UFRR

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

ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES

RELATÓRIO DO PROJETO: PROCESSADOR SPIRON

ALUNOS:

Francisco Pires Júnior – 1201424409
Felipe Derkian de Sousa Freitas – 1201424418

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 SPIRON

Janeiro de 2018
Boa Vista/Roraima

Resumo

Este trabalho aborda o desenvolvimento de um processador de 8 bits para a disciplina de Arquitetura e Organização de Computadores. O processador foi implementado usando a linguagem de descrição de hardware VHDL. E os testes executados foram feitos utilizando-se waveforms geradas que simulam o comportamento do hardware descrito, estes mostram um pouco do funcionamento e performance do processador

Conteúdo

1 Especificação

Nesta seção é apresentado o conjunto de itens para o desenvolvimento do processador Spiron, bem como a descrição detalhada de cada etapa da construção do processador. O processador é multiciclo, onde a execução das instruções se dá em etapas, onde cada etapa usa um ciclo de clock para a realização da mesma.

1.1 Plataforma de desenvolvimento

Para a implementação do processador Spiron foi utilizado a IDE: Quartus Prime Versão 17.0 Lite Edition

Figura 1 – Tela inicial do Quartus

1.2 Conjunto de instruções

O processador Spiron possui 4 registradores: $Zero, $s0, $s1, $s2. Assim como 3 formatos de instruções de 8 bits cada. Instruções do **tipo** R, I, 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;

Tipo de Instruções:

- Formato do tipo R: Abrange instruções de operações aritméticas, tais como: soma, soma de imediatos, subtração e etc. E a divisão de bits é descrita a seguir

Formato para escrita de código:

Tipo da Instrução	Reg1	Reg2

Formato para escrita em código binário:

4 bits	2 bits	2 bits
7-4	3-2	1-0
Opcode	Reg1	Reg2

- Formato do tipo I: Abrange instruções de operações de carregamento e gravação de dados da memória, tais como: Load e Store. E a divisão de bits é descrita a seguir:

4 bits	2 bits	2 bits
7-4	3-2	1-0
Opcode	Reg1	Reg2

- Formato do tipo J: Abrange instruções de operações de JUMP. E a divisão de bits é descrita a seguir:

4 bits	4 bits
7-4	3-0
Opcode	ADDRESS

Visão geral das instruções do Processador Spiron:

O campo de Opcode de cada instrução é de 4 bits, logo temos 15 opcodes disponíveis para identificação de instruções na UC, 2^4-1 = 15.

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

Opcode	Nome	Formato	Breve Descrição	Exemplo
0000	ADD	R	Soma	**add** $S0, $S1 ,ou seja, $S0 := $S0+$S1
0001	ADDI	R	Soma	**addi** $S0, 1 ,ou seja, $S0 := $S0+1
0010	BEQ	I	Branch	**beq** $S0, $S1
0011	LW	I	Load Word	**Lw** $S0,$S1
0100	SW	I	Store Word	**Sw** $S0,$S1
0101	LI	I	Load Immediately	**li** $S0, 31
0110	JUMP	J	Jump	**J** L1

1.3 Descrição do Hardware

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

1.3.1 ULA X8

O componente ULAx8 (Unidade Lógica Aritmética) tem como principal objetivo efetuar as principais operações aritméticas, dentre elas: soma, subtração, divisão e multiplicação. Adicionalmente o ULA X8 efetua operações de comparação de valor como maior ou igual, menor ou igual, somente maior, menor ou igual. O componente ULA X8 recebe como entrada três valores: **A** – dado de 8 bits para operação; **B** - dado de 8 bits para operação e **UALOp** – identificador da operação que será realizada de 4bits. A ULA X8 também possui três saídas: **ZERO** – identificador de resultado (1bit) para instruções de branch (1 se verdade e 0 caso contrário).

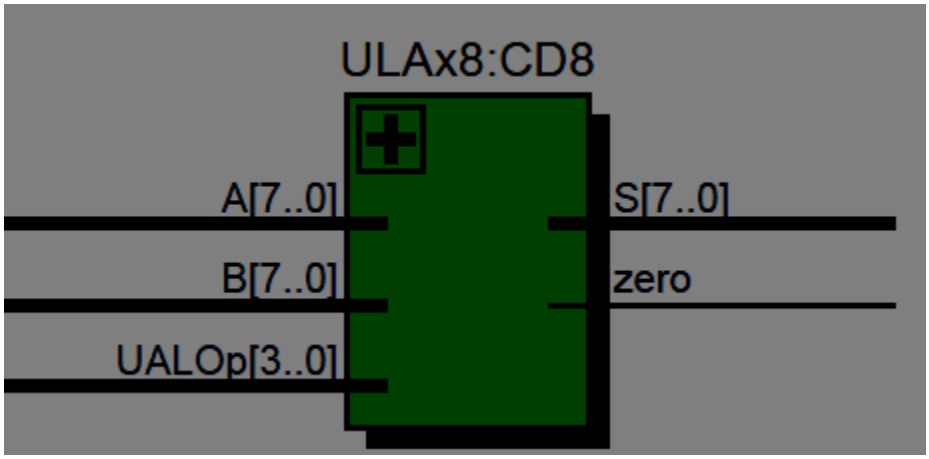


Figura 2 - Bloco simbólico do componente ULAx8 gerado pelo Quartus.

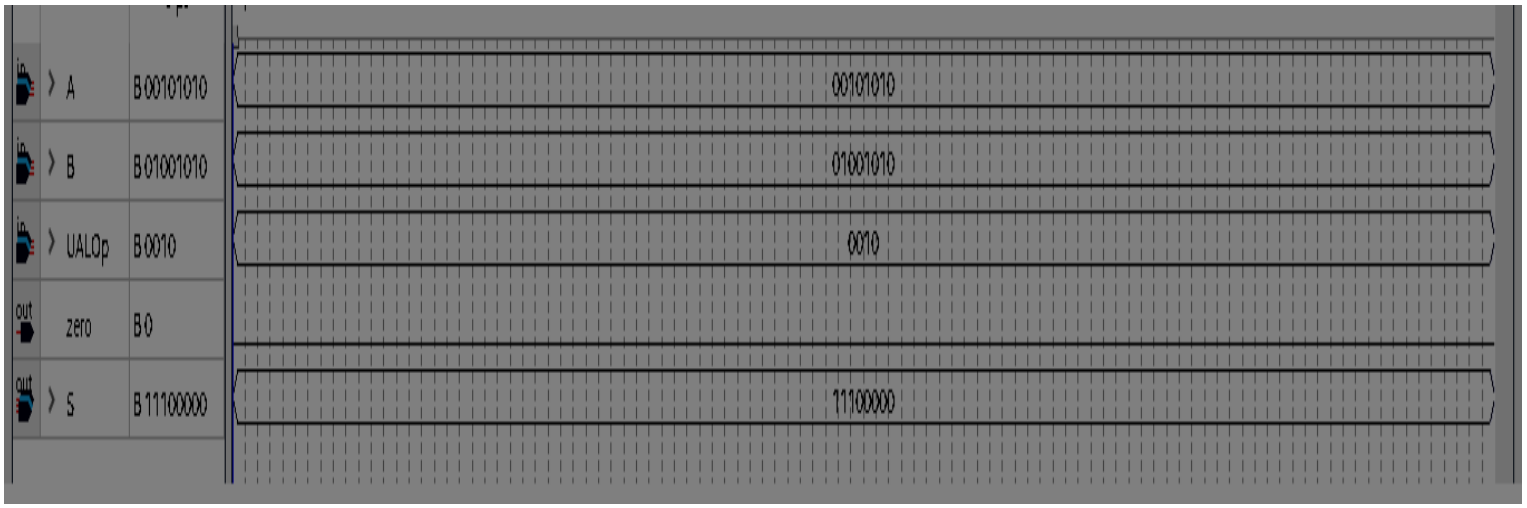


Figura 3 - waveform testando BEQ com valores diferentes.

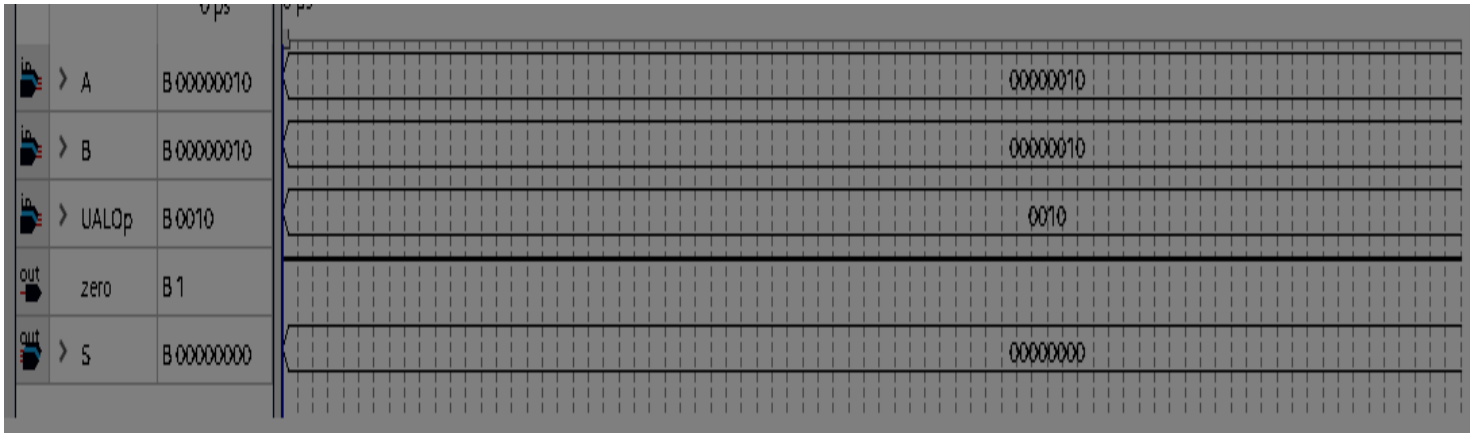


Figura 4 - waveform testando BEQ com valores iguais.

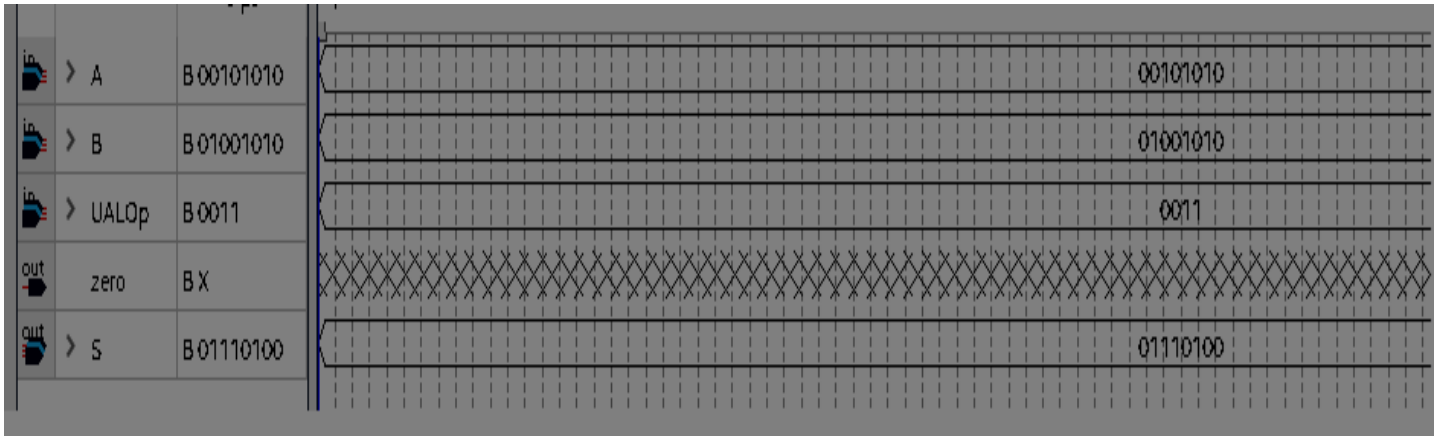


Figura 5 - waveform testando operação de cálculo de endereço.

1.3.2 Banco de Registradores

O componente Banco tem a função de ler e armazenar os valores em registradores sendo 4 no total. Sendo eles **$ZERO, $S1, $S2, $S3, o $ZERO** tem um valor constante 0, usado para fazer comparações, e mover valores para outros registradores. Os registradores $S1 ate $s3 são para armazenamentos de valores calculados ou carregados da memória.

Ele possui as seguintes sinais de entrada:

Clock: recebe o clock do sistema.

DadoAserEscrito(7..0): Dado que será escrito em registrador.

EscReg: Flag de sinal que aciona a escrita no registrador.

RegAserEscrito(1..0): Recebe o endereço do registrador a ser escrito.

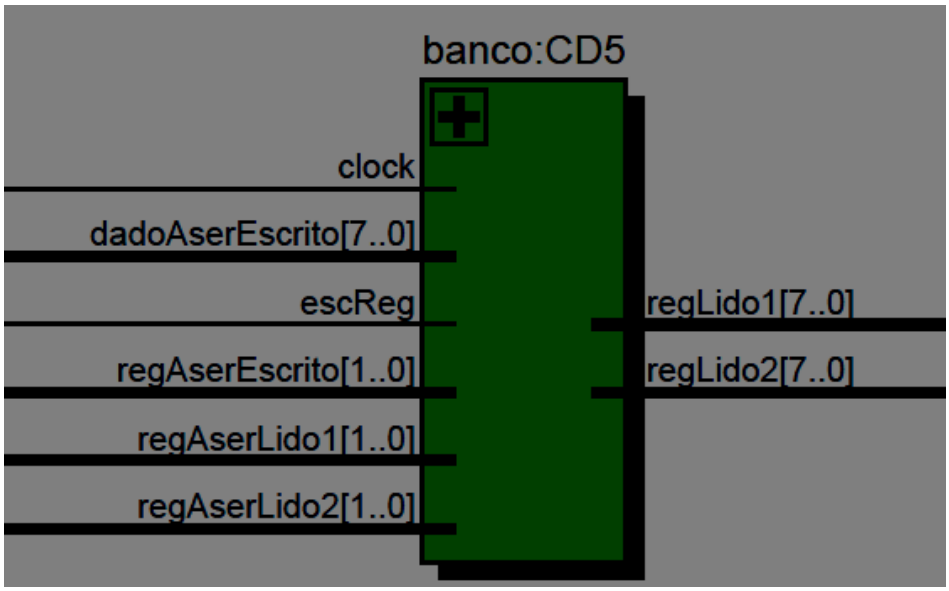
RegAserLido1(1..0): Recebe o endereço do registrador a ser lido.

RegAserLido2(1..0) : Recebe o endereço do registrador a ser lido.

Sinais de saida:

regLido1(7..0) : Recebe o valor lido para do primeiro registrador.

RegLido2 (7..0): Recebe o valor lido do segundo registrador.



1.3.3

Figura 6 - Bloco simbólico do componente banco gerado pelo Quartus.

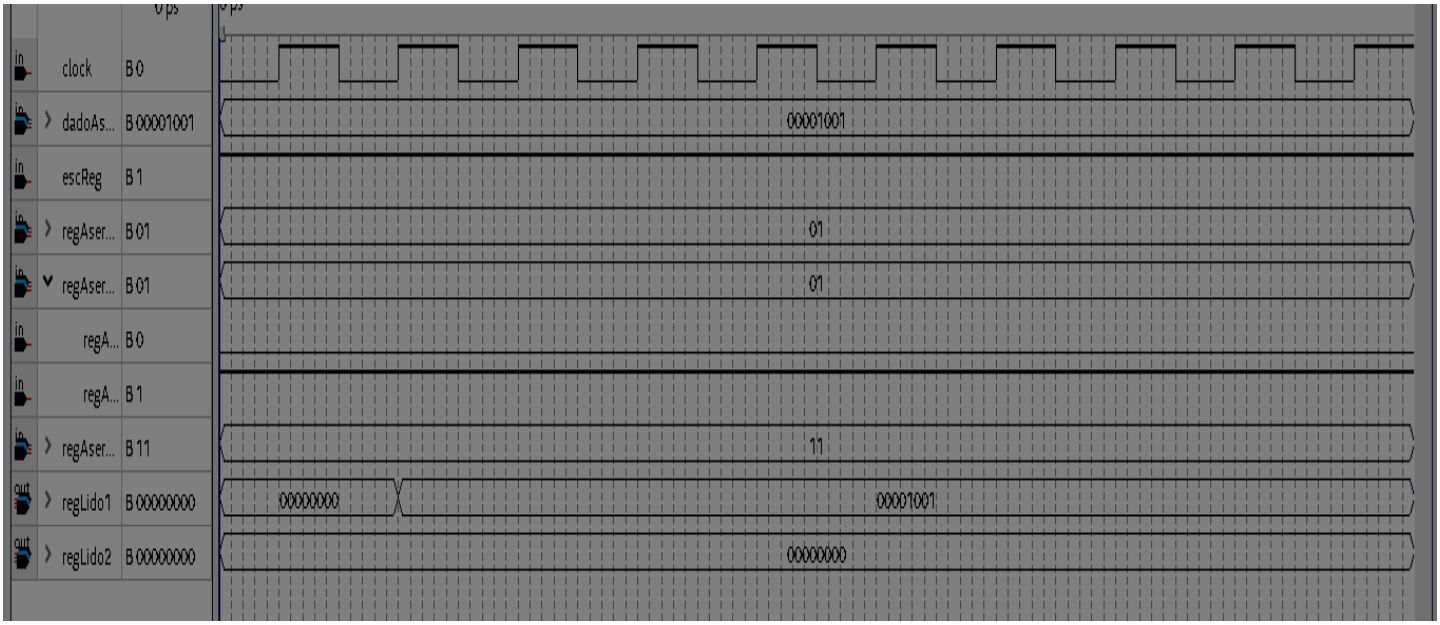


Figura 7 - waveform teste banco de registradores.

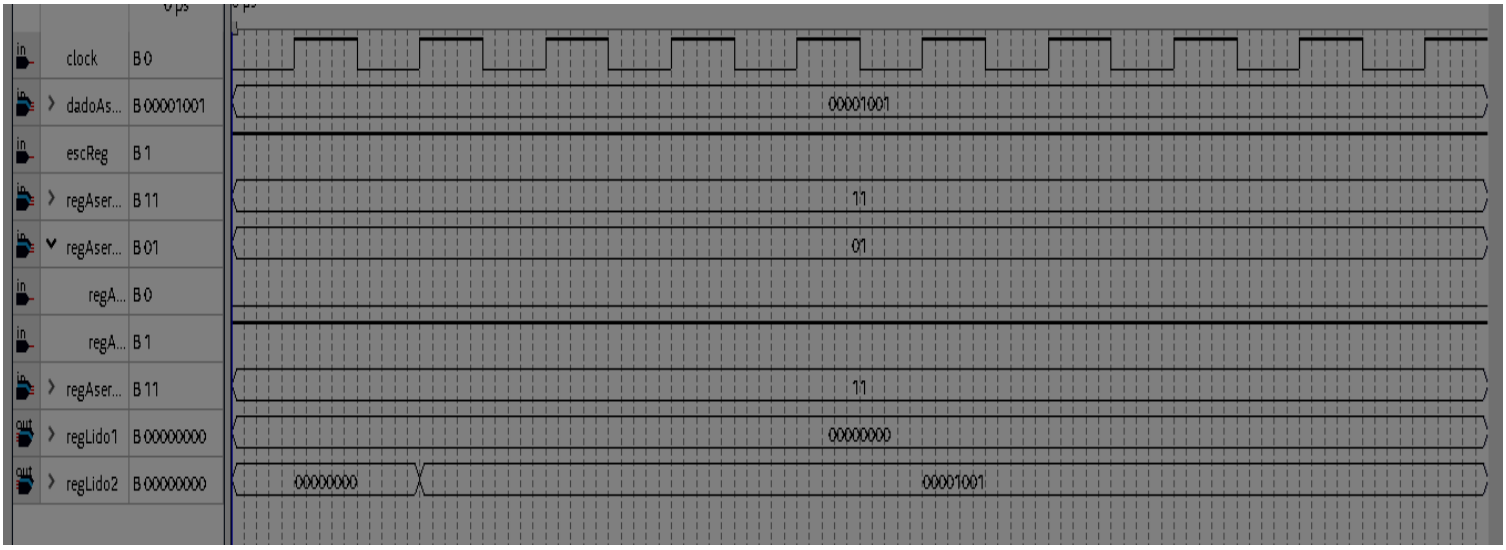


Figura 8 - waveform teste banco de registradores.

1.3.1 Extensor de Sinal 2x8

O componente extensor de sinal tem como funcao fazer a extensão de bits necessários completando os bits mais significados com 0's. No caso esse extensor é de 2 bits para 8 bits.

Sinal de entrada:

Entrada(1..0) : Recebe o valor a ser extendido o sinal.

Sinal de saida:

Saída(7..0): Recebe o valor com sinal extendido de 2 para 8 bits.

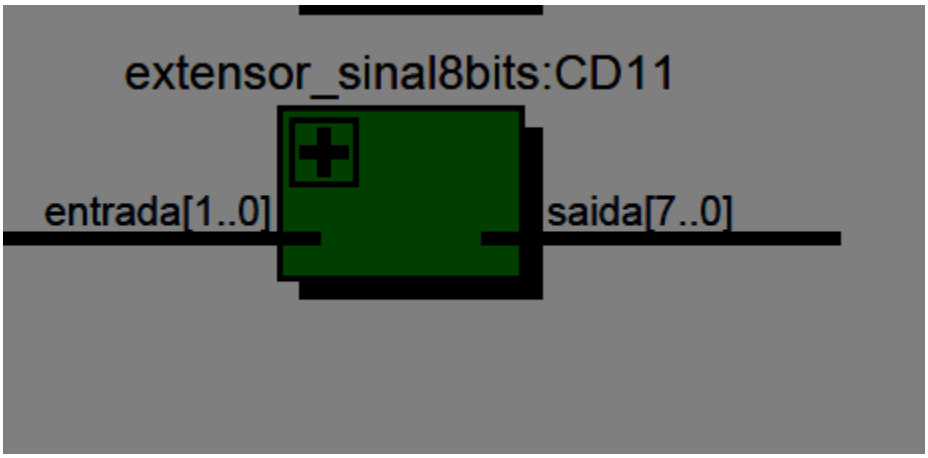


Figura 9 - Bloco simbólico do componente extensor_sinal8bits gerado pelo Quartus.

1.3.2 Unidade de Controle

O componente (Ucontrol), tem como função o controle da execução de instruções através da ativação das flags necessárias em cada etapa de execução. É ele quem gerencia e controla todo o caminho de dados do Processador.

Os sinais de entrada são:

Clock: Recebe o clock para ativar o processo.

inst_part7..4(3..0) (OPCODE): Recebe o opcode da instrução a ser executada.

Os sinais de saida são:

EscMeM: Flag de ativação para escrever na memoria.

EscReg: Flag de ativação para a realização da escrita no registrador de destino do banco de registradores.

IREsc: Flag para ativar o modo de decodificação da instruções dentro do registrador de instruções.

Loud: Loud é um sinal selector vindo da UC para seleção de entradas em um multiplexador do qual seleciona entre a entrada vinda da saida do PC ou da entrada vinda da saida do registrador UALSaida.

PCEscCond: Flag que aciona que um desvio foi tomado.

PCEsc: Flag que ativa a escrita no Program counter.

RegDest: Flag que seleciona qual o registrador de destino de uma instrução afim de salvar o resultado.

UALFonteA: Flag de seleção do multiplexador, tendo a função de selecionar a saida do Registrador A ou Endereço do Program Counter.

UALFonteB(1..0): Flag de seleção do multiplexador, tendo a função de selecionar a saida do Registrador B ou de outros tipos de sinais, tais como de endereço, desvio e etc.

FontePC(1..0): Flag seletora que seleciona qual endereço de memória irá compor o PC.

MemParaReg(1..0): Flag seletora para selecionar sinais de instruções Lw, Li, e escrita em registrador.

ULAOp(3..0): Codigo de seleção da operação a ser executada na ULA.

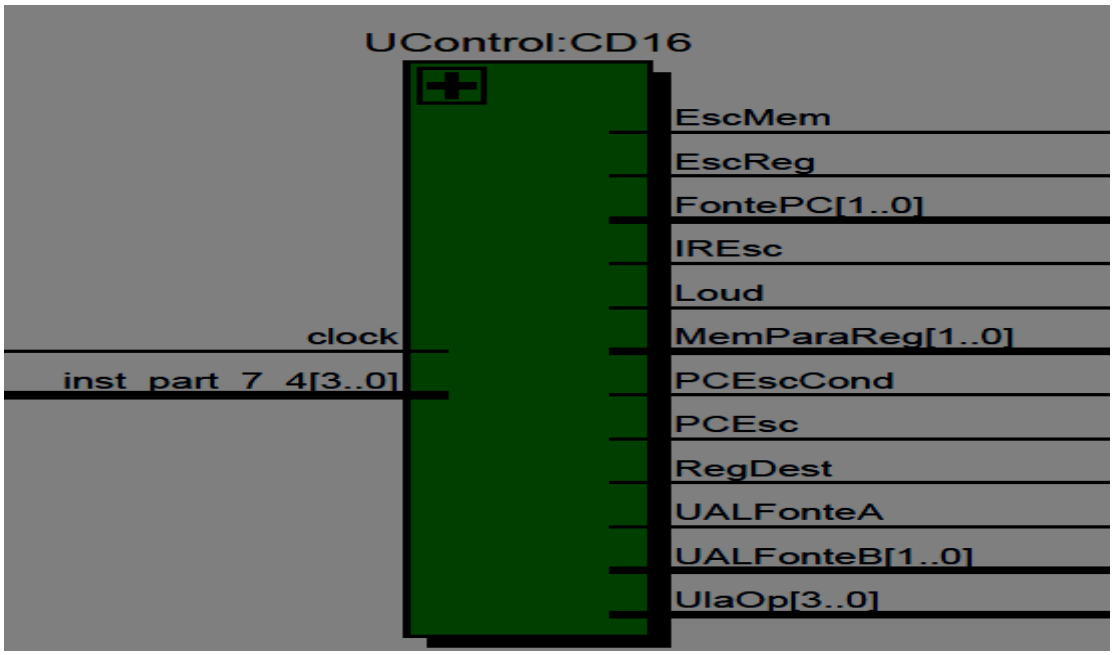


Figura 10 - Bloco simbólico do componente UControl gerado pelo Quartus.

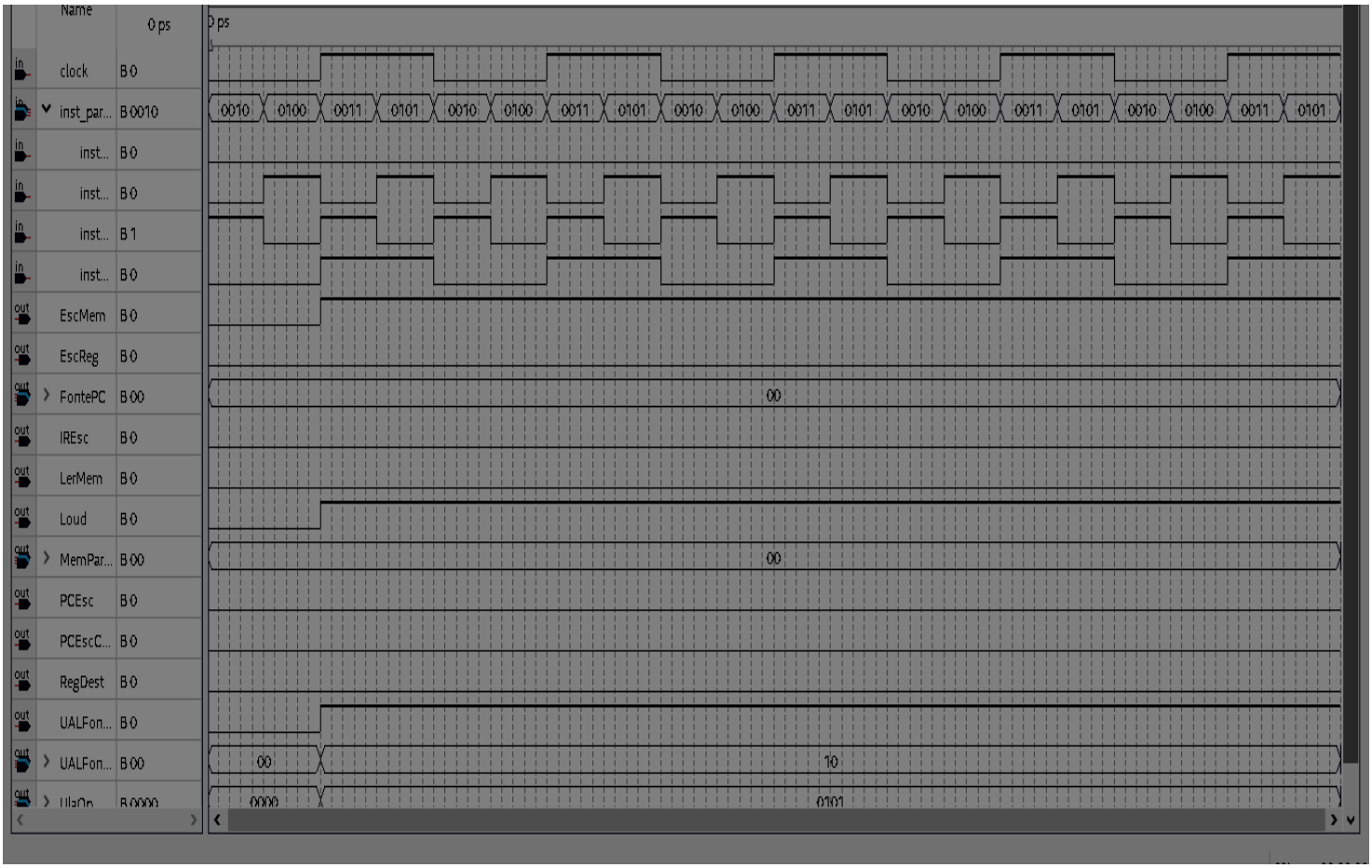


Figura 11 - waveform teste Unidade de controle.

Tabela 2 - Detalhes das Flags de controle do processador.

Nome do Sinal	Formato R	Lw	Sw	Beq	Jump
RegDest	0	1	X	X	X
MemPRegist	0	1	X	X	X
EscRegist	1	1	0	0	X
LerMem	0	1	0	0	X
EscMem	0	0	1	0	X
UALFonteA	1	1	1	1	X
UALFonteB	00	X	X	00	X

Loud	0	0 ou 1	0 ou 1	0	X
IREsc	1	1	1	1	X
FontePC	X	X	X	01	10
PCEsc	1	1	1	1	1

1.3.3 Memória RAM

O componente Memória RAM é uma memoria de Dados, onde está localizada todos os operandos que podem ser utilizados por instruções. A memoria RAM pode ser utilizada por instruções de Load ou Store.

Sinais de entrada:

E_data(7..0): Recebe o dado a ser colocado na memoria.

Clk: Recebe o clock do sistema para ativar o processo.

Endereco(7..0): Recebe o endereço onde encontra-se o operando.

FlagEscMem: Flag que ativa a escrita na memoria.

Sinais de Saida:

S_data(7..0): Recebe a saida com o valor operando buscado na memoria.

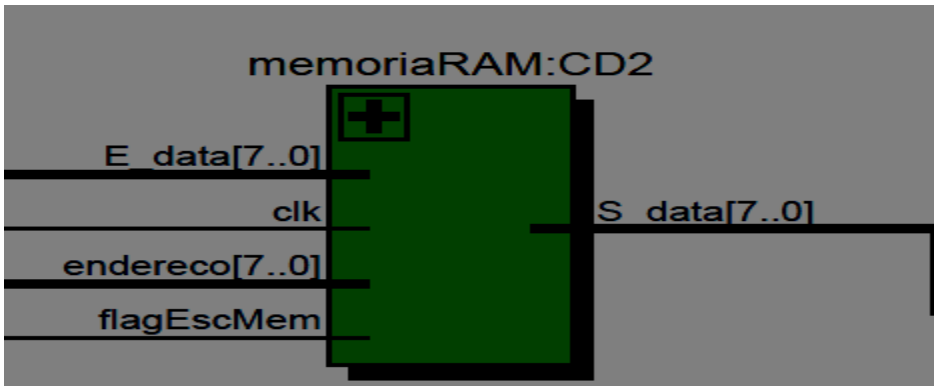


Figura 12 - Bloco simbólico do componente memoriaRAM gerado pelo Quartus.

1.3.4 Memória ROM

O componente Memória ROM é utilizada para armazenar as instruções a serem executadas.

Sinais de entrada:

endereco(7..0): Recebe o endereço da instrução.

Sinais de saida:

data(7..0): Saida com a instrução buscada em memoria.

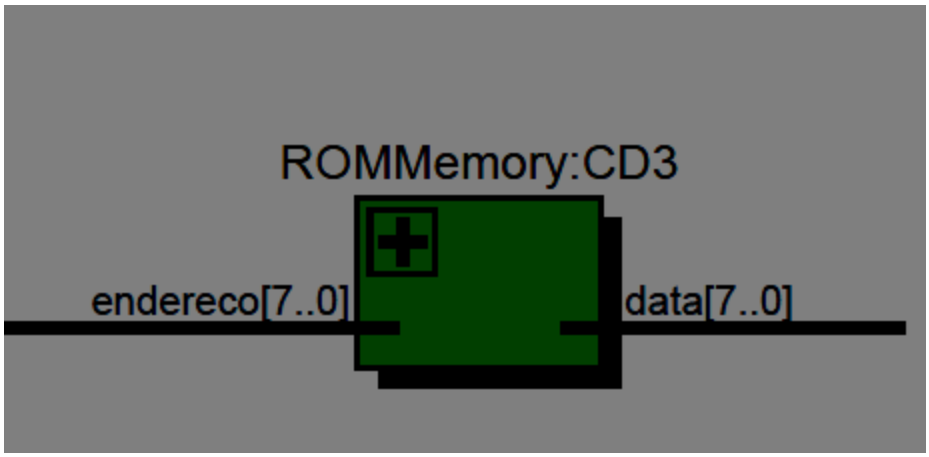


Figura 13 - Bloco simbólico do componente ROMMemory gerado pelo Quartus.

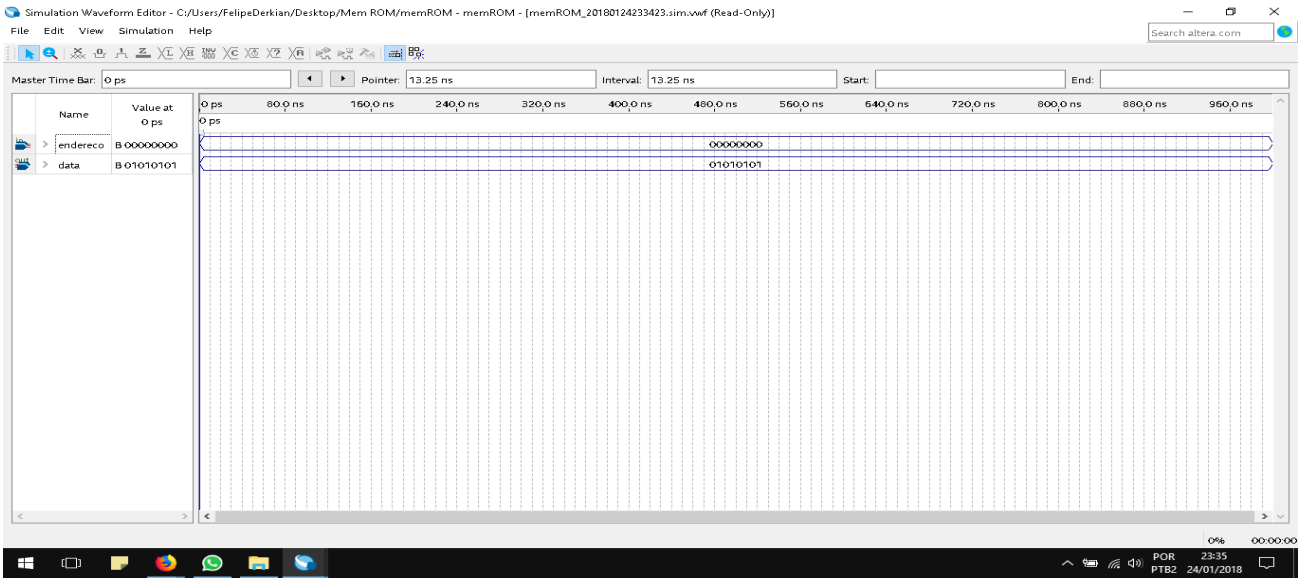


Figura 14 - waveform teste Memória ROM.

1.3.5 Multiplexador 2x1

O componente multiplexador serve como um IF ou seja, dependo do seletor ele seleciona determinada operação a ser executada dentre as possiveis.

Sinais de entrada:

A(7..0): Recebe a primeira opcao que é selecionada caso o seletor seja 0.

B(7..0): Recebe a segunda opcao que é selecionada caso o seletor seja 1.

Seletor: recebe da unidade de controle um valor para ser selecionado.

Sinais de saida:

Saida(7..0): Recebe o valor da opção selecionada.

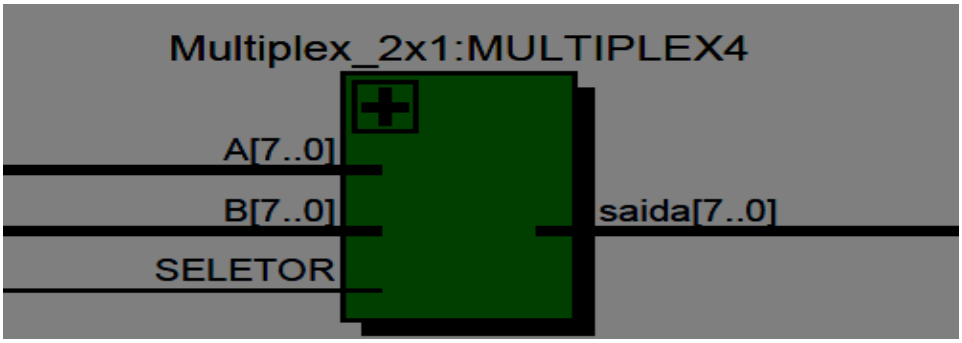


Figura 15 - Bloco simbólico do componente multiplex_2x1 gerado pelo Quartus.

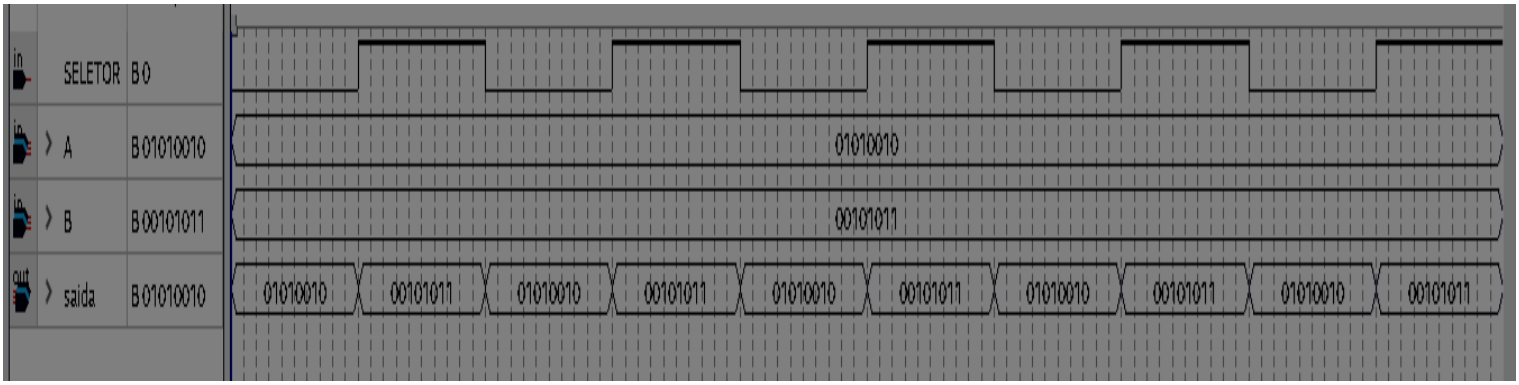


Figura 16 - waveform teste Multiplexador 2x1.

1.3.6 Multiplexador 3 x1

Sinais de entrada:

A(7..0): Recebe a primeira opcao que é selecionada caso o seletor seja 00.

B(7..0): Recebe a segunda opcao que é selecionada caso o seletor seja 01.

C(7..0): Recebe a terceira opcao que é selecionada caso o seletor seja 10.

Seletor(1..0): Recebe da unidade de controle um valor para ser selecionado.

Sinais de saida:

saida(7..0): Recebe o valor da opção selecionada.

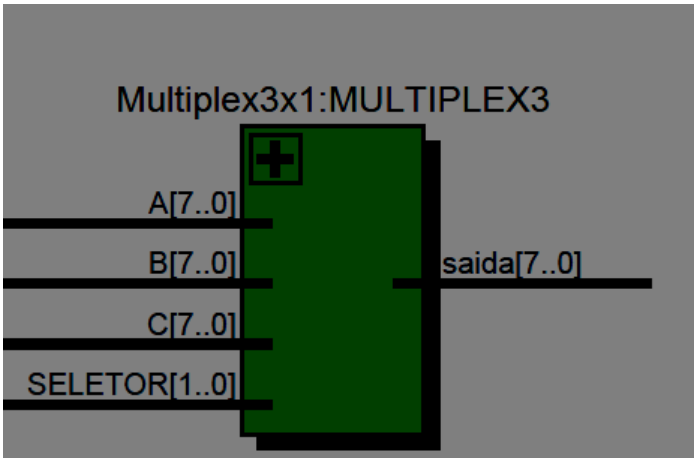


Figura 17 - Bloco simbólico do componente multiplexador_3x1 gerado pelo Quartus.

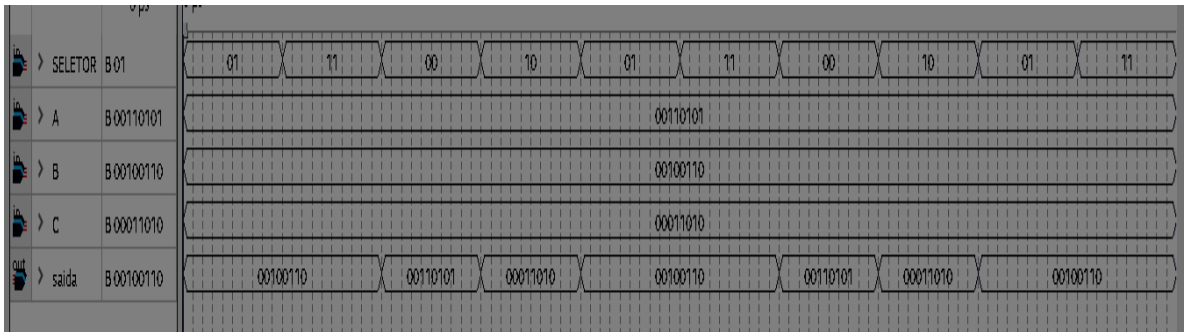


Figura 18 - waveform teste Multiplexador 3x1.

1.3.7 Multiplexador 4 x1

Sinais de entrada:

A(7..0): Recebe a primeira opcao que é selecionada caso o seletor seja 00.

B(7..0): Recebe a segunda opcao que é selecionada caso o seletor seja 01.

C(7..0): Recebe a terceira opcao que é selecionada caso o seletor seja 10.

D(7..0): Recebe a quarta opcao que é selecionada caso o seletor seja 11.

Seletor(1..0): Recebe da unidade de controle um valor para ser selecionado.

Sinais de Saida:

saida(7..0): Recebe o valor da opção selecionada.

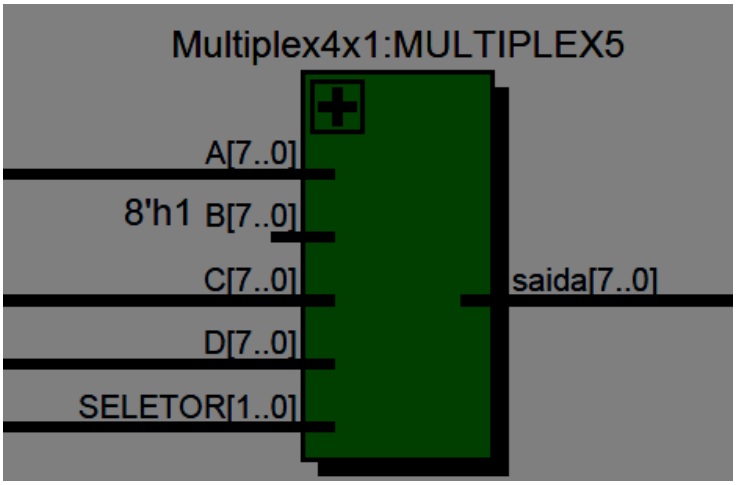


Figura 19 - Bloco simbólico do componente multiplexador_4x1 gerado pelo Quartus.

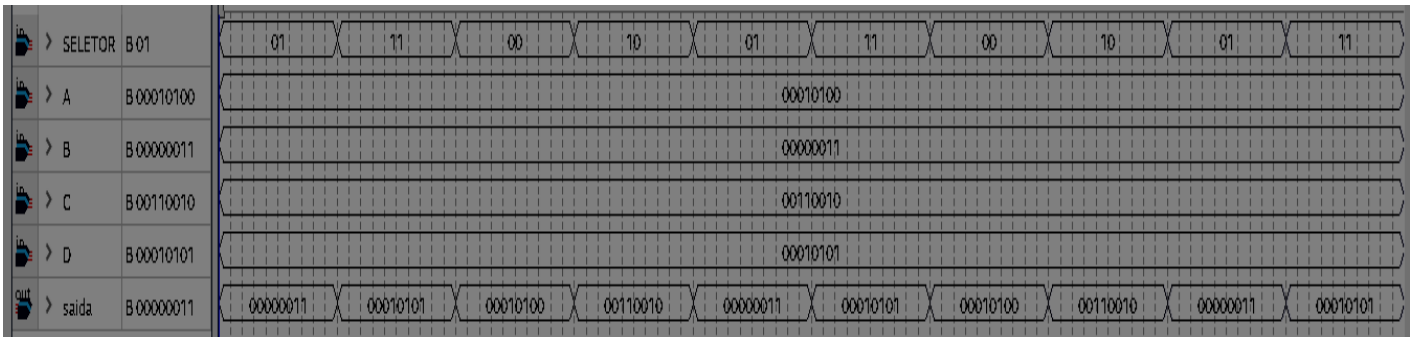


Figura 20 - waveform teste Multiplexador 4x1.

1.3.8 PC (PROGRAM COUNTER)

O componente PC é responsável por armazenar o endereço da instrução que está em execução.

Sinais de entrada:

Clock: Recebe o clock de sistema.

InputOR: Flag de ativação do PC, sendo recebido da unidade de controle.

InputPC(7..0): O endereço da próxima instrução a ser executada.

Sinais de saida:

output(7..0): Recebe o endereço instrução atual.

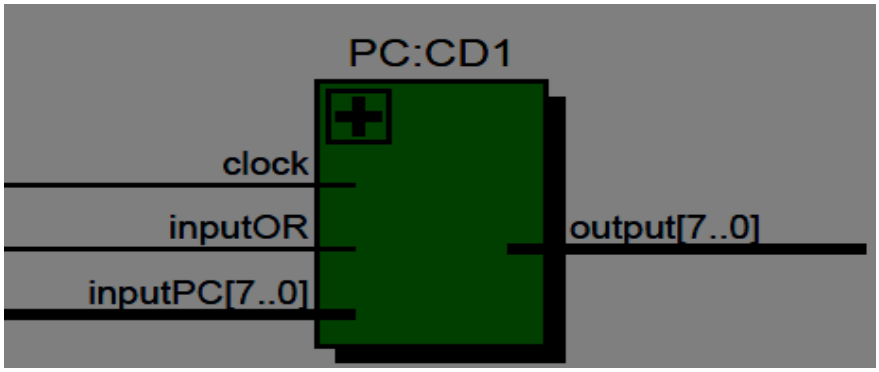


Figura 21 - Bloco simbólico do componente Program Counter gerado pelo Quartus.

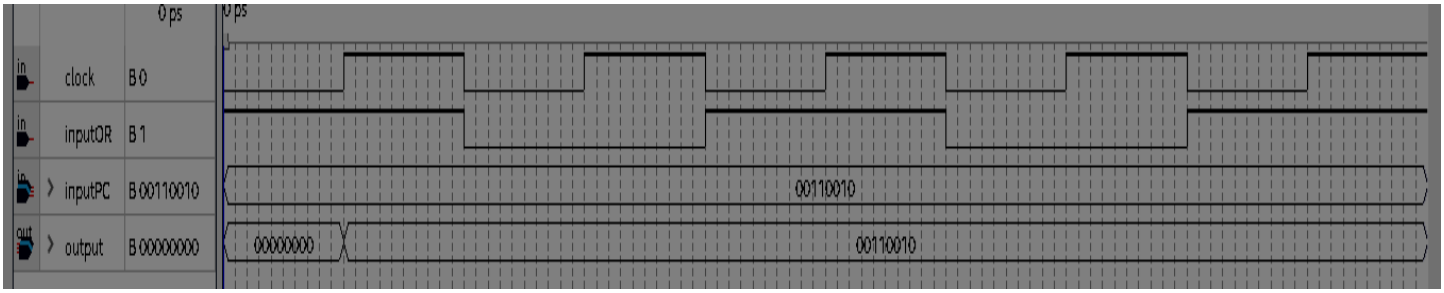
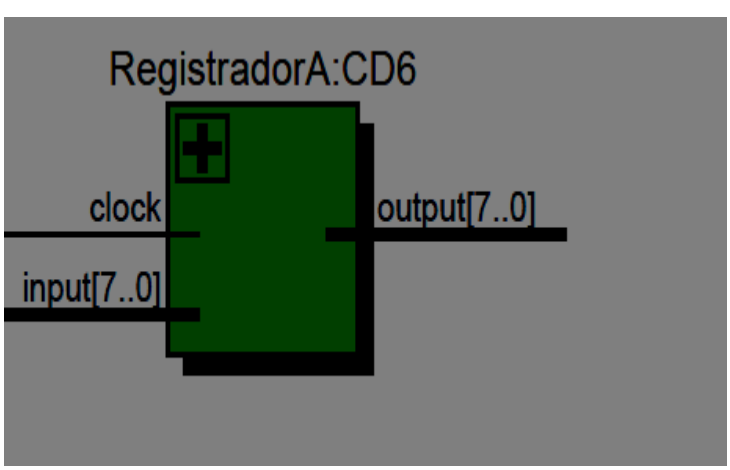


Figura 22 - waveform teste Program Counter

1.3.9 Registradores A e B

O componente Registrador A e B servem para armazenar o valores que foram lidos do banco de registradores. E garantem que eles não irão se perder com a mudança do Clock.



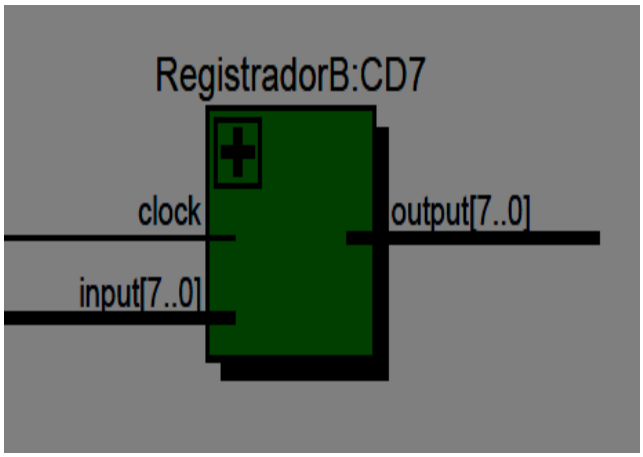


Figura 23 - Bloco simbólico do componentes Registrador A e B gerado pelo Quartus.

1.3.10 Registrador de dados da memória

O registrador de dados da memoria registra o valor do operando buscado em memoria e garante que o mesmo não irá se perder na mudança de ciclo de clock.

Sinais de entrada:

Clock: Recebe o clock do sistema.

input(7..0): Recebe o dado que foi buscado na memoria de dados.

Sinais de saida:

output(7..0): Saida do valor lido da memoria.

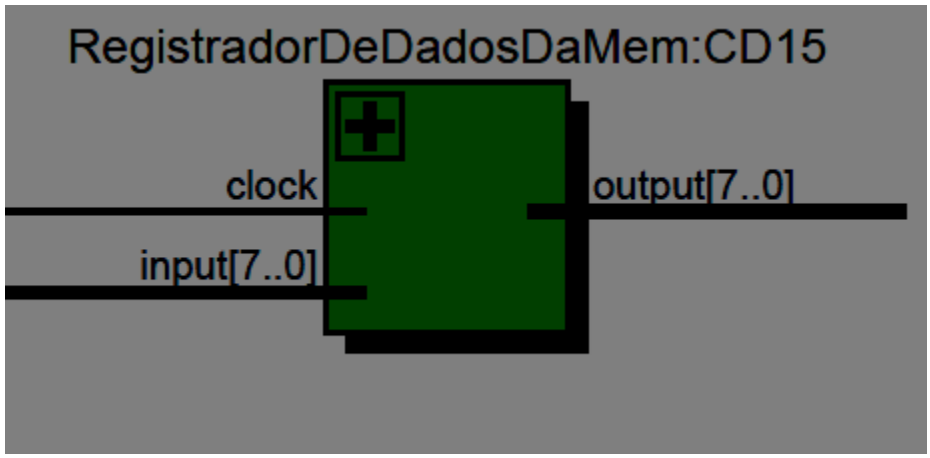


Figura 24 - Bloco simbólico do componente Registrador de dados da memória gerado pelo Quartus.

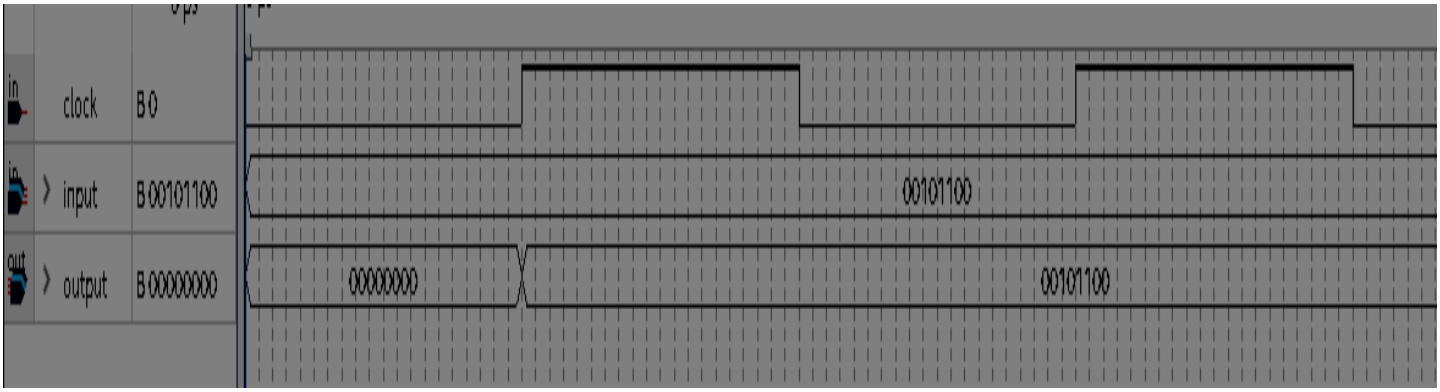


Figura 25 - waveform teste registrador de dados da memória.

1.3.11 Registrador de Instruções

O componente registrador de instruções tem como objetivo salvar a instrução e decodificar a mesma, em opcode, registrador1 e registrador2.

Sinais de entrada:

Clock: Recebe o clock do sistema.

Input(7..0): Recebe a instrução vinda da memoria de instruções.

Sinal: Flag que ativa a decodificação da instrução no registrador.

Sinais de Saida:

output_OPCODE(3..0): Recebe o opcode para enviar a unidade de controle.

output_r1: Recebe o endereco do registrador1.

output_r2: Recebe o endereco do registrador2.

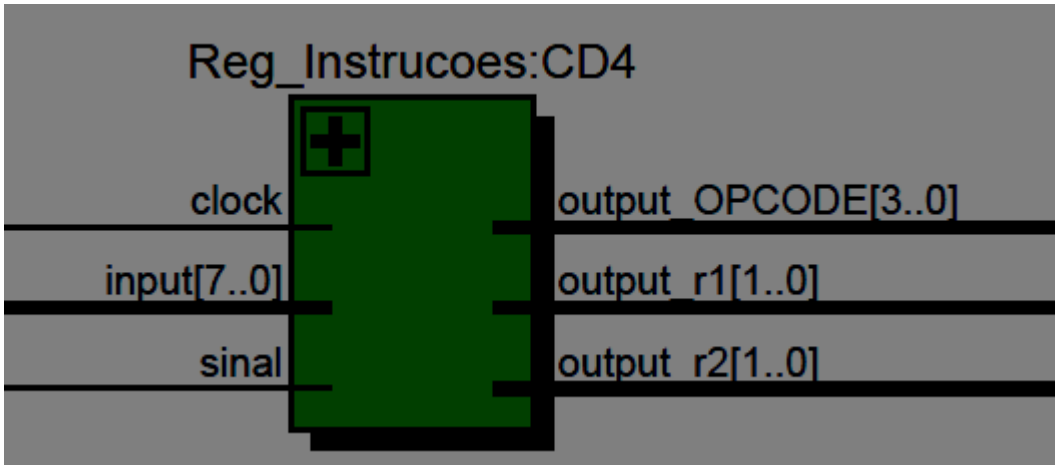


Figura 26 - Bloco simbólico do componente Registrador de Instruções gerado pelo Quartus.

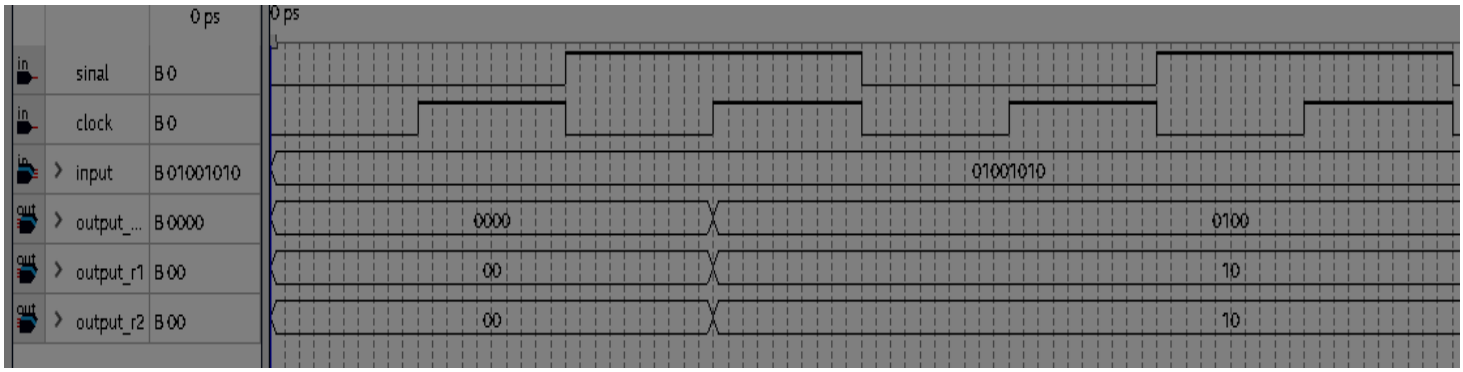


Figura 27 - waveform teste registrador Instruções.

1.3.12 Registrador Saída da ULA

O componente registrador saida da ULA recebe o resultado da ULA.

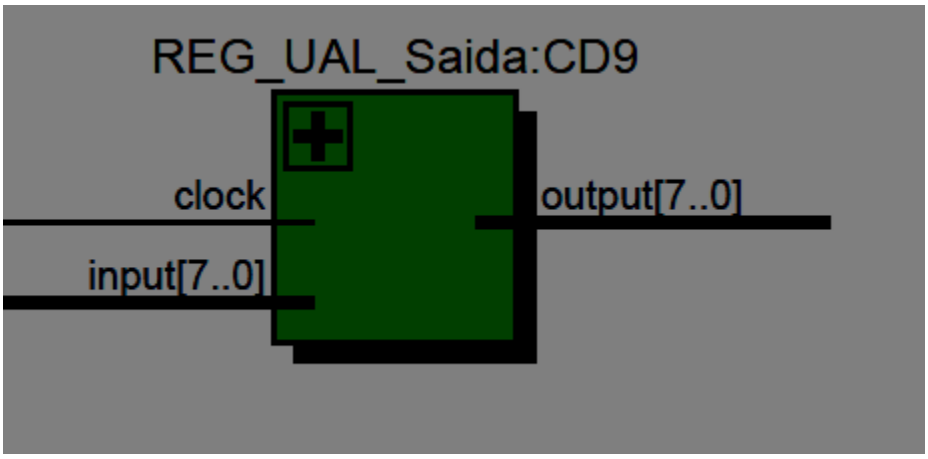


Figura 28 - Bloco simbólico do componente Registrador saída da ULA gerado pelo Quartus.

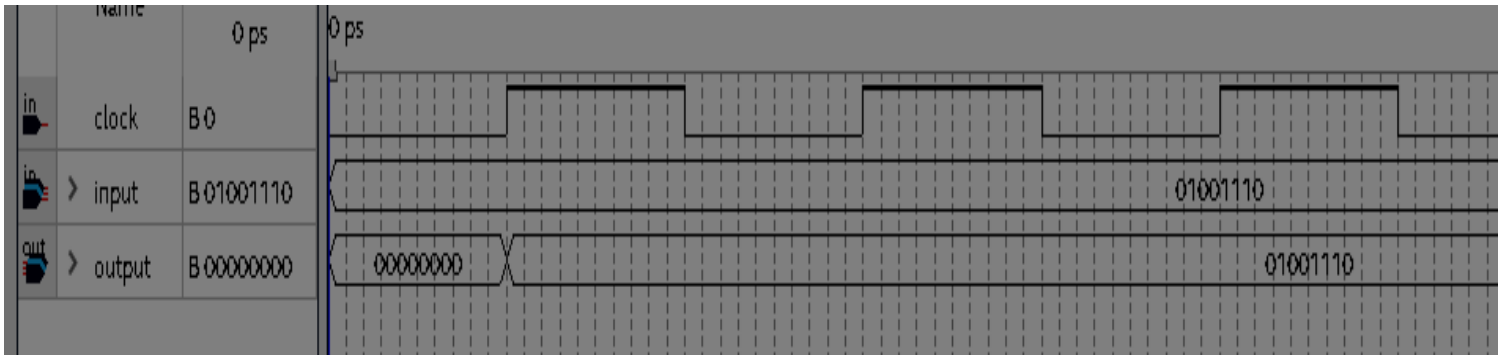


Figura 29 - waveform teste registrador Saída da ULA.

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.

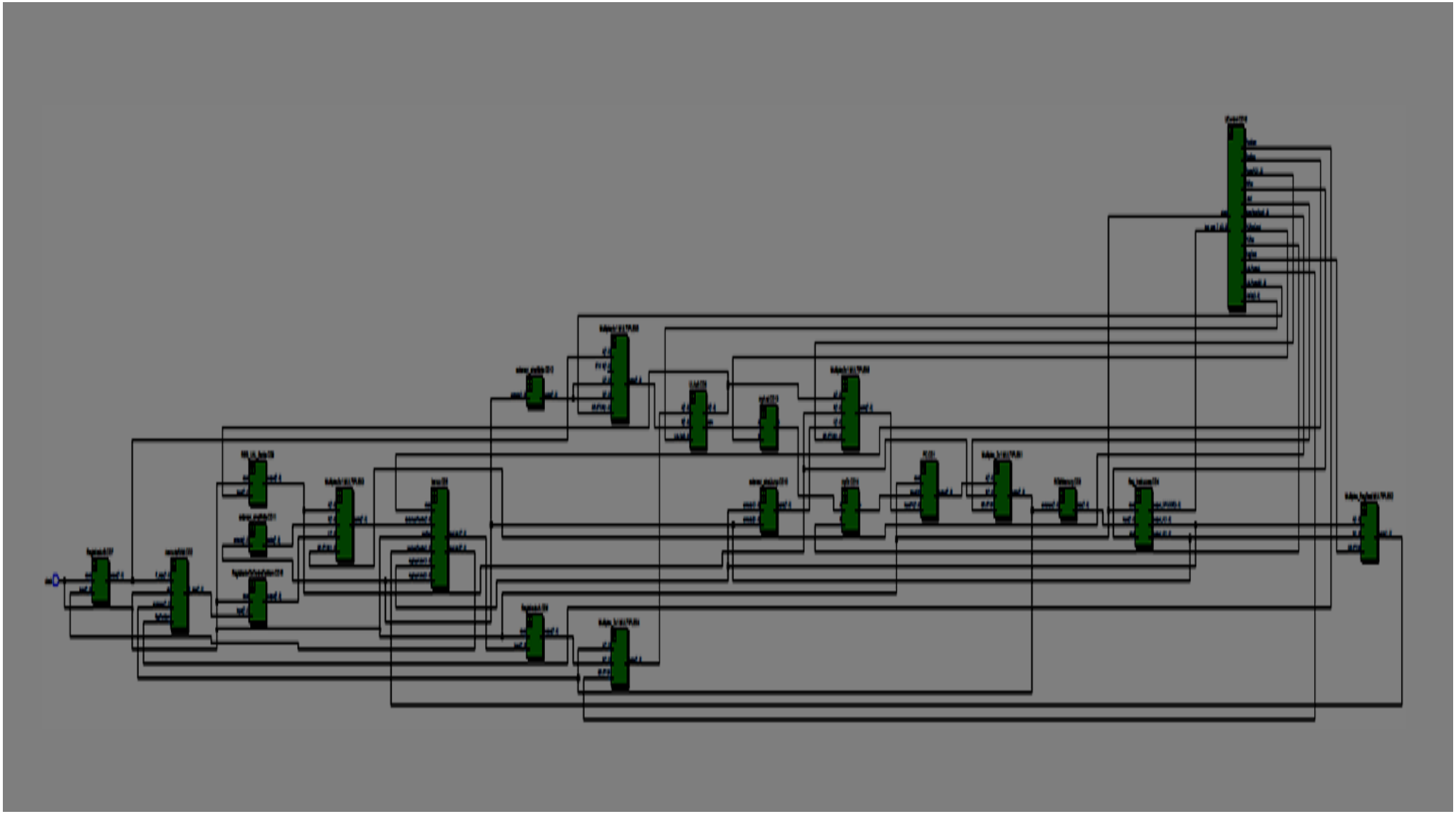


Figura 30 - Bloco simbólico do DATAPATH gerado pelo Quartus.

2 Simulações e Testes

O processador Spiron foi implementado e testado todos os componentes, porém a ultima fase de testes não pode ser realizada, portanto, não abordaremos o teste final do processador, ficando para um outro momento fazermos o teste final.

3 Considerações finais

Este trabalho veio a contribuir bastante para a equipe, podemos então ter compreendido como funciona um caminho de dados em um determinado processador, como é feita a execução de instruções no mesmo, como a ULA executa operações de soma, subtração, entre outras operações e deixou bem claro como é a Arquitetura de um Hardware e como ele se comporta. Desde já agradecemos pelo ótimo apoio do professor que contribuiu para que o trabalho desempenhado por nós fosse de sucesso.