



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

ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES

RELATÓRIO DO PROJETO: PROCESSADOR ML21_processor_8bits

ALUNOS:

Matheus de Souza Melo (2019007565).

Lucas Araldi (2019039139).

**Maio de 2021
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 ML21_processor_8bits

Relatório apresentado como para obtenção de nota na disciplina de Arquitetura e Organização de Computadores, ofertada pelo curso de Ciência da Computação da Universidade Federal de Roraima.

Prof. Dr. Herbert Oliveira Rocha.

**Outubro de 2019
Boa Vista/Roraima**

Resumo

O projeto **ML21** aborda a desenvolvimento de um processador **uniciclo** de 8 bits baseado na arquitetura de um processador **MIPS**. Ao longo do relatório será feito uma descrição minuciosa dos componentes utilizador no projeto, assim como, os testes realizados. O processador possui uma divisão de instrução(bits) 4, 2, 2 pra operações tipo **R** e **I** (aritméticas e imediato respectivamente) e 3, 4 para operações tipo **J** (salto), podendo executar até 13 instruções. Foi utilizada a linguagem **VHDL** e o software **Quartus Prime Lite** versão 20.1.1 e para a simulação da **WaveForm** foi utilizado o software ModelSim Altera.

Conteúdo

1	Especificação	5
1.1	Plataforma de desenvolvimento	5
1.2	Conjunto de instruções	6
1.3	Descrição do Hardware	7
1.3.1	ALU ou ULA	7
1.3.2	BDRegister	8
1.3.3	Clock	9
1.3.4	Controle	9
1.3.5	Memória de dados	11
1.3.6	Memória de Instruções	11
1.3.7	Somador	12
1.3.8	And	13
1.3.9	Mux_2x1	13
1.3.10	PC	14
1.3.11	ZERO	14
1.4	Datapath	15
2	Simulações e Testes	17
3	Considerações finais	19

Lista de Figuras

5

FIGURA 2 - BLOCO SIMBÓLICO DO COMPONENTE QALU GERADO PELO QUARTUS **ERROR! BOOKMARK NOT DEFINED.**

FIGURA 19 - RESULTADO NA WAVEFORM.18

Lista de Tabelas

TABELA 1 – TABELA QUE MOSTRA A LISTA DE OPCODES UTILIZADAS PELO PROCESSADOR XXXX.7

TABELA 2 - DETALHES DAS FLAGS DE CONTROLE DO PROCESSADOR.10

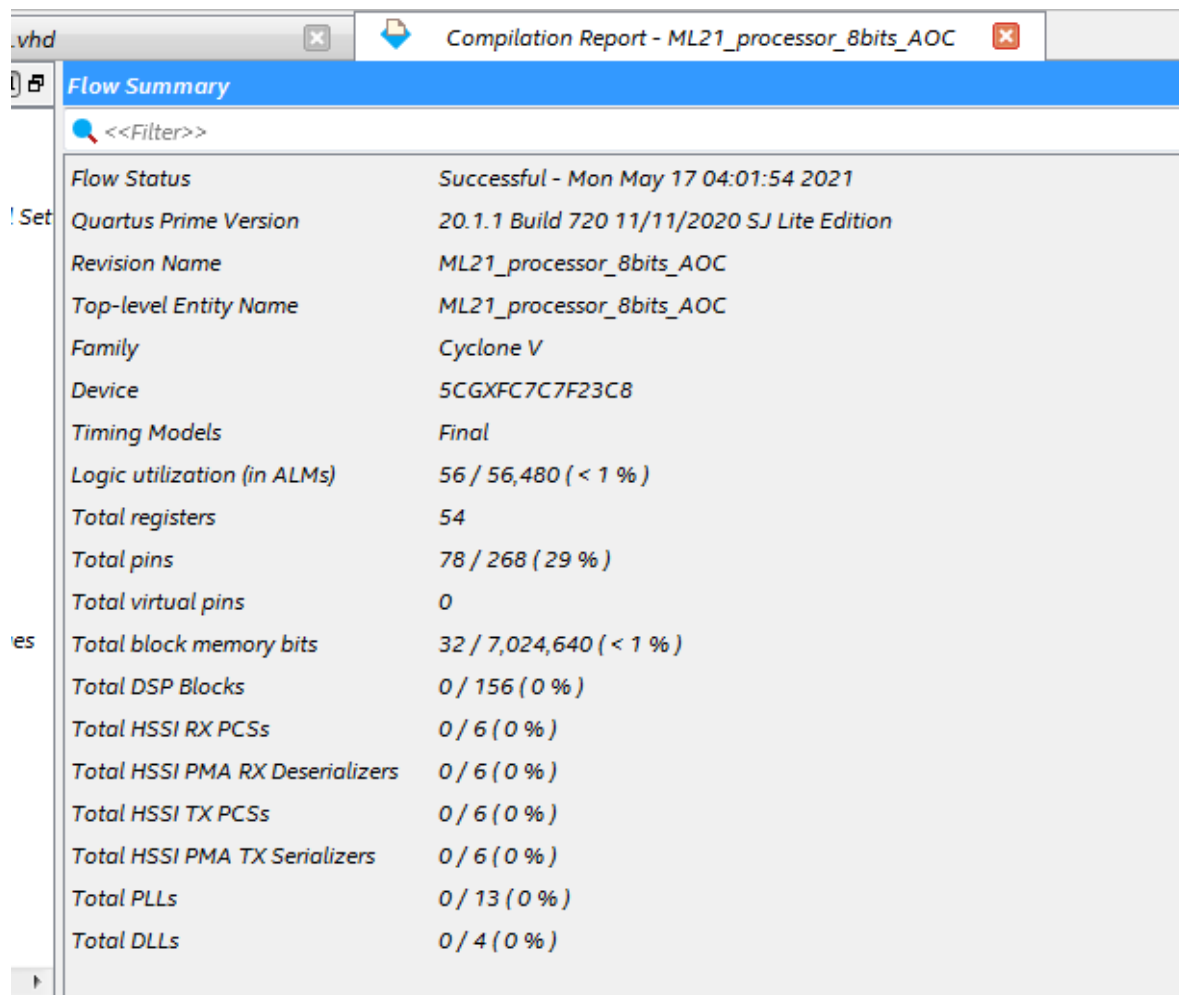
TABELA 3 - CÓDIGO FIBONACCI PARA O PROCESSADOR QUANTUM/EXEMPLO.17

1 Especificação

Nesta seção é apresentado o conjunto de itens para o desenvolvimento do processador ML21, bem como a descrição detalhada de cada etapa da construção do processador.

1.1 Plataforma de desenvolvimento

Para a implementação do processador ML21 foi utilizado a IDE: QuartusPrime Lite, versão 20.1.1 e o simulador ModelSim Altera.



Flow Status		Successful - Mon May 17 04:01:54 2021
Quartus Prime Version		20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name		ML21_processor_8bits_AOC
Top-level Entity Name		ML21_processor_8bits_AOC
Family		Cyclone V
Device		5CGXFC7C7F23C8
Timing Models		Final
Logic utilization (in ALMs)		56 / 56,480 (< 1 %)
Total registers		54
Total pins		78 / 268 (29 %)
Total virtual pins		0
Total block memory bits		32 / 7,024,640 (< 1 %)
Total DSP Blocks		0 / 156 (0 %)
Total HSSI RX PCSs		0 / 6 (0 %)
Total HSSI PMA RX Deserializers		0 / 6 (0 %)
Total HSSI TX PCSs		0 / 6 (0 %)
Total HSSI PMA TX Serializers		0 / 6 (0 %)
Total PLLs		0 / 13 (0 %)
Total DLLs		0 / 4 (0 %)

Figura 1 - Especificações no Quartus

1.2 Conjunto de instruções

O processador ML21 possui 4 registradores: \$S0 e \$S1. 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:** 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 as instruções do tipo **R**, é o registrador de destino;
- **Reg2:** o registrador contendo o segundo operando fonte e adicionalmente para instruções tipo **I**, é uma constante;

Tipo de Instruções:

Formato tipo R: este formato aborda as instruções baseadas em operações aritméticas e lógicas, soma, subtração, multiplicação.

Formato tipo I: este formato aborda as instruções baseadas em operações com valores imediatos, desvios condicionais e operações relacionadas à memória, soma e subtração imediata, BEQ, BNE, Load e Store.

Formato tipo J: este formato aborda as instruções de desvios incondicionais, exemplo Jump

Formato para escrita em código binário

OPERAÇÕES TIPO R (REG PADRÃO)

OPCODE	REGIS 1	REGIS 2
4 BITS	2 BITS (4 registradores)	2 BITS (4 registradores)
7-4	3-2	1-0

Tabela 1

Visão geral das instruções do Processador ML21:

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 2.

Tabela 2 – Tabela que mostra a lista de Opcodes utilizadas pelo processador ML21.

OPCODE	NAME	REG	TYPE	DESCRIPTION	EX
0000	ADD	4	R	SOMA	add \$s0, \$s1
0001	SUB	4	R	SUBTRAÇÃO	sub \$s0, \$s1
0010	ADDI	4	I	SOMA DIRETA	addi \$s0 2
0011	SUBI	4	I	SUB. DIRETA	subi \$
0100	SW	4	I	STORE WORD	sw \$s0 memo (00)
0101	LW	4	I	LOAD WORD	lw \$s0 memo (00)
0110	BEQ	8	J	CONDICIONAL	beq address
0111	MUL	4	R	MULIPLICAÇÃO	mul \$s0 \$s1
1000	LI	4	I	LOAD IMEDIATO	li \$s0 2
1001	BNE	8	J	CONDICIONAL	bne address
1010	JUMP	8	J	DESVIO	Jump address (0101)
1011	MOVE	4	R	MOVER	Move \$s0 \$s1
1111	IF BNE BEQ	4	R	CONDIÇÃO	If \$s0 \$s1

TABELA 2

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

A ULA é responsável por realizar as operações aritméticas, dentre elas: soma, subtração, multiplicação e eventualmente a ULA efetua operações de comparação de valor para realizar desvios condicionais, a ULA possui 4 entradas, o clock, ula_op que recebe qual operação a mesma vai realizar, um in_port_A e in_port_B onde vão ser recebidos os dois dados de 8 bits cada para realizar as operações, uma saída out_ula_result onde sairá os 8 bits do resultado das operações, zero onde sairá se ocorrerá um desvio condicional e a saída overflow que sai 1 se ocorrer um overflow durante a execução das operações.

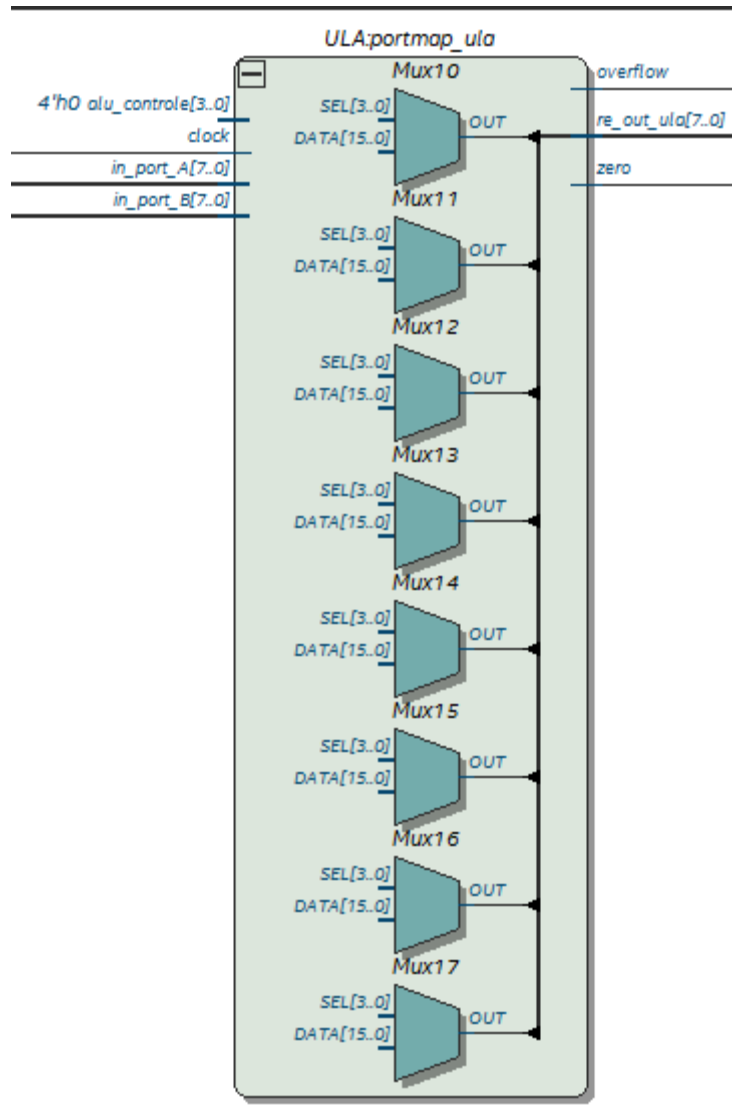


Figura 2

- Bloco simbólico do componente ALU gerado pelo Quartus

1.3.2 Banco de registradores:

O banco de registradores é responsável por armazenar os dados que são usados na execução das operações que utilizam o mesmo, possui 4 registradores que podem armazenar valores de 8 bits, possui como entrada, o clock, que vai ativar o componente, reg_write que ativa a opção de escrever dados no registrador, write_data que recebe o valor de 8 bits do dado a ser escrito no registrador de destino, o address_1 recebe 2 bits para o endereço do primeiro registrador, o address_2 recebe 2 bits para o endereço do segundo registrador, reg_out_A resulta em uma saída de 8 bits do valor armazenado no address_1 e o reg_out_B vai ter uma saída de 8 bits do valor armazenado no address_2 direto para o multiplexador 2x1.

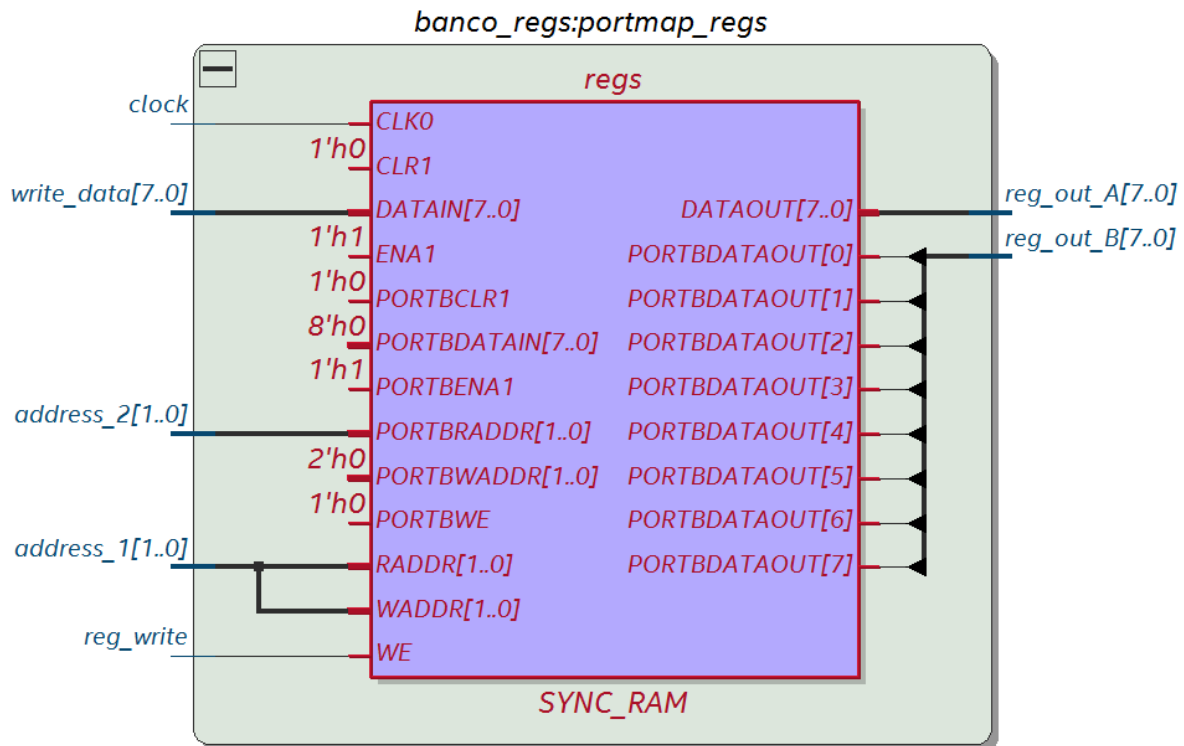


Figura 3

1.3.3 Unidade de Controle

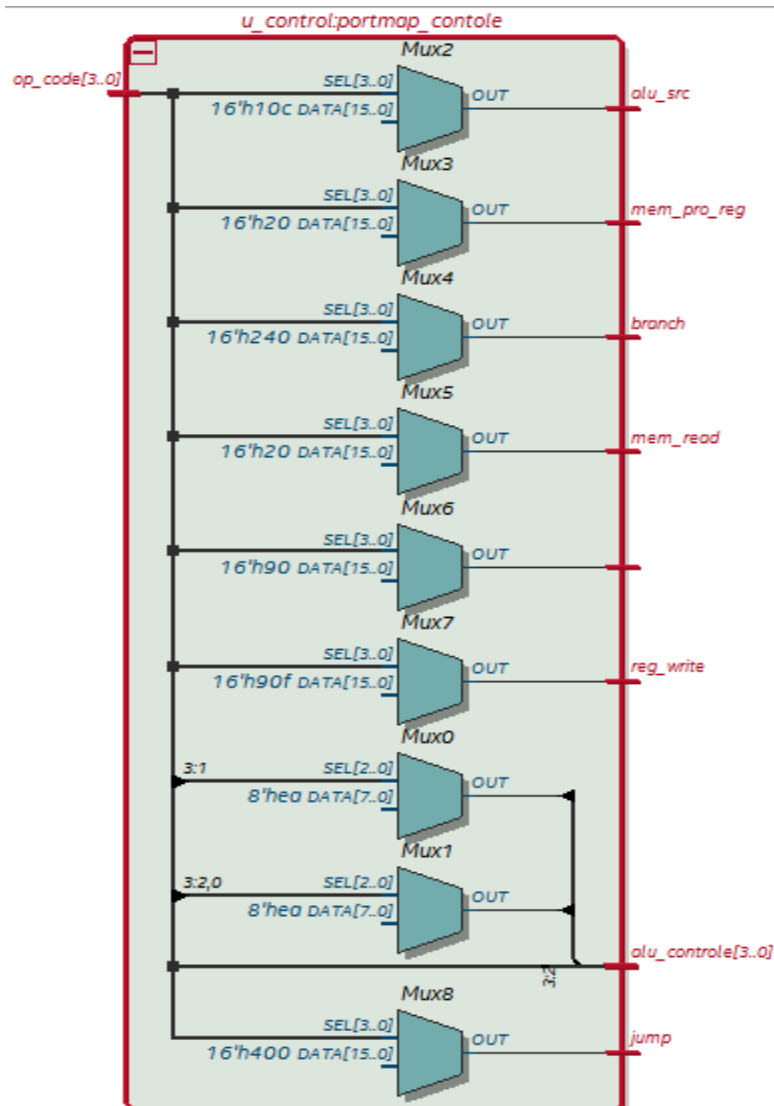
O componente Controle tem como objetivo realizar o controle de todos os componentes do processador de acordo com o opcode de 4 bits que é recebido na unidade. Esse controle é feito através das flags de saída abaixo:

- **mem_pro_reg** : Decide se o dado que será escrito no banco de registradores vai ser enviado pela ULA ou pela memória RAM.
- **alu_controle** : saída de 4 bits que será enviado para a ULA decidir qual operação será realizada.
- **alu_src**: Decide se o dado que entrará na ULA vai ser enviado pelo banco de registradores ou pelo extensor de sinal.
- **branch**: sinal para decidir se vai ocorrer um desvio condicional.
- **mem_read** : Decide se será lido um dado da memória RAM.
- **mem_volta** : Sinal para decidir se será escrito um dado da memória RAM.
- **Reg_write** : Sinal para decidir se o banco de registradores vai escrever um dado na posição do registrador de destino.
- **jump** : Sinal que vai decidir se vai ocorrer um desvio incondicional.
-

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.

COMMAND	mem_to_reg	alu_op	alu_src	branch	mem_read	mem_write	reg_write	jump
ADD	0	0000	0	0	0	0	1	0
SUB	0	0001	0	0	0	0	1	0
ADDI	0	0010	1	0	0	0	1	0
SUBI	0	0011	1	0	0	0	1	0
SW	0	0100	0	0	0	1	0	0
LW	1	0101	0	0	1	0	1	0
BEQ	0	0110	0	1	0	0	0	0
MUL	0	0111	0	0	0	0	1	0
LI	0	1000	1	0	0	0	1	0
BNE	0	1001	0	1	0	0	0	0
JUMP	0	1010	0	0	0	0	0	1
MOVE	0	1011	0	0	0	0	1	0
IF BNE BEQ	0	1111	0	0	0	0	0	0



- Figura 4

RTL viewer do componente Unidade de Controle

1.3.4 Memória Ram

A memória RAM é responsável por armazenar temporariamente os dados que são usados durante a execução das instruções, possui 5 entradas, o clock que ativa o componente, in_port_ram recebe o dado de 8 bits que será armazenado temporariamente na RAM, mem_volta recebe 1 bit para saber se vai armazenar dados na RAM, mem_read recebe 1 bit para saber se será lido algum dado da memória RAM, endereço recebe 8 bits da posição da memória RAM onde o dado deve ser escrito ou lido e possui 1 saída out_port onde sai um dado de 8 bits da posição que foi recebida no endereço, se a mem_read estiver setada em 1.

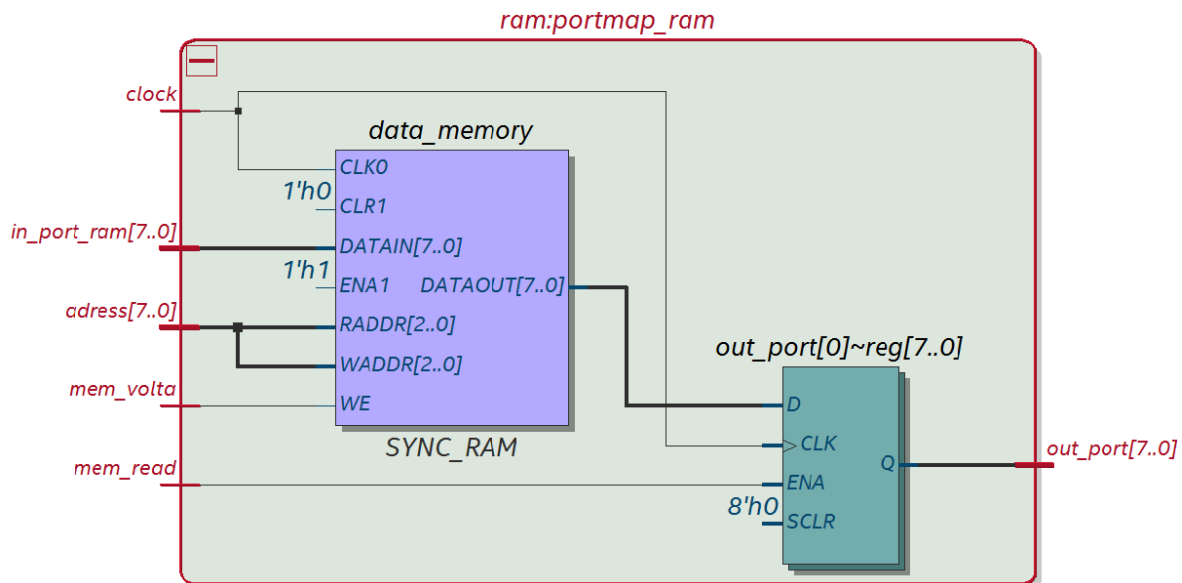


Figura 5

1.3.5 Memória Rom

A memória ROM tem a função de armazenar as instruções que vão serem executadas pelo processador, possui duas entradas, o clock e o in_port, endereço de 8 bits da instrução que será enviada para a execução, possui uma saída out_port de 8 bits da instrução que será executada.

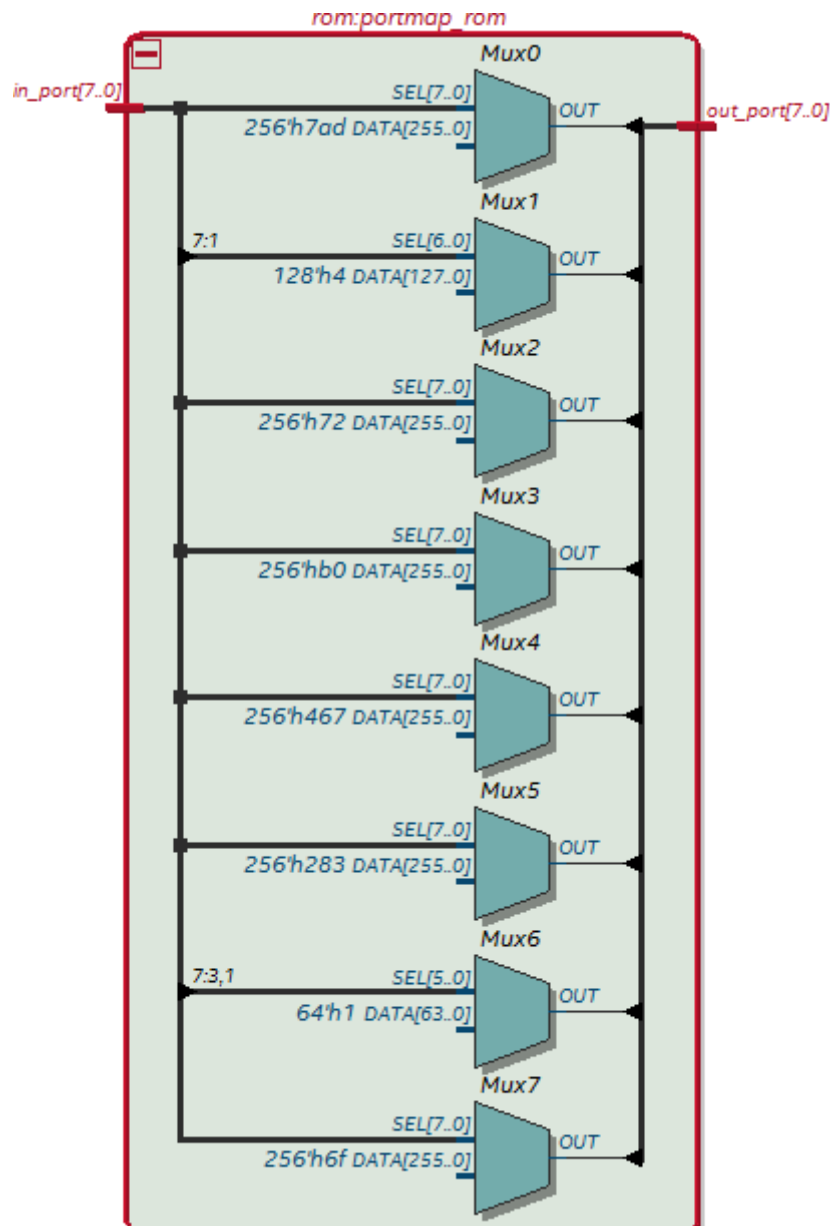


Figura 6

1.3.6 Somador

O somador do PC é responsável por somar o valor da instrução atual do PC com o valor 1 em binário, contém duas entradas, o clock, e o *in_port*, recebe o endereço de 8 bits da instrução atual do PC e possui uma saída *out_port* que receber o resultado da operação de soma do endereço atual com o valor 1, resultando no próximo endereço de instrução que será armazenado no PC.

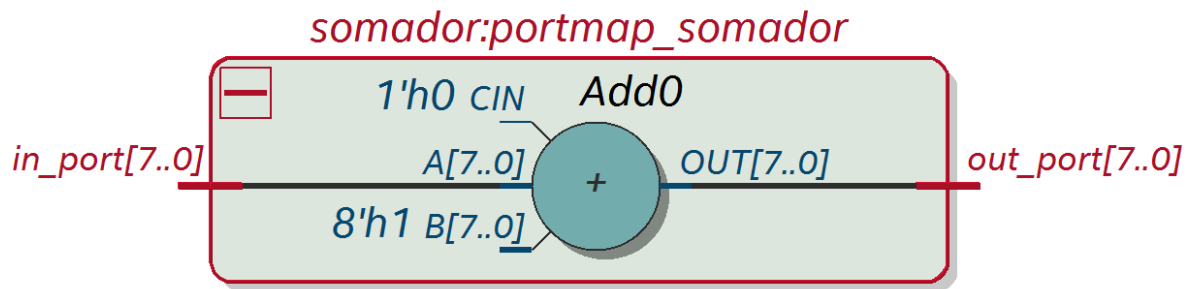


Figura 7

1.3.7 And

O componente AND tem a função de decidir se vai ocorrer um desvio condicional se as duas entradas estiverem recebendo o valor 1.

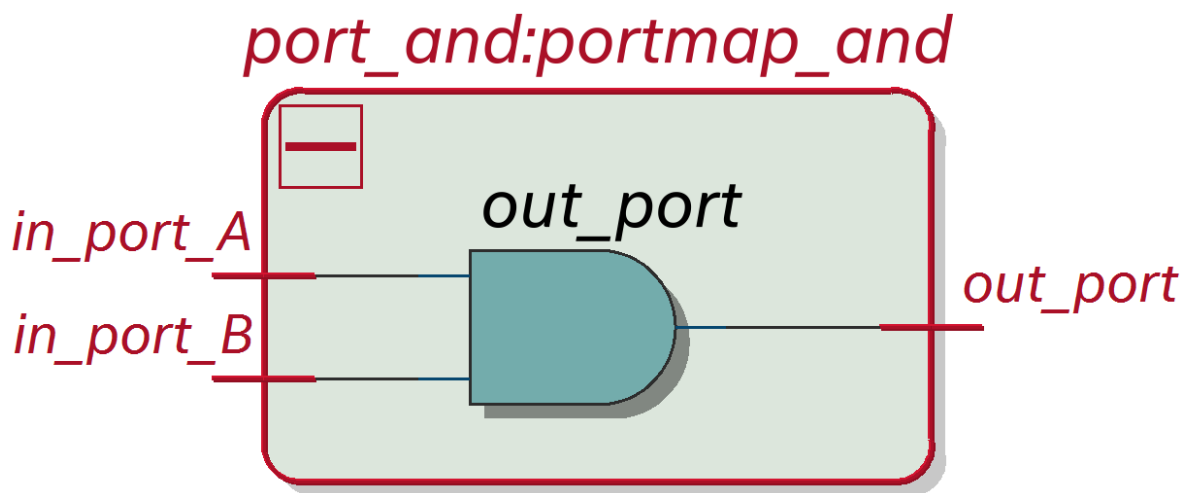


Figura 8

1.3.8 Mux_2x1

Um multiplexador tem duas entradas de 8 bits, *in_A* e *in_B* que receberá os dados e uma entrada *in_port* que decidirá qual dado vai sair, caso o *in_port* for 0 o dado que sairá na *out_port* será o dado *in_A* se for 1 sairá o dado *in_B*.

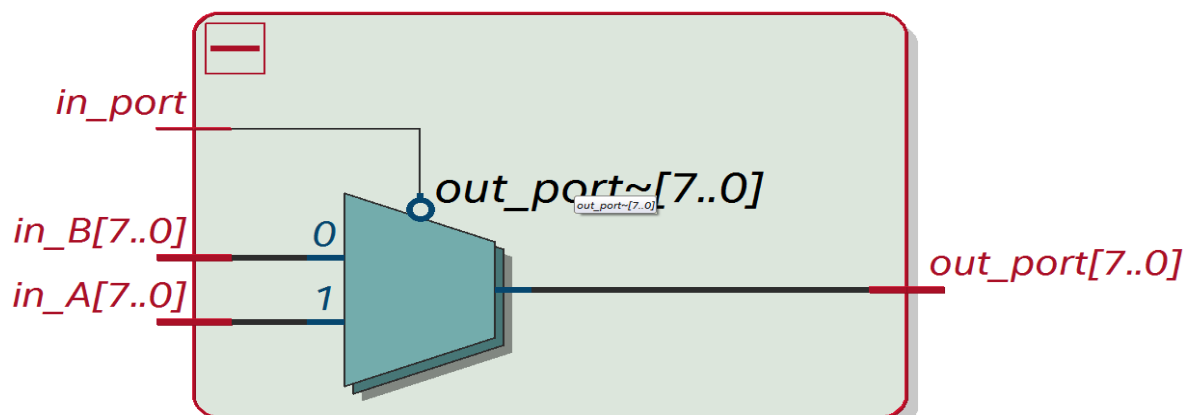


Figura 9

1.3.9 PC

A função do componente PC é armazenar o endereço de 8 bits da instrução que será executada, possui dois valores de entrada, o clock, que é o responsável por ativar o componente e o in_port, onde entra o endereço da instrução a ser executada e possui uma saída, out_port, onde sai o endereço da instrução a ser executada.

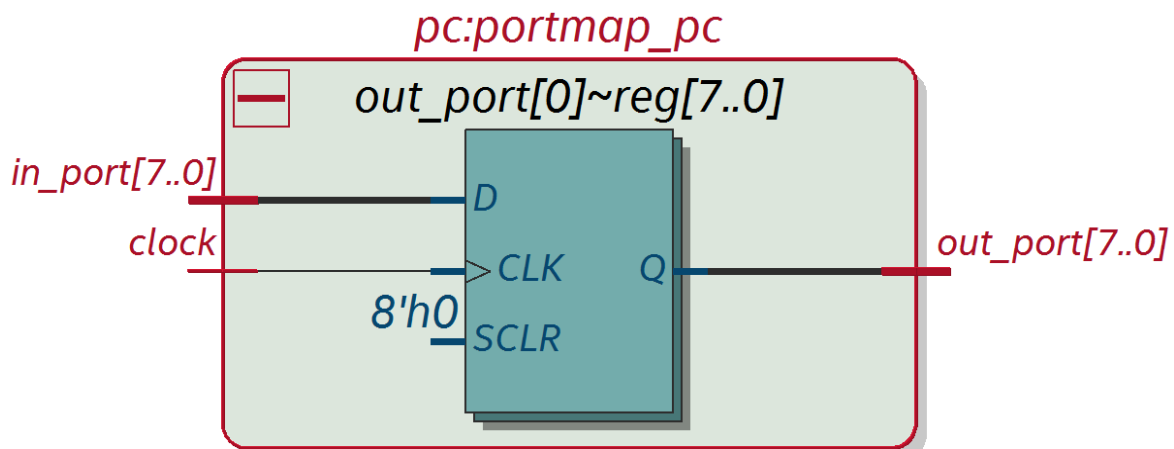


Figura 10

1.3.10 Extensor de Sinal 4 X 8

O extensor de sinal é responsável por estender o número de bits da entrada *in_port* de 4 bits para 8 bits.

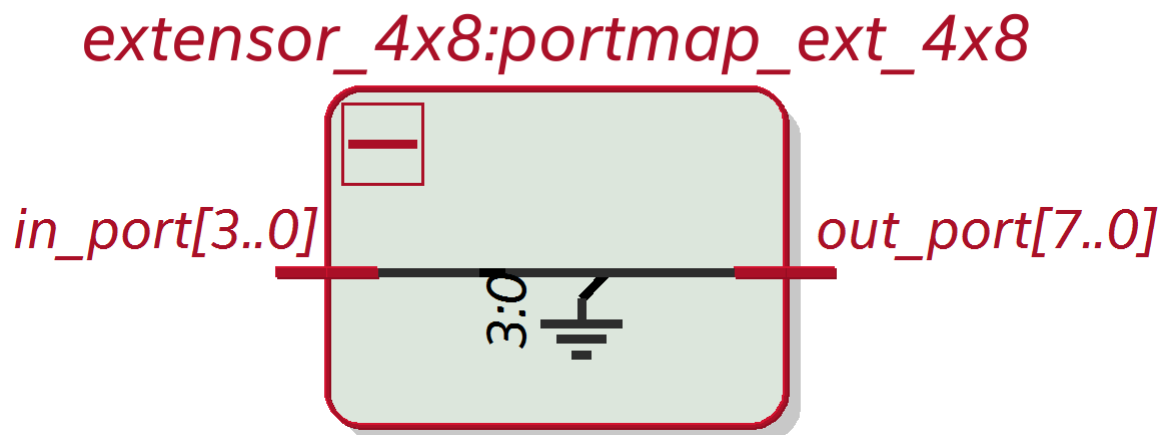


Figura 21

1.3.11 Extensor de Sinal 2 X 8

O extensor de sinal é responsável por estender o número de bits da entrada *in_port* de 2 bits para 8 bits.

extensor_2x8:portmap_ext_2x8

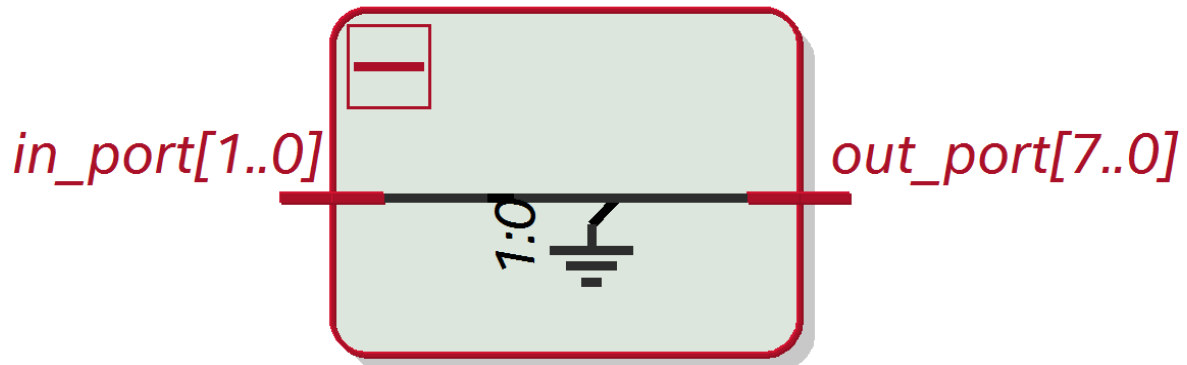
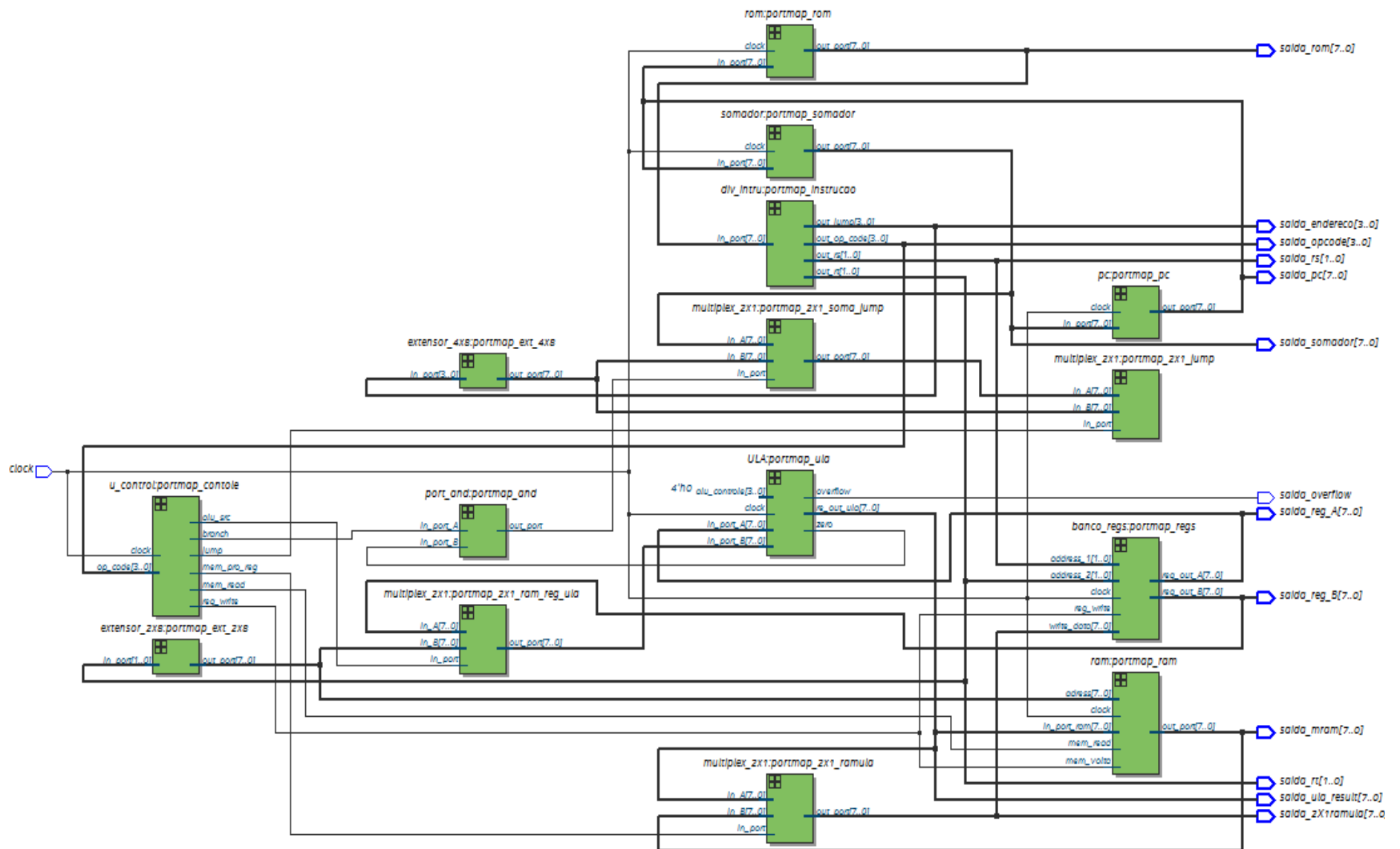


Figura 32

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. Veja na figura seguir.



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 ML21 utilizaremos como exemplo o código para calcular o número da sequência de **Fibonacci**.

Tabela 1 - Código Fibonacci para o processado.

LINGUAGEM MIPS	OPCODE	ADDRESS	REG 1	REG 2
li S3 3	1000	0	11	11
mul S3 S3	0111	1	11	11
addi S3 1	0010	2	11	01
addi S3 2	0010	3	11	10
addi S3 2	0010	4	11	10
li S2 0	1000	5	10	00
li S2 0	1000	6	00	00
sw S0 ram(00)	0100	7	00	00
Li S0 1	1000	8	00	01
sw S0 ram(01)	0100	9	00	01
lw S0 ram(00)	0101	10	00	00
add S1 S0	0000	11	01	00
lw S0 ram(01)	0101	12	00	01
add S1 S0	0000	13	01	00
sw S0 ram(00)	0100	14	00	00
sw S0 ram(01)	0100	15	01	01
addi S2 1	0010	16	10	01
if S2 == S3	1111	17	10	11

bne S2 != S3	1010	18	1010
---------------------	-------------	-----------	-------------

Tabela 5 - Código do fibonacci

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

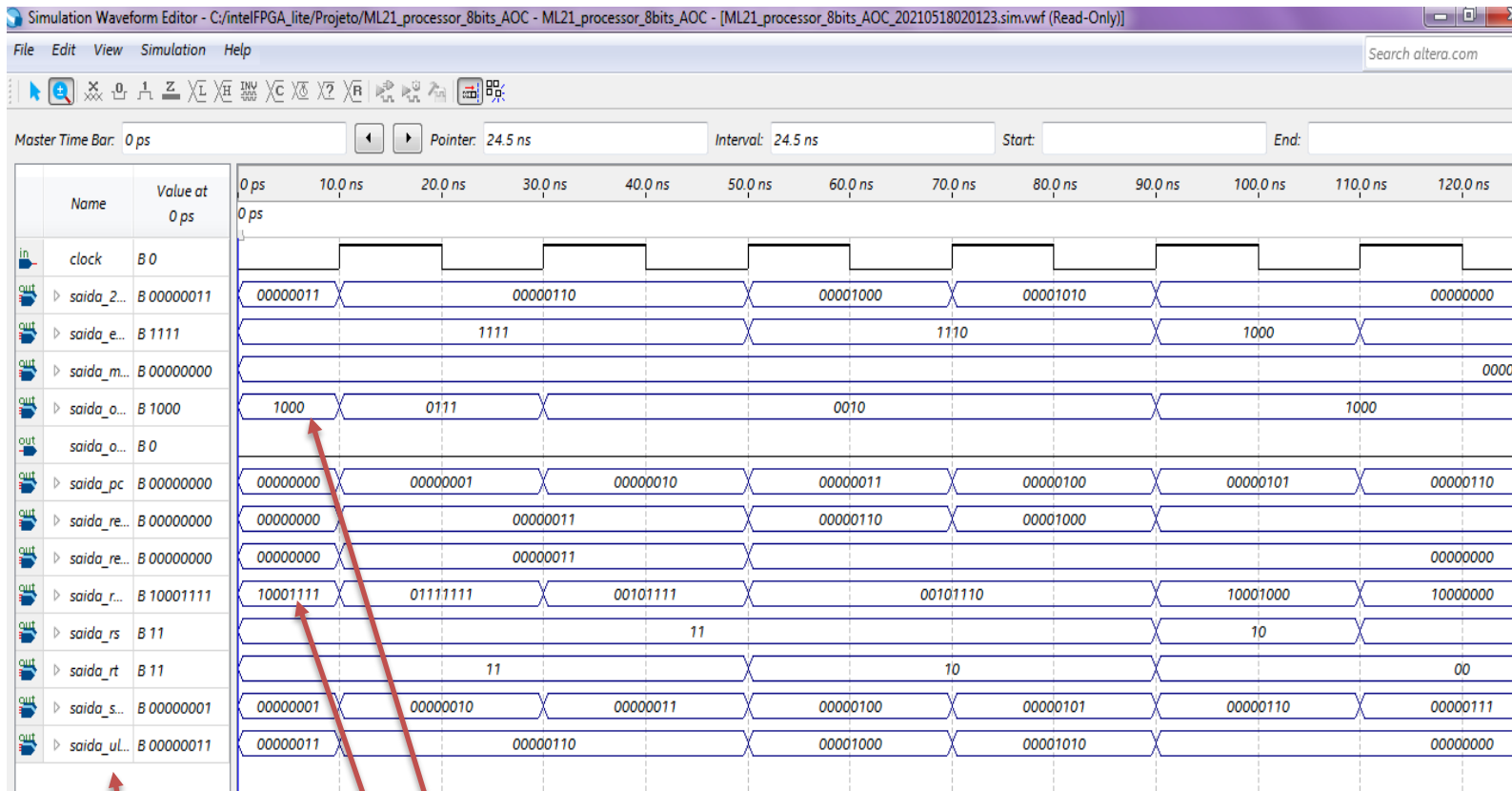


Figura 13

- Resultado na waveform

Estes são os
pinos de saída

Nestes pontos o
processador
inicia a execução
das instruções.

3 Considerações finais

Este trabalho apresentou o projeto e implementação do processador de 8 bits baseado em MIPS denominado de ML21. Todo o projeto foi realizado na modalidade EAD (educação a distância), devido a pandemia da COVID-19, muitas dificuldades foram encontradas ao longo de todo o projeto, porém, buscando referencias e testando vários códigos foi possível superar as mesmas. Todo o processador teve êxito na implementação, mas, foi encontrada uma última dificuldade que foi a conversão da saída da ULA de binário para inteiro que ainda não foi possível resolvida no processador em questão. Vale lembrar de uma limitação em que o load e store funcionam em somente uma instrução e devido a limitação de 8 bits, somente dois bits são usados para acessar a memória RAM, o que limita o processador a poder acessar somente as 4 primeiras posições da memória RAM.

REFERENCIAS:

PATTERSON, D.; HENESSY, J. L. Organização e projeto de computadores: a interface hardware/software. 3ª Edição. São Paulo: Elsevier, 2005, 484 p.

STALLINGS, William; FIGUEIREDO, Carlos Camarão De; MIDORIKAWA, Edson Toshimi. **Arquitetura e organização de computadores: projeto para o desempenho**. 5. ed. São Paulo: Prentice hall, 2006. 786p: il. ISBN: 8587918532.

TANENBAUM, Andrew S. 1944. **Organização estruturada de computadores**. 5. ed. São Paulo: Pearson/Prentice Hall, 2007. 449 p: il. ISBN: 8576050676.