# Relatório Prática 03 – 01/09/2023

# Arthur Souza/João Paulo – PN1

Iniciou a prática criando uma entidade nomeado Comparador no software QUARTUS II 13.0, especificando qual chip está sendo utilizado, no caso o EP2C35F672C6.

A partir disso, criou o primeiro arquivo em VHDL de mesmo nome que a entidade no qual foi colado uma parte do código do testbench disponibilizado e adicionado uma seção para realizar comparação das palavras considerando-as com complemento de 2:

***library IEEE;***

***use IEEE.STD\_LOGIC\_1164.all;***

***use ieee.numeric\_std.all;***

***entity Comparador is***

***generic***

***(***

***DATA\_WIDTH : natural := 16***

***);***

***port***

***(***

***a : in std\_logic\_vector ((DATA\_WIDTH-1) downto 0);***

***b : in std\_logic\_vector ((DATA\_WIDTH-1) downto 0);***

***maior : out std\_logic;***

***menor : out std\_logic;***

***igual : out std\_logic***

***);***

***end Comparador;***

***architecture dataflow of Comparador is***

***begin***

***igual<= '1' when signed(a)=signed(b) else '0';***

***maior<= '1' when signed(a)>signed(b) else '0';***

***menor<= '1' when signed(a)<signed(b) else '0';***

***end dataflow;***

Utilizou-se a estrutura logica WHEN-ELSE para comparação de cada palavra.

Simulado e atestado ausência de erros, verificou-se a esquematização do código em circuito pela opção RTL viewer e Technology Map Viewer. Segue abaixo as imagens:

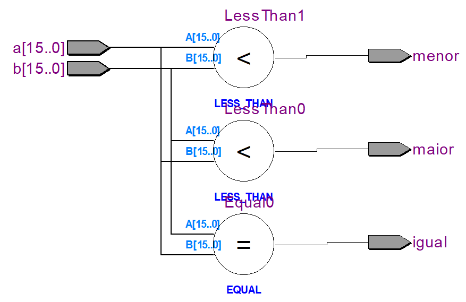


Figura : Circuito Comparadorr

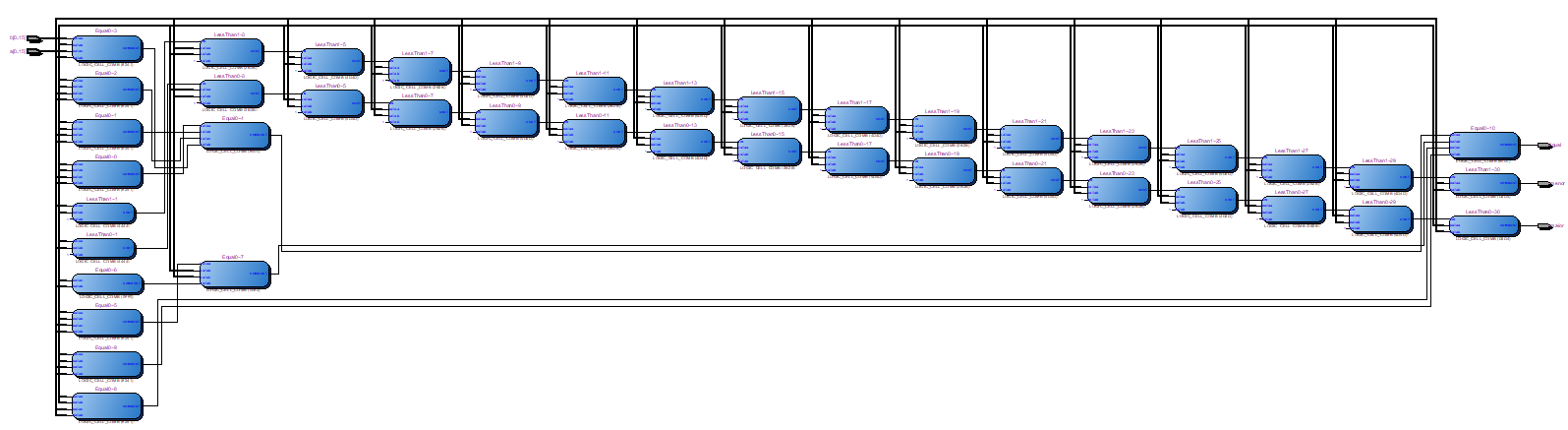


Figura : Diagrama do circuito Comparador

No map viewer foi possível observar a organização interna de cada bloco comparador, como visto no exemplo abaixo, inclusive vendo o nível de complexidade para um circuito de 16bits.

Diagrama, Desenho técnico

Descrição gerada automaticamente

Figura : Circuito interno do Bloco

Seguiu-se para a compilação do testbench disponibilizado, este chamado de tb\_comparador, e que irá definir os testes do projeto. Ele está descrito abaixo:

***library IEEE;***

***use IEEE.STD\_LOGIC\_1164.all;***

***use ieee.numeric\_std.all;***

***entity tb\_comparador is***

***end tb\_comparador;***

***architecture teste of tb\_comparador is***

***component Comparador is***

***generic***

***(***

***DATA\_WIDTH : natural := 16***

***);***

***port***

***(***

***a : in std\_logic\_vector ((DATA\_WIDTH-1) downto 0);***

***b : in std\_logic\_vector ((DATA\_WIDTH-1) downto 0);***

***maior : out std\_logic;***

***menor : out std\_logic;***

***igual : out std\_logic***

***);***

***end component;***

***signal fio\_A, fio\_B: std\_logic\_vector(3 downto 0);***

***signal fio\_maior, fio\_menor, fio\_igual: std\_logic;***

***begin***

***-- Note que o componente é instanciado com apenas 4 bits nas entradas para facilitar a simulação:***

***instancia\_comparador: Comparador generic map (DATA\_WIDTH => 4) port map(a=>fio\_A,b=>fio\_B,maior=>fio\_maior, menor=>fio\_menor,igual=>fio\_igual);***

***-- Dados de entrada de 4 bits são expressos em "hexadecimal" usando "x":***

***fio\_A <= x"0", x"8" after 20 ns, x"7" after 40 ns, x"4" after 60 ns;***

***fio\_B <= x"0", x"7" after 10 ns, x"8" after 30 ns, x"1" after 50 ns;***

***end teste;***

Com ele, podemos simular o funcionamento do circuito através do software MULTISIM. Basta apenas indica o arquivo como testbench em simulation e começar a simulação em RTL. Irá abrir um gráfico de sinais com valores determinados pelo testbench e irá auxiliar na verificação da lógica.

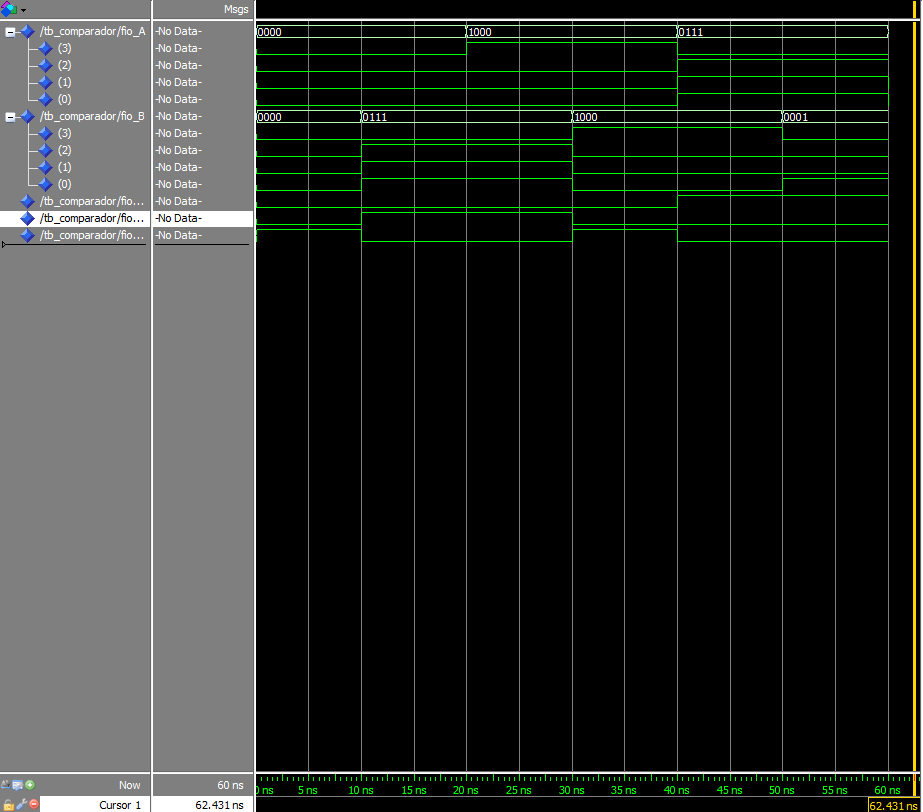


Figura : Simulação Comparador no Multisim (signed)

Além disso, foi feito uma alteração na entidade comparador para que agora as palavras fossem consideradas como valores absolutos, sendo apenas uma troca na conversão de signed para unsigned. Abaixo segue a simulação no multisim para comparação

Gráfico

Descrição gerada automaticamente com confiança média

Figura : Simulação Comparador no Multisim (unsigned)

Para facilitar o processo e possibilitar o uso do arquivo antigo de pinagem, foi determinado que os pinos que não seriam usados fossem relacionados ao gnd para não haver interferência durante os testes na placa FPGA.

***a[3] <=*** ***PIN\_N25***

***a[2] <= PIN\_N26***

***a[1] <= PIN\_P25***

***a[0] <= PIN\_AE14***

***b[3] <= PIN\_AC13***

***b[2] <= PIN\_C13***

***b[1] <= PIN\_B13***

***b[0] <= PIN\_A13***

***maior <= PIN\_AE23***

***igual <= PIN\_AF23***

***menor <= PIN\_AB21***

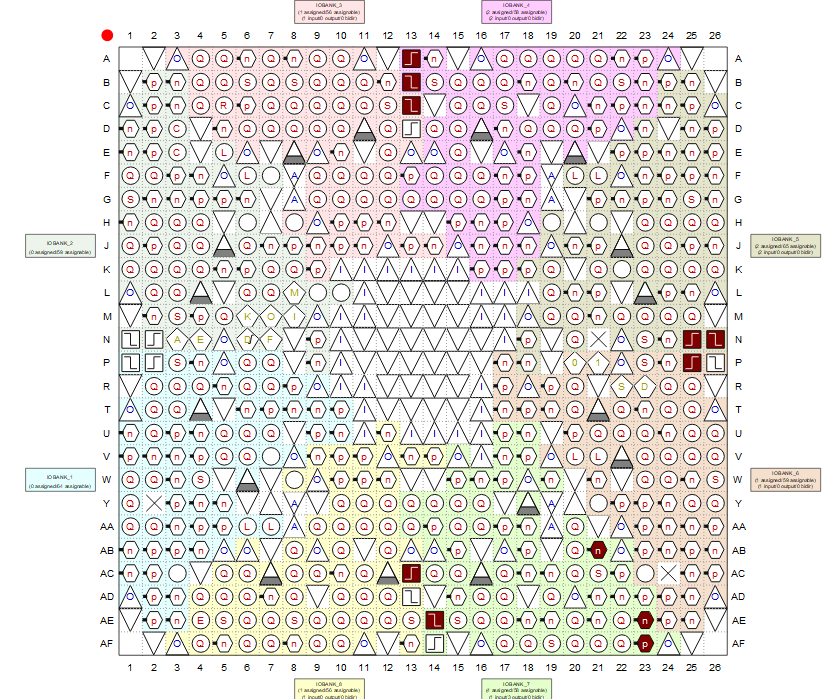


Figura 6: Pin Planner

Simulando novamente para verificação dos pinos, podemos enviar o projeto para o FPGA usando a função PROGRAMMER e o usb blaster.

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Figura : Programmer