ARTHUR BRENO DOS REIS PAULA - 12547382
CARLOS NERY RIBEIRO - 12547698
FELIPE CECATO - 12547785
FERNANDO CLARINDO CRISTÓVÃO - 12547573
GABRIEL RIBEIRO RODRIGUES DESSOTTI - 12547228
PEDRO MANICARDI SOARES - 12547621

Contador BCD em Verilog

Trabalho de Conclusão de Curso apresentado à
Escola de Engenharia de São Carlos, da
Universidade de São Paulo

Curso de Engenharia de Computação

Orientador: Maximiliam Luppe

São Carlos - SP
2022

Sumário

1. Resumo

O estudo de *Verilog* é de suma importância para as engenharias, uma vez que ele se comporta como uma linguagem que descreve os sistemas digitais, ajudando sua implementação por meio de softwares, como o DigitalJS, o qual foi utilizado neste relatório. Desse modo, essa linguagem de descrição de hardware é usada para moldar os circuitos eletrônicos. O projeto descrito neste trabalho usou diretamente dessa ferramenta para que fosse implementado um contador de três dígitos, que ia de 0 a 999, usando três displays de 7 segmentos. Após sua implementação em *Verilog*, foi desenhado o circuito RTL equivalente pelo software DigitalJS, o qual foi prontamente inserido no presente relatório. Além disso, foram organizados nas posições corretas os LEDs referentes aos três displays para que ficassem no formato onde se pode visualizar os números emitidos. Por fim, foram simulados a partir do software *8bitworkshop* os sinais de saída desse circuito.

2. Introdução

No código em *Verilog*, a primeira coisa que se fez foi colocar os dois *inputs* do circuito: o sinal de *clock* e o *reset*. A partir disso, fez-se três contador de 4 bits iniciando em zero, em que o segundo só avançava o número quando o primeiro atingisse ‘a’ e o terceiro só avançava o número quando o segundo atingisse ‘a’. Após isso, foi inserido um registrador para guardar os valores expostos pelo contador e, em seguida, implementou-se três decodificadores BCD para Display de 7 segmentos, os quais recebiam as entradas a partir do registrador e as convertia na representação por 7 segmentos, que era o objetivo inicial. Por fim, tinha-se o botão de *reset* que servia para reiniciar a contagem logo que o usuário assim desejasse.

3. Código

Assim, o código em Verilog do funcionamento, copiado do arquivo .v desenvolvido pelos autores é o seguinte:

```
// Trabalho de Sistemas Digitais - Contador BCD em Verilog
// Autores: Arthur Breno dos Reis Paula
//                    Carlos Nery Ribeiro
//                    Felipe Cecato
//                    Fernando Clarindo Cristovao
//                    Gabriel Ribeiro Rodrigues Dessotti
//                    Pedro Manicardi Soares
// SEL0628 - Sistemas Digitais - USP

module contador4b(clk, reset, q, qe, qee, qs, qes, qees, a1, b1, c1, d1, e1, f1,
g1, a2, b2, c2, d2, e2, f2, g2, a3, b3, c3, d3, e3, f3, g3);
// Entradas do circuito
  input clk;
  input reset;

// Saidas do circuito
// As variaveis q e qs referem-se ao primeiro digito, qe e qes ao segundo e qee e
qees ao terceiro, contando da direita para esquerda
  output reg[3:0] q;
  output reg[3:0] qe;
  output reg[3:0] qee;
  output reg[3:0] qs;
  output reg[3:0] qes;
  output reg[3:0] qees;
  output a1, b1, c1, d1, e1, f1, g1;
  output a2, b2, c2, d2, e2, f2, g2;
  output a3, b3, c3, d3, e3, f3, g3;

// As variaveis sao todas inicializadas com o valor 0 em binario de 4 bits
  initial begin
    q = 4'b0000;
    qe = 4'b0000;
    qee = 4'b0000;
  end

// Bloco de funcionamento do contador de 4 bits
  always @(posedge clk) begin

    // Botao de reset que zera todos os digitos
    if (reset) begin
       q = 4'b0000;
       qe = 4'b0000;
       qee = 4'b0000;
    end

    // Inicializa a contagem do primeiro digito
      q = q + 4'b0001;

    // Quando chega em um numero da forma xx9, o primeiro digito zera e soma 1 no
segundo digito
```

```
      if (q == 4'b1010) begin
        q = 4'b0000;
        qe = qe + 4'b0001;
      end

    // Quando chega em um numero da forma x99, o segundo digito zera e soma 1 no
terceiro digito
      if (qe == 4'b1010) begin
        qe = 4'b0000;
        qee = qee + 4'b0001;
      end

    // Quando chega no 999, o terceiro digito zera
      if (qee == 4'b1010) begin
        qee = 4'b0000;
      end
  end

  // Bloco de funcionamento do registrador de 4 bits
  // O registrador armazena os valores das saidas dos 3 digitos do contador
  always @(posedge clk) begin
    qs <= q;
    qes <= qe;
    qees <= qee;
  end


  // Bloco de funcionamento do decodificador BCD de 7 segmentos
  // O decodificar recebe as variaveis armazenadas no registrador e as trasnforma
no formato de 7 segmentos para cada um do tres digitos

  assign {a1, b1, c1, d1, e1, f1, g1} = ( qs == 4'b0000 ) ? 7'b1111110 :
//representacao do 0
                                        ( qs == 4'b0001 ) ? 7'b0110000 :
//representacao do 1
                                        ( qs == 4'b0010 ) ? 7'b1101101 :
//representacao do 2
                                        ( qs == 4'b0011 ) ? 7'b1111001 :
//representacao do 3
                                        ( qs == 4'b0100 ) ? 7'b0110011 :
//representacao do 4
                                        ( qs == 4'b0101 ) ? 7'b1011011 :
//representacao do 5
                                        ( qs == 4'b0110 ) ? 7'b1011111 :
//representacao do 6
                                        ( qs == 4'b0111 ) ? 7'b1110000 :
//representacao do 7
                                        ( qs == 4'b1000 ) ? 7'b1111111 :
//representacao do 8
                                        ( qs == 4'b1001 ) ? 7'b1111011 :
//representacao do 9

              7'b1001111 ; // em caso de algum erro no sistema

  // As transformacoes sao analogas para os demais digitos

    assign {a2, b2, c2, d2, e2, f2, g2} = ( qes == 4'b0000 ) ? 7'b1111110 :
```

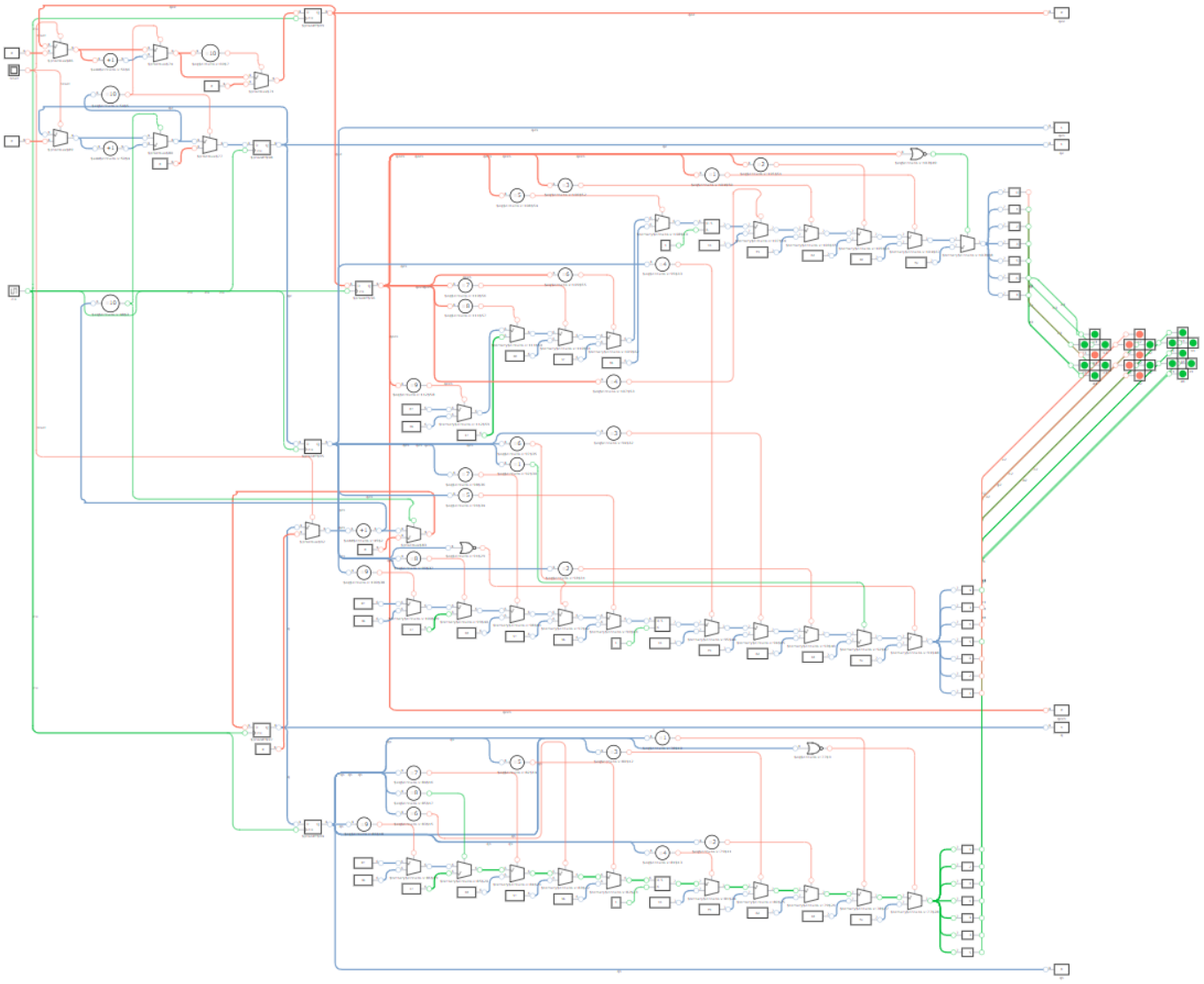
```
                                        ( qes == 4'b0001 ) ? 7'b0110000 :
                                        ( qes == 4'b0010 ) ? 7'b1101101 :
                                        ( qes == 4'b0011 ) ? 7'b1111001 :
                                        ( qes == 4'b0100 ) ? 7'b0110011 :
                                        ( qes == 4'b0101 ) ? 7'b1011011 :
                                        ( qes == 4'b0110 ) ? 7'b1011111 :
                                        ( qes == 4'b0111 ) ? 7'b1110000 :
                                        ( qes == 4'b1000 ) ? 7'b1111111 :
                                        ( qes == 4'b1001 ) ? 7'b1111011 :

                7'b1001111 ;

    assign {a3, b3, c3, d3, e3, f3, g3} = ( qees == 4'b0000 ) ? 7'b1111110 :
                                        ( qees == 4'b0001 ) ? 7'b0110000 :
                                        ( qees == 4'b0010 ) ? 7'b1101101 :
                                        ( qees == 4'b0011 ) ? 7'b1111001 :
                                        ( qees == 4'b0100 ) ? 7'b0110011 :
                                        ( qees == 4'b0101 ) ? 7'b1011011 :
                                        ( qees == 4'b0110 ) ? 7'b1011111 :
                                        ( qees == 4'b0111 ) ? 7'b1110000 :
                                        ( qees == 4'b1000 ) ? 7'b1111111 :
                                        ( qees == 4'b1001 ) ? 7'b1111011 :

                7'b1001111 ;
endmodule
```

O código pode ser baixado no seguinte link utilizando o email da USP: https://drive.google.com/file/d/1oowYPhYu8W3HlBd2uHVLCfJzDnjo1b-L/view?usp=sharing

4. Circuito RTL

Com base no código descrito anteriormente é possível gerar o circuito RTL correspondente ao projeto. Esse circuito é ilustrado na Figura 1:

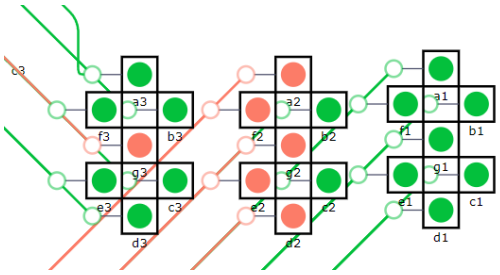
Figura 1 - Circuito RTL do contador



Fonte: Compilado pelos autores

A elaboração do circuito foi feita no software *DigitalJS Online* e ajustada para que os LEDs do decodificador BCD de 7 segmentos ficassem num formato visualmente melhor para identificar os números. O display pode ser visualizado de forma ampliada na Figura 2:

Figura 2 -Visor do circuito

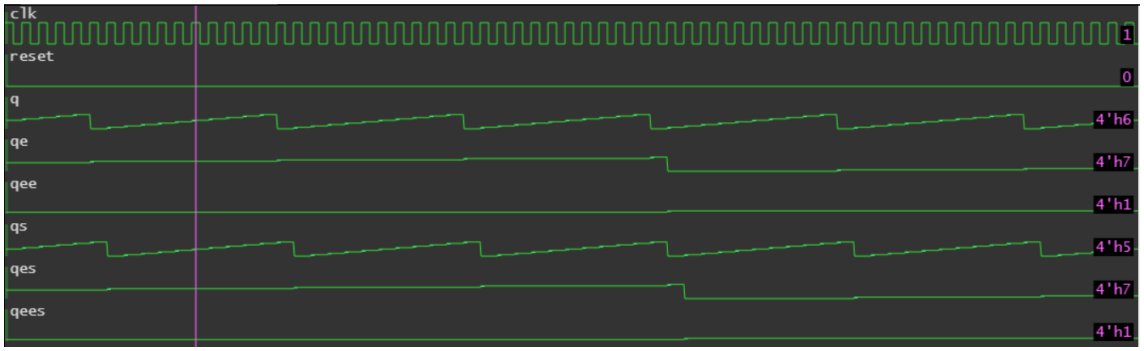


Fonte: Compilado pelos autores

5. Resultado da Simulação

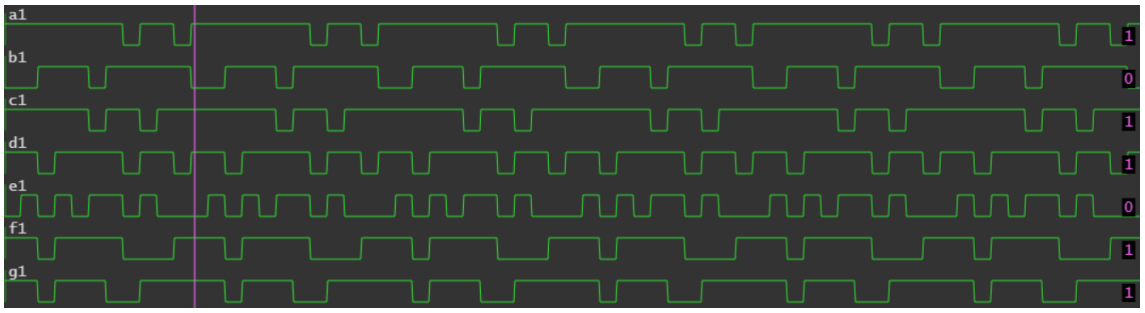
Por fim, é possível simular o circuito e os sinais de saída estão demonstrados nas figuras a seguir. Para fins de melhor visualização escolheu-se um valor qualquer de clock, sendo “clk 385” o decidido:

Figura 3 - Sinais da simulação do contador e registrador de 4 bits



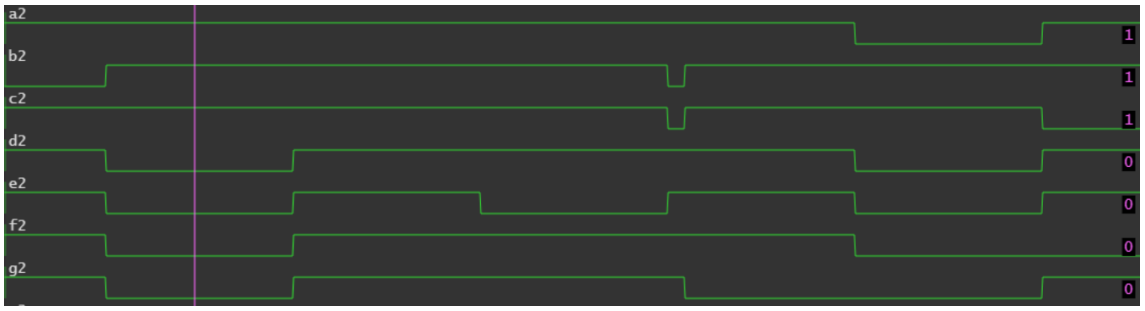
Fonte: Compilado pelos autores

Figura 4 - Sinais da simulação do decodificador BCD para o primeiro dígito



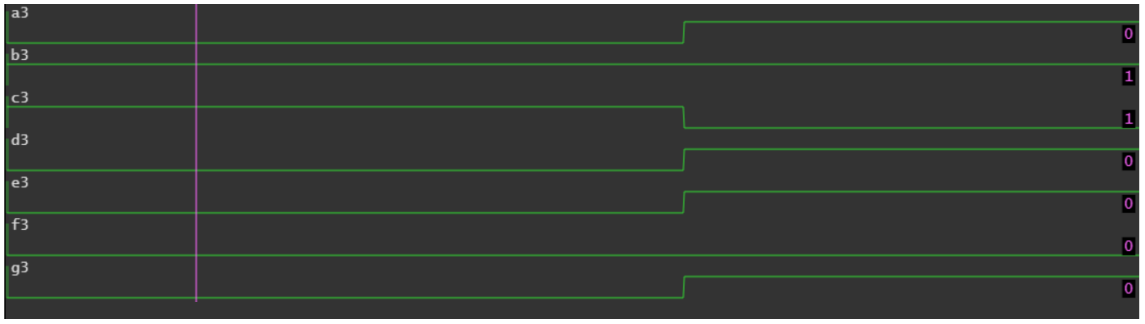
Fonte: Compilado pelos autores

Figura 5 - Sinais da simulação do decodificador BCD para o segundo dígito



Fonte: Compilado pelos autores

Figura 6 - Sinais da simulação do decodificador BCD para o terceiro dígito



Fonte: Compilado pelos autores

As simulações foram feitas utilizando o software *8bitworkshop* a partir do código desenvolvido no item *3. Código.*