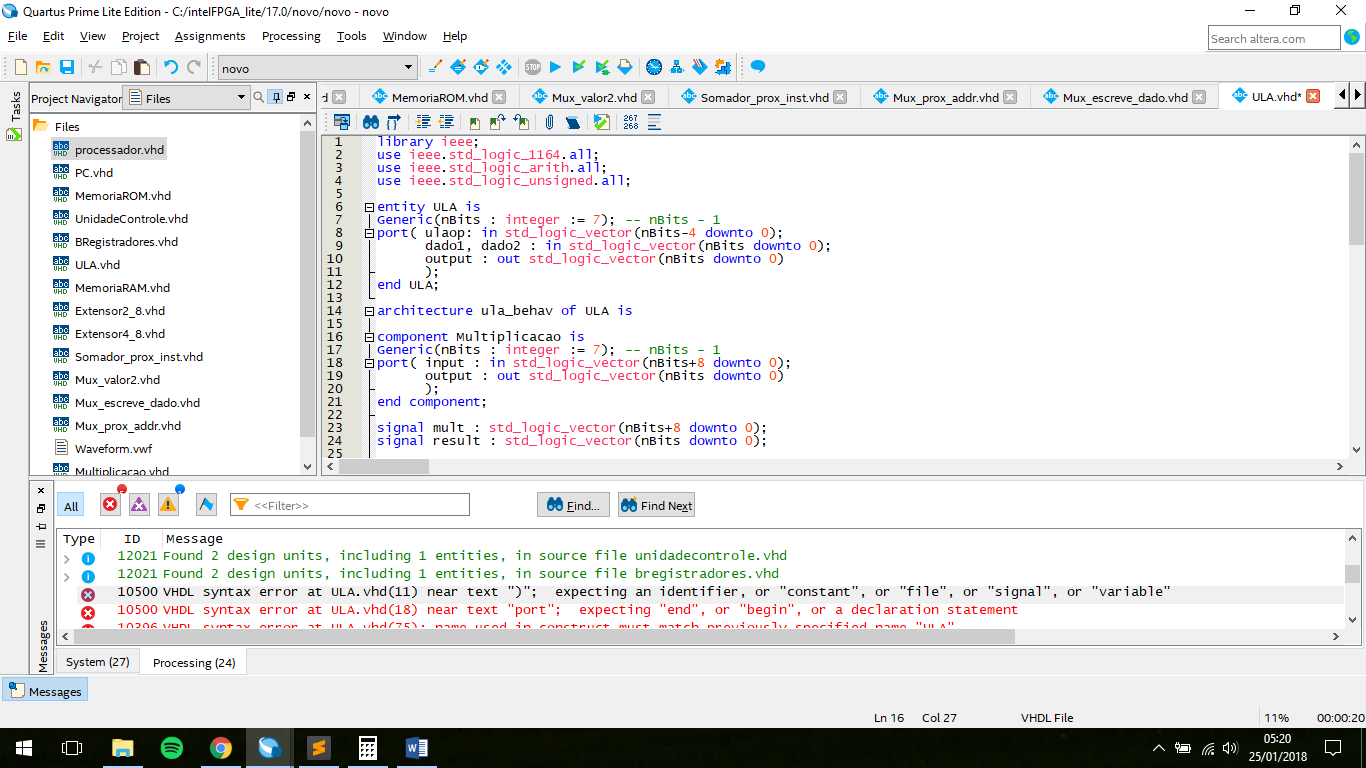


Figure 6 - Declaração da Unidade Lógica e Aritmética

* **ulaop** – Seletor de operação da ula
* **dado1** **e dado2** – Valores de 8 bits que poderão fazer parte das operações
* **output** – Resultado da operação

### Memória de dados (RAM)

A memória de dados foi implementada como um vetor de 256 espaços para valores de 8 bits

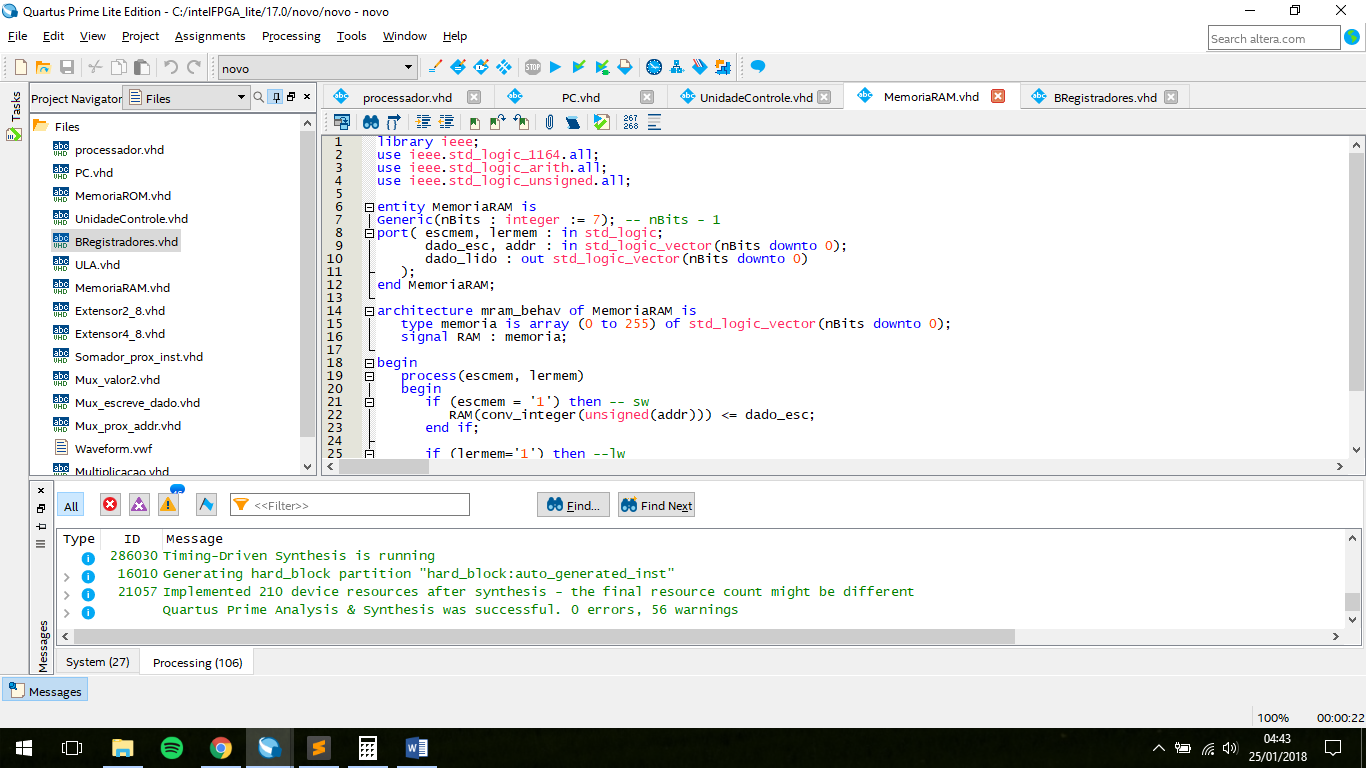


Figure 7 - Declaração da memória de dados

* **escmem** – Quando ‘1’, escreve o valor no endereço passado
* **lermem** – Quando ‘1’, ler o valor no endereço passado e ecrever em dado\_lido
* **dado\_esc** – Dado a ser escrito
* **addr** – Endereço de memória que está sendo escrito ou escrito
* **dado\_lido** – Dado escrito na memória no endereço **addr**

### Extensores de sinais

Existem dois extensors de sinal, um que recebe 2 bits e outro que recebe 4 bits, ambos transformam a entrada em uma saída de 8 bits. Os extensores de 2 e 4 bits para 8 bits vão preencher os bits extras com 0’s.

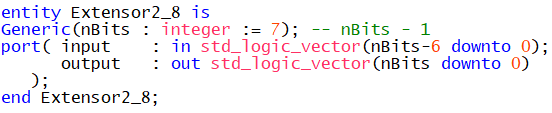


Figure 8 - Declaração de um extensor de sinal de 2 bits para 8 bits

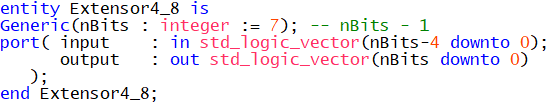
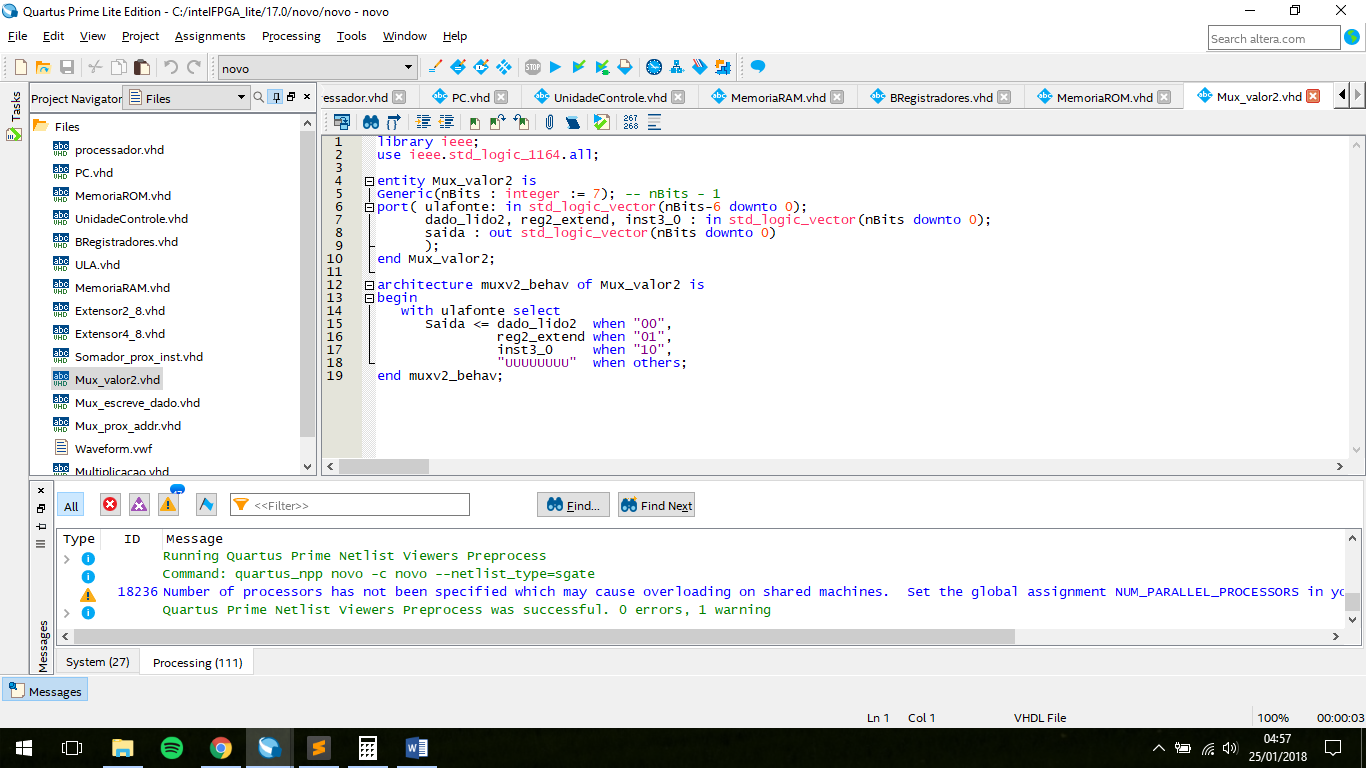


Figure 9 - Declaração de um extensor de sinal de 4 bits para 8 bits

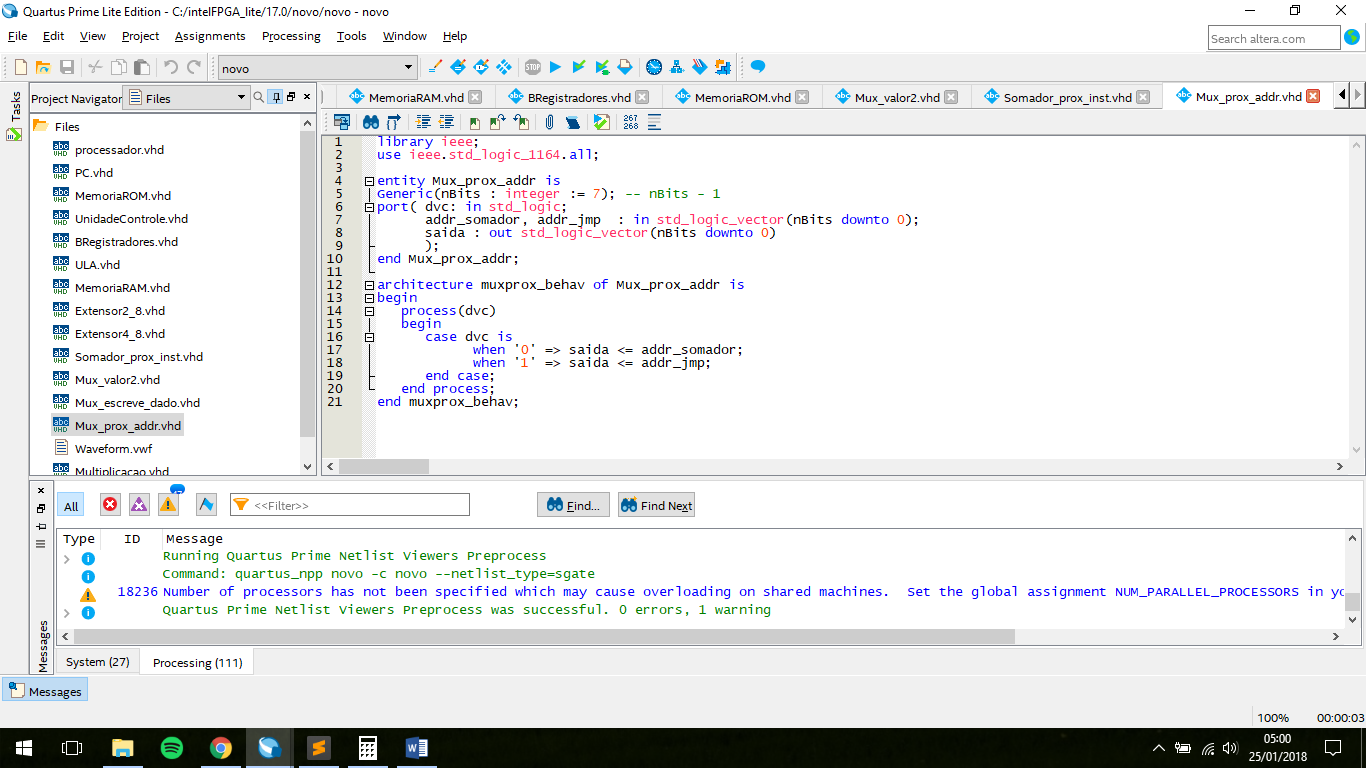
* **input** – Sinal de entrada em 2 / 4 bits
* **output** – Sinal de saída em 8 bits

### Multiplexadores

Existem 3 multiplexadores, um para selecionar qual o valor que será utilizado pela ULA (Mux\_valor2), outro para selecionar o endereço da próxima instrução (Mux\_prox\_addr) e outro para selecionar o dado que será escrito no banco de registradores (Mux\_escreve\_dado).

Figure 10 - Declaração do multiplexador Mux\_valor2

* **ulafonte** – Seletor do multiplexador
* **dado\_lido2** – Dado lido no banco de registradores
* **reg2\_extend** – Os 2 últimos bits da instrução extendidos para operações imediatas
* **inst3\_0** - Os 4 últimos bits da instrução extendidos para operações de salto
* **saida** – Valor selecionado de acordo com a ulafonte

Figure 11 - Declaração do multiplexador Mux\_prox\_addr

* **dvc** – Seletor do multiplexador
* **addr\_somador** – Endereço para a próxima instrução
* **addr \_jump** – Endereço de uma instrução de salto
* **saida** – Valor selecionado de acordo com a dvc

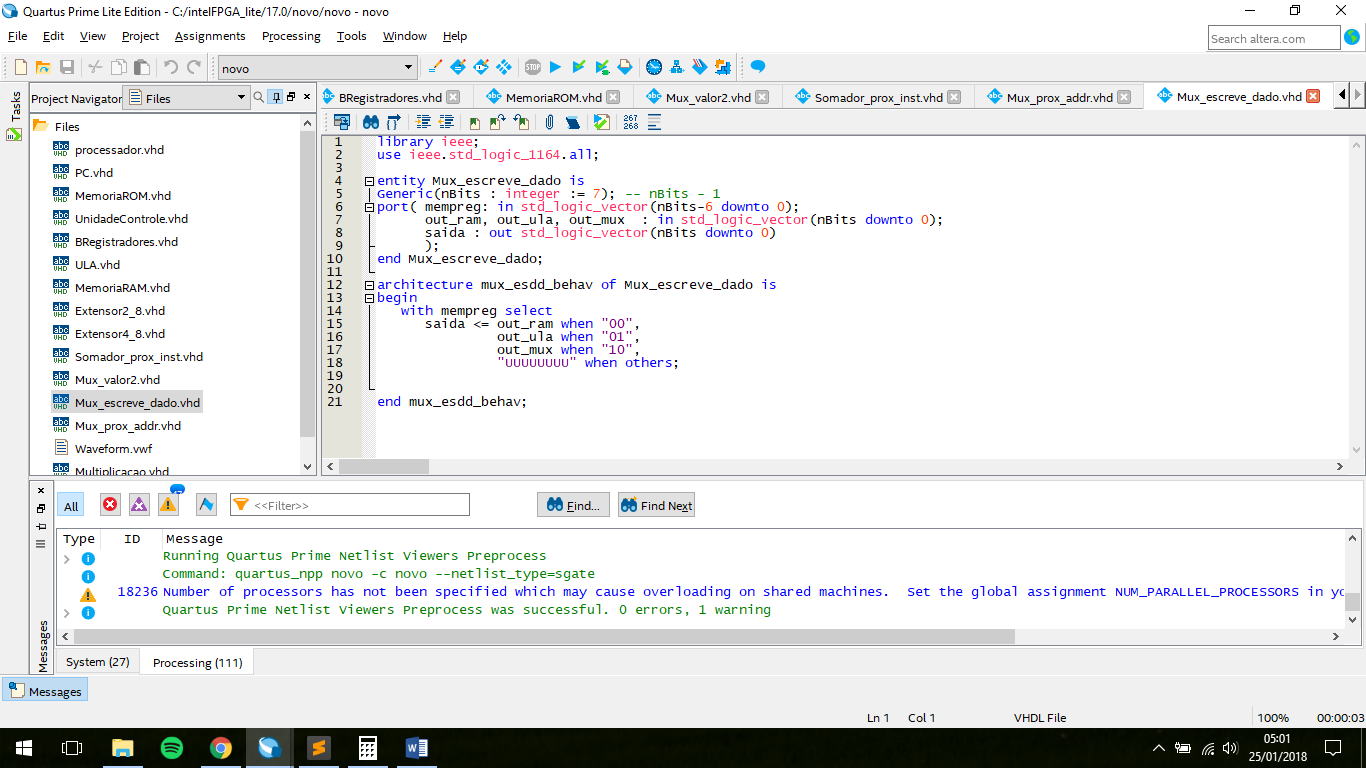
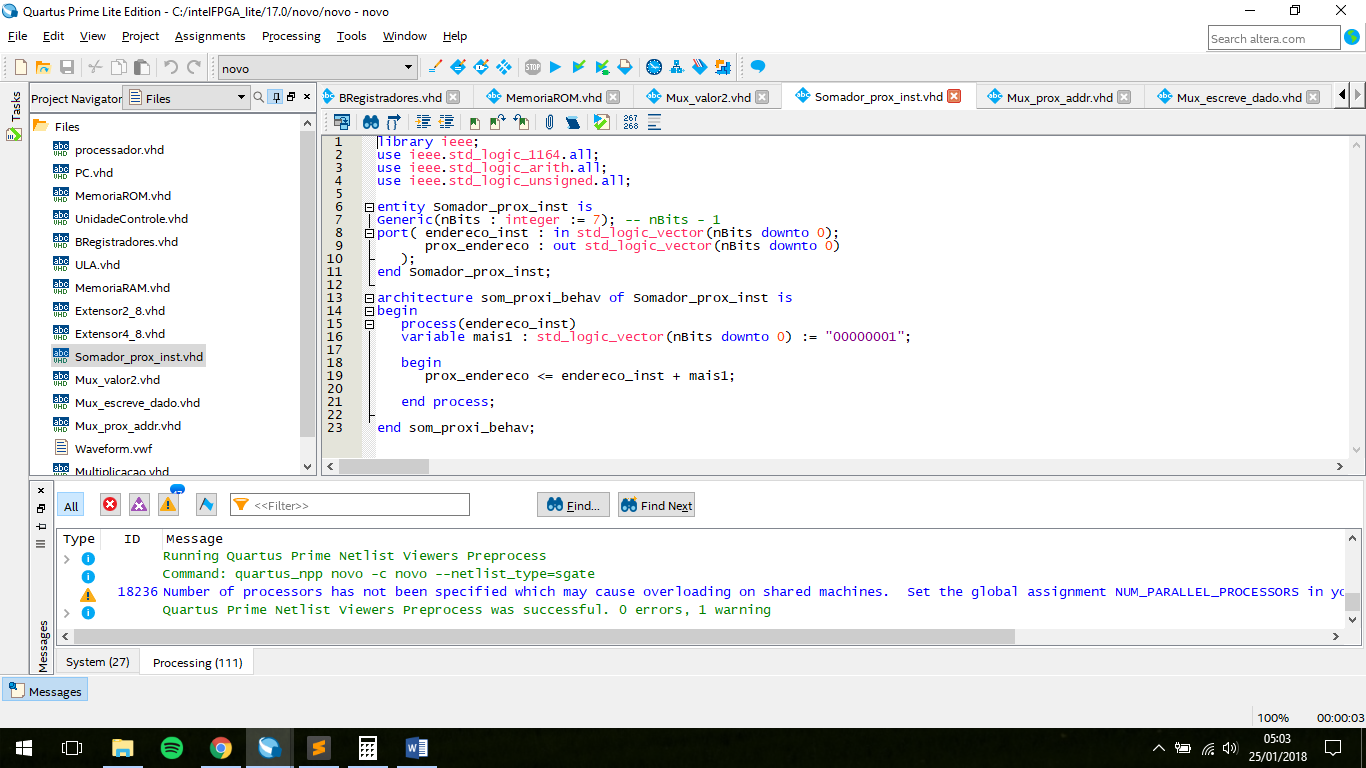


Figure 12 - Declaração do multiplexador Mux\_escreve\_dado

* **mempreg** – Seletor do multiplexador
* **out\_ram** – Valor recebido da Memória RAM
* **out \_ula** – Valor recebido da ULA
* **out \_mux** – Valor recebido do multiplexador Mux\_valor2 para load imediato
* **saida** – Valor selecionado de acordo com a mempreg

### Somador

Um somador simples para calcular o próximo endereço de instrução ir a próxima instrução.

 Figure 13 - Declaração do somador

## Datapath

No Datapth é onde irá ocorrer a conexão de todos os componentes descritos anteriormente para que o processador funcione e execute as instruções armazenadas.

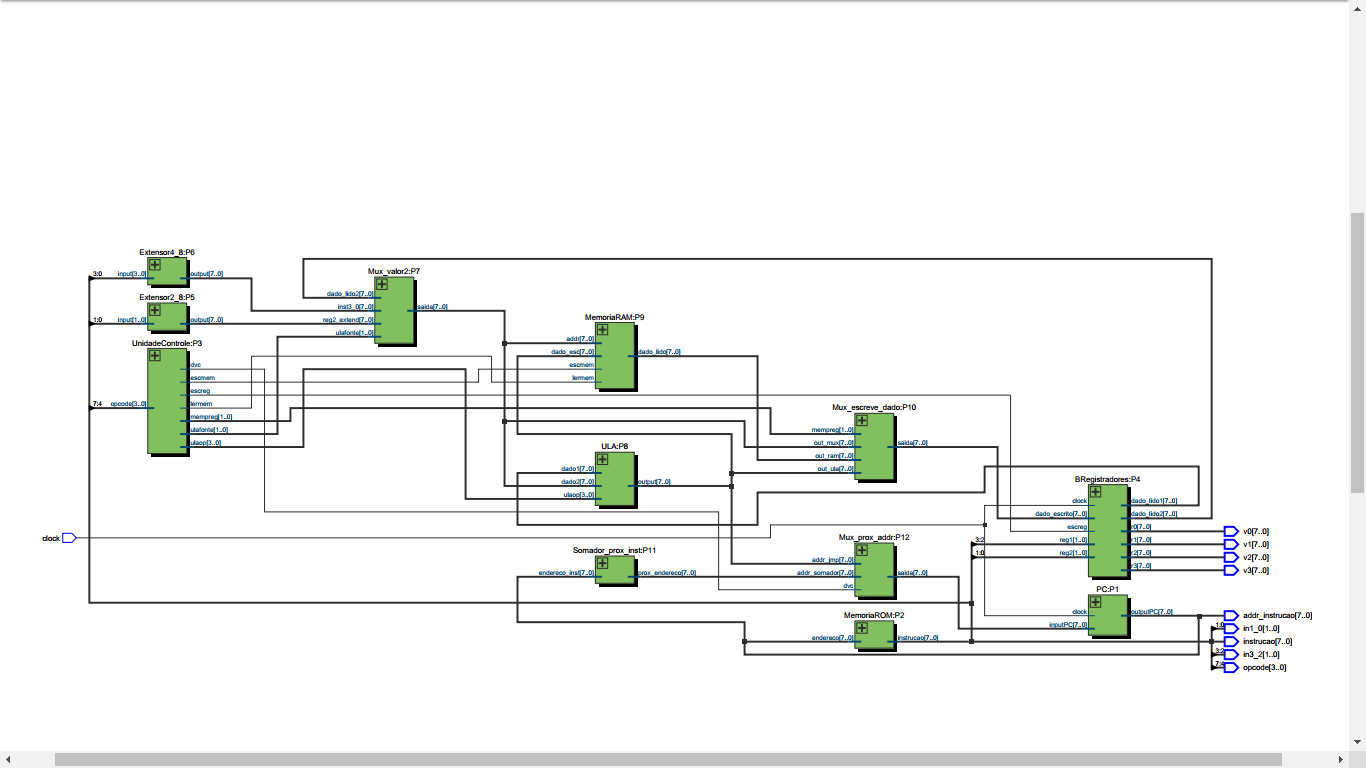


Figure 14 – RTL Viewer do Datapath

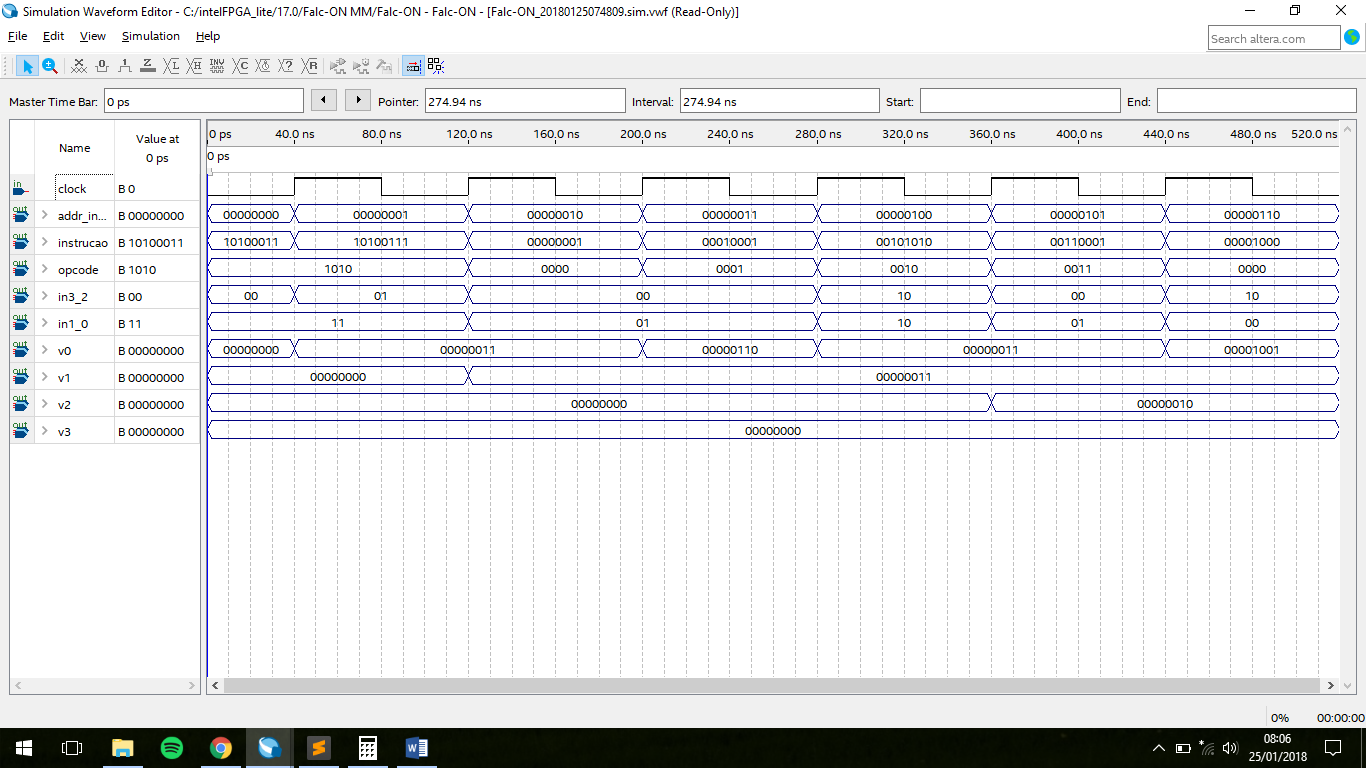
# Simulações e Testes

Para realizar as simulações do processador Falc-ON MM foram utilizadas Waveforms dentro do próprio Quartus Prime. Nos testes, instruções foram armazenadas na Memória ROM para serem executadas pelo processador.

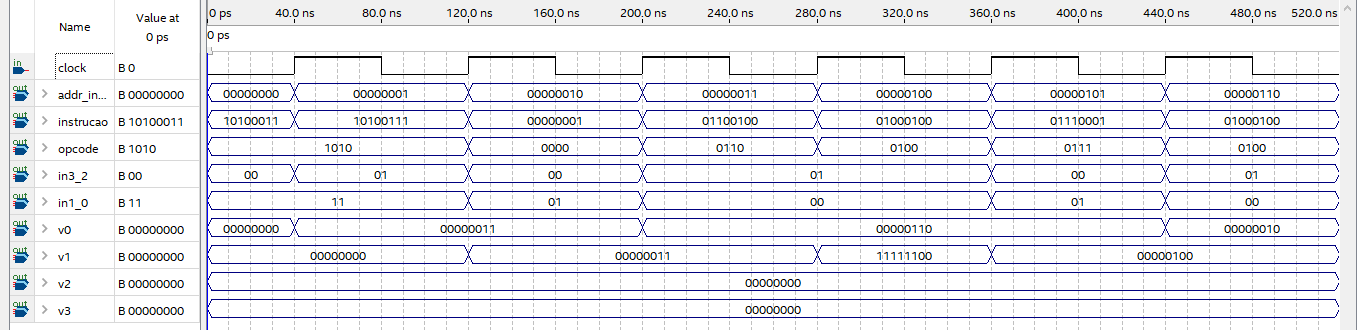
O conjunto e instruções abaixo testa todas as instrunções de tipo R que o processador pode executar. O processador irá carregar valores nos registradores e somar, subtrair e multiplicar os valores armazenados.

* 10100011 - li reg0,3 -> reg0 = 00000011 = 3
* 10100111 - li reg1,3 -> reg1 = 00000011 = 3
* 00000001 - add reg0,reg1 -> reg0 = 00000110 = 6
* 00010001 - sub reg0,reg1 -> reg0 = 00000011 = 3
* 00101010 - addi reg2,2 -> reg2 = 00000010 = 2
* 00110001 - mult reg0,reg1 -> reg0 = 00001001 = 9

Abaixo está a waveform com o resultado dos testes:



O próximo conjunto testa as portas lógicas do processador



# Considerações finais

O objetivo desse trabalho foi relatar todo o processo de criação do processador Falc-On MM de 8 bits.

O nome Falc-On MM foi inspirado na *Millenium Falcon* da franquia *Star Wars*, como ela tinha o nome "*Millenium*" que representa o mil, decidimos usar dois M's para representar o dois mil, pois estamos nos anos 2000. Pegamos o nome *Falcon* e o separamos em Falc e On, para termos uma representação computacional do *ON* e *OFF*, logo, o nome ficou Falc-On MM.

Foi bem trabalhoso fazer boa parte do projeto, os componentes do processador feitos separadamente não foram tão complicados assim pois todos funcionavam perfeitamente já que todos, sem excessão, seguem uma “receita de bolo” para funcionarem, os testes dos componentes provam que os mesmos estão em correto funcionamento. O problema surgiu quando foi para interliga-los pelo Datapath, pois tivemos dificuldade em conseguir mais um ciclo de clock, pois todos os componentes pareciam estar executando ao mesmo tempo, extrapolando as 5000 iterações que o Quartus permitia. Infelizmente, não conseguimos implementar um processador de multiciclo.

Mas sem dúvidas, foi um trabalho muito proveitoso, não apenas no conhecimento do funcionamento de um processador, mas em aperfeiçoamento em pesquisar e procurar respostas como um cientista.