UFRR

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

ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES

RELATÓRIO DO PROJETO: PROCESSADOR PU8B

ALUNOS:

Ilem Lima dos Santos - 2019028402
Paulo César Pereira Belmont - 2019002863

Maio de 2021
Boa Vista/Roraima

UFRR

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

ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES

RELATÓRIO DO PROJETO: PROCESSADOR PU8B

Maio 2021
Boa Vista/Roraima

Resumo

O relatório a seguir apresenta as principais características que definem o processador de 8 bits nomeado PU8B (Processador Uniciclo de 8 Bits). Serão descritas as informações julgadas pertinentes para uma visão não apenas geral, mas também a nível técnico, do processador. O componente é capaz de executar instruções básicas, como load, store, soma, subtração, beq e bne(saltos condicionais) e salto incondicional. Na ausência de uma linguagem de montagem ou alto nível, os programas ficam armazenados na descrição do componente "MemoriaRom" (Uma alternativa para armazenar os programas é mantê-los comentados no código do componente relativo à memória ROM, separados por blocos).

O processador é capaz de executar os 3 tipos de instrução encontrados em um processador de arquitetura MIPS: instruções R, as quais acessam os registradores, I, as quais acessam um registrador e um valor imediato, e J, que alteram diretamente o valor do PC, condicionalmente ou não.

O processador é Uniciclo, ou seja, cada instrução é executada em um ciclo completo, sem a quebra da mesma em passos.

Palavras-chave: Processador, VHDL, Uniciclo.

Conteúdo

Lista de Figuras

Lista de Tabelas

1. Especificação

Nesta seção é apresentado o conjunto de itens para o desenvolvimento do processador PU8B, bem como a descrição detalhada de cada etapa da construção do processador.

1.1. Plataforma de Desenvolvimento

Para a implementação do processador PU8B foi utilizado o software Quartus Prime Lite Edition 20.1, desenvolvido pela Intel. O software inclui IDE, gerador de waveforms, visualizador de RTL e outras funções úteis para o desenvolvimento do processador.

Flow Status	Successful - Sun May 16 15:22:33 2021
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	ProcessadorUniciclo
Top-level Entity Name	ProcessadorUniciclo
Family	Cyclone V
Device	5CGXFC7C7F23C8
Timing Models	Final
Logic utilization (in ALMs)	N/A
Total registers	72
Total pins	70
Total virtual pins	0
Total block memory bits	0
Total DSP Blocks	0
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0
Total DLLs	0

Figura 1 - Especificações no Quartus

1.2. Conjunto de Instruções

O processador PU8B possui 4 registradores: S0, S1, S2 e S3. Assim como 3 formatos de instruções de 8 bits cada. Seguem algumas considerações sobre as estruturas contidas nas instruções:

Tipo R:

- **Opcode (Primeiros 4 bits)**: Define qual operação em geral será executada pelo processador.
- **Reg1 (5º e 6º bits)**: Endereço do primeiro registrador a ser acessado.
- **Reg2 (7º e 8º bits)**: Endereço do segundo registrador a ser acessado.

Tipo I:

- **Opcode (Primeiros 4 bits)**: Define qual operação em geral será executada pelo processador.
- **Reg1 (5º e 6º bits)**: Endereço do registrador a ser acessado.
- **Imediato (7º e 8º bits)**: Valor utilizado como imediato.

Tipo J:

- **Opcode (Primeiros 4 bits)**: Define qual operação em geral será executada pelo processador.
- **Endereço (Últimos 4 bits)**: Endereço a ser acessado no em caso de Jump ou Branch.

Descrição dos tipos de Instruções:

Instrução do tipo R: Este formato aborda instruções de load(exceto load immediate), store e instruções baseadas em operações aritméticas que utilizam 2 registradores.

Formato para escrita de código de alto nível:

Tipo da Instrução	Reg1	Reg2

Formato para escrita em código binário:

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

Instrução do tipo I: Este formato aborda instruções que utilizam um valor carregado no próprio código (imediato).

Formato para escrita de código em alto nível:

Tipo da Instrução	Reg1	Reg2

Formato para escrita em código binário:

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

Formato do tipo J: O formato é associado à instruções de salto incondicional(jump) e salto condicional porr Branch (BEQ e BNE).

Formato para escrita de código em alto nível:

Tipo da Instrução	Endereço

Formato para escrita em código binário:

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

Visão geral das instruções do Processador PU8B:

O número de bits do campo **Opcode** das instruções é quatro, assim, obtemos um máximo $(Bit(0e1)^4 \therefore 2^4 = 16)$ de 16 Opcodes (0 a 15) que podem ser implementados. A Tabela 1 apresenta as instruções associadas aos Opcodes.

Tabela 1 – Tabela que mostra a lista de Opcodes utilizadas pelo processador PU8B.

Opcode	Nome	Formato	Nome	Exemplo
0000	LW	R	Load	lw S0, memória(00)
0001	SW	R	Store	sw S0, memória(00)
0010	ADD	R	Soma	add S0, S1, ou seja: S0 = S0 + S1
0011	SUB	R	Subtração	sub S0, S1, ou seja: S0 = S0 - S1
0100	ADDI	I	Soma imediata	addi S0, 11, ou seja: S0 = S0 + 3
0101	SUBI	I	Subtração imediata	subi S0, 11, ou seja: S0 = S0 - 3
0110	MOVE	R	Move	move S0 S1, ou seja: S0 = S1
0111	LI	I	Load Imediato	li S0, 11, ou seja: S0 = 3
1000	BEQ	J	Branch if equal	beq 0000
1001	BNE	J	Branch if not equal	bne 0000
1010	CMP	R	Comparação	cmp S0, S1

1011	JUMP	J	Salto incondicional	jump 0000
1100 a 1111	NÃO IMPLEMENTADAS	----------	----------	----------

1.3. Descrição do Hardware

Nesta seção são descritos os componentes do hardware que compõem o processador PU8B, incluindo uma descrição de suas funcionalidades e valores de entrada e saída.

1.3.1. PC

O PC, ou Program Counter, é o componente responsável por sequenciar o código. Em resumo, ao receber um clock igual a 1, o PC recebe um valor de 8 bits (0 a 255), referente ao endereço de uma instrução, e o envia para 2 componentes: o PC Counter e a Memória de Instruções.

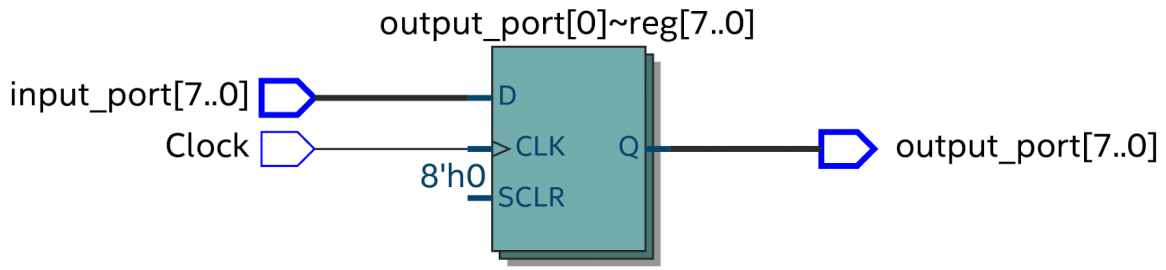

Figura 2 - RTL do PC

1.3.2. PC Counter

O PC Counter é o componente que adiciona 1 passo ao PC. Em operações normais, ou seja, sem saltos, ele apenas adiciona 1 ao endereço recebido pelo PC, fazendo o programa armazenado avançar 1 passo.O componente recebe o endereço do PC e adiciona 1 a esse valor.

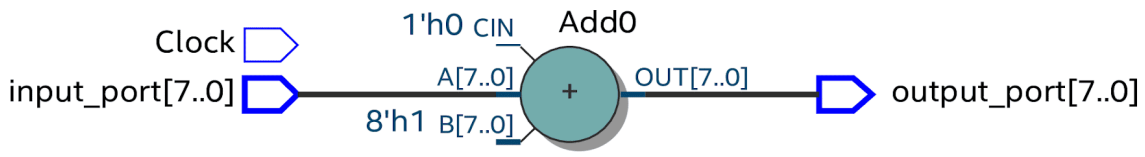

Figura 3 - RTL do PC Counter

1.3.3. Divisor de Instruções

O divisor de instruções é o principal componente em termos de barramento. É garantida a ele a responsabilidade de enviar os bits de uma instrução a seus determinados componentes. Ao receber 8 bits, ele direciona os primeiros 4 bits (7 downto 4) à Unidade de Controle, os próximos 2 bits (3 downto 2) para o Banco de Registradores, entrada Reg1, e os últimos 2 bits (1 downto 0) para a entrada Reg 2 do Banco de Registradores. Também envia esses últimos 2 bits (1 downto 0) para um multiplexador que decide se utilizará a 2ª entrada da ALU como um valor imediato ou um valor carregado do Banco de Registradores. Em caso de jump, só os últimos 4 bits funcionarão como o endereço a ser acessado.

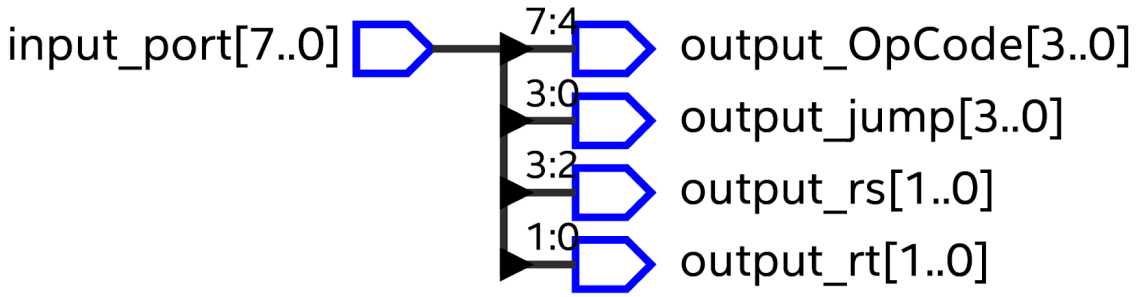

Figura 4 - RTL do Divisor de Instruções

1.3.4. Unidade de Controle

O componente Unidade de Controle controla outros componentes do processador através do valor relacionado às suas flags, que por sua vez estão relacionadas com o Opcode recebido pelo componente.

As flags da Unidade de Controle são as seguintes:

- **Jump**: Define se o próximo endereço será o do PC Counter ou um endereço acessado diretamente por salto.
- **Branch**: Semelhante ao Jump, mas depende de uma condição para saltar.
- **MemRead**: Carrega um valor acessado na RAM.
- **MemToReg**: Selecionar de onde vem o valor a ser escrito em um registrador: RAM ou resultado da ALU
- **ALUOp**: Define a operação a ser executada na ALU.
- **MemWrite**: Define que será registrado na RAM um valor vindo da ULA ou de um registrador.

- **AluSRC**: Define se a segunda entrada da ALU será o dado de um registrador. Ou um valor imediato.
- **RegWrite**: Ativa a escrita de dados no registrador.

A seguinte tabela faz a associação entre os Opcodes e suas respectivas combinações de flags.

Tabela 2 - Detalhes das flags de controle do processador.

Instrução	Jump	Branch	MemRead	MemtoReg	ALUOp	MemWrite	ALUSrc	RegWrite
lw	0	0	1	1	000	0	0	1
sw	0	0	0	0	000	1	0	0
add	0	0	0	0	001	0	0	1
sub	0	0	0	0	010	0	0	1
addi	0	0	0	0	001	0	1	1
subi	0	0	0	0	010	0	1	1
move	0	0	0	0	011	0	0	1
li	0	0	0	0	011	0	1	1
beq	0	1	0	0	100	0	0	0
bne	0	1	0	0	101	0	0	0
cmp	0	0	0	0	110	0	0	0
j	1	0	0	0	111	0	0	0

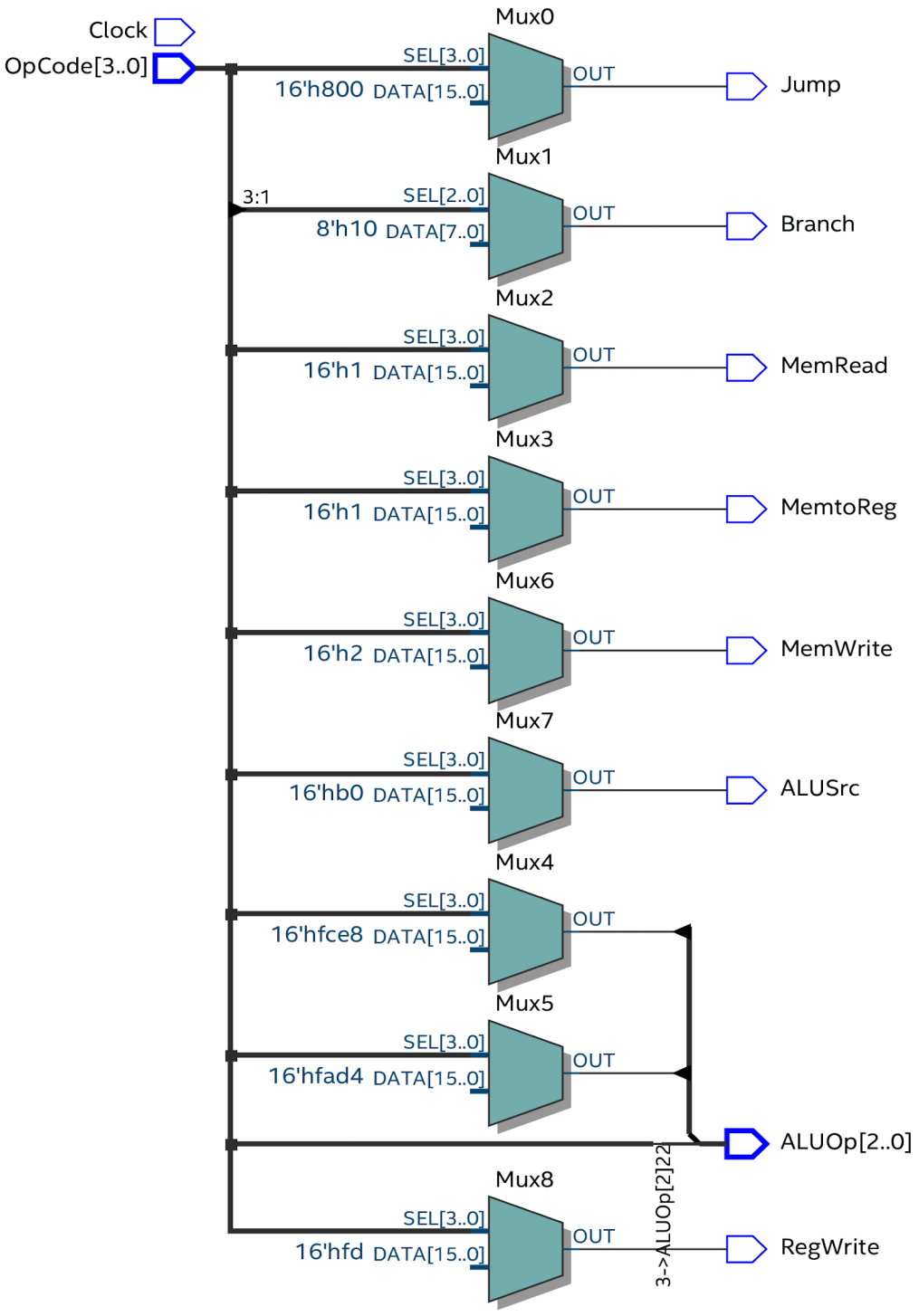

Figura 5 - RTL da Unidade de Controle

1.3.5. Memória de Instruções

A memória de instruções, conhecida também por ROM, é o componente responsável por armazenar os passos e instruções relativas aos mesmos. Será a ROM que enviará para o divisor uma instrução associada ao passo recebido pelo PC.

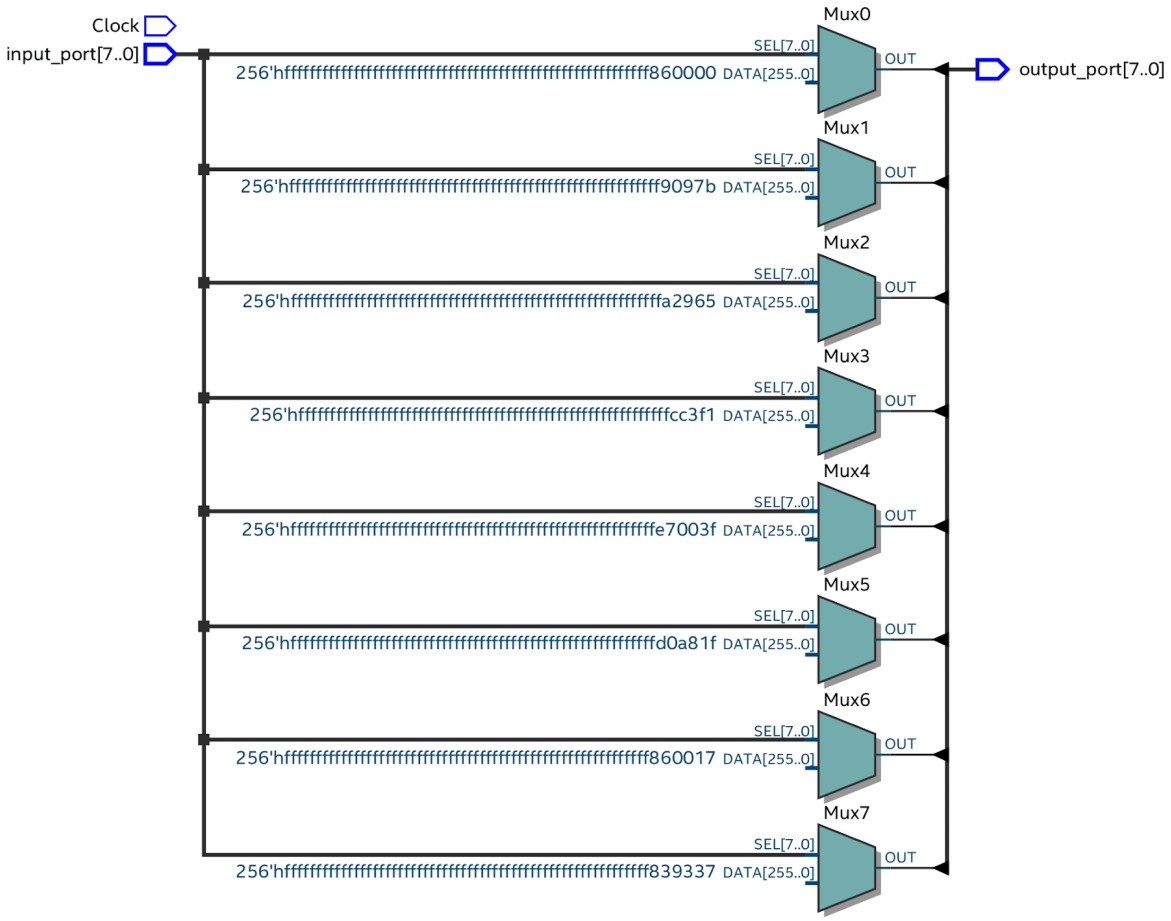

Figura 6 - RTL da Memória de Instruções

1.3.6. Memória de Dados

A memória de dados ou é o componente associado ao armazenamento de dados em tempo de execução, lembrando a memória RAM. Existem apenas 4 registradores para armazenamento de valores. Assim, para ampliar o alcance de armazenamento, existe tal componente. A memória RAM tem 2 entradas: Uma recebe um dado imediato, enquanto a outra recebe um resultado da ALU. A sua saída é um dado a ser lido.

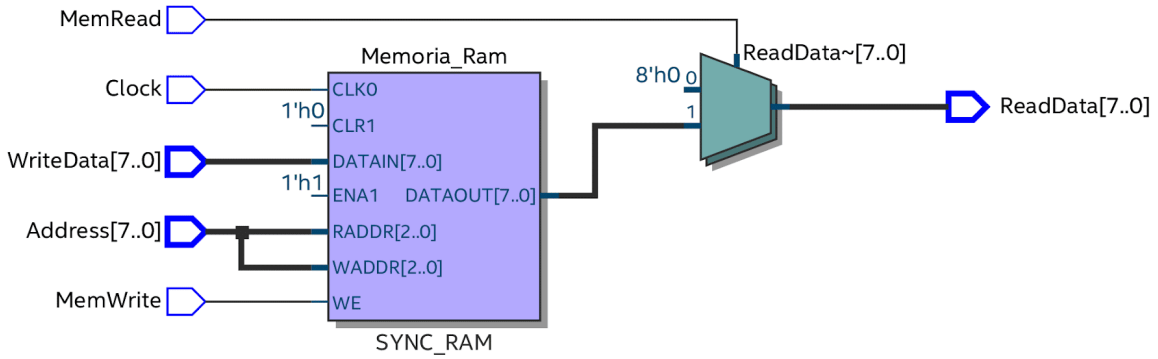

Figura 7 - RTL da Memória de Dados

1.3.7. Banco de Registradores

O banco de registradores acessa um registrador a partir do endereço recebido. Nesse processador, existem 4 (2^2) registradores, por haver 2 bits disponíveis para o acesso de cada registrador.

Em instruções do tipo R, 4 dos 8 bits são endereçados a este componente, que buscará o registrador por seu endereço, obterá os dados desse, e os enviará para o próximo componente.

Cada par de bits se relaciona a uma entrada, Reg1 ou Reg2.

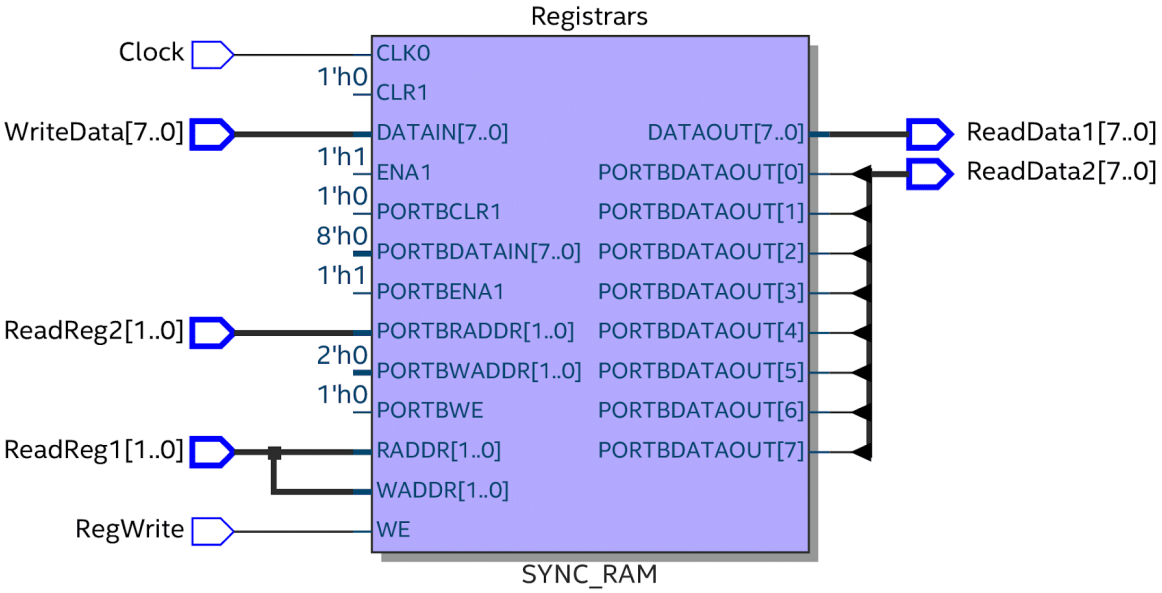

Figura 8 - RTL do Banco de Registradores

1.3.8. ALU

A **ALU** (Arithmetic Logical Unity) ou **ULA** (Unidade lógico-aritmética) tem como objetivo principal efetuar as operações de cunho aritmético e lógico. Ao receber a flag **ALUOp** (3 bits), enviada pela Unidade de Controle, o componente sabe qual operação executar. A ALU recebe 3 valores de entrada: A flag ALUOp, o primeiro valor, vindo de um registrador, e o segundo valor, acessado ou de um segundo registrador (Tipo R), ou da instrução (Tipo I). Após as operações, a ALU envia o resultado para ser escrito na memória RAM(quando a flag MemWrite está ativada) e para o registrador(quando a flag MemToReg estiver habilitada). Também há uma saída **Zero** , utilizada em comparações, de 1 bit. Essa saída define se ocorrerá um salto de endereço, dada uma operação de comparação (beq, bne ou cmp).

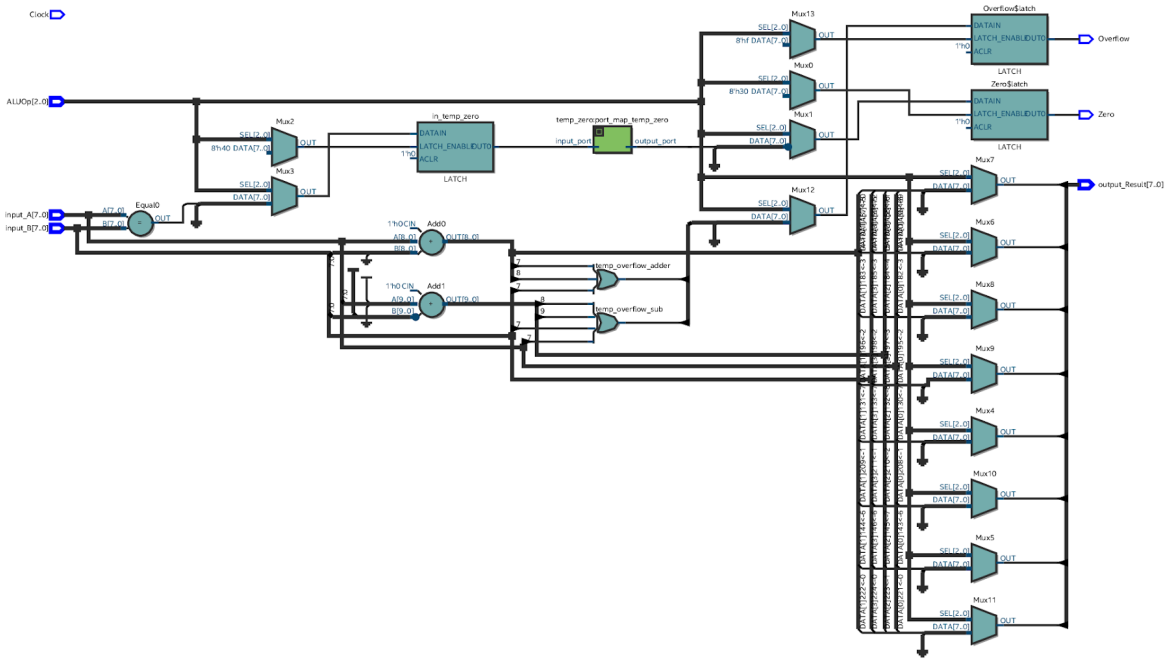

Figura 9 - RTL da ALU

1.3.9. Subcomponentes

Os subcomponentes são os componentes que foram avaliados suficientemente simples para serem indexados no mesmo escopo dos outros. Porém, isso não diminui a importância de sua operação no processador como um todo.

1.3.9.1. AND_GATE

Esse subcomponente executa a mesma função de uma porta AND, sendo positiva apenas se receber 2 bits que valham 1 cada.

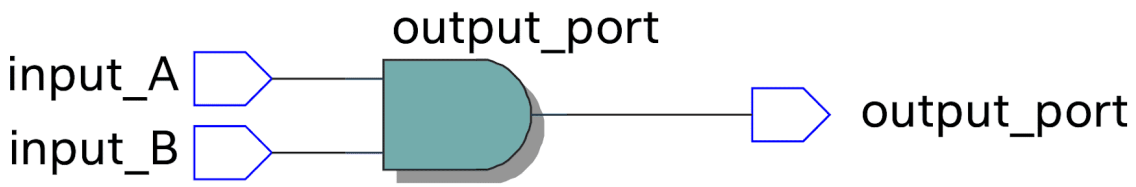

Figura 10 - RTL do AND_Gate

1.3.9.2. Extensores

Os extensores servem para promover a compatibilidade das informações, provendo o tamanho necessário em bits para que os componentes recebam o comprimento de informação esperado. Foram implementados extensores 2 to 8 e 4 to 8.

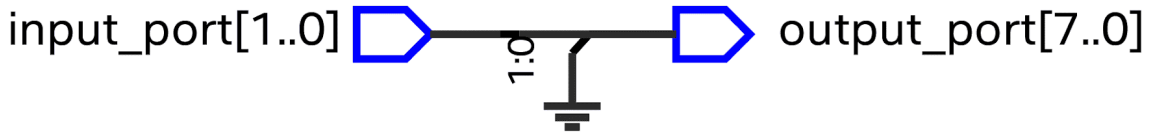

Figura 11 - RTL do extensor 2 to 8

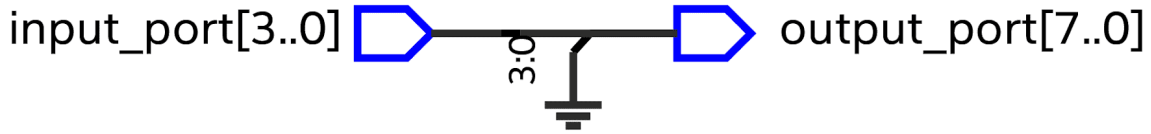

Figura 12 - RTL do extensor 4 to 8

1.3.9.3. Zero

O zero fornece uma flag que indica se um valor é igual ou diferente do que foi comparado. O componente Zero fica dentro da ULA, e é utilizado apenas no caso de operações comparativas. Sua função é apenas inicializar a flag necessária para realizar a comparação.

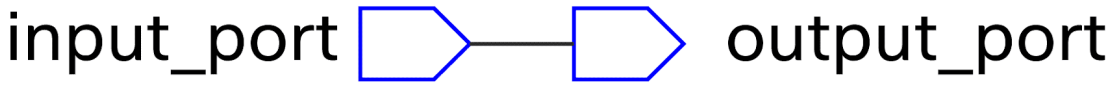

Figura 13 - RTL do temp_zero

1.3.9.4. Mux 2 to 1

Os multiplexadores são utilizados na decisão de valores baseados em uma flag, que decidem qual valor sairá no output.

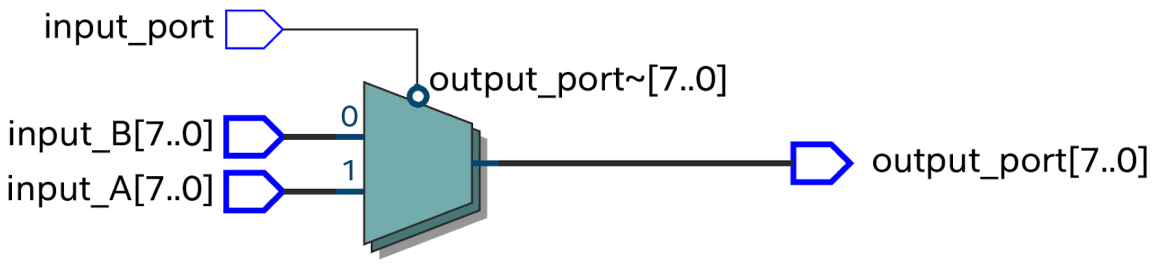

Figura 14 - RTL do Multiplexador 2 to 1

1.3.10. Clock

O clock não foi implementado como um componente físico, porém, é de suma importância para o funcionamento do processador, já que fica responsável pelo controle de ciclos da unidade, simulando os clocks. O clock fica interligado a maioria dos componentes, indicando quando o processador está em operação.

1.4. Datapath

O Datapath é a representação das conexões e barramentos presentes no processador, ligados a uma unidade que gerencia o funcionamento do mesmo (clock). A ferramenta Quartus disponibiliza uma visão em datapath do processador gerada a partir dos códigos que o descreveram, representada em RTL (Register Transfer Language ou Linguagem de Transferência entre Registradores). A figura a seguir exibe a visão geral do processador.

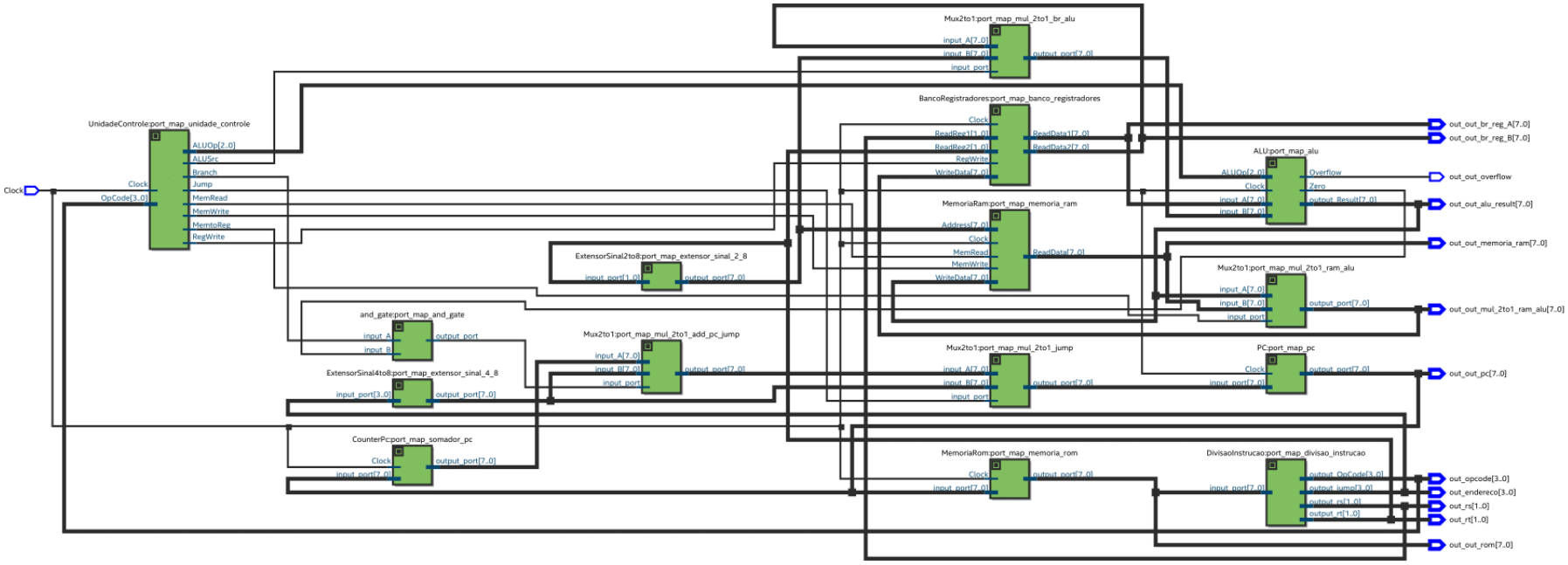

Figura 15 - RTL do PU8

2. Simulações e Testes

Objetivando demonstrar o funcionamento do processador PU8B, foi aplicado a ele um programa bastante conhecido, que exibe a sequência de Fibonacci.

Tabela 3 - Código Fibonacci para o processador PU8B.

Endereço	Instrução	Alto Nível	Binário		
			Opcode	Reg2 / Imediato / Endereço	Reg1
0	01111111	li S3, 3	0111	11	11
1	01001111	addi S3, 3	0100	11	11
2	00101111	add S3, S3	0010	11	11
3	01001100	addi S3, 0	0100	11	00
4	01001101	addi S3, 1	0100	11	01
5	01111001	li S2, 1	0111	10	01
6	01110000	li S0, 0	0111	00	00
7	00010000	sw S0, ram(00)	0001	00	00
8	01110001	li S0, 1	0111	00	01
9	00010001	sw S0, ram(01)	0001	00	01
10	00000000	lw S0, ram(00)	0000	00	00
11	01100100	move S1, S0	0110	01	00
12	00000001	lw S0, ram(01)	0000	00	01
13	00100100	add S1, S0	0010	01	00
14	00010000	sw S0, ram(00)	0001	00	00
15	00010101	sw s1, ram(01)	0001	01	01
16	01001001	addi s2, 1	0100	10	01
17	10101011	cmp S2, S3	1010	10	11
18	10011010	bne 1010	1001	10	10
19	01110000	li S0, 0	0111	00	00
20	01110100	li S1, 0	0111	01	00
21	01111000	li S2, 0	0111	10	00
22	01111100	li S3, 0	0111	11	00

2.1. Descrição do Programa

O programa acima descrito representa o cálculo da sequência de Fibonacci. Os 4 registradores foram utilizados nesse programa, sendo que S0 foi utilizado para acessar os valores da RAM, S1 como auxiliar da soma, S2 como contador e, finalmente, S3 para o

número Fibonacci objetivado. Enquanto o programa avança, são armazenados na memória RAM o último número da sequência e seu anterior.

2.2. Waveform

Verificação dos resultados no relatório da simulação: Após a simulação ser concluída, temos a seguinte waveform:

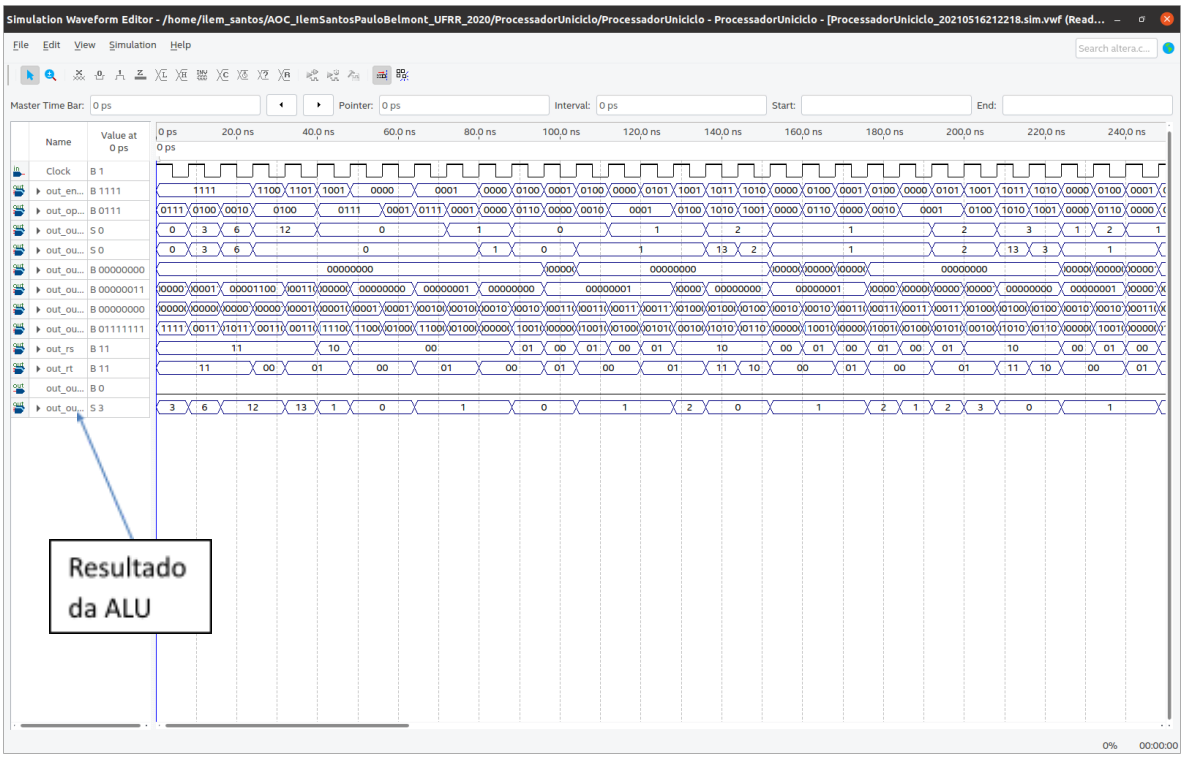

Figura 16 - Resultado na Waveform.

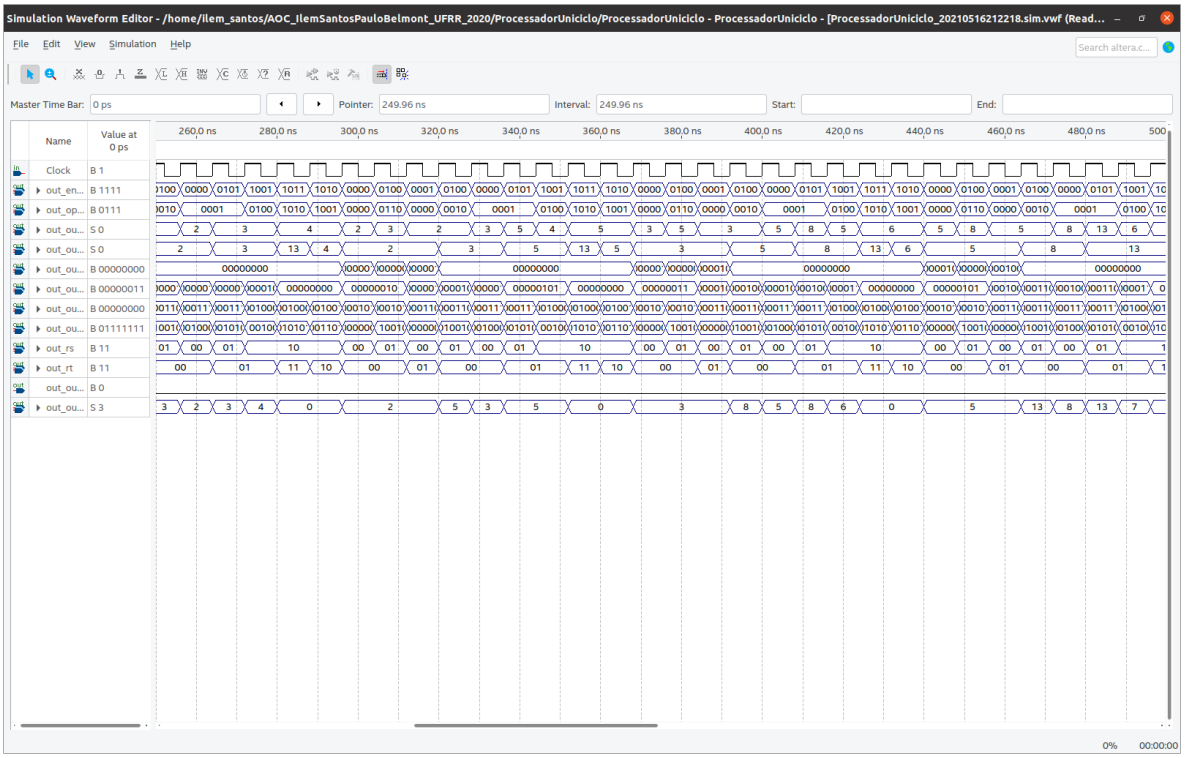

Figura 17 - Resultado na waveform.

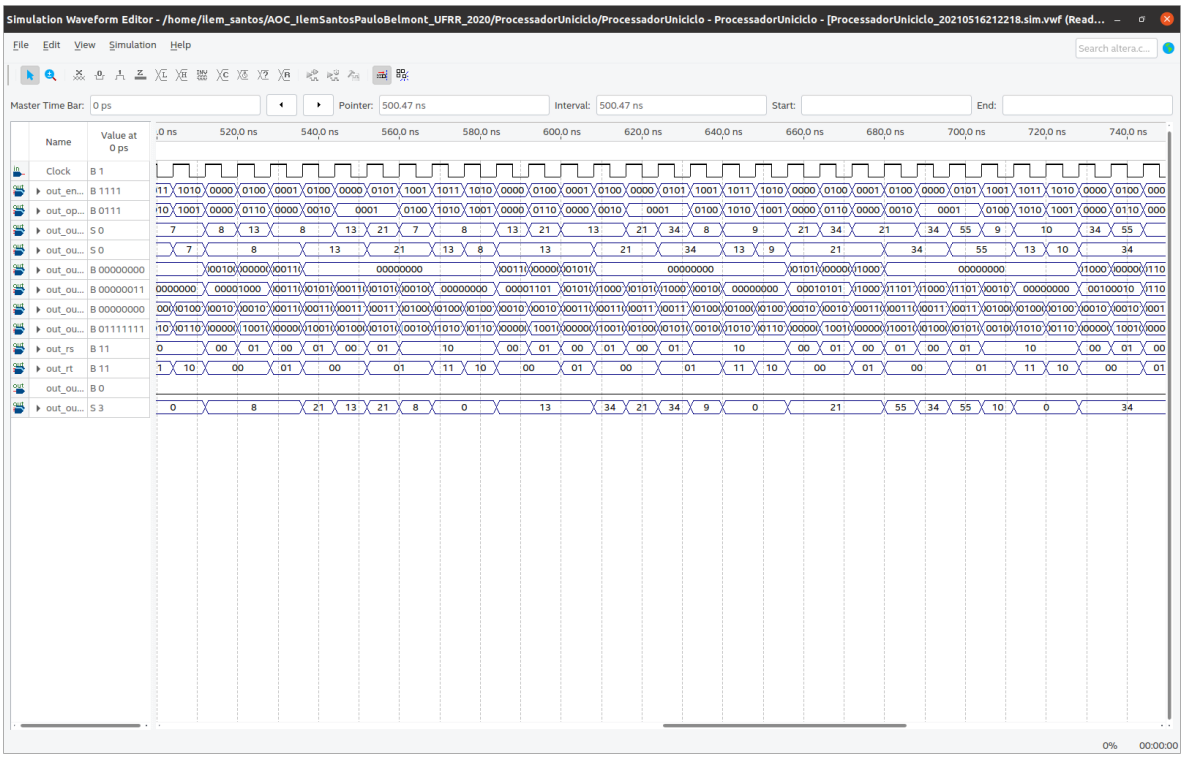

Figura 18 - Resultado na waveform.

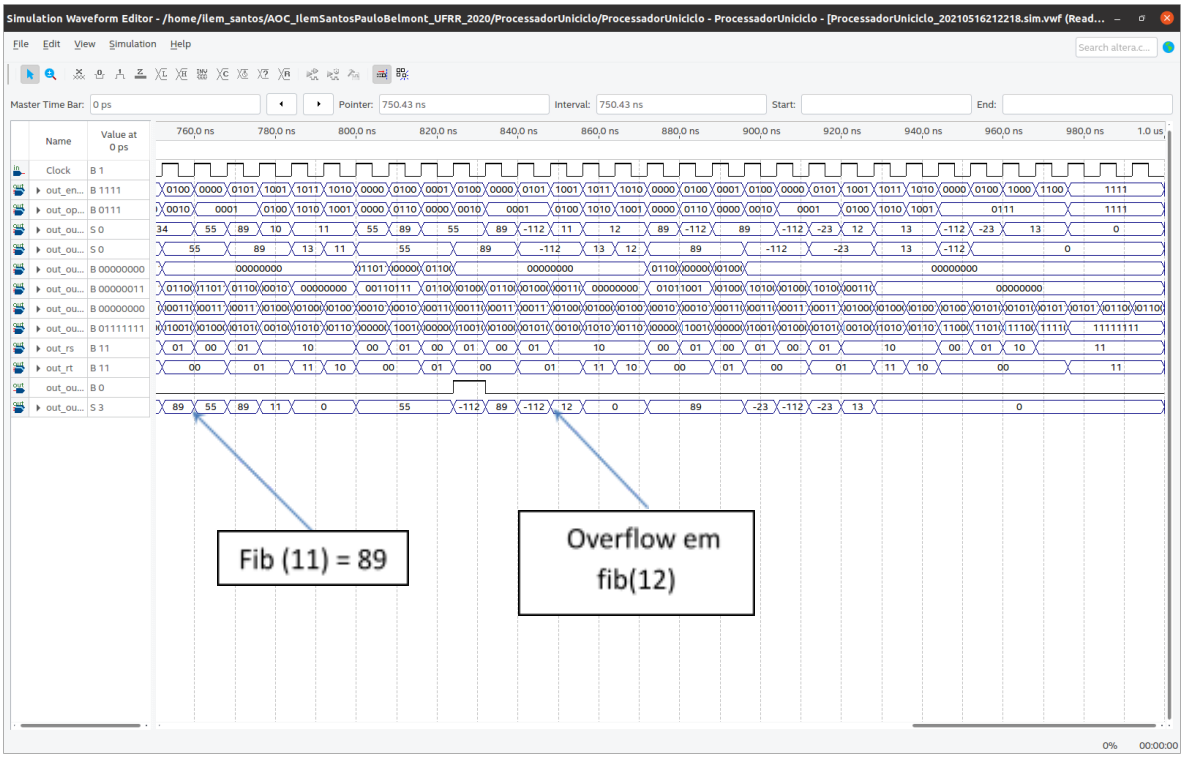

Figura 19 - Resultado na waveform.

3. Limitações Operacionais

O processador PU8B, como todo processador, possui alguns limites que o impedem de gerar certos resultados de forma correta, principalmente (nesse caso) por causa do seu tamanho de 8 bits suportados por instrução. Uma das limitações mais notáveis está no endereço a ser acessado pelos saltos, considerando que, mesmo que hajam 8 bits para armazenar até 256 passos, apenas 4 bits podem ser direcionados à seleção de um endereço, dando à instrução jump apenas o espaço entre os passos 0 e 15. Outra limitação é a que resulta em overflow, ou seja, o número resultado da ALU é maior que 8 bits, sendo impossível lidar com tal valor. Como foi optado por trabalharmos com números negativos, números que ultrapassem os valores decimais -128 ou 127 resultarão em overflow da ALU. Uma pino de nome overflow foi criado para exibir na waveform o overflow, caso ocorra.

4. Considerações Finais

Foi apresentado nesse relatório o processador PU8B(Processador Uniciclo de 8 bits) que, apesar de suas limitações, executa corretamente as 11 operações nele implementadas. As limitações mais notáveis, como a baixa janela de valores que podem sair da ALU e a baixa capacidade de transmissão de informações em bits reduzem o escopo de programar que podem ser implementados no PU8B. Todavia, o processador age exatamente como um processador Uniciclo deveria agir em suas instruções, podendo armazenar dados de até 8 bits em seus registradores, além de ter um espaço de armazenamento na memória de dados suficiente para suprir as operações que forem executadas nele.

5. Repositório do projeto

https://github.com/IlemSantos/AOC_IlemSantosPauloBelmont_UFRR_2020