Arquitetura de Sistemas Digitais – Prof. Lucas Vinicius Hartmann
Exercícios SystemVerilog

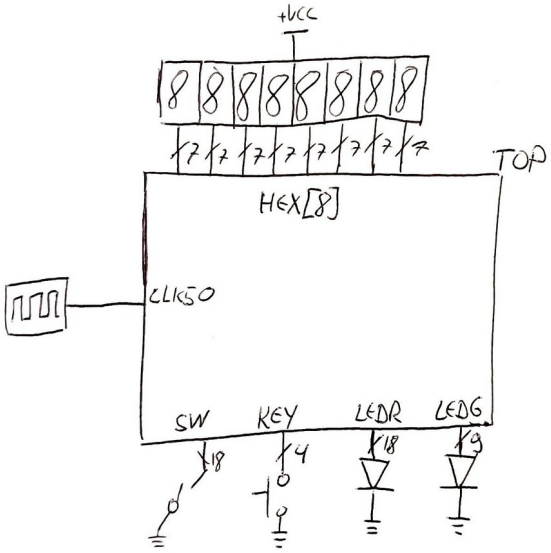
Aluno: _____ Matrícula:_____

MATERIAIS

- Computador com o programa Altera Quartus II 13.0sp1 instalado (disponível no LEAD).
- Placa de desenvolvimento Altera DE2 (disponível no LEAD).
- Manual do usuário para a placa de desenvolvimento (baixar).

PREPARAÇÃO

1. Iniciar o Quartus II versão 13.0sp1. Versões mais recentes não suportam a placa DE2 disponível no laboratório.
2. Criar um novo projeto pelo menu *File → New Project Wizard*.
   a) Em *working directory* colocar o nome de uma pasta nova, vazia, para armazenar o projeto.
      - Não usar espaços ou acentos no caminho.
      - Não colocar em "Desktop" ou "Área de Trabalho."
   b) No nome do projeto escolha algo representativo, sem espaços ou acentos. O último nome de pasta pode ser uma boa opção.
   c) Em *top-level design entity* insira "`top`" para ajudar na organização.
   d) Em *Family & Device Settings* escolher *Family = Cyclone II, Pin count = 672, Speed grade = C6*, e selecionar o chip *EP2C35F672C6*.
3. Criar um arquivo de código fonte SystemVerilog:
   a) Selecione no menu *File → New → Design Files → SystemVerilog HDL File*.
   b) Salvar utilizando "`top.sv`" como nome de arquivo.
   c) Definir as portas do módulo top considerando os nomes dos dispositivos disponíveis na placa, como mostrado na figura abaixo:
      - Porta `CLK50`: Entrada, um bit.
      - Porta `SW`: Entrada, um número de 18 bits.
      - Porta `KEY`: Entrada, um número de 4 bits.
      - Porta `LEDR`: Saída, um número de 18 bits.
      - Porta `LEDG`: Saída, um número de 9 bits.
      - Porta `HEX`: Saída, 8 números de 7 bits cada.



4. Compilar o projeto no menu *Processing* → *Start Compilation*.
5. Mapear os pinos pelo menu *Assignments* → *Pin Planner*, de acordo com a especificação do manual da placa Altera DE2.
6. Criar um módulo e arquivo fonte SystemVerilog para cada exercício abaixo, utilizando o módulo `top` já construído para conectar os módulos dos exercícios aos componentes da placa.

EXERCÍCIOS

Programar, simular e testar cada um dos seguintes experimentos utilizando o Quartus II e a placa de desenvolvimento DE2.

1. Utilizando apenas instanciação, construa um módulo `Maioria` considerando:
   a) O módulo deve ter 4 portas de entrada A,B,C,X, e uma saída Y.
   b) Quando a maioria dos sinais A,B e C forem verdadeiros, Y deve ter o mesmo valor de X, ou zero em caso contrário.
   c) Construa a tabela verdade, esboce o circuito lógico, dê nomes aos sinais internos, e crie o módulo Maioria em SystemVerilog utilizando apenas instanciação de portas primitivas.
   d) Instancie `Maioria` em `top`, conectando às chaves e LEDs como necessário.
   e) Compile, grave e teste o teste o circuito.
2. Construir um decodificador de hexadecimal para 7 segmentos com sinal de habilitação.
   a) O módulo deve ser chamado `d7s`, possuir uma entrada X com 4 bits, uma entrada EN com 1 bit, e uma saída Y com 7 bits.

b) Quando EN estiver ativo o número da entrada deve ser mostrado no display. Quando inativo o display deve apagar.

c) Para testar, conecte via `top` as chaves `SW[3:0]` como X, a chave `SW[17]` como EN, e o o display `HEX[0]` como Y.

3. Construir um codificador de 16:4 com gate.

a) O módulo deve ser chamado `cod16_4`, possuir uma entrada X com 16 bits, uma saída Y com 4 bits e uma saída GATE com um bit.

b) Quando houver uma ou mais bits ativos em X, a saída Y deve fornecer o número de bit ativo mais significativo, e GATE deve estar em verdadeiro.

c) Quando nenhum bit estiver ativo em X o valor de Y não importa, mas GATE deve ser falso.

d) Para testar conecte `SW[15:0]` em X do codificador, `LEDR[3:0]` em Y, e utilize o decodificador de display do exercício anterior para mostrar Y em hexadecimal.

4. Construir um demux 4:16.

a) O módulo deve ser chamado `demux4_16`, possuir entrada X com um bit, uma entrada seletora S com 4 bits, e saída Y com 16bit.

b) O valor da entrada X deve aparecer na saída Y selecionada, e as demais saídas devem permanecer em zero.

c) Conectar à placa e testar.

5. Construir um contador auto-iniciante que implemente a sequência 1-2-7-3-6-1.

a) O módulo deve ser chamado `cnt_127361`, possuir entrada CLK com um bit, e saída Y com 3 bits.

b) Caso inicie fora de sequência o contador deve retornar para a posição 1.

c) Para testar utilize `KEY[0]` como clock, e o decodificador do primeiro exercício para mostrar o número em hexadecimal.

6. Construir um divisor de frequência de 50MHz para 1Hz utilizando um contador.

a) O módulo deve ser chamado `div50M`, e ter uma entrada `CLK_IN` e uma saída `CLK_OUT`, ambas com um bit.

b) A saída `CLK_OUT` deve permanecer ativa por apenas 1 a cada 50M ciclos de `CLK_IN`.

c) Para testar utilizar o sinal de 50MHz da placa como fonte de clock, e alternar o estado do `LEDR[0]` sempre que `CLK_OUT` for verdadeiro.

7. Fazer um contador de 8 bits que:

a) Mostre a saída em hexadecimal em dois displays de 7 segmentos, e em binário em `LEDR[7:0]`.

b) Incremente uma unidade quando o usuário pressionar KEY[0] e soltar em menos de um segundo.

c) Retorne a zero quando o usuário segurar KEY[0] por mais de um segundo.