



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

ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES

RELATÓRIO DO PROJETO: PROCESSADOR BORI

ALUNOS:

**Rosialdo Queivison Vidinho de Queiroz Vicente – 20200188122
Venícius Jacob Pereira de Oliveira - 2020014633**

**Março de 2022
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 BORI

**Março de 2022
Boa Vista/Roraima**

Resumo

Este trabalho aborda o projeto e implementação de um processador uniciclo 8 bits nomeado de BORI, processador esse baseado na arquitetura MIPS. Para o seu desenvolvimento foi utilizada a linguagem VHDL (VHSIC Hardware Description Language), o programa para escrever os componentes e rodar os testes foi o Quartus Prime Lite 20.1. Sendo esse processador capaz de executar as instruções básicas como load, store, soma, subtração e os saltos condicionais (beq e bne). Efetua instruções do tipo R, I e J que estão no MIPS.

Sumário

1. Especificação	7
1.1 Plataforma de desenvolvimento	7
1.2 Conjunto de instruções	8
1.3 Descrição do Hardware	10
1.3.1 And	10
1.3.2 Divisor de instruções	10
1.3.3 Banco de registradores.....	11
1.3.4 Counter PC	11
1.3.5 Memória RAM	12
1.3.6 Temp Zero	12
1.3.7 Multiplexador 2x1	12
1.3.8 ULA.....	13
1.3.9 Program Counter.....	13
1.3.10 Unidade de Controle	14
1.3.11 Clock.....	15
1.3.12 Memória de instrução	16
1.3.13 Extensor de sinal	17
1.4 Datapath	17
2 Simulações e testes	18
3 Considerações Finais	19
4 Repositório	19

Lista de Figuras

77

FIGURA 2 - PORT AND **ERRO! INDICADOR NÃO DEFINIDO.**10

FIGURA 3 – DIVISOR DE INSTRUÇÕES **ERRO! INDICADOR NÃO DEFINIDO.**10

711

FIGURA 5 – COUNTER PC **ERRO! INDICADOR NÃO DEFINIDO.**11

FIGURA 6 – MEMÓRIA DE DADOS. **ERRO! INDICADOR NÃO DEFINIDO.**2

712

FIGURA 8 – MULTIPLEXADOR 2x1 **ERRO! INDICADOR NÃO DEFINIDO.**12

FIGURA 9 - ULA **ERRO! INDICADOR NÃO DEFINIDO.**

713

FIGURA 11 – UNIDADE DE CONTROLE **ERRO! INDICADOR NÃO DEFINIDO.**15

FIGURA 12 – MEMÓRIA DE INSTRUÇÃO **ERRO! INDICADOR NÃO DEFINIDO.**16

717

FIGURA 14 – EXTENSOR 4x8 **ERRO! INDICADOR NÃO DEFINIDO.**17

FIGURA 15 – DATAPATH RTL VIEWER. **ERRO! INDICADOR NÃO DEFINIDO.**7

718

FIGURA 17 – OVERFLOW FIBONACCI **ERRO! INDICADOR NÃO DEFINIDO.**19

Lista de Tabelas

TABELA 1 – TABELA LISTA DE OPCODES UTILIZADAS PELO PROCESSADOR BORI..**ERRO! INDICADOR NÃO DEFINIDO.**9

TABELA 2 - FLAGS DE CONTROLE DO PROCESSADOR..**ERRO! INDICADOR NÃO DEFINIDO.**14

1. Especificação

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

1.1 Plataforma de desenvolvimento

Para a implementação do processador BORI foi utilizado a IDE: Intel Quartus Prime Lite 20.1, tanto para o desenvolvimento dos códigos dos componentes, como também para os testes realizados.

Flow Status	Successful - Tue Mar 08 18:37:38 2022
Quartus Prime Version	20.1.0 Build 711 06/05/2020 SJ Lite Edition
Revision Name	BORI
Top-level Entity Name	Main
Family	Cyclone V
Device	5CGXFC7C7F23C8
Timing Models	Final
Logic utilization (in ALMs)	N/A
Total registers	32
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 BORI possui 4 registradores: S0, S1, S2 e S3. 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:

Tipo de Instruções:

- **Formato do tipo R:** Este formatado aborda instruções de Load (exceto *load Immediately*), Store e instruções baseadas em operações aritméticas.

- **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;

Formato para escrita de código na linguagem Quantum:

Tipo da Instrução	Reg1	Reg2
-------------------	------	------

Formato para escrita em código binário:

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

- **Formato do tipo I:** As instruções do tipo I tem como proposito utilizar instruções com valores imediatos, do qual é um valor que é carregado do código.

- Opcode: a operação básica a ser executada pelo processador, tradicionalmente chamado de código de operação;
- Reg1: Endereço do primeiro registrador;
- Reg2: vai ter o valor para uso do imediato.

Formato para escrita de código na linguagem Quantum:

Tipo da Instrução	Reg1	Reg2
-------------------	------	------

Formato para escrita em código binário:

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

- **Formato do tipo J:** Esse formato é responsável pelas ações de salto incondicional, como por exemplo o Jump.

- **Opcode:** a operação básica a ser executada pelo processador, tradicionalmente chamado de código de operação;
- **Endereço:** Esse vai ser o endereço que será utilizado para Jump e Branch.

Formato para escrita de código na linguagem Quantum:

Tipo da Instrução	Reg1	Reg2
-------------------	------	------

Formato para escrita em código binário:

4 bits	4 bits
7-4	3-0
Opcode	Endereço

Visão geral das instruções do Processador BORI:

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

Tabela 1 – Tabela lista de Opcodes utilizadas pelo processador BORI.

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
0011	SUB	R	Subtração	sub S0, S1
0100	ADDI	I	Soma imediata	addi S0, 11
0101	SUBI	I	Subtração imediata	subi S0, 11
0110	MOVE	R	Move	move S0, S1
0111	LI	I	Load Imediato	li S0, 11
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

1.3 Descrição do Hardware

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

1.3.1 And

A porta AND é um sub componente que recebe dois bits e somente retorna um bit igual a 1 se os dois bits recebidos forem iguais a 1

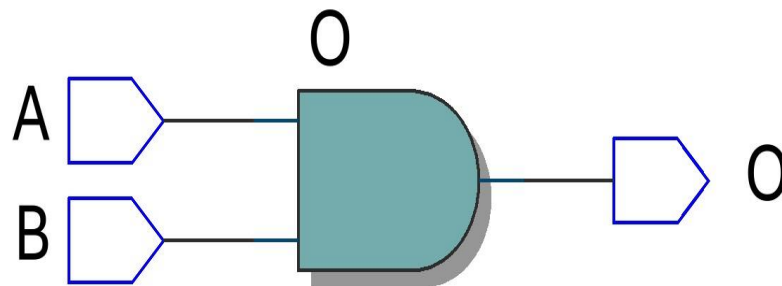


Figura 2 – Porta And

1.3.2 Divisor de instruções

É o componente que fica responsável pela distribuição de bits para os outros componentes, ele recebe 8 bits e direciona eles para os respectivos fazendo com que a instrução seja realizada. No caso de ser uma instrução de Jump, apenas os últimos 4 bits irão funcionar como um endereço a ser acessado.

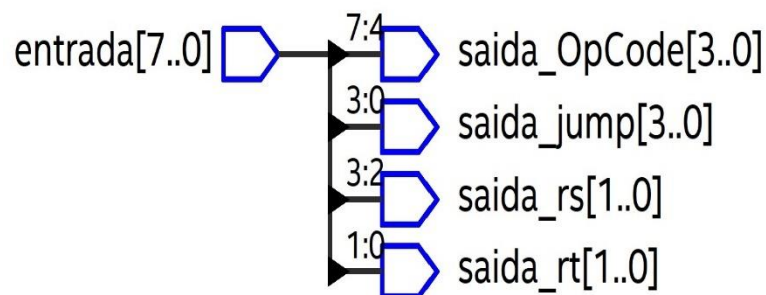


Figura 3 – Divisor de Instruções

1.3.3 Banco de registradores

Ele acessa um determinado registrador através do endereço que recebeu, nossa escolha por deixar 2 bits disponível para cada registrador, fez com que fosse possível ter $4(2^2)$ registradores. Nas instruções do tipo R, 4 dos 8 bits disponíveis vão ser direcionados para esse componente, fazendo com que ele busque os registradores pelo endereço dele, obtenha seus dados e envie eles para o próximo componente.

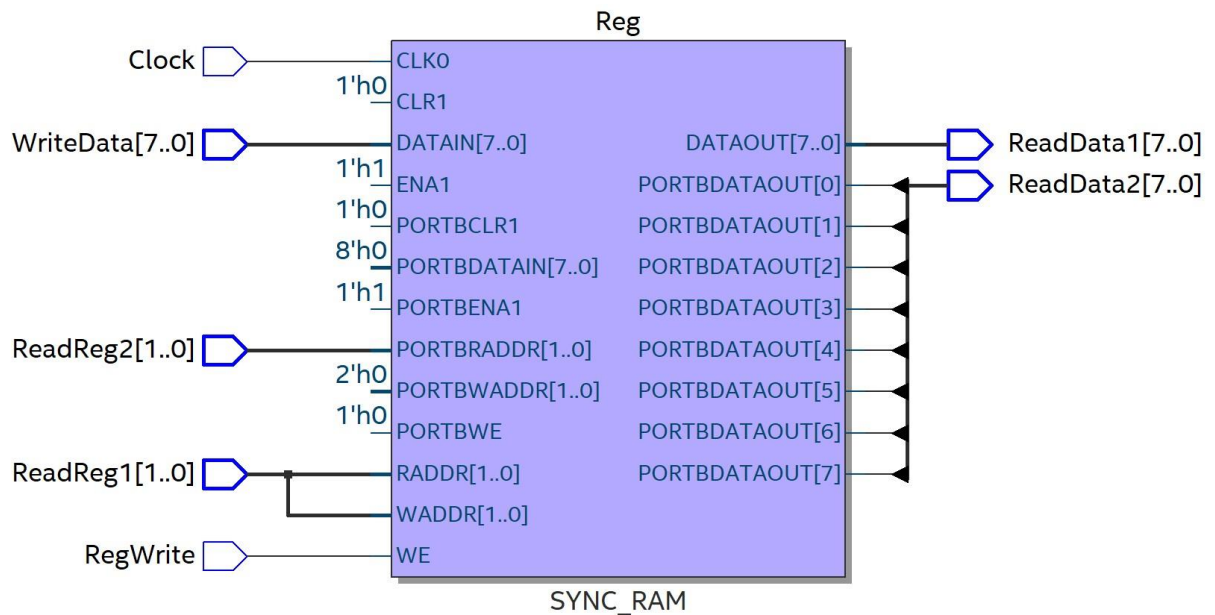


Figura 4 – Banco de Registradores

1.3.4 Counter PC

O Counter PC é um componente que tem como objetivo adicionar 1 no PC. Em operações que não contem saltos, ele adiciona 1 no endereço que foi recebido pelo PC, fazendo com que o programa avance 1 passo.

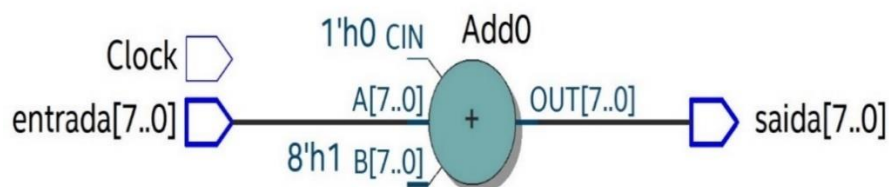


Figura 5 – Counter PC

1.3.5 Memória RAM

A Memória de dados ou Memória Ram, tem como função armazenar os dados temporariamente, dados esses que são usados durante a execução das instruções do processador.

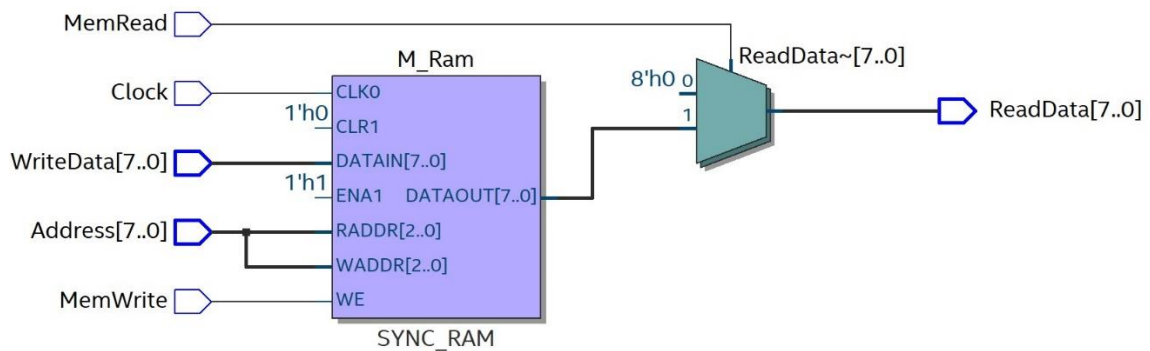


Figura 6 – Memória de dados

1.3.6 Temp Zero

O zero é um subcomponente que faz uma comparação e diz se os valores comparados eram iguais, ele fica dentro da ULA e somente é utilizado em operações comparativas, ele é responsável por inicializar o processo de comparação.

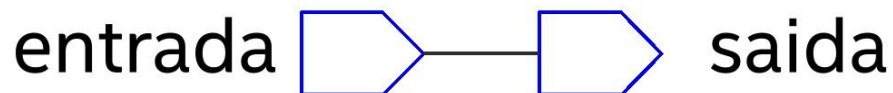


Figura 7 – Zero

1.3.7 Multiplexador 2x1

Os multiplexadores são um componente que determinam um valor baseados em um seletor, e esse valor vai ser o que irá sair do multiplexador.

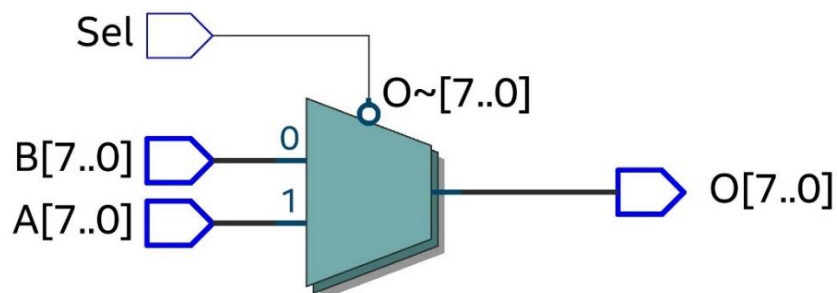


Figura 8 – Multiplexador 2x1

1.3.8 ULA

A ULA (Unidade Lógica Aritmética) é um componente do processador, que tem como função realizar as principais operações aritmética, por exemplo: soma, subtração e multiplicação. A ULA também pode fazer desvio condicional através da comparação de valor.

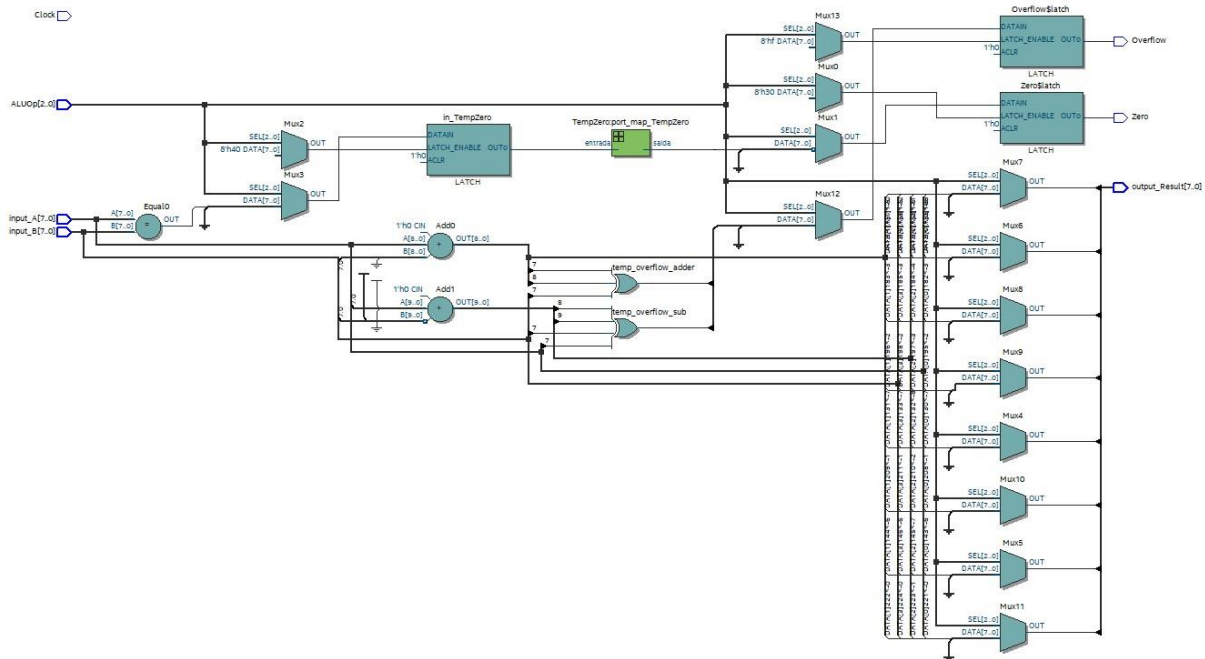


Figura 9 – ULA

1.3.9 Program Counter

O componente Program Counter (PC), tem como função a sequência do código, quando ele recebe o clock igual a 1, ele recebe um valor de 8 bits (0 a 255), o valor se é parte de um endereço de uma instrução, e é enviado para outros 2 componentes: o CounterPC e a Memória de Instruções.

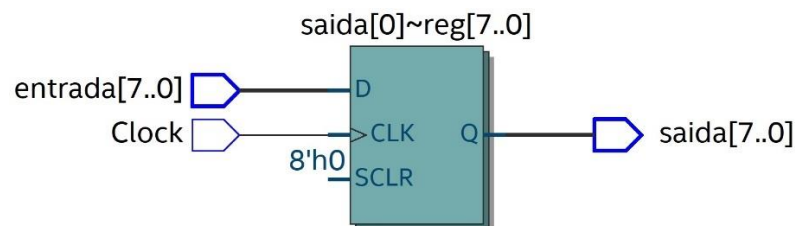


Figura 10 - PC

1.3.10 Unidade de Controle

A unidade de controle tem como função controlar os componentes do processador, por um valor relacionado as flags, essas que estão relacionadas com o Opcode recebido pelo componente.

jump: A função do jump é definir se próximo endereço vai ser do PC ou se ele será acessado diretamente por um salto;

Branch: Ela é parecida com o Jump, porém necessita de uma determinada condição para realizar o salto;

M_read: vai ficar responsável por disponibilizar um valor que foi acessado pela RAM;

M_to_reg: Decide de onde vai vir um valor que será escrito em algum registrador, RAM ou ULA;

ALUOp: Será a função que vai decidir qual operação vai ser realizada;

M_write: Registra um valor que veio da ALU ou de algum registrador na memória RAM;

ula_src: Ela vai define se na segunda entrada da ALU vai entrar um dado de algum registrador ou se será um valor imediato;

reg_write: Faz com que a escrita de dados em um registrador seja ativada.

Tabela 2 - Flags de controle do processador.

Instrução	Jump	Branch	MemRead	MemReg	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

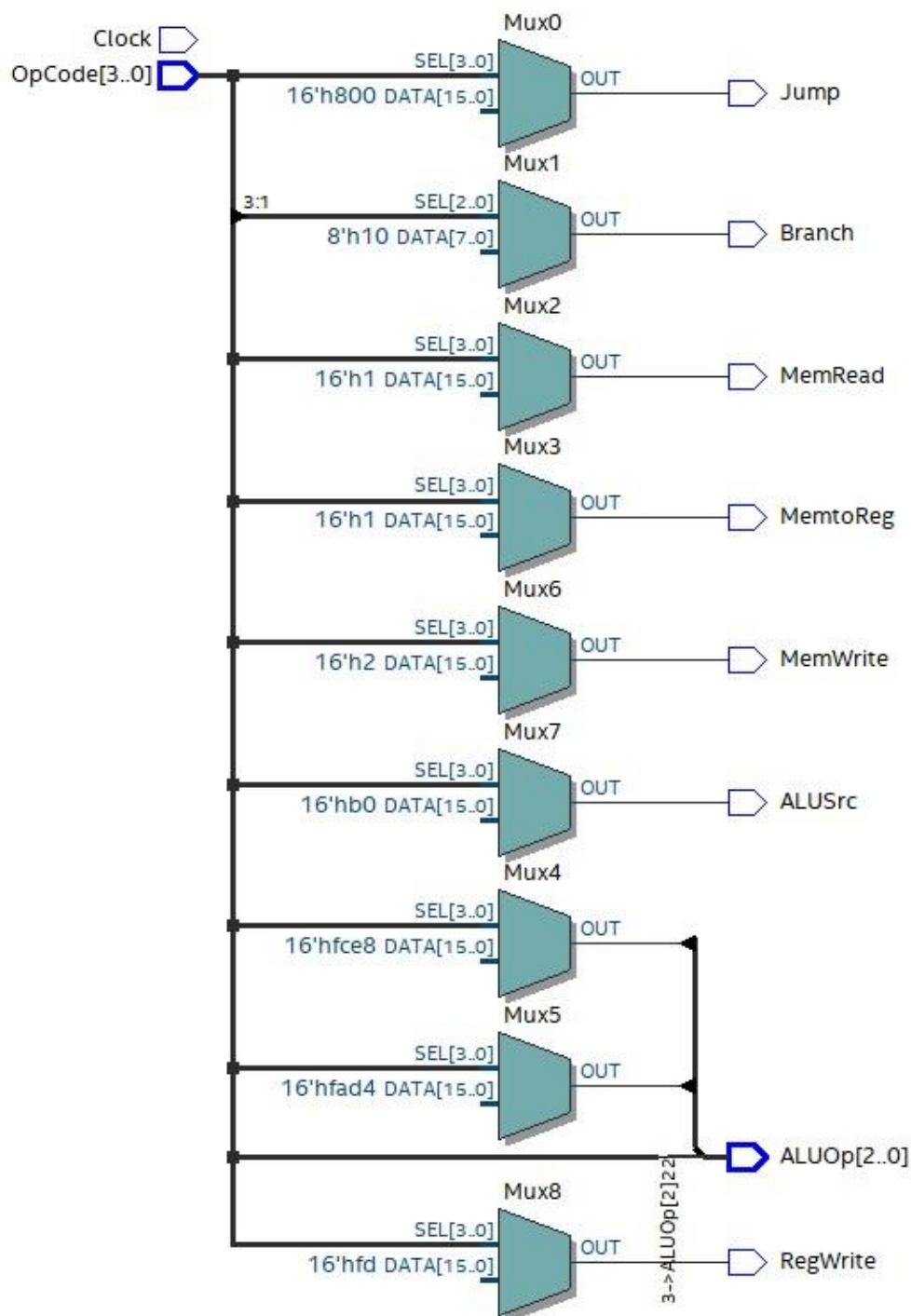


Figura 11 - Unidade de controle

1.3.11 Clock

Nesse projeto não foi implementado o clock como um componente específico, no entanto é de grande importância para o funcionamento do processador, uma vez que ele tem como função o controle do ciclo da unidade, simulando os clocks. Foi integrado o clock na maioria dos componentes, fazendo com que ele indique quando o processador está em operação.

1.3.12 Memória de instrução

É na memória de instrução também conhecida como (ROM) que fica armazenado os dados para a execução do programa que vai ser executado pelo processador, os dados que vão ser fornecidos são apenas para leitura.

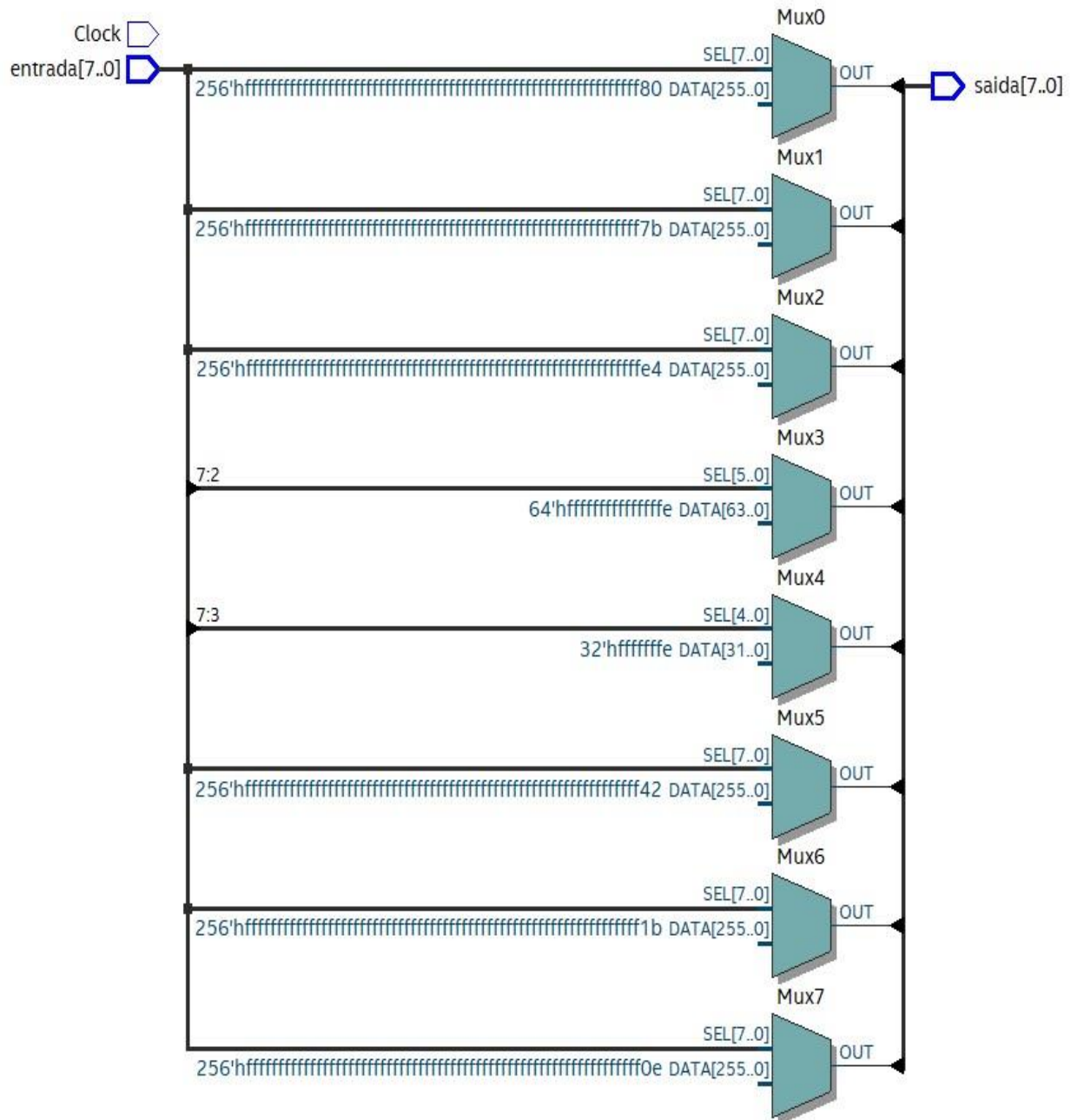


Figura 12 - Memória de instrução

1.3.13 Extensor de sinal

Foi implementado dois extensores, um de 2 bits para 8 bits e outro de 4 bits para 8 bits, eles têm uma função muito importante, pois servem para não permitir que uma instrução chegue a um componente com o tamanho inadequado.



Figura 13 – Extensor 2 para 8

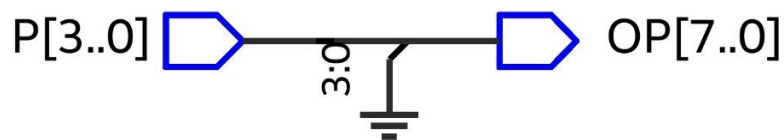


Figura 14 – Extensor 4 para 8

1.4 Datapath

É a visualização dos componentes interligados por barramentos fazendo um caminho de dados, e uma unidade para realizar o controle das operações. Ele foi retirado diretamente do programa Quartus, na função RTL Viwer.

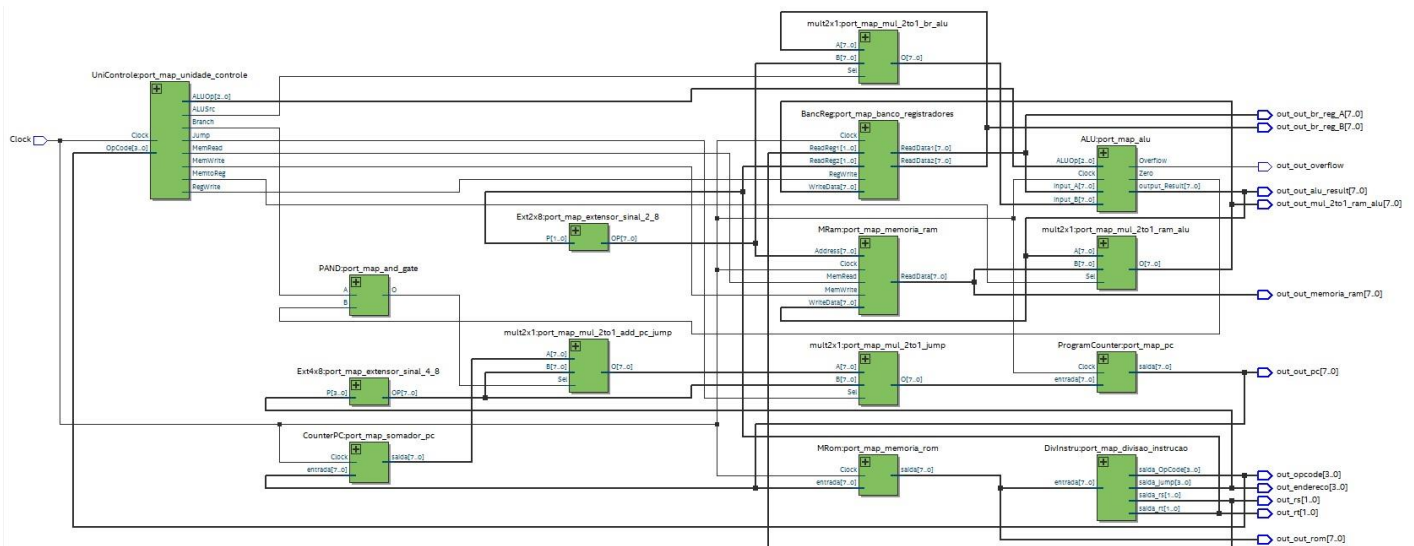


Figura 15 – Datapath RTL Viwer

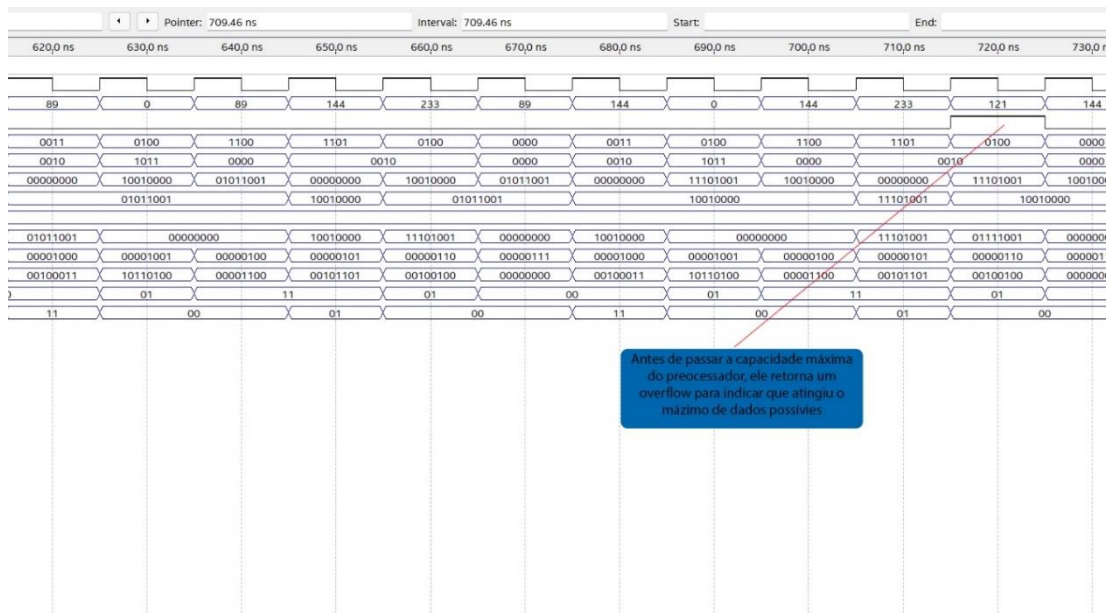


Figura 17 – Overflow Fibonacci

3 Considerações Finais

Esse trabalho apresentou o projeto e implementação do processador de 8 bits denominado de BORI. O processador tem a capacidade de executar as operações propostas, apesar das limitações como qualquer processador de 8 bits, os valores baixos que saem da ULA, por exemplo. O processador cumpre com o proposto, que neste caso é armazenar os dados em até 8 bits nos registradores, e ter espaço suficiente de armazenamento para conseguir armazenar as operações uma limitação bem evidente é quantidade de dados que ele pode armazenar, não podendo ultrapassar 256 bits. Para informar que esse limite foi ultrapassado foi implementado um overflow, que é um pino na ULA.

4 Repositório

https://github.com/veniciusjacob/AOC_Venicius_Jacob_Rosialdo_Vidinho_UFRR_2022