Departamento de Ciência da Computação

GCC113 – Circuitos Digitais

Projetos Hierárquicos e Verilog

1. Objetivos

- Conhecer novas características do Verilog;
- Usar o display de sete segmentos do kit do laboratório;
- Criar projetos hierárquicos no Quartus Prime contendo vários arquivos em Verilog.

2. Qual será o projeto desta prática?

Nesta prática, vamos desenvolver um circuito que utilizará as 10 chaves presentes no kit do laboratório (SW) e dois displays de sete segmentos (HEX0 e HEX1). O circuito deverá receber 10 bits de entrada por meio das chaves e verificar o número de uns presentes nos bits e exibir o valor em decimal no display de sete segmentos. Por exemplo, se a entrada binária nas dez chaves SW for *0000101100*, o display deverá exibir o número *3*, pois temos 3 bits iguais a 1. Se todos os bits forem iguais a 1, os displays devem exibir o número 10 na base decimal. A Figura 1 apresenta o diagrama de bloco desse circuito.

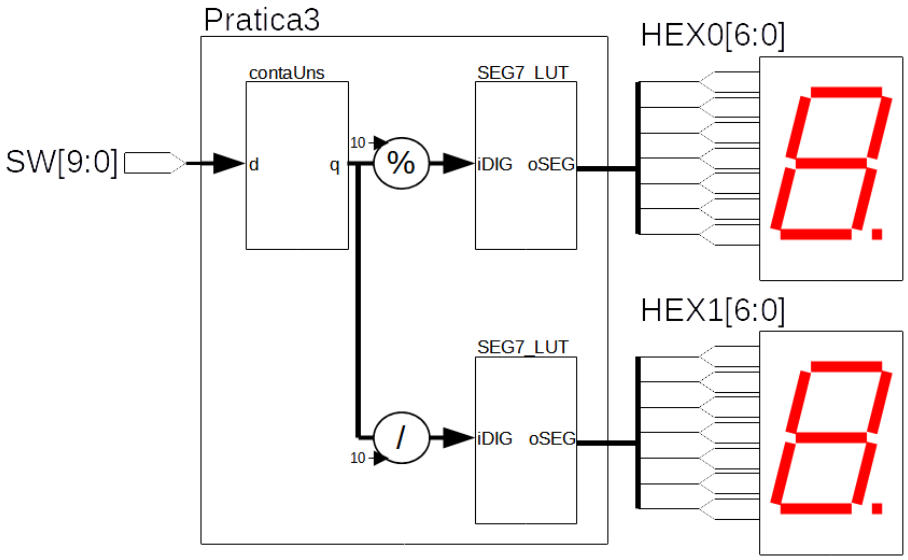

Figura 1: Diagrama de blocos do projeto.

3. Como o projeto deverá ser feito?

Para implementar este projeto, será usada a ideia de projetos hierárquicos. Isso ocorre quando temos diversos arquivos no projeto em que cada arquivo implementa parte do circuito (*modules*). E um dos arquivos é chamado de principal (ou *top-level*). Nesta prática, teremos três arquivos diferentes em Verilog. Haverá dois arquivos já prontos: *contaUns.v*, responsável por contar o número de chaves valendo 1, e *SEG7_LUT.v*, responsável por exibir o valor em decimal nos displays de sete segmentos. E um terceiro arquivo, o arquivo principal, será implementado de acordo com as instruções neste documento. O terceiro arquivo é o arquivo que irá usar os módulos implementados nos outros dois arquivos usando a descrição estrutural.

4. Qual o passo a passo para fazer o projeto?

As seguintes etapas devem ser realizadas para desenvolver o projeto:

- Crie uma nova pasta chamada Pratica3, na Área de Trabalho;
- Baixe o código SEG7_LUT.v em: https://github.com/brabreus/GCC260-UFLA/blob/main/Pratica3/SEG7_LUT.v e salve na pasta Pratica3;
- Baixe o código contaUns.v em: https://github.com/brabreus/GCC260-UFLA/blob/main/Pratica3/contaUns.v e salve na pasta Pratica3;
- Crie o projeto Pratica3 no Quartus Prime, lembrando de alterar o diretório para criar o projeto na pasta Pratica3, **adicionar no projeto os 2 arquivos já baixados** e definir a placa (board) como da família MAX10, kit DE10-Lite **deixando marcada a opção "Create top-level design file"**;
- Entender os códigos baixados;
- Implementar, no arquivo já criado **DE10_LITE_Golden_Top.v**, o circuito da Figura 1 usando a descrição estrutural do verilog;
- Compilar o projeto *Processing->Start Compilation*;
- Programar o kit do laboratório *Tools->Programmer*;
- Verificar seu funcionamento na placa;
- Apresentar ao Professor e enviar a pasta do projeto compactada para o Campus Virtual.

/////// SEG7_LUT.v

```
module SEG7_LUT (
        input   wire [3:0]      iDIG,
        output  reg [6:0]       oSEG
);

always @(iDIG)
begin
                case(iDIG)
                4'h0: oSEG = 7'b1000000;
                4'h1: oSEG = 7'b1111001;        // ---a----
                4'h2: oSEG = 7'b0100100;        // |       |
                4'h3: oSEG = 7'b0110000;        // f       b
                4'h4: oSEG = 7'b0011001;        // |       |
                4'h5: oSEG = 7'b0010010;        // ---g----
                4'h6: oSEG = 7'b0000010;        // |       |
                4'h7: oSEG = 7'b1111000;        // e       c
                4'h8: oSEG = 7'b0000000;        // |       |
                4'h9: oSEG = 7'b0011000;        // ---d----
                4'ha: oSEG = 7'b0001000;
                4'hb: oSEG = 7'b0000011;
                4'hc: oSEG = 7'b1000110;
                4'hd: oSEG = 7'b0100001;
                4'he: oSEG = 7'b0000110;
                4'hf: oSEG = 7'b0001110;
                endcase
end
endmodule
```

////// contaUns.v

```
module contaUns (
    input wire [9:0] d,
    output wire [3:0] q
);



    integer num_bits;

    always @ (d) begin
        integer i;

        num_bits = 0;
        for (i = 0; i < 10; i = i + 1) begin
            if (d[i] == 1)
                num_bits = num_bits + 1;
        end
    end

    assign q = num_bits;

endmodule
```