



Departamento de Ciência da Computação

Prof. Bruno de Abreu Silva

GCC260 – Laboratório de Circuitos Digitais

Circuito somador em Verilog

1. Objetivos

- Utilizar parâmetros em Verilog;
- Implementar um circuito somador em Verilog;
- Implementar um arquivo *Top-Level* que utiliza as *switches* como entrada e os displays de sete segmentos e LEDs como saída.

2. Projeto para implementação

Implemente um circuito usando as *switches* de 0 a 7 para receber dois números inteiros, A e B, de quatro bits como entrada e que exiba o resultado da soma deles com dois dígitos decimais nos displays HEX0 e HEX1. Além disso, o circuito deve possuir as seguintes características:

- Definir o A como sendo as *switches* SW[9:6] e B como SW[5:2];
- Exibir o valor de A em decimal nos displays HEX5 e HEX4;
- Exibir o valor de B em decimal nos displays HEX3 e HEX2;
- Exibir o sinal de *cout* (*carry out*) do somador no LEDR[0] para overflow de soma de inteiros sem sinal.

A Figura 1 apresenta o diagrama esquemático do circuito a ser implementado no arquivo top-level.

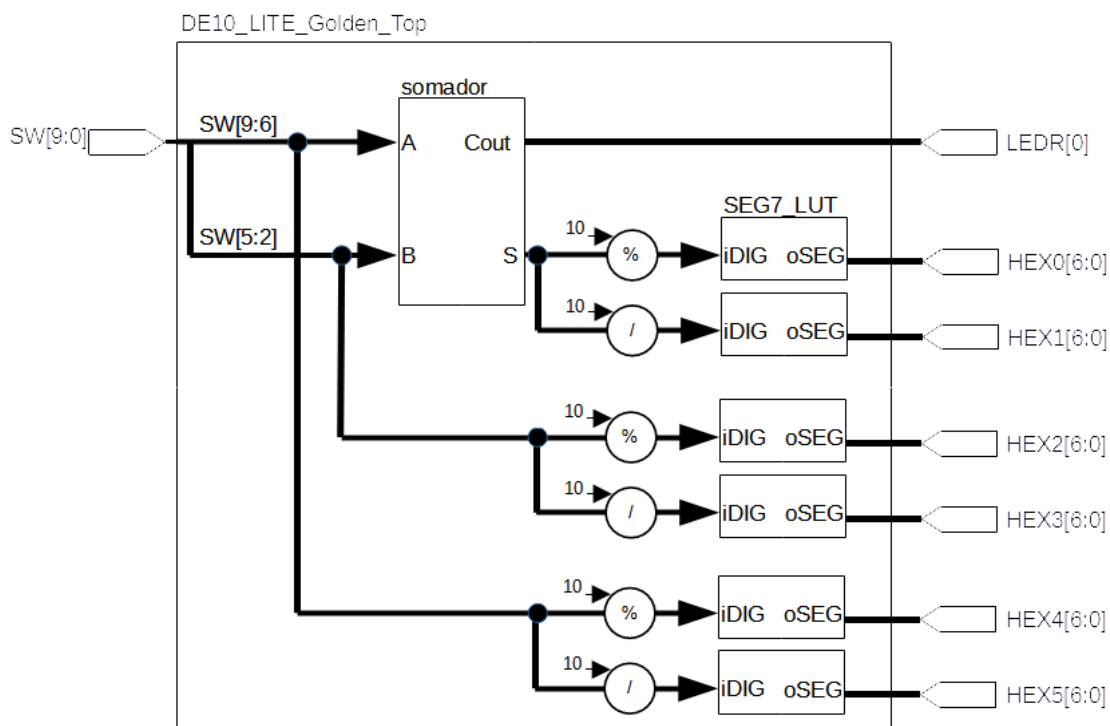


Figura 1: Diagrama esquemático do arquivo top-level.

2.1. Passo a passo

- Crie uma pasta nova, chamada Pratica5 na Área de Trabalho;
- Baixe o arquivo SEG7_LUT.v em https://github.com/brabreus/GCC260-UFLA/blob/main/Pratica5/SEG7_LUT.v e salve na pasta Pratica5;
- Baixe o arquivo somador.v em <https://github.com/brabreus/GCC260-UFLA/blob/main/Pratica5/somador.v> e salve na pasta Pratica5;
- No Quartus Prime, crie o projeto Pratica5, altere o diretório do projeto para a pasta Pratica5, na Área de Trabalho, adicione os dois arquivos baixados ao projeto, defina o *board* como família MAX10 e kit DE10-LITE e deixe marcada a caixinha “Create top-level design file”;
- Entenda o código do somador.v e tire suas dúvidas durante a explicação do Professor;
- Releia o início da seção 2, reveja a Figura 1 deste documento e complete o código DE10_LITE_Golden_Top.v para atender às características pedidas para o circuito;
- Compile o projeto (*Processing->Start Compilation*) e corrija os eventuais erros;
- Conecte a FPGA ao computador;
- Programa a FPGA em *Tools->Programmer*;
- Verifique o funcionamento do circuito na placa e o que se pode dizer sobre os resultados das somas? Estão todos corretos? Se acontecer algum problema, por que acontece?

3. Códigos

A seguir são apresentados os códigos dos arquivos somador.v e SEG7_LUT.v

3.1 somador.v

```
module somador
    #(
        parameter N = 32
    )
    (
        input wire [N-1:0] A, B,
        output reg [N-1:0] S,
        output reg cout
    );

    always@*
    begin
        {cout,S} = A + B;
    end

endmodule
```

3.2 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
```