Universidade Federal de Uberlândia
UFU
Curso de Ciência da Computação
AOC2

Relatório de FPGA

Aluno: Antonio Carlos Neto
Aluno: Higor Emanuel Souza Silva
Professor: Alexandro

Setembro
2017

Conteúdo

1 Introdução

Com o intuito de aprendermos a programar em VHDL e usar a placa Altera DE2-115, uma FPGA(Field Programmable Gate Array), assistimos todas as videoaulas na playlist chamada Curso de VHDL com URL "https://www.youtube.com/playlist?list=PLYE3wKnWQbHA9HVn6dq7cywyaHzkx-WOP", do usuário "Felipe Pfirmer". Com o auxilio do qsf fonercido pelo professor, a tradução do código fornecido pelo Felipe Pfimer foi facilitada, apenas alterando o nome das entradas e saídas do código em VHDL, e mudando o tipo da variável para STATIC_LOGIC e STATIC_LOGIC_VECTOR, que possuem o mesmo comportamento da usada nas videoaulas.

2 Porta NAND

Descrição do Código: Tomando como o inicio dos estudos, fomos introduzidos a sintaxe e a estrutura que um código em vhdl precisa possuir para funcionar. Foi feita apenas uma demostração simples, onde duas entradas eram combinadas através de uma porta nand. Fomos introduzidos aos conceitos de entity, architecture, e na declração de váriaveis de entrada e saída dentro do comando port. Código em VHDL:

```
Library ieee;
Use ieee.std_logic_1164.all;

Entity porta_nand is
Port(
SW : IN STD_LOGIC_VECTOR(1 downto 0);
LEDR : OUT STD_LOGIC_VECTOR(0 to 0)
);

End porta_nand;

Architecture Behaviour of porta_nand is
begin

LEDR(0) <= SW(1) nand SW (0);
end Behaviour;
```

3 Bit Vector

Descrição do Código: Na introdução aos vetores de bit utilizados pelo autor dos vídeos, notamos a necessidade de adequar nosso código, e pelo auxilio do professor, trocamos a váriavel para std_logic_vector. Neste circuito são instanciadas 6 entradas, e duas a duas foram três conceitos lógicos importantes, uma porta and, uma porta or e uma xor. Código em VHDL:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity bit_vector_ports is
port(
SW : in STD_LOGIC_VECTOR (5 downto 0);
LEDR : out STD_LOGIC_VECTOR (0 to 2)
);
end entity;

architecture circuito of bit_vector_ports is
begin
LEDR(0) <= SW(5) and SW(4);
LEDR(1) <= SW(3) or SW(2);
LEDR(2) <= SW(1) xor SW(0);
end architecture;
```

4 Mux 16

Descrição do Código: Neste circuito, simulamos um mux de 16 entradas, onde as chaves seletoras escolhiam de 4 em 4 entradas para serem transportadas para a saída. Fomos inseridos ao conceito de when, else. Código em VHDL:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity mux is
port(
SW : in STD_LOGIC_VECTOR (17 downto 0);
LEDR : out STD_LOGIC_VECTOR (3 downto 0)
);
end entity;

architecture circuit of mux is
begin
LEDR <= SW(17 downto 14) when SW(1 downto 0) = "11"else
SW(13 downto 10) when SW(1 downto 0) = "10"else
SW(9 downto 6) when SW(1 downto 0) = "01"else
SW(5 downto 2);

end architecture;
```

5 Mux 4

Descrição do Código: Fomos introduzidos ao conceito de with select, na implementação de um mux de entradas, onde os dois bits seletores, escolhiam um dos bits de entrada para serem transportados para a saída. Código em VHDL:

```
    library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.Numeric_Std.ALL;

entity mux_4 is
port(
SW : in STD_LOGIC_VECTOR (5 downto 0);
LEDG : out STD_LOGIC_VECTOR (7 downto 7)
);
end entity;

architecture circuits of mux_4 is

signal seletor : INTEGER;

begin
with SW(1 downto 0) select LEDG(7) <= SW(5) when "00",
SW(4) when "01",
SW(3) when "10",
SW(2) when others;

end architecture;
```

6 Paridade Par

Descrição do Código: Neste código somos introduzidos ao conceito de generate que é uma espécie de laço de repetição na linguagem VHDL. No entando, difere-se porque todas as iterações são feitas paralelamente, o que não acontece nas linguagens de programação comuns. O circuito se baseia em exprimir portas XOR em todas as entradas, combinando as da seguinte maneira, a primeira e a segunda são um XOR, o resultado faz a operação com a terceira, o resultado dessa com a quarta e assim por diante. São utilizados também, sinais auxiliares que nos ajudaram a descrever tais comportamentos. Código em VHDL:

```
library IEEE;
use IEEE.std_logic_1164.all;

entity paridade_par is

port(
SW : in std_logic_vector (4 downto 0);
LEDG : out std_logic_vector (7 to 7)
);
end paridade_par;

architecture circuito of paridade_par is

signal aux : std_logic_vector(0 to 3);

begin

aux(0) <= SW(0) xor SW(1);
Gen : for i in 1 to 3 generate
aux(i) <= SW(i+1) xor aux (i-1);
end generate;
LEDG(7) <= aux(3);
end architecture;
```

7 Somador Completo 1 bit

Descrição do Código:
Este código exemplifica a simulação de um somador completo. Foi utilizado 3 entradas, duas para os bits a serem somados e uma pra entrada de carry-in. E duas saídas foram utilizadas, uma delas com a saída do somador, e outra que extraia a existência ou não de um carry-out. Foi utilizado um sinal intermediário, nomeado como temporário. Código em VHDL:

```
library ieee;
use ieee.std_logic_1164.all;

entity full_adder is

port(
SW : in std_logic_vector (0 to 2);
LEDR : out std_logic_vector (0 to 1)
);
end entity;

architecture circuito of full_adder is

signal temp : std_logic;

begin

temp <= SW(0) XOR SW(1);
LEDR(0) <= temp XOR SW(2);
LEDR(1) <= (SW(0) and SW(1)) OR (temp and SW(2));

end architecture;
```