CHAPTER 6

INTRODUÇÃO AO PROJETO EM FPGA

6.1 Objetivos

Compreender os conceitos e se familiarizar com as etapas de projeto de circuitos digitais em FPGA utilizando programação VHDL.

6.2 Projeto em FPGA

6.2.1 Introdução

Um FPGA (*Field Programmable Gate Array* ou, em português, Arranjo de Portas Programáveis por Campo) é um tipo de dispositivo de lógica programável, ou seja, um *hardware* que pode ser configurado através de programação para realizar funções lógicas especificadas pelo usuário. A principal vantagem da lógica programável é a possibilidade de alterar projetos com facilidade sem alterações físicas no *hardware* ou substituição de componentes, tornando-a uma excelente ferramenta para a prototipagem e teste de circuitos digitais.

6.2.2 Estrutura do FPGA

O FPGA é um chip que suporta a implementação de circuitos lógicos relativamente grandes. Consiste de um grande arranjo de células lógicas ou blocos lógicos configuráveis contidos em um único circuito integrado.

Os três elementos básicos do FPGA são o bloco lógico configurável, as interconexões programáveis e os blocos de entrada e saída (I/O – in/out) como ilustrado no Fig. 6.1. Os blocos lógicos formam uma matriz bidimensional, e as interconexões programáveis são organizadas como canais de roteamento horizontal e vertical entre as linhas e colunas dos blocos lógicos. Os canais de roteamento possuem chaves de interligação programáveis que permitem conectar os blocos lógicos de maneira conveniente, em função das necessidades de cada projeto.

Os blocos de I/O estão nas bordas da estrutura e proporcionam acesso individualmente selecionável de entrada, saída ou bidirecional ao mundo externo. A matriz de interconexões programáveis distribuídas provê as interconexões entre os blocos lógicos e as conexões para as entradas e saídas. FPGAs de grande capacidade podem ter dezenas de centenas de blocos lógicos, além de memória e outros recursos.

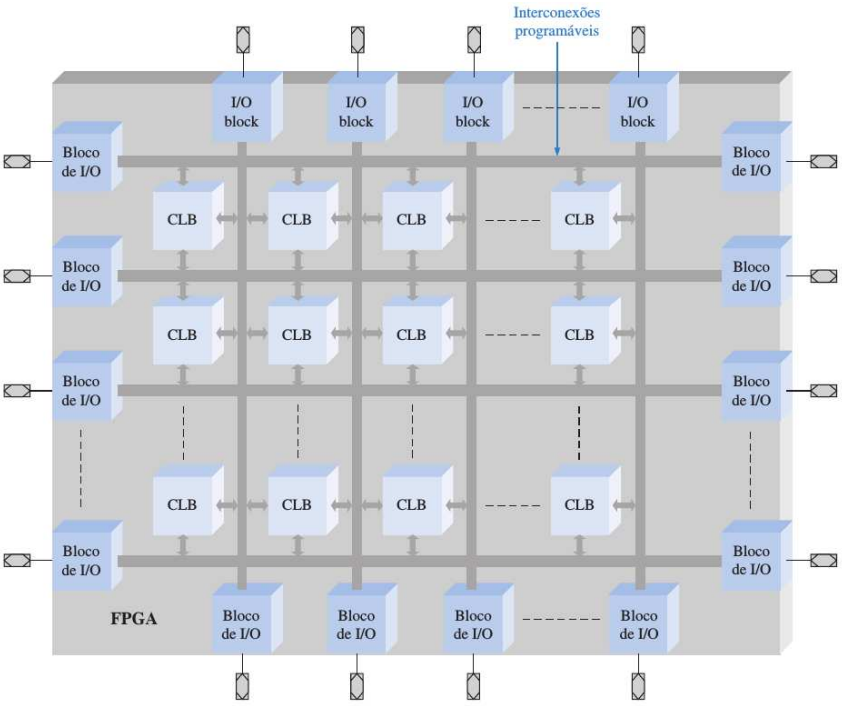

Figure 6.1: Estrutura básica de um FPGA. O bloco lógico configurável é o CLB.

6.2.3 Etapas de projeto em FPGA

Conforme discutido, um FPGA pode ser visto como um "quadro branco" no qual podemos implementar um determinado circuito ou sistema projetado fazendo uso de um certo processo de desenvolvimento. Esse processo necessita de um *software* instalado em um computador para implementar um projeto de circuito em um chip programável. O computador tem que ser interfaceado por meio de uma placa de desenvolvimento contendo o chip programável.

Vários passos, chamados de fluxo de projeto, estão envolvidos no processo de implementação de um projeto lógico em um FPGA:

Entrada do Projeto

Esse é o primeiro passo para implementação de um circuito em FPGA. O projeto do circuito ou sistema tem que ser inserido no *software* de desenvolvimento através de entrada textual, gráfica (desenho esquemático) ou descrição em diagrama de estado.

A inserção textual é realizada com uma linguagem de descrição de *hardware* (HDL - *hardware description language*), tal como VHDL, Verilog, AHDL ou ABEL. A inserção gráfica (esquemático) permite que funções lógicas pré-armazenadas sejam selecionadas a partir de um biblioteca, apresentadas na tela e então interconectadas para criar o projeto lógico. A inserção por diagrama de estados requer a especificação dos estados pelos quais um circuito lógico seqüencial passa e as condições que provocam a mudança de cada estado.

Uma vez que o projeto é inserido no *software*, ele é compilado. Um compilador é um programa que traduz o código-fonte em código-objeto num formato que pode ser testado logicamente.

Simulação Funcional

O projeto compilado é simulado por *software* para confirmar se os circuitos lógicos funcionam conforme esperado. A simulação irá verificar se as saídas adequadas são geradas para um conjunto de entradas

especificadas. Qualquer falha demonstrada pela simulação deve ser corrigida voltando-se à etapa de inserção do projeto e realizando as alterações apropriadas.

Síntese

A fase de síntese ocorre quando o projeto é traduzido em uma lista (*netlist*), a qual tem uma forma padronizada e é independente do dispositivo. A *netlist* é uma rede de portas, registradores e conexões que implementa as funções descritas pelo HDL.

Implementação

A implementação é onde a estrutura lógica descrita pela *netlist* é mapeada na estrutura real do dispositivo a ser programado. O processo de implementação é denominado *fitting* ou *place and route* e resulta em uma saída denominada sequência de bits (*bitstream*), a qual depende do dispositivo usado.

Simulação de Temporização

Esse passo ocorre após o projeto ser mapeado no dispositivo especificado. A simulação de temporização é basicamente usada para confirmar que não existem falhas no projeto ou problemas de temporização em função dos atrasos de propagação.

Download

Uma vez gerada uma seqüência de bits para um dispositivo programável específico, ele tem que ser transferido (operação de *download*) ao dispositivo para implementar o projeto no *hardware*. Alguns dispositivos são voláteis, o que significa que eles perdem o *bitstream* armazenado quando sofrem uma operação de *reset* ou quando a alimentação é desligada. Nesse caso, o *bitstream* tem que ser armazenado numa memória e recarregado no dispositivo após cada *reset*. Além disso, o conteúdo de um dispositivo ISP pode ser manipulado ou atualizado enquanto estiver operando no sistema.

6.3 Placa Digilent Basys3 Xilinx Artix-7

No Laboratório SS, estão disponíveis para uso na disciplina de Prática de Eletrônica Digital 1 um conjunto de placas da Digilent modelo Basys 3 (Fig. 6.2), que é uma plataforma de desenvolvimento de circuito completa para o chip FPGA Artix-7 da Xilinx. Estas placas serão usadas no restante do curso para programação do FPGA, em conjunto com o *software* de desenvolvimento Vivado Design Suite.

A Basys 3 uma coleção de portas e periféricos incluindo 16 chaves, 4 displays de 7 segmentos, 16 LEDs, 5 *pushbuttons*, saída VGA de 12 bits, flash serial, além de portas USB-JTAG para programação e comunicação da plataforma e etc.

A placa trabalha com o *software* Vivado Design Suite que inclui muitas ferramentas para facilitar o projeto do circuito a ser implementado no FPGA.

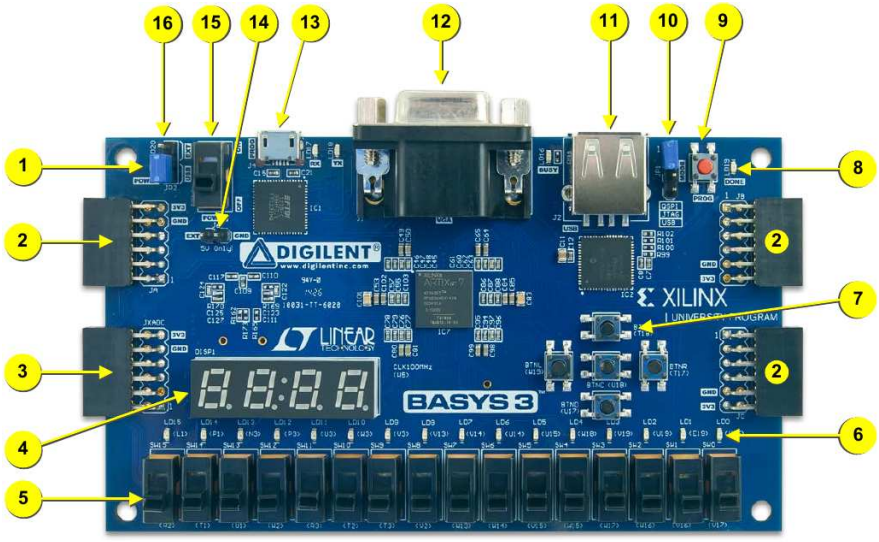

Figure 6.2: Placa Basys 3 da Digilent.

	Descrição do componente		*Descrição do componente*
1	LED da alimentação	**9**	Botão de reset da FPGA
2	Conectores Pmod	**10**	Jumper de modo de programação
3	Conector Pmod sinal analógico (XADC)	**11**	Conector host USB
4	Displays 7-segmentos (4)	**12**	Conector VGA
5	Chaves (16)	**13**	Porta partilhada UART/JTAG
6	LEDs (16)	**14**	Conector para alimentação externa
7	*Pushbuttons* (5)	**15**	Chave de alimentação
8	LED de programação completa	**16**	Jumper de seleção de alimentação

6.4 Pré-Relatório

6.4.1 Pesquisa bibliográfica

1) Leia o manual da placa Basys 3 (*Basys 3 FPGA Board Reference Manual*) e determine:

a) As formas de alimentação suportadas pela placa, os níveis de tensão e corrente envolvidos e como esta configuração é feita;

b) As formas de programação suportadas pela placa e como elas podem ser configuradas;

c) As formas de armazenamento do bitstream na placa (tipos de memória);

d) A quantidade e frequência dos clocks disponíveis;

e) A identificação (nome) dos pinos usados para as entradas da placa (chaves e *pushbuttons*) e para as saídas (LEDs e displays de 7-segmentos);

f) O tipo de display de 7-segmentos (anodo comum ou catodo comum) e os níveis lógicos dos sinais que devem ser aplicados nos pinos de anodo e catodo para que os segmentos se acendam (observe que esta placa usa transistores no pino comum).

6.4.2 Projetos e Simulações

Para a etapa de projetos, será necessário instalar o *software* do Vivado Design Suite (Vivado HL WebPACK Edition - versão 2016.2).

Você pode encontrar pacotes de instalação e instruções em: `https://www.xilinx.com/support/download.html`

6.4.2.1 Projeto e Simulação 1

Projete um sistema em que 7 chaves da placa ($SW0$ à $SW7$) serão usadas para ativar 7 LEDs ($LD0$ à $LD1$), tal que:

$LD0 = \overline{SW0}$
$LD1 = SW1 \,.\, \overline{SW2}$
$LD2 = LD1 + LD3$
$LD3 = SW2 \,.\, SW3$
$LD4 = SW4$
$LD5 = SW5$
$LD6 = SW6$
$LD7 = SW7$

Crie um projeto para implementar este sistema na placa Basys 3 utilizando o *software* Vivado. Você deverá realizar todas as etapas de projeto, deixando apenas a fase final de *download* para ser realizada em sala durante a parte prática deste roteiro.

Para auxiliá-lo na familiarização com o ambiente de desenvolvimento do Vivado, bem como as etapas para criação e configuração do projeto, use como referência o tutorial disponibilizado (*Vivado Tutorial*).

6.4.2.2 Projeto e Simulação 2

Adapte o código VHDL para multiplexação de dois display de 7-segmentos desenvolvido no Experimento 5. Assim como no experimento anterior, o sistema projetado terá duas entradas (A e B), cada uma representando um dígito BCD para a dezena e para a unidade, respectivamente. Uma terceira entrada, servirá como chave seletora para alternar entre os displays.

Com a chave em 0:

- o display das unidades é ligado,
- o valor de B é representado no display, e
- o display das dezenas é desligado.

Com a chave em 1:

- o display das dezenas é ligado,
- o valor de A é representado no display, e
- o display das unidades é desligado.

Produza um projeto completo no software Vivado para implementar o sistema especificado acima e realize todas as etapas de desenvolvimento deixando apenas a fase final de *download* para ser realizada em sala de aula.

Dica: algumas funcionalidades de VHDL serão bastante úteis para realizar este projeto. Uma delas é o uso de componentes para reaproveitamento de código e interconexão de blocos.

Por exemplo, o código VHDL a seguir representa o circuito ilustrado na Fig. 6.3:

Algorithm 6.1 Exemplo do uso de *component* no VHDL.

```
entity my_circuit is
   port (x,y:  in std_vetor;
         z,w:  out std_vetor);
end latch;

architecture structure of my_circuit is
   component my_component
      port (a,b:  in std_vetor;
            c:  out std_vetor);
   end component;
begin
   n1:  nor_gate port map (y,w,z);
   n2:  nor_gate port map (x,z,w);
end structure;
```

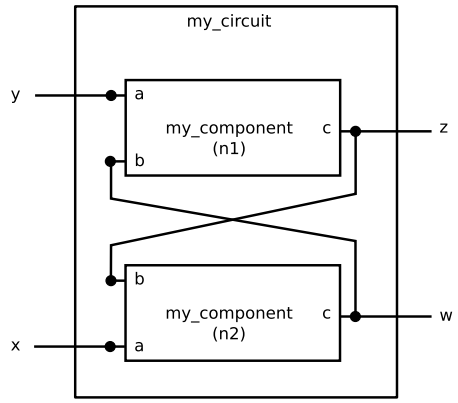

Figure 6.3: Circuito resultante do código apresentado no Algoritmo 6.1.

Outro conceito novo de VHDL que será explorado em sala durante este experimento é o uso de processos.

Até o momento, estivemos codificando os circuitos em VHDL de forma estrutural, em que componentes são interligados entre si e operam de forma concorrente (ao mesmo tempo). Para isso, utilizamos sentenças concorrentes para atribuição de um sinal através de equações booleanas, *with-select-when* ou *when-else*.

Um processo também se interliga com outros processos e elementos do circuito de forma concorrente. Entretanto, internamente, as sentenças de um processo são executadas de forma sequencial. Pesquise sobre o uso de *process* no VHDL e como ele pode ser usado para gerar um sinal de *clock* em seu projeto.