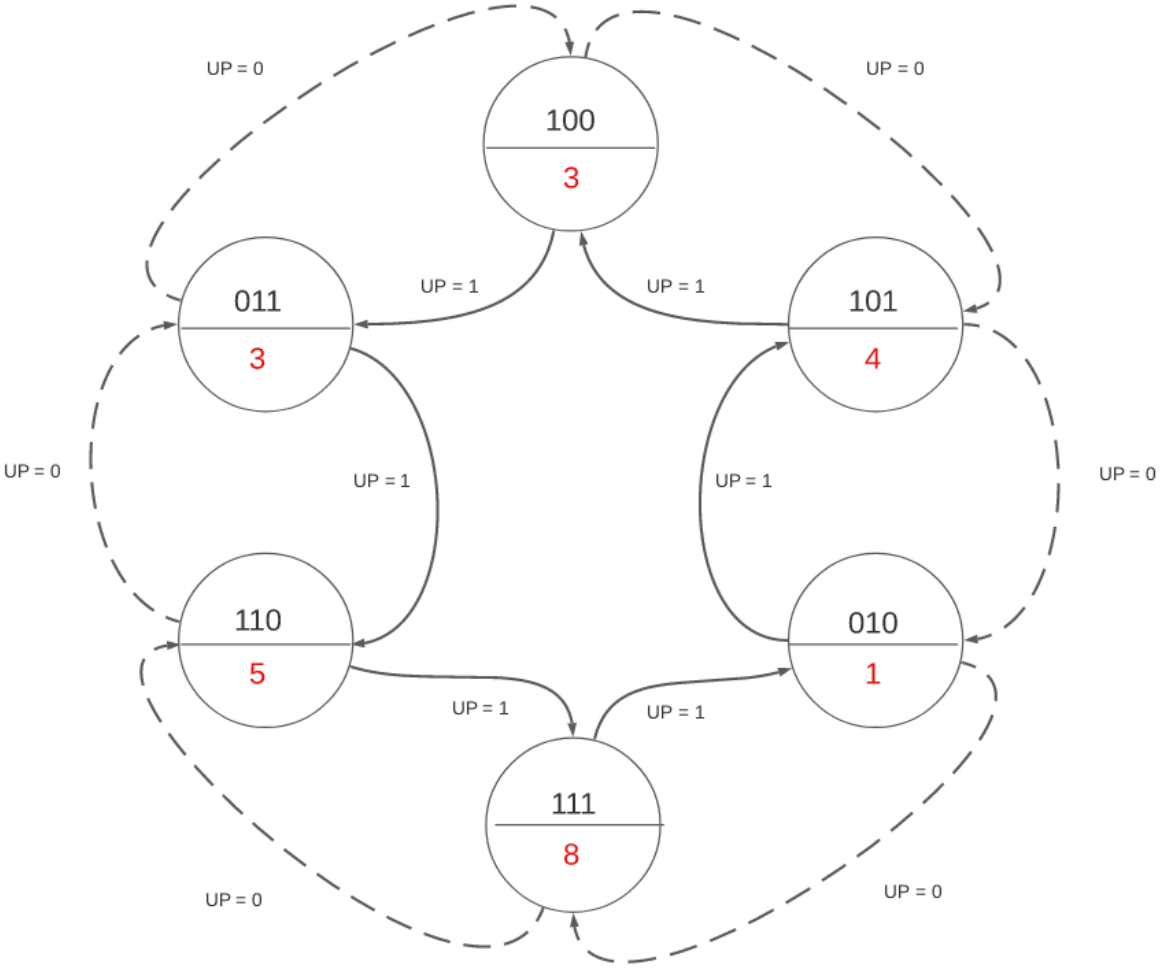
Relatório do Trabalho Final do curso de Circuitos Digitais

INF01058 - Prof. Andre Reis

Nome: Henrique Lorentz Trein
Matrícula: 00341853

1. Diagrama da máquina de estados com as devidas codificações da matrícula



2. Tabela verdade do circuito combinacional relativo ao próximo do estado (CC_PE)

UP (A)	**EA2 (B)**	**EA1 (C)**	**EA0 (D)**	PE2	PE1	PE0
0	0	0	0	X	X	X
0	0	0	1	X	X	X
0	0	1	0	1	1	1
0	0	1	1	1	0	0
0	1	0	0	1	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	1	0
1	0	0	0	X	X	X
1	0	0	1	X	X	X
1	0	1	0	1	0	1
1	0	1	1	1	1	0
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	1	1
1	1	1	1	0	1	0

3. Equações relativas ao CC_PE

Foi utilizado o software Karma para verificação das equações, por isso as entradas possuem um nome alternativo sinalizado entre ( ) referente a utilização do software.

PE2 = (!A*C*D)+(!B)+(A*C*!D)+(!A*!C*!D)+(A*!C*D)

PE1 = (!A*B*D)+(A*B*!D)+(A*C*D)+(!A*C*!D)

PE0 = (!D)

4. Tabela verdade do circuito combinacional relativo ao próximo do estado (CC_Saida)

EA2 (A)	**EA1 (B)**	**EA0 (C)**	a	b	b	d	e	f	g
0	0	0	X	X	X	X	X	X	X
0	0	1	X	X	X	X	X	X	X
0	1	0	0	1	1	0	0	0	0
0	1	1	1	1	1	1	0	0	1
1	0	0	1	1	1	1	0	0	1
1	0	1	0	1	1	0	0	1	1
1	1	0	1	0	1	1	0	1	1
1	1	1	1	1	1	1	1	1	1

5. Equações relativas ao CC_Saida

Foi utilizado o software Karma para verificação das equações, por isso as entradas possuem um nome alternativo sinalizado entre ( ) referente a utilização do software.

a = (A*!C)+(B*C)

b = (C)+(!A)+(!B)

c = 1

d = (A*!C)+(B*C)

e = (A*B*C)

f = (A*B)+(!B*C)

g = (C)+(A)

6. Versão final do projeto em verilog

DESCRIÇÃO

```
module me (
input   eck,
input   er,
input   es,
input   eena,
input   up,
input   ini,
output  [6:0] sq);

wire [2:0] ea;
wire [2:0] pe;
wire out, out_certo;
wire nini;
not(nini, ini);
and(out_certo, eena, out);

freq_div contador (.eck(eck),.er(er),.es(es),.eena(eena),.j(out));

cc_pe proximo_estado (.ea(ea), .up(up), .pe(pe));

reg3 meu_registrador_querido (.d(pe), .ck(eck), .reset(er),
.set(es), .enable(out_certo), .q(ea), .nini(nini));

cc_saida xololo (.ea(ea), .s(sq));

endmodule


module cc_pe (
input   [2:0] ea,
input up,
output  [2:0] pe);

wire nup;
wire nea[2:0];
wire w0, w1, w2, w3, w4, w5, w6, w7;

not(nup, up);
not(nea[2], ea[2]);
```

```
not(nea[1], ea[1]);
not(nea[0], ea[0]);

and(w0, nup, ea[1], ea[0]);
and(w1, up, ea[1], nea[0]);
and(w2, nup, nea[1], nea[0]);
and(w3, up, nea[1], ea[0]);

and(w4, nup, ea[2], ea[0]);
and(w5, up, ea[2], nea[0]);
and(w6, up, ea[1], ea[0]);
and(w7, nup, ea[1], nea[0]);

or(pe[2], w0, w1, w2, w3, nea[2]);
or(pe[1], w4, w5, w6, w7);
buf(pe[0], nea[0]);

endmodule


module cc_saida (
input   [2:0] ea,
output  [6:0] s);

wire nea0,nea1,nea2;
wire w0,w1,w2,w3,w4,w5;
not(nea2, ea[2]);
not(nea1, ea[1]);
not(nea0, ea[0]);

and(w0, ea[2], ea[1]);
and(w1, nea1, ea[0]);

and(w2, ea[2], nea0);
and(w3, ea[1], ea[0]);

and(w4, nea0, ea[2]);
and(w5, ea[1], ea[0]);

or(s[6], w4, w5);
or(s[5], ea[0], nea2, nea1);
buf(s[4], 1'b1);
or(s[3], w2, w3);
```

```
and(s[2], ea[2], ea[1], ea[0]);
or(s[1], w0, w1);
or(s[0], ea[0], ea[2]);

endmodule


module reg3 (
input   [2:0] d,
input   ck,
input   reset,
input   set,
input   enable,
output  [2:0] q,
input   nini);

dffrse armazena_ea2(
        .din(d[2]),
        .dout(q[2]),
        .clk(ck),
        .en(enable),
        .set(nini),
        .rst(reset));

dffrse armazena_ea1(
        .din(d[1]),
        .dout(q[1]),
        .clk(ck),
        .en(enable),
        .set(set),
        .rst(nini));

dffrse armazena_ea0(
        .din(d[0]),
        .dout(q[0]),
        .clk(ck),
        .en(enable),
        .set(set),
        .rst(nini));

endmodule
```

```
module dffrse(din, dout, clk, en, set, rst);
    input din, clk, en, set, rst;
    output reg dout;

    always@(posedge clk, posedge rst) begin
        if(rst)
            dout <= 1'b0;
        else if(set)
            dout <= 1'b1;
        else if(en)
            dout <= din;
        else
            dout <= dout;
    end
endmodule


//divisor
module freq_div (
input   eck,
input   er,
input   es,
input   eena,
output  j);

wire [1:0] ea;
wire [1:0] pe;

cc_pe2 proximo_estado (.ea(ea), .pe(pe));

reg4 meu_registrador_querido (.d(pe), .ck(eck), .reset(er),
.set(es), .enable(eena), .q(ea));

cc_saida2 xololo (.ea(ea), .s(j));

endmodule


module cc_pe2 (
input   [1:0] ea,
output  [1:0] pe);

buf(pe[1], ea[0]);
```

```
nor(pe[0], ea[1], ea[0]);

endmodule


module cc_saida2 (
input   [1:0] ea,
output  s);

buf(s, ea[0]);

endmodule


module reg4 (
input   [1:0] d,
input   ck,
input   reset,
input   set,
input   enable,
output  [1:0] q);

dffrse armazena_ea1(
        .din(d[1]),
        .dout(q[1]),
        .clk(ck),
        .en(enable),
        .set(set),
        .rst(reset));

dffrse armazena_ea0(
        .din(d[0]),
        .dout(q[0]),
        .clk(ck),
        .en(enable),
        .set(set),
        .rst(reset));
endmodule


module dffrse2(din, dout, clk, en, set, rst);
    input din, clk, en, set, rst;
    output reg dout;
```

```
    always@(posedge clk, posedge rst) begin
        if(rst)
            dout <= 1'b0;
        else if(set)
            dout <= 1'b1;
        else if(en)
            dout <= din;
        else
            dout <= dout;
    end
endmodule
```

PINOUT

- **Observação:** deve-se ligar sw9 (chave para começar no estado inicial correto) e sw2 (enable) para começar a contagem.

```
// (c) 2023 inPlace Design Automation
// Descrição : arquivo de pinagem contendo o mapeamento do módulo
counter2b para
//             a placa T (T-board).
//  MODULE PORT     BOARD COMPONENT

    eck          =   clk_1hz;
    up           =  sw3;
    eena          =   sw2;
    es          =   sw1;
    er          =   sw0;
    ini         =   sw9;
// Mapeamento dos bits de saída
    sq[6]       =   segd0.a_on;
    sq[5]       =   segd0.b_on;
    sq[4]       =   segd0.c_on;
    sq[3]       =   segd0.d_on;
    sq[2]       =   segd0.e_on;
    sq[1]       =   segd0.f_on;
    sq[0]       =   segd0.g_on;
```