



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

ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES

RELATÓRIO DO PROJETO: PROCESSADOR 8BIS

ALUNOS:

**Guilherme Lirioberto da Silva Alves - 2020021600
Glisbel de Las Nieves Aponte Lopez - 2019047346**

**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 8BIS

**Março de 2022
Boa Vista/Roraima**

Resumo

Este trabalho aborda o projeto e implementação de um processador 8 bits unicycle chamado 8BIS ou seja, cada instrução gasta 1 ciclo de relógio (clock) para executar, que tem como base a arquitetura de um processador MIPS. Para a efetuação dele utilizamos a linguagem voltada para design de circuitos digitais chamada VHDL com a IDE Quartus Prime Lite, onde os componentes presentes nele são: ALU, Banco de Registradores, CounterPC, Divisão de Instruções, Extensor de Sinal 2x8, Extensor de Sinal 4x8, Memória Ram, Memória Rom, Mux2x1, PC, Unidade de Controle, Porta And, Clock e o ZERO. 5 desses componentes apresentam unidades funcionais onde será possível fazer processos aritméticos, comparativos além de saltos incondicionais.

Palavras Chaves: MIPS, Processador, Instrução, VHDL, Unidades funcionais, Quartus Prime Lite

Conteúdo

- 1 Especificação7
 - 1.1 Plataforma de desenvolvimento7
 - 1.2 Conjunto de instruções8
 - 1.3 Descrição do Hardware9
 - 1.3.1 ALU ou ULA9
 - 1.3.2 BDRegister9
 - 1.3.3 Clock10
 - 1.3.4 Controle10
 - 1.3.5 Memória de dados11
 - 1.3.6 Memória de Instruções12
 - 1.3.7 Somador13
 - 1.3.8 And13
 - 1.3.9 Mux_2x114
 - 1.3.10 PC14
 - 1.3.11 ZERO14
 - 1.4 Datapath15
- 2 Simulações e Testes16
- 3 Considerações finais18

Lista de Figuras

7

FIGURA 2 - BLOCO SIMBÓLICO DO COMPONENTE QALU GERADO PELO QUARTUS9

FIGURA 19 - RESULTADO NA WAVEFORM.17

Lista de Tabelas

TABELA 1 – TABELA QUE MOSTRA A LISTA DE OPCODES UTILIZADAS PELO PROCESSADOR XXXX.9

TABELA 2 - DETALHES DAS FLAGS DE CONTROLE DO PROCESSADOR.11

TABELA 3 - CÓDIGO FIBONACCI PARA O PROCESSADOR QUANTUM/EXEMPLO.16

1 Especificação

Nesta seção é apresentado o conjunto de itens para o desenvolvimento do processador 8BIS, bem como a descrição detalhada de cada etapa da construção do processador.

1.1 Plataforma de desenvolvimento

Para a implementação do processador 8BIS foi utilizado a IDE: Quartus Prime Lite Edition 20.1, desenvolvido pela Intel Corporation, onde foi codificado e testado. Dentro do Software ele apresenta o gerador de WaveForms e Visualizador de RTL's entre outras ferramentas que auxiliaram no desenvolvimento do processador.

Flow Status	Successful - Mon Feb 28 12:52:59 2022
Quartus Prime Version	20.1.0 Build 711 06/05/2020 SJ Lite Edition
Revision Name	PortMap
Top-level Entity Name	unidadeControle
Family	Cyclone V
Device	5CGXFC7C7F23C8
Timing Models	Final
Logic utilization (in ALMs)	N/A
Total registers	0
Total pins	15
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 8BIS possui 4 registradores: S0, S1, S2, S3. Assim como 3 formatos de instruções de 8 bits cada, Instruções do **tipo** R, I e J seguem algumas considerações sobre as estruturas contidas nas instruções:

- **Opcode:** instrução que um determinado processador possui para conseguir realizar determinadas tarefas. Suas especificações e formatos são definidos no conjunto de instruções da arquitetura do processador em questão;
- **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:** Este formato aborda instruções de Load (exceto *load Immediately*), Store e instruções baseadas em operações aritméticas.

- **Formato do tipo I:** Formato que aborda instruções baseadas em operações com valores imediatos, desvios condicionais e operações relacionadas a memória, BNE, BEQ, Store e Load.

- **Formato do tipo J:** É um formato que aborda as instruções de desvios incondicionais como exemplo o Jump.

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

Visão geral das instruções do Processador 8BIS:

O número de bits do campo Opcode das instruções é igual a quatro, sendo assim obtemos um total ($Bit(0e1)^{NumeroTotaldeBitsdoOpcode} \therefore 4^2 = 16$) 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 8BIS.

Opcode	Nome	Formato	Breve Descrição	Exemplo
1111	LI	I	Load Immediately	li \$S0, 31
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	DIV_INT	R	Divisão	div \$S0, \$S1 ,ou seja, \$S0 := \$S0 / \$S1

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 ALU ou ULA

O componente QALU (Q Unidade Lógica Aritmética) tem como principal objetivo efetuar as principais operações aritméticas, dentre elas: soma, subtração, divisão (considerando apenas resultados inteiros) e multiplicação. Adicionalmente o QALU efetua operações de comparação de valor como maior ou igual, menor ou igual, somente maior, menor ou igual. O componente QALU recebe como entrada três valores: **A** – dado de 8bits para operação; **B** - dado de 8bits para operação e **OP** – identificador da operação que será realizada de 4bits. O QALU também possui três saídas: **zero** – identificador de resultado (2bit) para comparações (1 se verdade e 0 caso contrário); **overflow** – identificador de overflow caso a operação exceda os 8bits; e **result** – saída com o resultado das operações aritméticas.

Comentado [h1]: Um exemplo da descrição

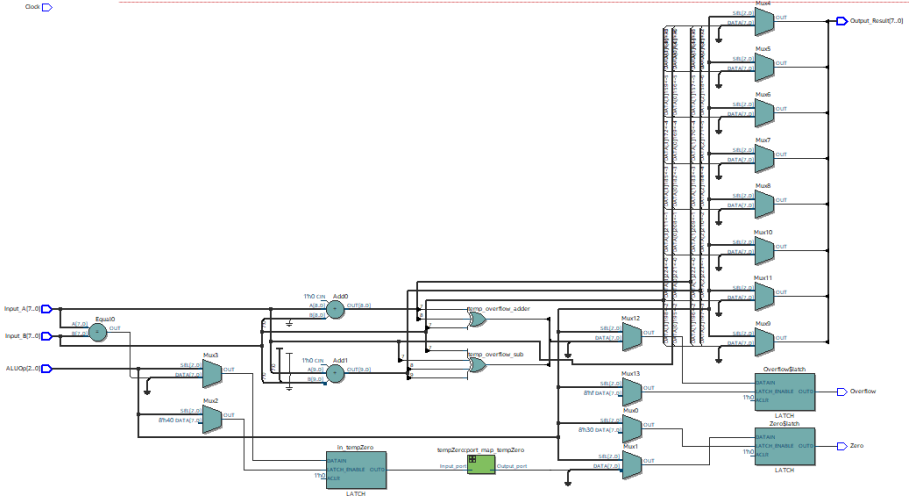


Figura 2 - Bloco simbólico do componente QALU gerado pelo Quartus

1.3.2 BDRegister

A unidade funcional banco de registradores é um componente que consiste em um conjunto de registradores que podem ser acessados de forma organizada.

O banco de registradores dividido é composto por quatro registradores \$S0, \$S1, \$S2 e \$S3, de 2 bits cada, organizados em uma estrutura com três portas de acesso.

- entrada_reg1 – recebe 2 bits para o endereço do primeiro registrador;
 - entrada_reg2 - recebe 2 bits para o endereço do segundo registrador;
 - entrada_escrita_dados – dado de 8 bits vindo da memória RAM que será escrito no registrador de destino.
 - reg_write - identificador de resultado da flag reg_write de 1 bit. Recebe o valor 1 se a flag estiver ativada e 0 se estiver desativada.
- O Banco de Registradores também possui duas saídas:
- saída_dado_lido1 - resulta em uma saída de 8 bits do valor armazenado na entrada_reg1
 - saída_dado_lido2 - resulta em uma saída de 8 bits do valor armazenado na entrada_reg

1.3.3 Clock

Ele é responsável pelo controle de ciclos da unidade, simulando os clocks. Ou seja, o clock é a frequência com que um processador é capaz de executar as tarefas incumbidas a ele. Desse modo, quanto maior a frequência, portanto o clock, menor é o tempo de execução, mais ágil o processador é.

1.3.4 Controle

O componente Controle tem como função controlar todos os componentes do processador de acordo com o recebimento de um opcode de 4 bits. Sua entrada específica é se a instrução é do tipo add, sub, mult, lw, sw, li, beq, bne ou J. Para cada instrução, cada flag terá valores específicos. Bandeiras:

- **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.

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.

instrução	Jump	Branch	MemRead	MemtoReg	ALUOp	MemWrite	ALUSrc	RegWrite
lw	0	0	1	1	000	0	0	1
sw	0	0	0	0	000	0	0	0
add	0	0	0	0	001	0	0	1
sub	0	0	0	0	010	0	0	0
addi	0	0	0	0	001	1	1	0
subi	0	0	0	0	010	1	1	0
move	0	0	0	0	011	0	0	0
li	0	0	0	0	011	1	1	0
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

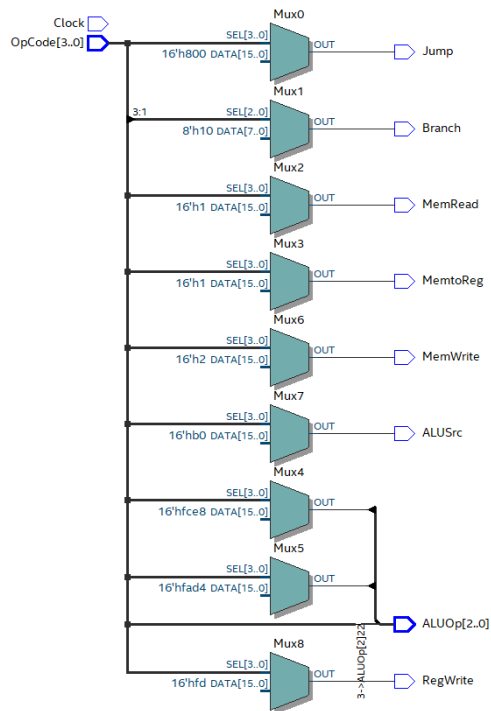


Figura 3 - Bloco simbólico do componente Controle gerado pelo Quartus

1.3.5 Memória de dados

Conhecido como memória RAM é responsável por armazenar informações temporariamente que só serão usados durante a execução de instruções.

A memória de dados recebe seis entradas:

- clock_mem_dados: que ativa representa a energia do componente, tem valor 1 se o processador está ligado 0 se desligado.
- M_read: recebe 1 bit da flag vinda da unidade de controle informando se será lido algum dado da Memória De Dados - RAM (caso o valor da entrada seja 1) ou não (caso o valor da entrada seja zero)
- M_write: recebe 1 bit da flag vinda Unidade De Controle para saber se vai armazenar dados na Memória de Dados - RAM
- entrada_endereco: recebe 8 bits da posição da memória RAM onde o dado deve ser escrito ou lido
- escrita_dados: recebe o dado de 8 bits que será armazenado temporariamente na RAM

E uma saída:

- dado_lido: onde sai um dado de 8 bits da posição que foi recebida no endereço, se a M_read estiver setada em 1.

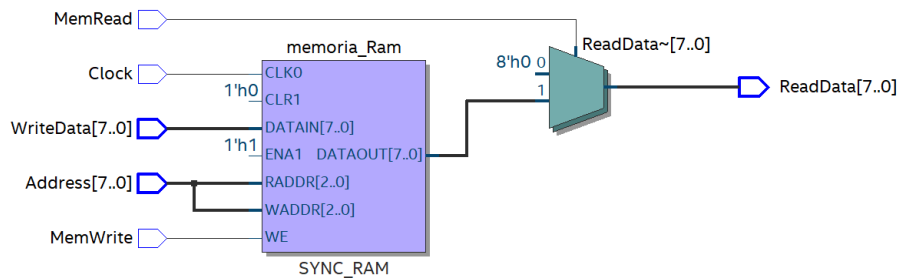


Figura 4 - Bloco simbólico do componente Memória de Dados gerado pelo Quartus

1.3.6 Memória de Instruções

A memória de instruções fornece dados somente para leitura e é onde o programa vai ser executado pelo processador e onde é armazenado. Memória que permite apenas a leitura de dados e não a escrita. Isso porque suas informações são gravadas pelo fabricante uma única vez e não podem ser alteradas ou apagadas, somente acessadas, sendo classificadas como memória não volátil.

Entradas:

- Clock: Recebe o sinal com a borda alta para ativar o componente.
- input_port: Recebe um endereço de 8 bits do componente PC (Program Counter) que será enviado para a execução.

Uma saída:

- output_port: Uma saída de 8 bits da instrução que será executada

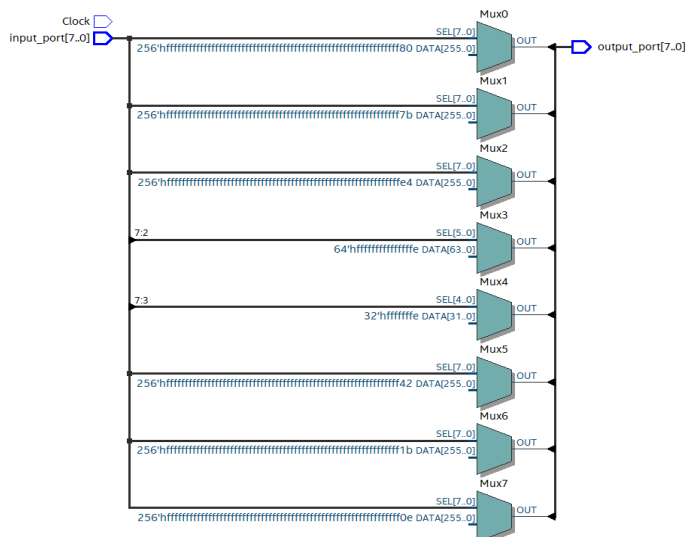


Figura 5 - Bloco simbólico do componente Memória de Instruções gerado pelo Quartus

1.3.7 Somador (countPC)

O somador (countPC) tem a função de adicionar uma unidade de bit à instrução atual permitindo que o PC receba a instrução seguinte da memória.

Uma entrada:

- input_port: a única entrada do componente somador (countPC) é a instrução de 8 bits que está no PC.

Uma saída:

- output_port: a única saída do somador é uma instrução de 8 bits, sendo a instrução seguinte da que entrou pelo saída_pc

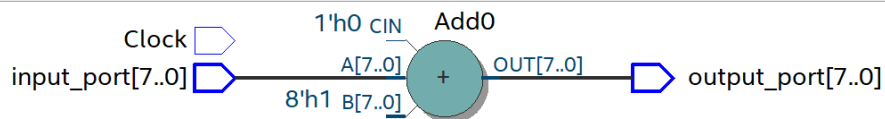


Figura 6 - Bloco simbólico do componente Somador gerado pelo Quartus

1.3.8 AndGate

O bloco lógico AND assume 1 quando todas as duas entradas forem 1 e assume 0 nos demais casos.

Duas entradas:

- entrada_and_1: a primeira entrada do componente AND diz respeito à flag Branch vinda da unidade de controle e possui 1 bit, se o valor fornecido for 1 significa que a flag de desvio condicional está ativa

- entrada_and_2: a segunda entrada do componente AND possui 1 bit e recebe 1 caso a flag Zero que significa desvio condicional vinda da ULA esteja ativada e 0 caso esteja inativa

Uma saída:

- saída_and: caso ambas as flags de desvio condicional estiverem ativas, saíra 1 bit no valor 1, o que significa que será feita uma instrução de desvio

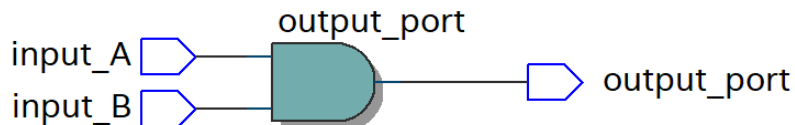


Figura 7 - Bloco simbólico do componente AndGate gerado pelo Quartus

1.3.9 Mux_2x1

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

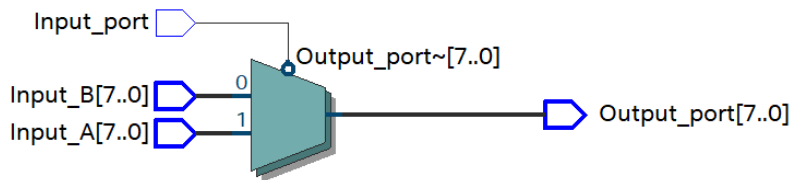


Figura 8 - Bloco simbólico do componente Mux_2x1 gerado pelo Quartus

1.3.10 PC

O Program Counter é um registrador de propósito especial usado pelo processador para armazenar o endereço de 8 bits da próxima instrução a ser executada. Ele recebe a entrada de um bit chamado clock que indica se a unidade está ligada e o computador de entrada é uma instrução de 8 bits.

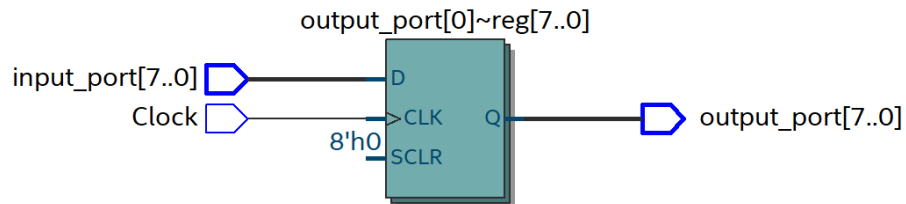
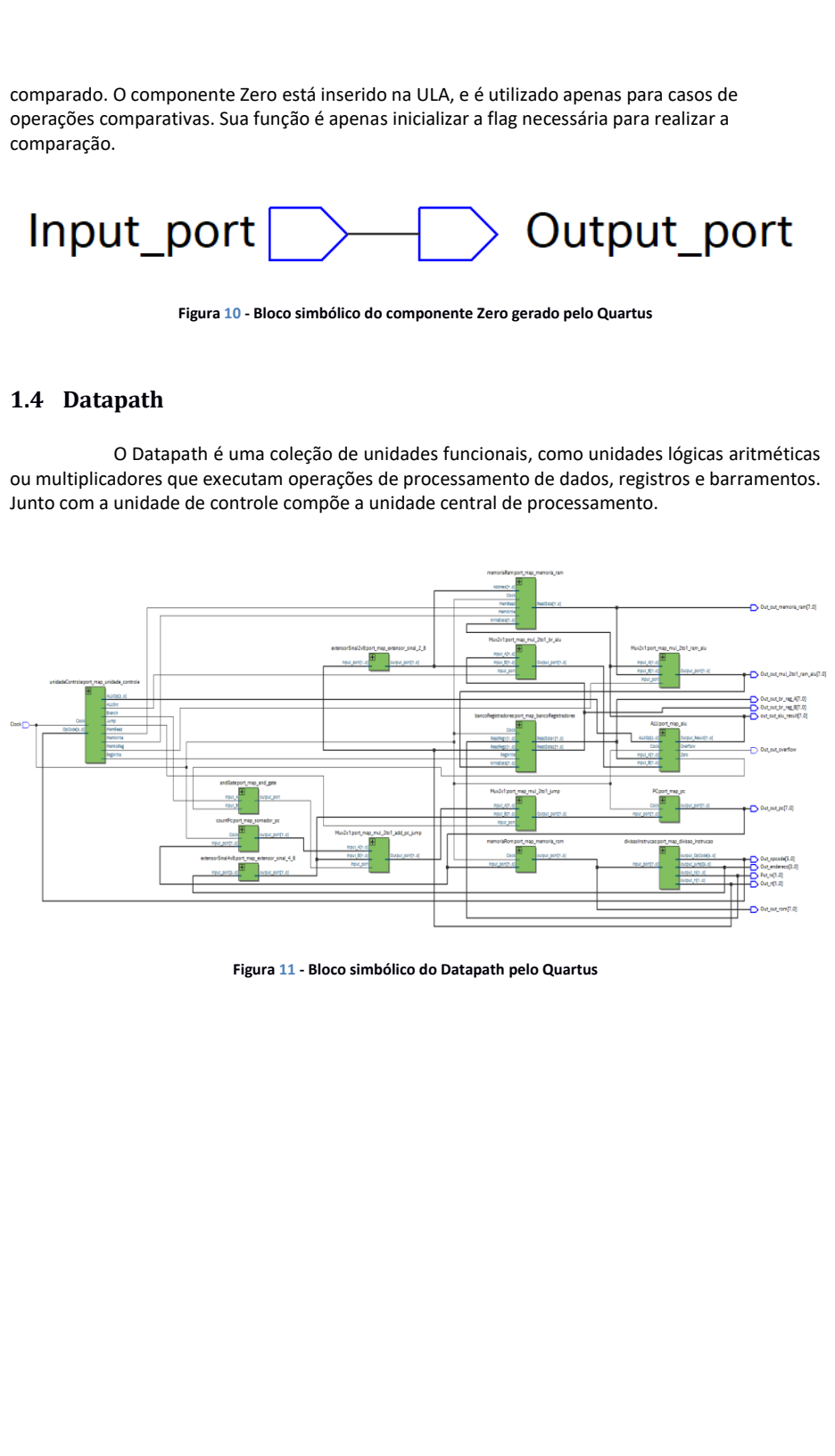


Figura 9 - Bloco simbólico do componente PC gerado pelo Quartus

1.3.11 ZERO

O zero fornece uma flag que indica se um valor é igual ou diferente do que foi



comparado. O componente Zero está inserido na ULA, e é utilizado apenas para casos de operações comparativas. Sua função é apenas inicializar a flag necessária para realizar a comparação.

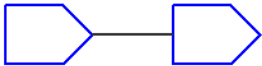
Input_port  **Output_port**

Figura 10 - Bloco simbólico do componente Zero gerado pelo Quartus

1.4 Datapath

O Datapath é uma coleção de unidades funcionais, como unidades lógicas aritméticas ou multiplicadores que executam operações de processamento de dados, registros e barramentos. Junto com a unidade de controle compõe a unidade central de processamento.

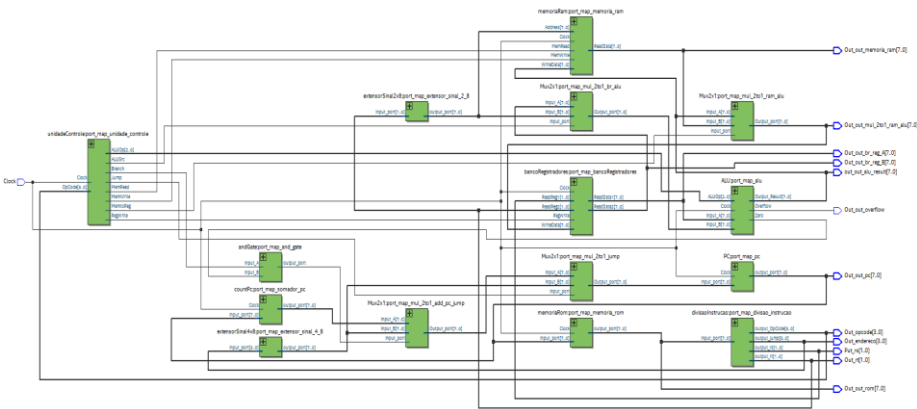


Figura 11 - Bloco simbólico do Datapath pelo Quartus

comparado. O componente Zero está inserido na ULA, e é utilizado apenas para casos de operações comparativas. Sua função é apenas inicializar a flag necessária para realizar a comparação.

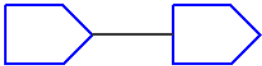
Input_port  **Output_port**

Figura 10 - Bloco simbólico do componente Zero gerado pelo Quartus

1.4 Datapath

O Datapath é uma coleção de unidades funcionais, como unidades lógicas aritméticas ou multiplicadores que executam operações de processamento de dados, registros e barramentos. Junto com a unidade de controle compõe a unidade central de processamento.

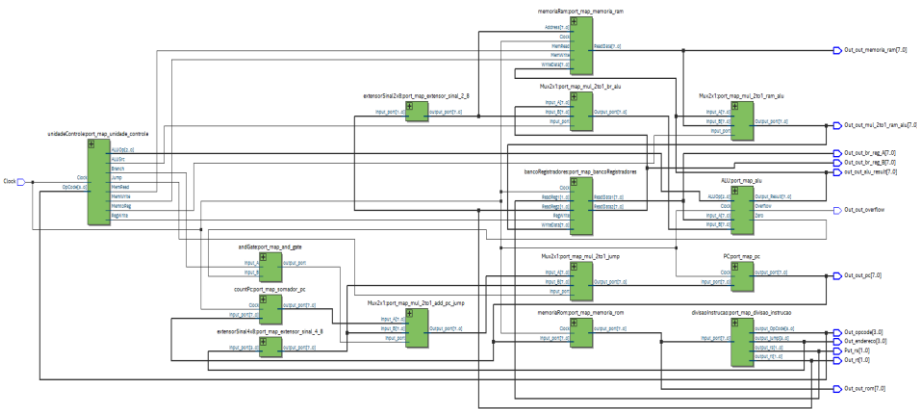


Figura 11 - Bloco simbólico do Datapath pelo Quartus

comparado. O componente Zero está inserido na ULA, e é utilizado apenas para casos de operações comparativas. Sua função é apenas inicializar a flag necessária para realizar a comparação.

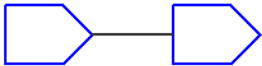
Input_port  **Output_port**

Figura 10 - Bloco simbólico do componente Zero gerado pelo Quartus

1.4 Datapath

O Datapath é uma coleção de unidades funcionais, como unidades lógicas aritméticas ou multiplicadores que executam operações de processamento de dados, registros e barramentos. Junto com a unidade de controle compõe a unidade central de processamento.

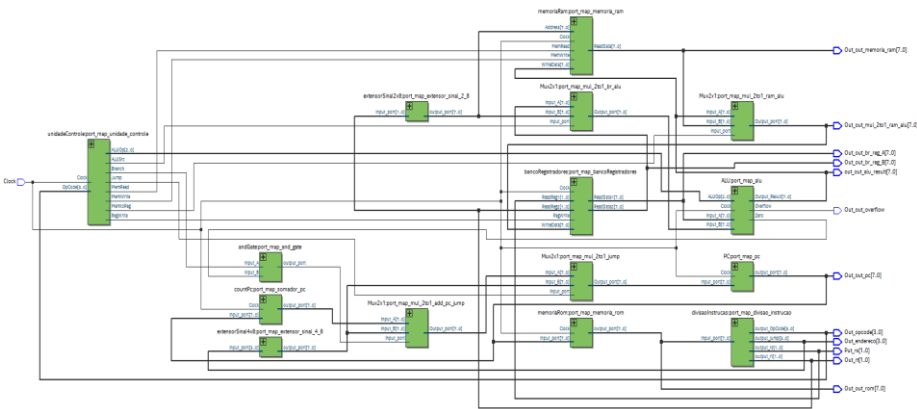
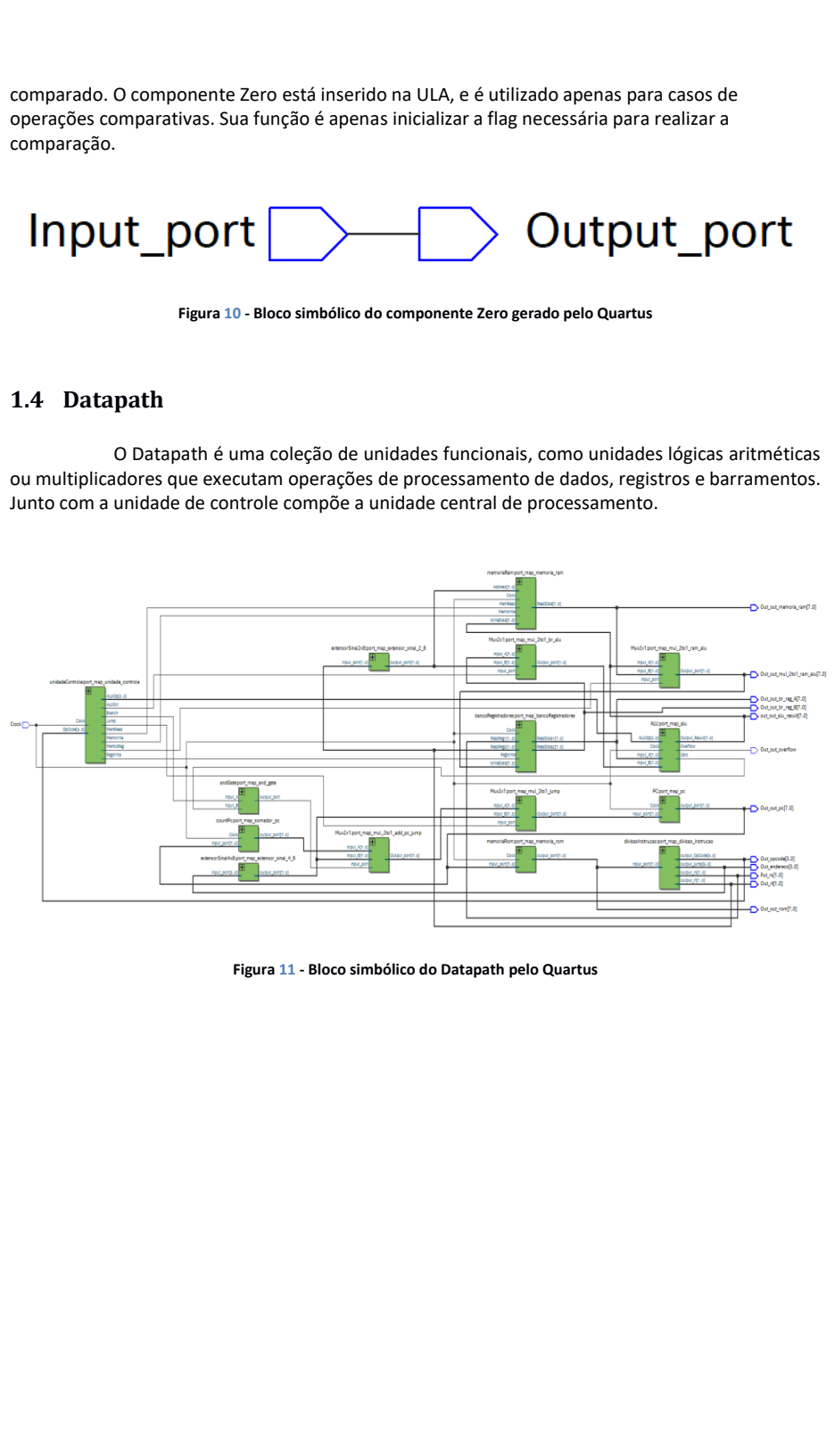


Figura 11 - Bloco simbólico do Datapath pelo Quartus



comparado. O componente Zero está inserido na ULA, e é utilizado apenas para casos de operações comparativas. Sua função é apenas inicializar a flag necessária para realizar a comparação.

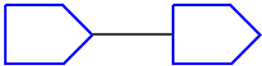
Input_port  **Output_port**

Figura 10 - Bloco simbólico do componente Zero gerado pelo Quartus

1.4 Datapath

O Datapath é uma coleção de unidades funcionais, como unidades lógicas aritméticas ou multiplicadores que executam operações de processamento de dados, registros e barramentos. Junto com a unidade de controle compõe a unidade central de processamento.

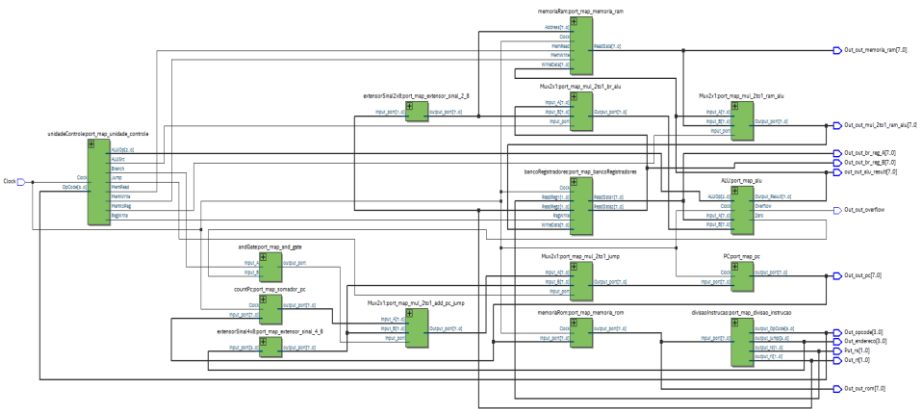


Figura 11 - Bloco simbólico do Datapath pelo Quartus

2 Simulações e Testes

Objetivando analisar e verificar o funcionamento do processador, efetuamos alguns testes analisando cada componente do processador em específico, em seguida efetuamos testes de cada instrução que o processador implementa. Para demonstrar o funcionamento do processador 8BIS utilizaremos como exemplo o código para calcular o número da sequência de Fibonacci.

Tabela 3 - Código Fibonacci para o processador Quantum/8BIS.

Endereço	Linguagem de Alto Nível	Binário		
		Opcode	Reg2	Reg1
			Endereço	
Dado				
0	LI \$S0, 0	1111	00	00
1		00000000		
2	LI \$S3, 6	1111	00	11
3		00000110		
4	SW \$S3, \$S0	0111	00	11
5	LI \$S1, 1	1111	00	01
6		00000001		
7	LRT \$S2, \$S1	0110	01	10
8	LI \$S3, 3	1111	00	11
9		00000011		
10	LW \$S0, \$S0	0101	00	00
11	CMPG \$S3,\$S0	1010	00	11
12	JMP fim	1101	0000	
13		00011010		
14	loop_fib: LI \$S0, 1	1111	00	00
15		00000001		
16	ADD \$S3, \$S0	0010	00	11
17	LRT \$S0, \$S2	0110	10	00
18	ADD \$S2, \$S1	0010	01	10
19	LRT \$S1, \$S0	0110	00	01
20	LI \$S0, 0	1111	00	00
21		00000000		
22	LW \$S0, \$S0	0101	00	00
23	CMPLE \$S3,\$S0	1001	00	11
24	JMP loop_fib	1101	0000	
25		00001110		
26	Fim: DEBUG \$S2, \$S2	0001	10	10

Na Sequência de Fibonacci, há uma regra na qual cada termo subsequente corresponde à soma dos dois anteriores, e no processador isso acontece a cada 3 instruções onde ao passar por elas ocorre a soma dos anteriores pelo resultado, onde logo a baixo será exibido.

Verificação dos resultados no relatório da simulação: Após a compilação e execução da simulação, o seguinte relatório é exibido.

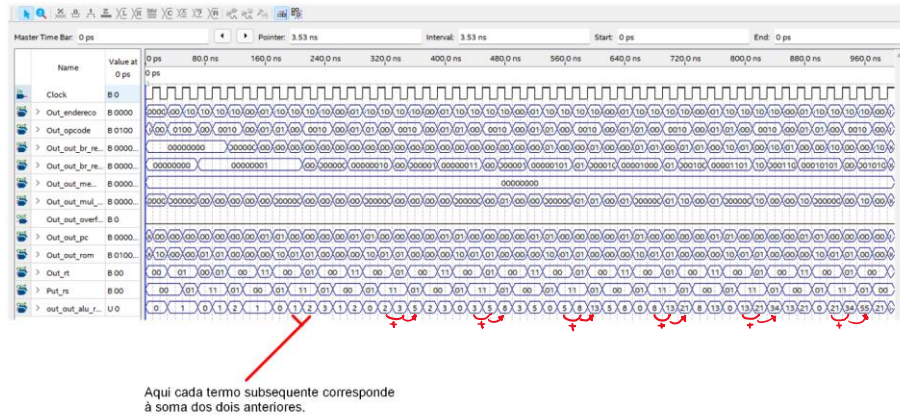


Figura 12 – Sequencia Fibonacci para teste na waveform.

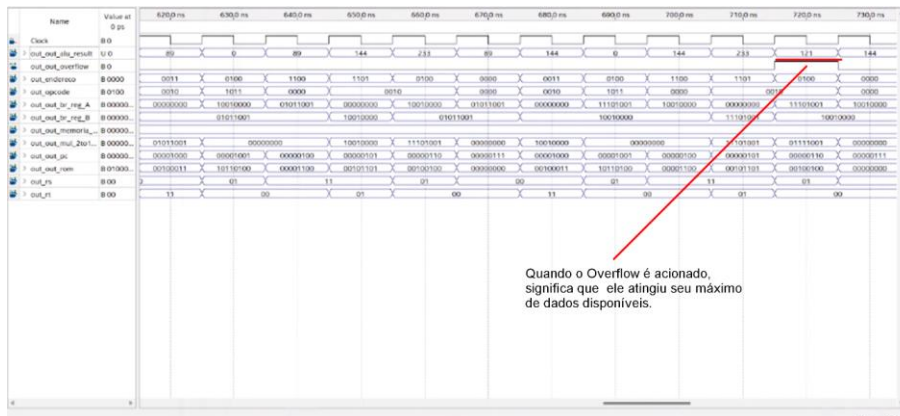


Figura 13 – Overflow da sequência de Fibonacci para teste na waveform.

3 Considerações finais

Este trabalho apresentou o projeto e implementação do processador de 8 bits denominado de 8BIS onde ocorreu uma ampla pesquisa sobre a arquitetura MIPS para base da construção do processador, além de pesquisas relacionadas a antigos processadores que já utilizaram o formato 8 bits.

No final acabamos procurando exemplos específicos de trabalhos anteriores que tinham o mesmo tema. Isso foi de vital importância para a conclusão do nosso projeto, pois por conta da situação atual do mundo, nós ficamos limitados a uma tela na hora de tirarmos nossas dúvidas, embora tivesse disponibilidade algumas vezes na semana para tal, ainda assim um encontro síncrono não substitui um real.

E a conclusão que tivemos depois de todas essas dificuldades que nos deparamos, é a seguinte, aulas remotas desestimulam nossa vontade de ir atrás de algo, pois esse algo está logo a nossa frente a um click de distância, claro que metaforicamente.