UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

Emanuel de Oliveira
Gabriel Bazon
Giovanna Velasco
Guilherme Schmidt
João Pedro Gomes
João Victor Alves
Luana Hartmann

Projeto: Terceira Parte

São Carlos

2023

Emanuel de Oliveira
Gabriel Bazon
Giovanna Velasco
Guilherme Schmidt
João Pedro Gomes
João Victor Alves
Luana Hartmann

Projeto: Terceira Parte

Trabalho apresentado à disciplina de Sistemas Digitais da Escola de Engenharia de São Carlos, Universidade de São Paulo, como parte dos requisitos para a aprovação na disciplina.

Área de concentração: Engenharia de Computação

Orientador: Prof. Dr. Maximiliam Luppe

São Carlos
2023

Sumário

1 Introdução

O presente documento descreve o desenvolvimento e a implementação em Verilog de um módulo contador digital BCD de 000 a 999, com saída para displays de 7 segmentos.

Em suma, o módulo em questão deve receber entradas de clock, reset e enable, além de uma adicional para habilitar a atualização do display. Como saída, este fornecerá 21 saídas para o controle de 3 displays de 7 segmentos e uma saída adicional para indicar quando chegou à contagem máxima.

Tal módulo foi construído através da união de módulos menores previamente desenvolvidos no projeto, sendo estes 3 contadores de 0 a 9, 3 Flip-Flops Tipo D de 4 bits com entrada enable, e 3 decodificadores BCD para display de 7 segmentos.

Portanto, passemos à descrição do módulo.

2 Contador digital BCD de 000 a 999

O contador em questão produz valores BCD de 0 a 999, os quais armazena conforme determinado pelo sinal *ld* em um conjunto de 3 Flip-Flops tipo D de 4 bits, consequentemente seguindo para três decodificadores BCD para display de 7 segmentos que permitem a visualizam dos dados.

Desse modo, o contador possui 4 entradas:

1. *clk*: o sinal de clock, que determina a taxa de crescimento do contador;
2. *enb*: habilita ou desabilita a contagem;
3. rst_s: reseta o valor do contador para 0;
4. *ld*: atua como sinal de enable dos Flip-Flops de armazenamento.

Além de 2 saídas:

1. *display*: 21 saídas correspondentes aos 3 displays de 7 segmentos necessários, em ordem crescente;
2. cnt_max: indica que o contador está em seu valor máximo (999).

Portanto, descrita sua função, e suas entradas e saídas, partimos à implementação.

3 Implementação do Contador em Verilog

Para a implementação do circuito, será utilizada a linguagem de descrição de hardware Verilog. Destaca-se que o código pode ser também encontrado no repositório do projeto.

Portanto, a implementação consistiu primeiramente na importação dos módulos previamente construídos para o Contador de décadas, o Flip-Flop tipo D, e o decodificador BCD para display de 7 segmentos.

A seguir, os contadores foram organizados de forma que as saídas cnt_max de cada um tornem-se entradas do sinal *enb* no contador seguinte, garantindo que cada um estivesse uma ordem de grandeza acima do anterior. Ademais, a saída *cnt_max* do último contador de 0 a 9 corresponde à do módulo de 0 a 999, e a entrada *enb* geral corresponde à do primeiro contador. Por fim, o Clock atua normalmente nos 3 contadores de 0 a 9.

Consecutivamente a saída *qx* de 4 bits correspondente a cada contador entra em um Flip-Flop tipo D também de 4 bits, o qual altera seu valor apenas quando sua entrada Clock Enable *ld* é igual a 1, armazenando valores desejados do contador.

Por fim, a saída *qs* de cada um dos Flip-Flops procede ao decodificador BCD para display de 7 segmentos (com ânodo comum), permitindo então visualizar em um diplay o valor de 3 casas decimais produzidos pelo contador e armazenados pelo Flip-Flop.

Abaixo segue o código criado e circuito esquemático correspondente ao módulo descrito (Figura 1).

```
// Decodificador BCD para display 7 segmentos, produzido na parte 1
module BCDto7seg #(parameter common_cathode=1)(output a, b, c, d, e, f,
↪  g, input [3:0] BCD);
  reg [6:0] seg;

  generate
    if (common_cathode == 1)
      always @ (BCD)
        begin
          case(BCD)
            0: seg = 7'b1111110;
            1: seg = 7'b0110000;
            2: seg = 7'b1101101;
            3: seg = 7'b1111001;
```

```
            4: seg = 7'b0110011;
            5: seg = 7'b1011011;
            6: seg = 7'b1011111;
            7: seg = 7'b1110000;
            8: seg = 7'b1111111;
            9: seg = 7'b1111011;
            default: seg=7'b1001111;
          endcase
        end
        else
      always @ (BCD)
        begin
          case(BCD)
            0: seg = 7'b0000001;
            1: seg = 7'b1001111;
            2: seg = 7'b0010010;
            3: seg = 7'b0000110;
            4: seg = 7'b1001100;
            5: seg = 7'b0100100;
            6: seg = 7'b0100000;
            7: seg = 7'b0001111;
            8: seg = 7'b0000000;
            9: seg = 7'b0000100;
            default: seg=7'b0110000;
          endcase
        end
  endgenerate
  assign {a,b,c,d,e,f,g} = seg;
endmodule

// Flip-Flop tipo D com sinal 'enb'
module dff_4bit(output reg [3:0] q, input [3:0] d, input clk, enb);
  always @ (posedge clk)
    if (enb)
      q = d;
endmodule
```

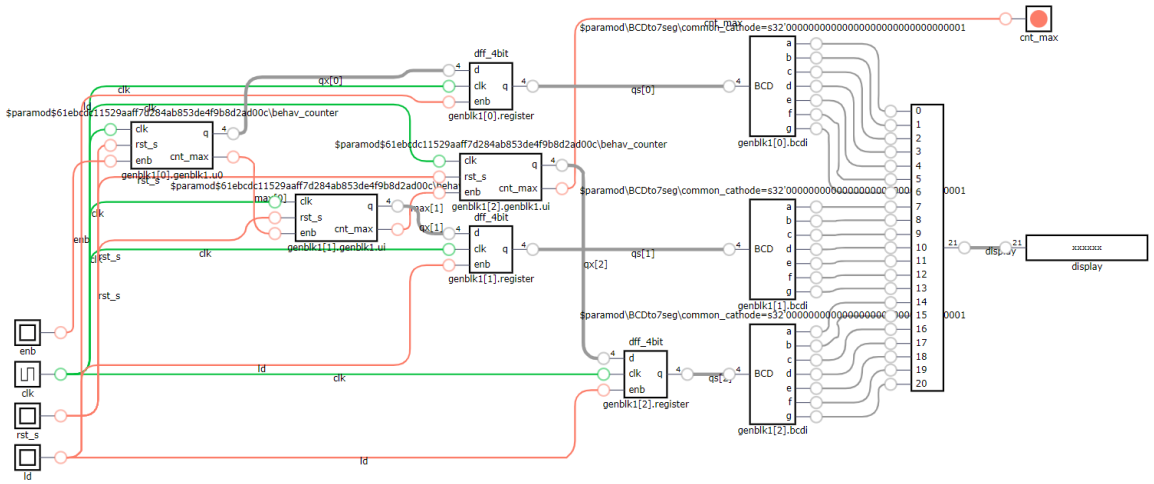
```
// Contador parametrizavel comportamental, produzido na parte 2
module behav_counter #(parameter width=4, max_count=10)(input clk, rst_s,
↪  enb, output reg [width-1:0] q, output cnt_max);

  always @ (posedge clk, posedge rst_s)
    begin
      cnt_max<=0;
      if (rst_s)
        q <= 0;
      else if (enb)
        begin
          if (q < max_count-2)
            q<=q+1;
          else if (q == max_count-2)
            begin
              q<=q+1;
              cnt_max<=1;
            end
          else
            q<=0;
        end
     end

endmodule

// Contador digital BCD de 000 a 999, com 21 fios de output para 3
↪  displays de 7 segmentos
module counter_999 #(parameter module_width=4, module_count=10)
(output [20:0] display, output cnt_max, input clk, enb, rst_s, ld);

  reg [3:0] qx[3];
  reg [3:0] qs[3];
  wire max[3];

  // Gera 3 contadores de 0 a 9 em sequência
  genvar i;
  generate
```

```
    for (i=0; i<3; i=i+1)
      begin
        if (i == 0)
          // Primeiro contador
          behav_counter #(.width(module_width), .max_count(module_count))
          u0 (.q(qx[0]), .cnt_max(max[0]), .clk(clk), .enb(enb),
↪   .rst_s(rst_s));
        else
          // Contadores seguintes, com a saida 'cnt_max' do anterior
↪   entrando em 'enb' no seguinte
          behav_counter #(.width(module_width), .max_count(module_count))
          ui (.q(qx[i]), .cnt_max(max[i]), .clk(clk), .enb(max[i-1]),
↪   .rst_s(rst_s));

        dff_4bit register (.q(qs[i]), .clk(clk), .d(qx[i]), .enb(ld));

        // Decodificador BCD para 7 segmentos recebendo 'q' do contador
↪   atual e retornando os 7 fios de display
        BCDto7seg #(.common_cathode(1))
        bcdi (.a(display[i*7]), .b(display[1+i*7]), .c(display[2+i*7]),
↪   .d(display[3+i*7]),
         .e(display[4+i*7]), .f(display[5+i*7]), .g(display[6+i*7]),
↪   .BCD(qs[i]));
      end
  endgenerate

  assign cnt_max = max[2];


endmodule
```

4 Conclusão

A partir do exposto, pode-se concluir que o circuito foi implementado de modo adequado, uma vez que as representações a partir da linguagem Verilog fornecem resultados corretos para os testes realizados.

Figura 1 – Circuito esquemático do contador digital BCD de 0 a 999



No mais, ressaltam-se certas dificuldades na compreensão da finalidade de certos aspectos do contador, como da entrada *ld*, a qual só foram elucidadas na parte 4, onde sua função dentro Conversor Analógico-Digital de Rampa Dupla tornou clara sua utilidade.

5 Bibliografia

V. P. Nelson. *Digital logic circuit analysis and design*. Prentice Hall, 1995