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: Segunda Parte

São Carlos
2023

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

Projeto: Segunda 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 da segunda parte do projeto de Sistemas Digitais, dividido em três etapas principais. Primeiramente, apresentamos o passo a passo para a implementação de um Contador Síncrono Crescente de Década, partindo desde a tabela verdade ao circuito lógico implementado.

Em seguida descrevemos o funcionamento um Flip Flop Tipo-D com reset e clock enable ativos em '1', um elemento de circuitos sequenciais utilizado para armazenamento de informação.

Por fim, implementamos, em Linguagem de Descrição de Hardware (HDL) Verilog, contadores binários crescentes parametrizáveis, seguindo três sugestões: Contador assíncrono, utilizando o Flip-flop Tipo-D descrito anteriormente, com generate e Rede de Ligações; Contador síncrono, utilizando o Flip-flop Tipo-D descrito anteriormente, com generate e Rede de Ligações; e Contador com incremento, utilizando Declaração Procedural ou Comportamental (always if-else).

2 Contador Síncrono Crescente de Década

Para a implementação de um contador síncrono, é comum seguir alguns passos que facilitem a construção do contador. Primeiro, a partir da ideia do circuito, montamos um diagrama de estados/valores, de onde se pode extrair a tabela de transição de estados. Com essas informações, escolhemos o elemento de memória para implementação do circuito e montamos a tabela de excitação, da qual obtemos as expressões booleanas com o auxílio de mapas de Karnaugh para, finalmente, desenharmos o circuito lógico.

2.1 Diagrama de Estados

Como o exemplo trata da implementação de um contador crescente de década, queremos um contador de 4 bits que inicie em 0 (0000 em BCD) e termine em 9 (1001 em BCD), reiniciando a contagem após isso. Dessa forma, podemos representar esse objetivo com um Diagrama de Estados (Figura 1).

2.2 Tabela de Transição de Estados

Analisando o Diagrama de Estados, notamos que a mudança de um estado para o outro está na própria ordem de contagem, ou seja, é fácil montar a Tabela de Transição de Estados, basta seguir a sequencia de *minterms* de 1 a 9 e finalizar com o 0, como mostrado na Tabela 1.

Figura 1 – Diagrama de Estados para Contador Síncrono Crescente de Década

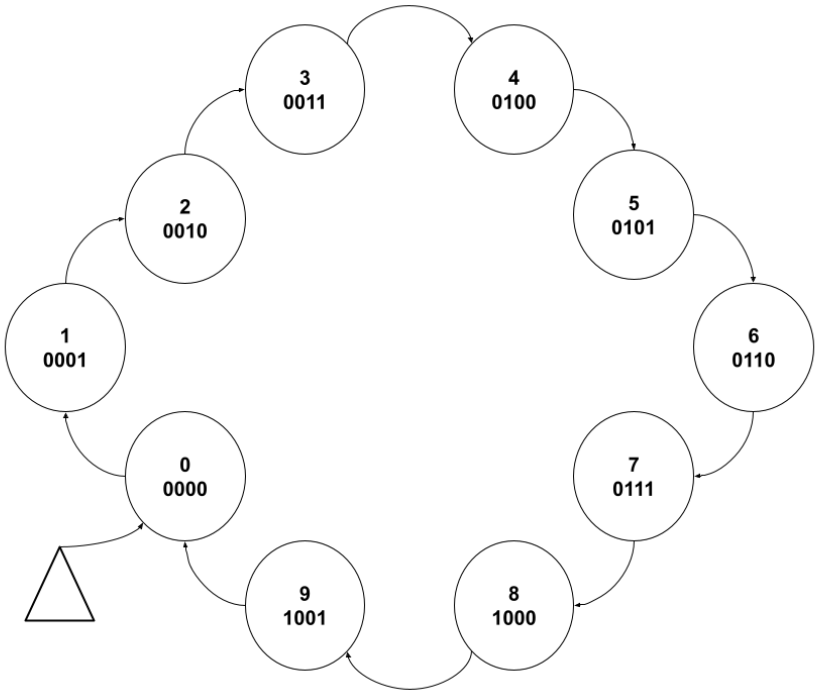


Tabela 1 – Tabela de Transição de Estados para Contador Síncrono de Década

Q_3	Q_2	Q_1	Q_0	Q_3^*	Q_2^*	Q_1^*	Q_0^*
0	0	0	0	0	0	1	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0

2.3 Tabela de Excitação para Flip Flop Tipo-D

De modo a facilitar a abordagem do projeto, escolhemos o flip flop Tipo-D para implementação do contador em questão, tendo em vista que o mesmo é utilizado novamente ao longo do projeto. Relembrando o funcionamento do FF Tipo-D, a Tabela 2 demonstra seu funcionameno para um bit, note que este não passa de um dispositivo que armazena o estado em questão.

A partir dessa lógica e da tabela de transição obtida anteriormente, estedemos o conceito do FF Tipo-D para 4 bits, para obtenção de uma contagem de década crescente

Tabela 2 – Tabela de Excitação 1 bit FF Tipo-D

Qa	Q^*	D
0	0	0
0	1	1
1	0	0
1	1	1

(Tabela 3).

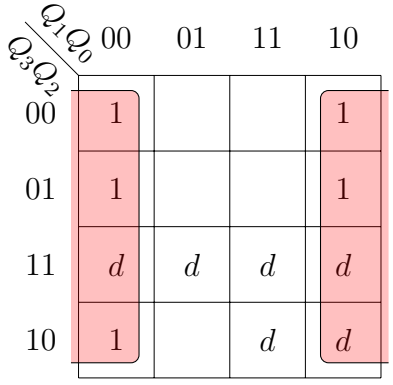
Tabela 3 – Tabela de Excitação FF Tipo-D 4 bits para Contagem de Década

Q_3	Q_2	Q_1	Q_0	Q_3^*	Q_2^*	Q_1^*	Q_0^*	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	1	0	0	0	1	0
0	0	1	0	0	0	1	1	0	0	1	1
0	0	1	1	0	1	0	0	0	1	0	0
0	1	0	0	0	1	0	1	0	1	0	1
0	1	0	1	0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	1	0	1	1	1
0	1	1	1	1	0	0	0	1	0	0	0
1	0	0	0	1	0	0	1	1	0	0	1
1	0	0	1	0	0	0	0	0	0	0	0

2.4 Equações Booleanas de Excitação

A partir da tabela de excitação (Tabela 3), utilizamos de mapas de Karnaugh para obter as Expressões Booleanas para cada Excitação dos FFs.

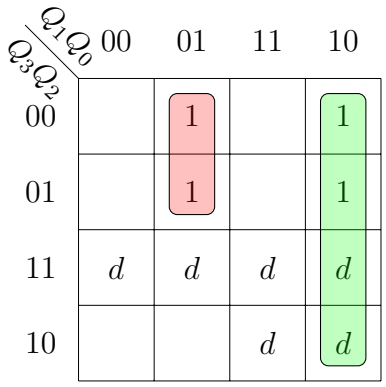
Mapa de Karnaugh para excitação D_0



Portanto, a Expressão Booleana para o FF D_0 é dada por:

$$D_0 = Q_0 \oplus 1 = \overline{Q_0} \tag{1}$$

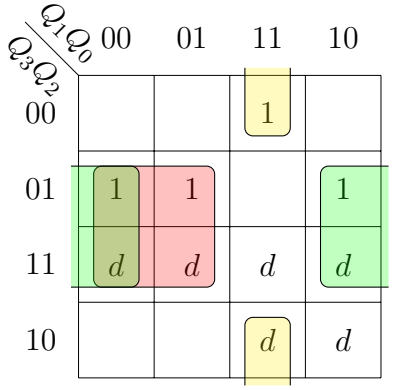
Mapa de Karnaugh para excitação D_1



Assim, a Expressão Booleana para o FF D_1 é dada por:

$$D_1 = (\overline{Q_3} \cdot \overline{Q_1} \cdot Q_0) + (Q_1 \cdot \overline{Q_0}) \tag{2}$$

Mapa de Karnaugh para excitação D_2



Logo, a Expressão Booleana para o FF D_2 é dada por:

$$D_2 = Q_2 \oplus (Q_1 \cdot Q_0) \tag{3}$$

Mapa de Karnaugh para excitação D_3

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00				
01			1	
11	d	d	d	d
10	1		d	d

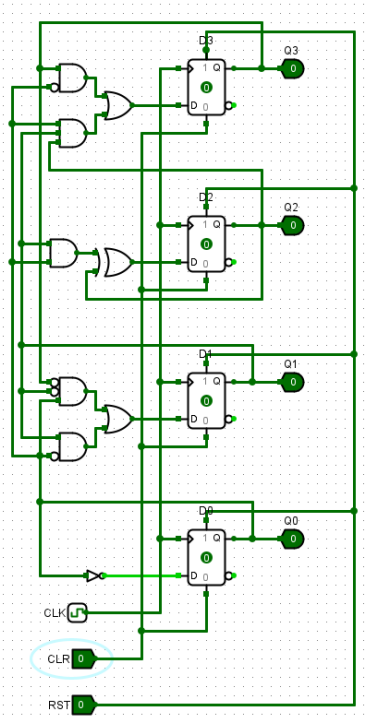
Dessa forma, a Expressão Boolean para o FF D_3 é dada por:

$$D_3 = (Q_3 \cdot \overline{Q_0}) + (Q_2 \cdot Q_1 \cdot Q_0) \tag{4}$$

2.5 Circuito Lógico

Com as Expressões Booleanas obtidas anteriormente, podemos então desenhar o circutio lógico para implementação do contador de década utilizando Flip Flops Tipo-D (Figura 2).

Figura 2 – Circuito Lógico de Contador de Década Síncrono com FF Tipo-D



3 Flip Flop Tipo D com reset e clock enable ativos em '1'

A implementação de um FF Tipo-D pode ser obtida a partir de um FF JK-MS, onde a sua entrada K recebe o complemento da entrada J, gerando uma nova entrada chamada D. A entrada Reset (ou Clear) é adicionada ao Flip Flop com objetivo de força a saída $Q = 0$ quando desejado, e a entrada Clock enable (enb) é adicionada de forma a habilitar (ou negar) o sinal de CLK.

A seguir, foi construída uma tabela (Tabela 4) de excitação para o Flip-Flop tipo D com sinais 'rst' e 'enb', permitindo descrever em sua completude o funcionamento do circuito sequencial desejado.

Tabela 4 – Tabela de Excitação FF Tipo D com entradas 'rst' e 'enb'

Qa	Q^*	rst	enb	D
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

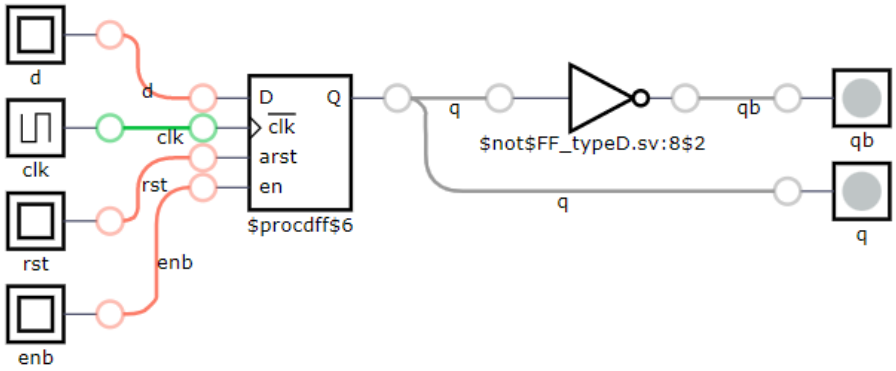
A partir da tabela, percebe-se que o sinal $rst = 1$ em obrigatoriamente torna a saída $D = 0$, independentemente das demais entradas. Sendo $rst = 0$ o sinal $enb = 0$ torna a saída $D = Qa$, enquanto $enb = 1$ a torna $D = Q^*$.

A partir da tabela construída e das constatações observadas, foi produzido o código de descrição do circuito. Abaixo segue o código em Verilog (HDL) implementando o Flip-Flop tipo D com 'reset' e 'clock enable' e o circuito correspondente (Figura 3).

```
module dff(output reg q, qb, input clk, d, enb, rst);
  always @ (negedge clk or posedge rst)
    if (rst)
```

```
      q = 1'b0;
          else if (enb)
            q = d;

  assign qb = ~q;
endmodule
```

Figura 3 – Circuito esquemático do Contador Assíncrono



4 Implementação de Contadores Parametrizáveis em HDL

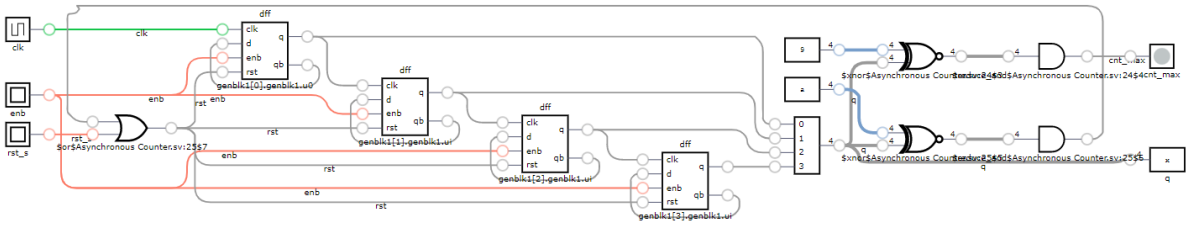
A partir dos Flip-Flops tipo D descritos anteriormente, serão implementados agora três tipos distintos de contadores binários crescentes parametrizáveis de 'width' bits e valor máximo 'max_value', com entradas 'enb' (para habilitar ou desabilitar o sinal de Clock), 'rst_s' (para resetar o contador de volta para 0), 'clk' (entrada de Clock) e saída 'cnt_max' (informando que o contador está em seu valor máximo) . Para a implementação dos contadores, será utilizada a linguagem de descrição de hardware Verilog (HDL). Cada implementação também pode se encontrada no repositório do projeto.

4.1 Assíncrono com generate e Rede de Ligações

Nesta seção, será implementado um contador assíncrono empregando Flip-Flops tipo D e contendo as entradas previamente descritas. Tal tipo de contador é caracterizado pelo fato de que apenas o primeiro Flip-Flop está ligado ao sinal de Clock, sendo que a entrada 'clk' dos demais é populada pela saída 'q' do Flip-Flop anterior. Abaixo segue o código criado e circuito esquemático correspondente (Figura 4).

```
module dff(output reg q, qb, input clk, d, enb, rst);
  always @ (negedge clk or posedge rst)
    if (rst)
      q = 1'b0;
    else if (enb)
      q = d;

  assign qb = ~q;
endmodule

module counter_a #(parameter width=4, max_count=10)(output [width-1:0] q,
↪  output cnt_max, input clk, enb, rst_s);
  wire [width-1:0] qb;
  wire rst;
  parameter max = max_count - 1;

  genvar i;
  generate
    for (i=0; i<width; i=i+1)
      if (i == 0)
        dff u0 (.q(q[i]), .qb(qb[i]), .clk(clk), .d(qb[i]), .enb(enb),
↪  .rst(rst));
      else
              dff ui (.q(q[i]), .qb(qb[i]), .clk(q[i-1]), .d(qb[i]),
↪  .enb(enb), .rst(rst));
  endgenerate
  assign cnt_max = &(max[width-1:0] ~^ q[width-1:0]);
  assign rst = &(max_count[width-1:0] ~^ q[width-1:0]) | rst_s;
endmodule
```

Figura 4 – Circuito esquemático do Contador Assíncrono



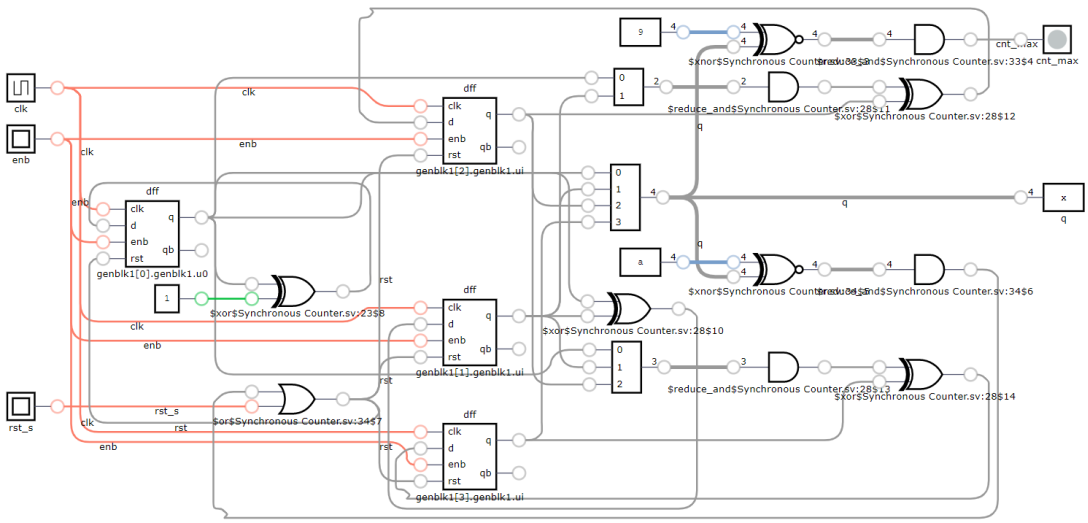
4.2 Síncrono com generate e Rede de Ligações

Nesta seção, será implementado um contador síncrono novamente empregando Flip-Flops tipo D e contendo as entradas previamente descritas. Este diferencia-se do anterior pelo fato de todos os Flip-Flops receberem o sinal de Clock em sua entrada 'clk', sendo necessário então um circuito combinacional auxiliar para definir as mudanças de sinal em Flip-Flops em níveis maiores do circuito. A seguir, seguem a implementação e circuito esquemático (Figura 5).

```
module dff(output reg q, qb, input clk, d, enb, rst);
  always @ (negedge clk or posedge rst)
    if (rst)
      q = 1'b0;
          else if (enb)
            q = d;

  assign qb = ~q;
endmodule

module counter_s #(parameter width=4, max_count=10)(output [width-1:0] q,
↪  output cnt_max, input clk, enb, rst_s);
  wire [width-1:0] qb;
  wire [width-1:0] de;
  wire rst;
  parameter max = max_count - 1;

  genvar i;
  generate
    for (i=0; i<width; i=i+1)
      begin
        if (i == 0)
          begin
            assign de[i] = q[i]^1;
            dff u0 (.q(q[i]), .qb(qb[i]), .clk(clk), .d(de[i]), .enb(enb),
↪  .rst(rst));
          end
        else
          begin
```

```
                assign de[i] = (&q[i-1:0])^q[i];
            dff ui (.q(q[i]), .qb(qb[i]), .clk(clk), .d(de[i]), .enb(enb),
↪   .rst(rst));
          end
      end
  endgenerate
  assign cnt_max = &(max[width-1:0] ~^ q[width-1:0]);
  assign rst = &(max_count[width-1:0] ~^ q[width-1:0]) | rst_s;
endmodule
```

Figura 5 – Circuito esquemático do Contador Síncrono

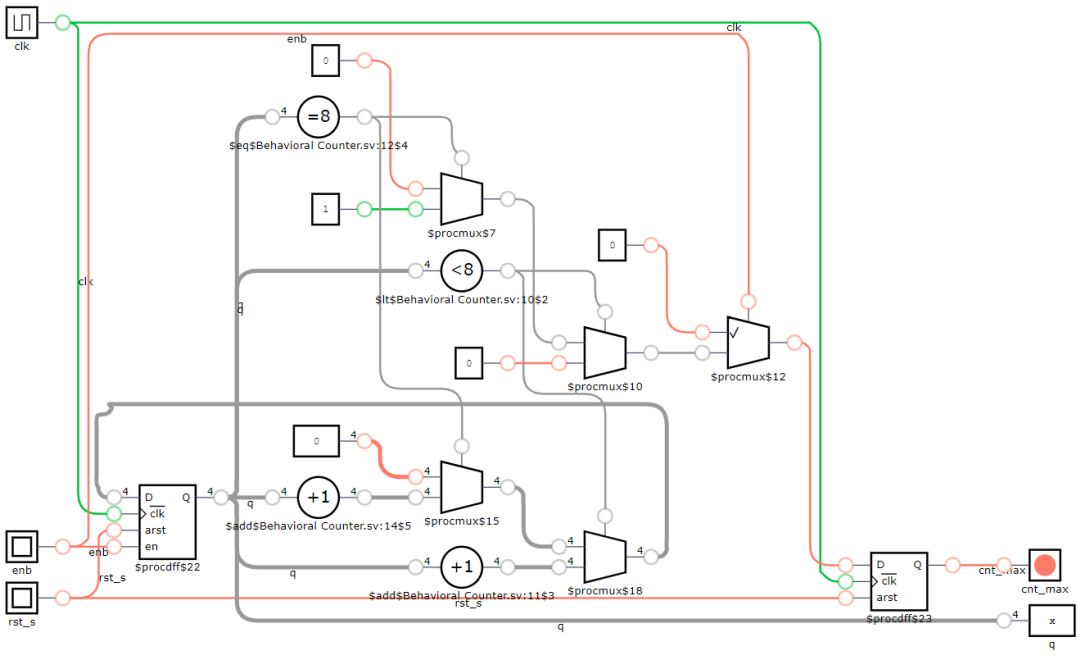


4.3 Com incremento, utilizando Declaração Procedural ou Comportamental

Por fim, a última implementação utilizada foi através do emprego de Declarações comportamentais, com implementação mais simples e intuitiva que os anteriores, não sendo necessário nem mesmo definir o Flip-Flop. Segue o código e esquemático (Figura 6) produzidos.

```
module behav_counter #(parameter width=4, max_count=10)(input clk, rst_s,
↪  enb, output reg [width-1:0] q, output cnt_max);

  always @ (negedge 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
```

Figura 6 – Circuito esquemático do Contador pelo método de Declaração Procedural ou Comportamental



5 Conclusão

A partir do exposto, pode-se concluir que os circuitos contadores foram implementados de modo adequado, em todas as variações realizadas, uma vez que as representações a partir da linguagem Verilog fornecem resultados corretos para os testes realizados. Ademais, acredita-se que a descrição e implementação anterior do contador síncrono crescente de década e do Flip-Flop Tipo D com sinais rst e enb também foram satisfatórias.

No mais, ressalta-se um certo nível de dificuldade encontrado na implementação dos contadores, especialmente no que concerne à implementação de um limite superior para estes. Nesse sentido, haviam dúvidas iniciais (já sanadas) acerca do modo mais adequado de promover a emissão do sinal cnt_max e o reinício da contagem, solucionada através da emissão do sinal cnt_max quando $q = max\_value - 1$ e de um sinal rst assíncrono quando $q = max\_value$.

6 Bibliografia

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