UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

INSTITUTO DE INFORMÁTICA

DISCIPLINA: SISTEMAS DIGITAIS

Prof. Fernanda Lima Kastensmidt

**Aula Remota NEANDER – parte I**

**Projeto do Processador Neander em VHDL**

O computador NEANDER foi criado com intenções didáticas pelo prof. Raul Weber da UFRGS. Neste site há referencias e link para o simulador: <http://www.dcc.ufrj.br/~gabriel/neander.php>

O objetivo deste trabalho de SD é implementar o NEANDER usando a linguagem de descrição de hardware VHDL, simular esse circuito em um simulador lógico sem atraso, depois e realizar a síntese lógica, mapeamento tecnológico, posicionamento e roteamento para um FPGA, realizar a simulação com atraso e prototipar o processador em uma placa de prototipação.

O computador NEANDER tem as seguintes características:

• Largura de dados e endereços de 8 bits

• Dados representados em complemento de dois

• 1 acumulador de 8 bits (AC)

• 1 apontador de programa de 8 bits (PC)

• 1 registrador de estado com 2 códigos de condição: negativo (N) e zero (Z)



**A aula 1 remota do NEANDER pede para descrever em VHDL o Datapath do Neander, ou seja, tudo que está na figura, menos a parte de controle (que será uma FSM e a memória BRAM).**

1. INSIRA AQUI O VHDL DO DATAPATH

-- Sistemas Digitais Para Computadores A

-- Matheus Almeida Silva - 00316326

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

use IEEE.std\_logic\_unsigned.ALL;

use IEEE.std\_logic\_arith.ALL;

entity neander is

Port ( clk : in STD\_LOGIC;

rst : in STD\_LOGIC;

selMux : in STD\_LOGIC;

cargaREM : in STD\_LOGIC;

cargaAC : in STD\_LOGIC;

cargaNZ : in STD\_LOGIC;

cargaRI : in STD\_LOGIC;

cargaPC : in STD\_LOGIC;

incPC : in STD\_LOGIC;

selULA: in STD\_LOGIC\_VECTOR (2 downto 0);

Z : out STD\_LOGIC;

N : out STD\_LOGIC;

AC : out STD\_LOGIC\_VECTOR (7 downto 0);

PC : out STD\_LOGIC\_VECTOR (7 downto 0);

outputDecod : out STD\_LOGIC\_VECTOR (3 downto 0);

outputData : out STD\_LOGIC\_VECTOR (7 downto 0));

end neander;

architecture Behavioral of neander is

-- Mux 2x1

signal outputMux: STD\_LOGIC\_VECTOR (7 downto 0);

--Decod

signal instruction: STD\_LOGIC\_VECTOR(15 downto 0);

signal decod: STD\_LOGIC\_VECTOR(3 downto 0);

-- REM

signal regREM: STD\_LOGIC\_VECTOR(7 downto 0);

signal outputREM: STD\_LOGIC\_VECTOR(7 downto 0);

-- ULA

signal input1ULA : STD\_LOGIC\_VECTOR(7 downto 0);

signal input2ULA : STD\_LOGIC\_VECTOR(7 downto 0);

signal outputULA: STD\_LOGIC\_VECTOR(7 downto 0);

signal regULA: STD\_LOGIC\_VECTOR(7 downto 0);

-- PC

signal regPC, outputPC: STD\_LOGIC\_VECTOR (7 downto 0);

--AC

signal inputAC: STD\_LOGIC\_VECTOR(7 downto 0);

signal outputAC: STD\_LOGIC\_VECTOR(7 downto 0);

--NZ

signal regN: STD\_LOGIC;

signal regZ: STD\_LOGIC;

signal outputN: STD\_LOGIC;

signal outputZ: STD\_LOGIC;

--RI

signal inputRI: STD\_LOGIC\_VECTOR(7 downto 4);

signal outputRI: STD\_LOGIC\_VECTOR(7 downto 4);

-- Memória

signal writeMem: STD\_LOGIC\_VECTOR(0 to 0) :="0";

signal outputMem: STD\_LOGIC\_VECTOR(7 downto 0);

begin

--Mux

process (selMux, outputPC, outputMem)

begin

if (selMux = '0') then

outputMux <= outputPC;

else

outputMux <= outputMem;

end if;

end process;

--PC

process (clk, rst)

begin

if rst ='1' then

regPC <= "00000000";

elsif (clk'event and clk='1') then

if (cargaPC='1') then

regPC<= outputMem;

elsif(incPC='1') then

regPC <= regPC + 1;

else

regPC <= regPC;

end if;

end if;

end process;

outputPC <= regPC;

--REM

process(clk, rst)

begin

if rst='1' then

regREM <= "00000000";

elsif (clk'event and clk='1') then

if (cargaREM ='1') then

regREM <= outputMux;

else

regREM<= regREM;

end if;

end if;

end process;

outputREM <= regREM;

--ULA

input1ULA <= outputAC;

input2ULA <= outputMem;

process(selULA, input1ULA, input2ULA)

begin

case selULA is

when "000" => regULA <= (input1ULA + input2ULA);

when "001" => regULA <= (input1ULA AND input2ULA);

when "010" => regULA <= (input1ULA OR input2ULA);

when "011" => regULA <= (NOT input1ULA);

when "100" => regULA <= input2ULA;

when others => regULA <= "00000000";

end case;

end process;

outputULA <= regULA;

--AC

process (clk, rst)

begin

if rst='1' then

inputAC <= "00000000";

elsif (clk'event and clk='1') then

if (cargaAC='1') then

inputAC <= outputULA;

else

inputAC <= inputAC;

end if;

end if;

end process;

outputAC <= inputAC;

--RI

process (clk, rst)

begin

if rst='1' then

inputRI<= "0000";

elsif (clk'event and clk='1') then

if (cargaRI='1') then

inputRI <= outputMem(7 DOWNTO 4);

else

inputRI <= inputRI;

end if;

end if;

end process;

outputRI <= inputRI;

--NZ

process (clk, rst)

begin

if rst='1' then

regN <= '0';

regZ <= '0';

elsif(clk'event and clk='1') then

if outputAC = "00000000" then

regZ <= '1';

else

regZ <= '0';

end if;

regN <= outputAC(7);

end if;

end process;

outputZ <= regZ;

outputN <= regN;

--DECOD

decod <= outputRI (7 downto 4);

process(decod)

begin

instruction <= "0000000000000000";

case decod is

when "0000" => instruction(0) <= '1'; --NOP

when "0001" => instruction(1) <= '1'; --STA

when "0010" => instruction(2) <= '1'; --LDA

when "0011" => instruction(3) <= '1'; --ADD

when "0100" => instruction(4) <= '1'; --OR

when "0101" => instruction(5) <= '1'; --AND

when "0110" => instruction(6) <= '1'; --NOT

when "0111" => instruction(7) <= '1'; --NOP

when "1000" => instruction(8) <= '1'; --JMP

when "1001" => instruction(9) <= '1'; --JZ

when "1010" => instruction(10) <= '1'; --JN

when "1011" => instruction(11) <= '1'; --NOP

when "1100" => instruction(12) <= '1'; --SUB

when "1101" => instruction(13) <= '1'; --XOR

when "1110" => instruction(14) <= '1'; --NOP

when "1111" => instruction(15) <= '1'; --HLT

when others => instruction <= "0000000000000000";

end case;

end process;

Z <= outputZ;

N <= outputN;

AC <= outputAC;

PC <= outputPC;

outputData <= outputMem;

outputDecod <= decod;

end Behavioral;

Preencha:

**Dados de Area do Datapath do Neander**

FPGA device:

Numero de 4-LUTs:

Numero de ffps:

Numero de MULT e ADD DSP

Tabela

Descrição gerada automaticamente

Interface gráfica do usuário, Aplicativo, Tabela

Descrição gerada automaticamente