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

ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES

RELATÓRIO DO PROJETO: NBSOULESSPROCESSOR DE 8 BITS

ALUNOS:
BARBARA ZAMPERETE OLIVEIRA – 2017012486
NATALIA RIBEIRO DE ALMADA - 2017009364

Novembro de 2019
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: NBSOULESSPROCESSOR DE 8 BITS

Novembro de 2019
Boa Vista/Roraima

Resumo

Este trabalho aborda o projeto e implementação dos componentes um processador de 8 bits uniciclo/monocycle, que nomeamos NBSouless.

Conteúdo

Lista de Figuras

Lista de Tabelas

1 Especificação

Nesta seção serão apresentados os itens para o desenvolvimento do processador e a descrição de cada etapa da construção do processador.

1.1 Plataforma de desenvolvimento

Para a implementação do processador NBSOULESSPROCESSOR foi utilizada a IDE: Quartus Prime 18.1 Lite Edition

1.2 Conjunto de instruções

O processador NBSOULESSPROCESSOR possui os registradores S0 e S1, que combinam-se a 3 formatos de instruções do tipo R, I e J de 8 bits:
Opcode : operação básica executada pelo processador.
Reg1 : o registrador contendo o primeiro operando, é o registrador de destino;
Reg2 : o registrador contendo o segundo operando, é a fonte;
Funct: Tipo de Instruções:

Tipo de Instruções:
Formato do tipo R: EInstruções de operações aritméticas.
Tipo da Instrução Reg1 Reg2 funct
Formato para escrita em código binário:

3 bits	1 bits	1 bits	3
7-5	4	3	2-0
Opcode	Reg2	Reg1	Funct

Formato do tipo I: Instruções de Load, Store, Load Immediately e Branch.

Tipo da Instrução Reg1 funct
Formato para escrita em código binário:

3 bits	1 bits	4 bits
7-5	4	3
Opcode	Reg1	Funct

3
Formato do tipo J: Instrução de Jumps.

Formato para escrita em código binário:

3 bits	3 bits
7-5	4-0
Opcode	Endereço

Visão geral das instruções do Processador XXXX:

São 4 bits do campo das instruções Opcod, então: ($Bit(0e1)NumeroTodaldeBitsdoOpcode \therefore 2X = X$) totalizam 8 Opcodes que preenchem de 0-7

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

Opcode	Nome	Formato	Breve Descrição	Exemplo
000000	ADD	R	SOMA	**add** $S0,$S1
000001	ADD	R	Soma	**sub** $S0, $S1 ,ou seja, $S0 := $S0-$S1
000100	SUB	R	Subtração	**sub** $S0, $S1 ,ou seja, $S0 := $S0 - $S1
000100	MULT	R	Multiplicação	**mult** $S0, $S1 ,ou seja, $S0 := $S0 * $S1
001	LW	I	Load	**Lw** $S0,8
110	LI	I	Load Immediataly	**Li** $S0, 10
010	SW	I	Store	**sw** $S0, 8
011	BNE	I	Branch Not Equal	**bne** $S0,$S1,TESTE
100	BEQ	I	Branch Equal	**beq** $S0,$S1,TESTE
111	JUMP	J	Jump	**jump** Endereço

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 ULA (Unidade Lógica Aritmética) tem como principal objetivo efetuar as principais operações aritméticas de números inteiros, e também efetua operações de comparação de valor como igual ou diferente. O ULA recebe três valores: **a** - dado de

8bits para operação, **b** - dado de 8bits para operação e **ula_controle** - identificador da operação que será realizada de 3bits. A ULA também possui duas saídas: **zero** - identificador de
resultado (1bits) para comparações (1 se verdade e 0 caso contrário); e **ula_resultado** -saída com o resultado das operações aritméticas

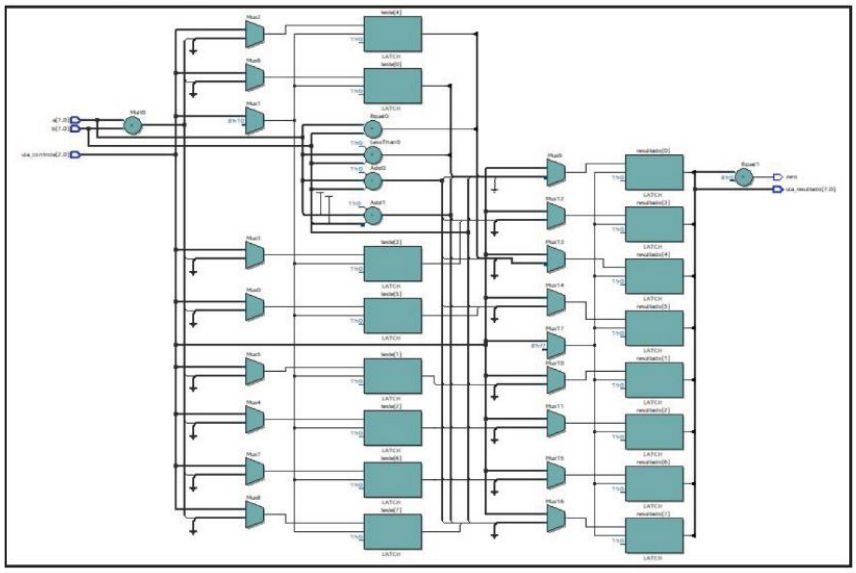
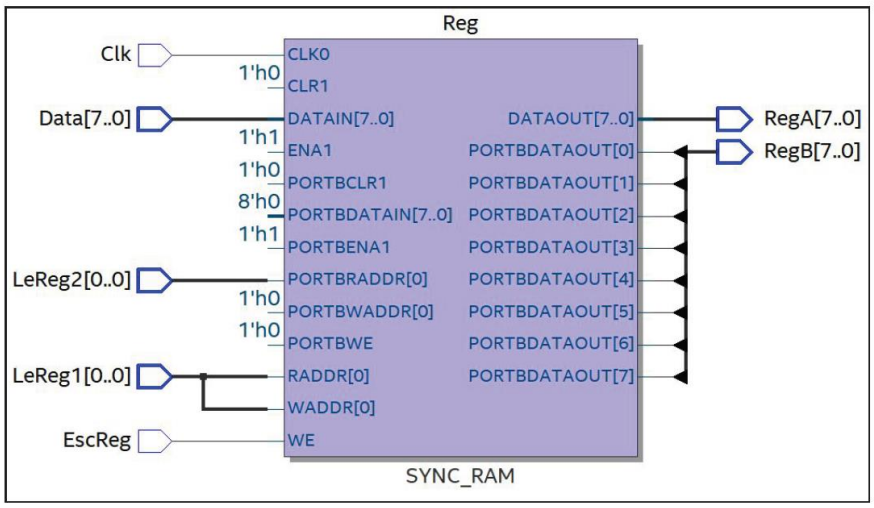


Figura 1 - Bloco simbólico do componente QALU gerado pelo Quartus

Comentado [h1]: Figura do RTL viewer

1.3.2 Banco de Registradores

O BDRegister guarada valores e/ou resultados das operações realizadas pela



ULA.

1.3.3 Clock

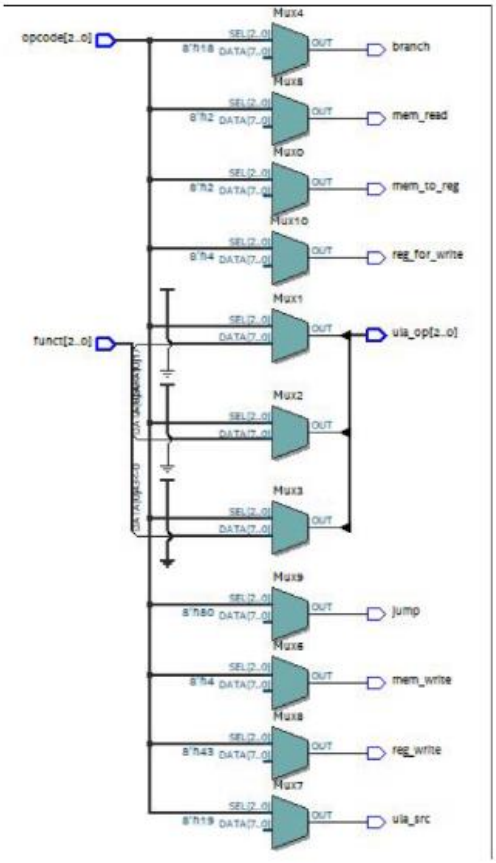
Serve para manter os components funcionando apenas quando ee estiver ligado, portanto, se for True/1

1.3.4 Controle

Este componente controla todos os componentes do processador de acordo com o opcode, através das flags de saída:
mem_to_reg: XXXX.
ula_op: XXXX.
branch: XXXX.
mem_read: XXXX.
mem_write: XXXX.
ula_src: XXXX.
reg_write: XXXX.
jump: XXXX.
reg_for_write: XXXX.

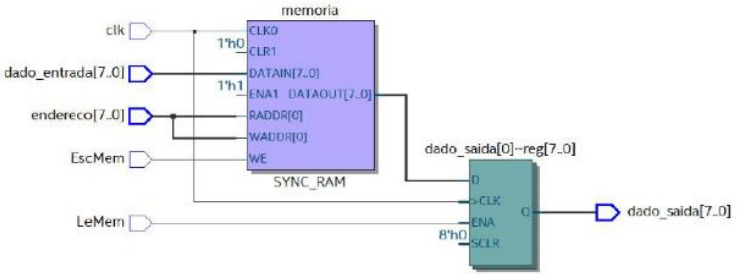
Abaixo segue a tabela, onde é feita a associação entre os opcodes e as flags de controle

Comando	mem_to_reg	ula_op	branch	mem_read	mem_write	ula_src	reg_write	jump	reg_for_write
add	0	000	0	0	0	1	1	0	0
sub	0	001	0	0	0	1	1	0	0
mult	0	100	0	0	0	1	1	0	0
lw	1	110	0	1	0	0	1	0	0
li	0	110	0	0	0	0	1	0	0
sw	0	110	0	0	1	0	0	0	1
bne	0	101	1	0	0	1	0	0	0
beq	0	010	1	0	0	1	0	0	0
jump	0	111	0	0	0	0	0	1	0



1.3.5 Memória de dados ou RAM

Armazena os valores na memória e os disponibiliza para serem salvos nos registradores. Tendo como entrada **dado_entrada** (que define o que vai ser salvo na memória); **endereco** - o endereço (onde o valor será salvo na memória); **EscMem** - flag que define se algo vai ser escrito na memória ou não; **LeMem** - flag que define se algo vai ser lido da memória e passada para o registrador; **clk** - clock. E ele retorna com o que foi lido na memória - **dado_saida**.



1.3.6 Memória de Instruções ou ROM

Onde todas as instruções serão executadas e é a partir dela que o opcode vai para a unidade de controle e que sabemos quais registradores usar. A ROM recebe um endereço que vem do PC e executa o que está nesta instrução.

1.3.7 Somador

Soma cada PC para que ele execute as instruções

1.3.8 And

Verifica se vai ou não ocorrer um branch

1.3.9 Mux_2x1

O multiplexador auxilia a decidir que trilha será usada dependendo do valor da flag que está recebendo.

1.3.10 PC

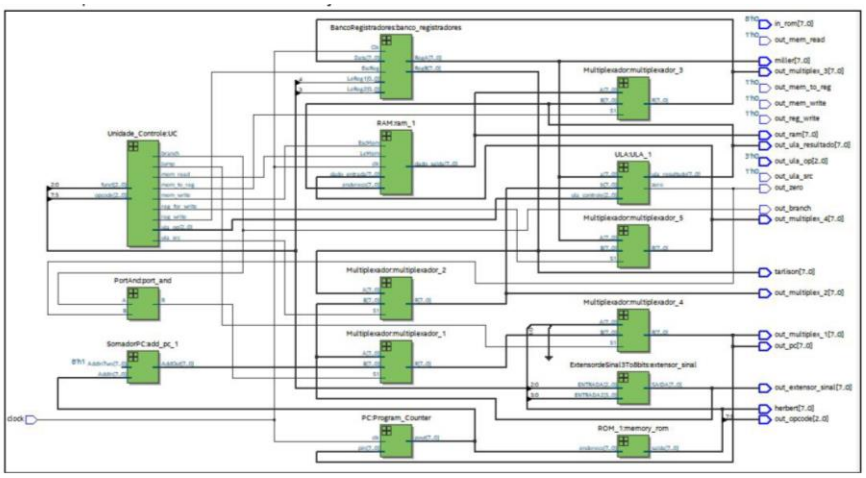
Responsável em mandar o endereço para a próxima instrução para a memória de instruções.

1.3.11 ZERO

Identificador de resultado (1bit) para comparações (1 se verdade e 0 se falso).

1.4 Datapath

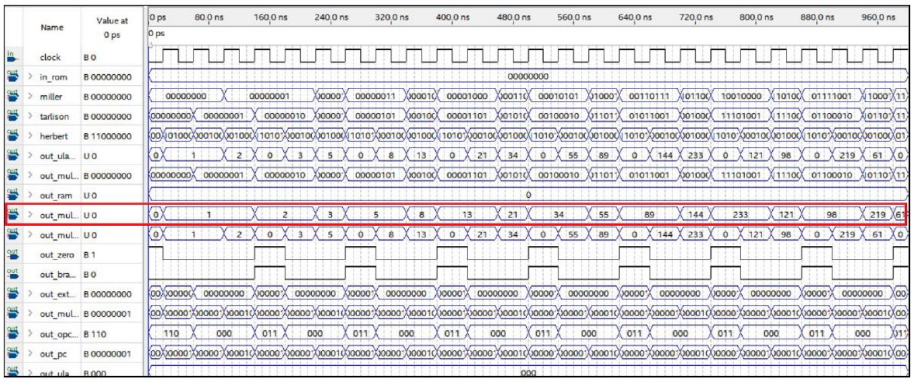
É a conexão entre as unidades funcionais formando um único caminho de dados e acrescentando uma unidade de controle responsável pelo gerenciamento das ações que serão realizadas para diferentes classes de instruções.



2 Simulações e Testes

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

Endereço	Linguagem de Alto Nível	Binário			
		Opcode	Reg2	Reg1	Funct
			Endereço		
			Dado		
0	**LI** $S0, 0	110	0	0000	
			00000000		
1	**LI** $S1, 1	110	1	0001	
			00000001		
2	**ADD** $S0, $S1	000	0	1	000
3	**ADD** $S0, $S1	000	1	0	000
4	**BNE** $S0, SS1	011	0	1	010



Este trabalho apresentou o projeto e implementação do processador uniciclo de 8 bits referente ao projeto final de disciplina de Arquitetura e Organização de computadores. O processador contém todos os requisitos para que tenha essa alcunha pois contém operações aritméticas, memória de dados, condicionais, controle de dados e afins.