centro universitário FEI

CURSO DE ENGENHARIA ELÉTRICA

NE7720 – LABORATÓRIO DE SISTEMAS DIGITAIS II

1° SEMESTRE DE 2021

RELATÓRIO DO PROJETO 2: SUPERVISÃO DE PEDÁGIO

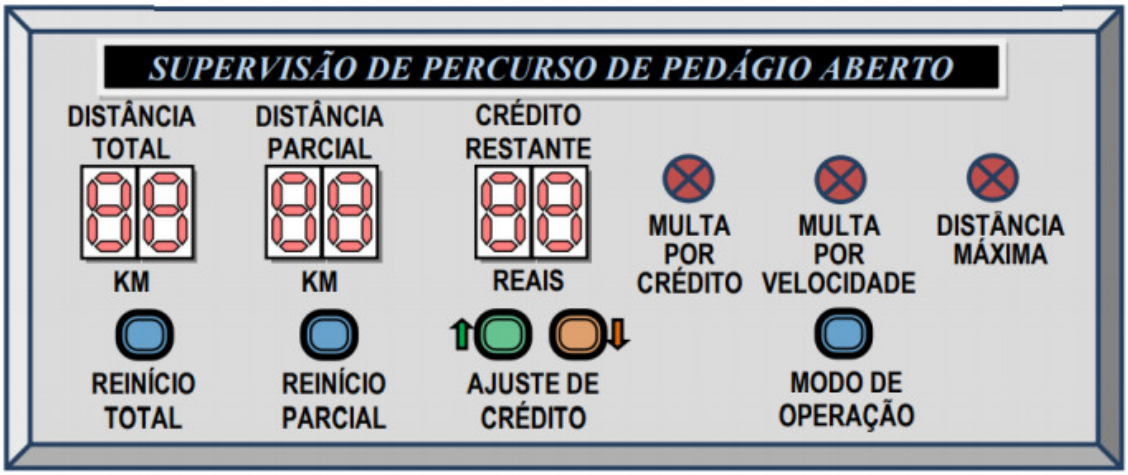
Nome dos Alunos	Número	Turma
DANILO H. B. DOS SANTOS	12.218.079-7	615

1. INTRODUÇÃO

O objetivo geral deste projeto 2 do Laboratório de Sistemas Digitais II é desenvolver um sistema digital de controle para Supervisão de Percurso de Pedágio (Aberto), similar a um sistema SCADA. Devemos desenvolver uma máquina de estado finita (FSM) e a lógica necessária para os demais comandos em FPGA, utilizando a linguagem de descrição VHDL implementada no software Quartus Prime Lite Edition 16.1 e e utilizando a metodologia RTL

Esse laboratório também tem como objetivos exercitar a metodologia de desenvolvimento de projetos de engenharia apoiada em computador (CAE) onde todo o desenvolvimento do software embarcado e modelo final será feito no software Quartus Prime Lite Edition 16.1 onde será feita a simulação funcional e interface final do sistema de supervisão totalmente implementado utilizando o FPGA da família MAX 10 (modelo: 10M50DAF484C7G) existente na placa de desenvolvimento Altera DE10-Lite. Esse processo envolve a compreensão do problema, seu planejamento, desenvolvimento da solução lógica, integração dos subsistemas, implementação no ambiente computacional, simulação, testes, depuração do projeto, implementação física e registro dos resultados.

Figura 1: Esquema do painel do SCADA para Percurso e Pedágio Aberto
(Fonte: Roteiro do Projeto 2)



Desta forma, vamos realizar o desenvolvimento utilizando os conceitos abordados na Teoria, como o projeto do Fluxo de Dados (FD), bem como a Unidade de Controle para o sistema, como podemos verificar na Figura 2 abaixo.

Figura 2: Diagrama de Blocos do SCADA para Percurso e Pedágio Aberto
(Fonte: Roteiro do Projeto 2)

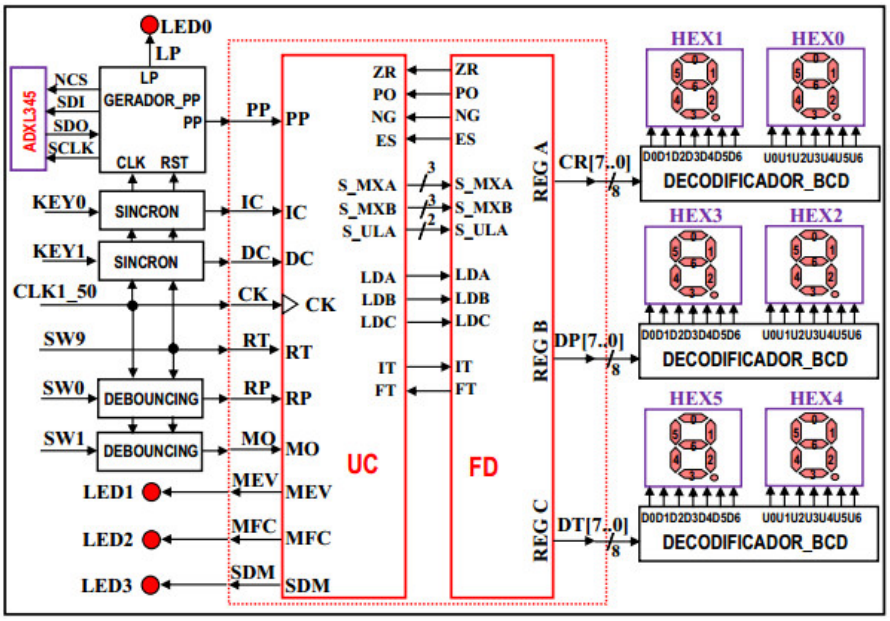
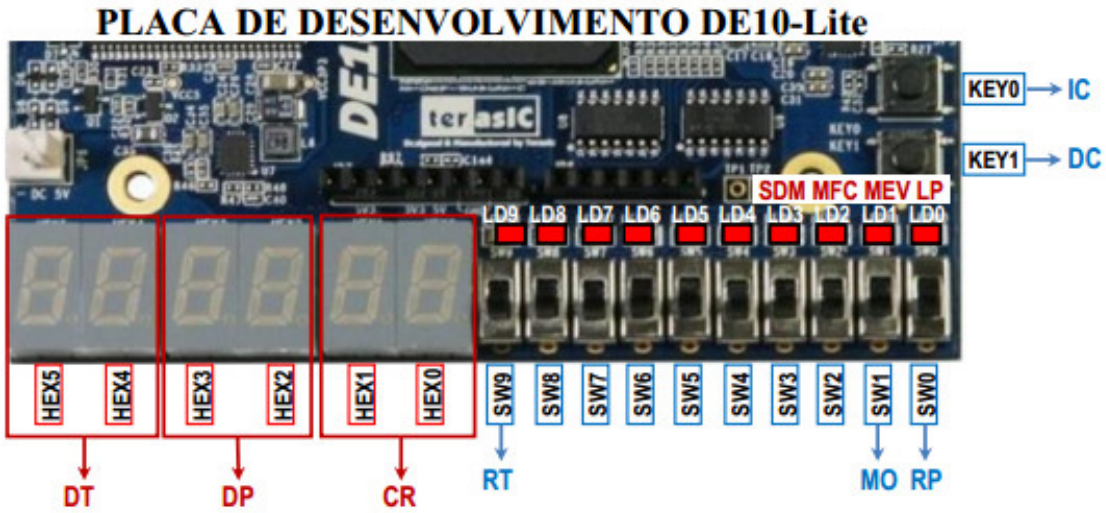
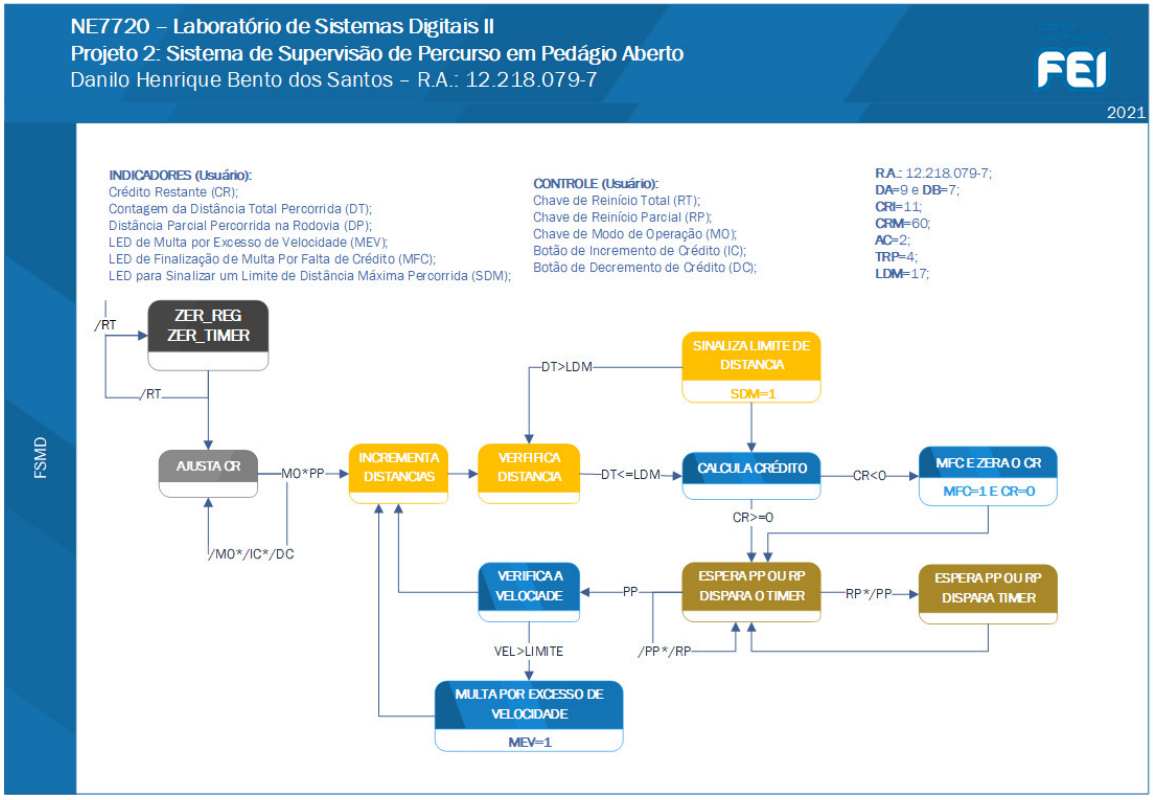


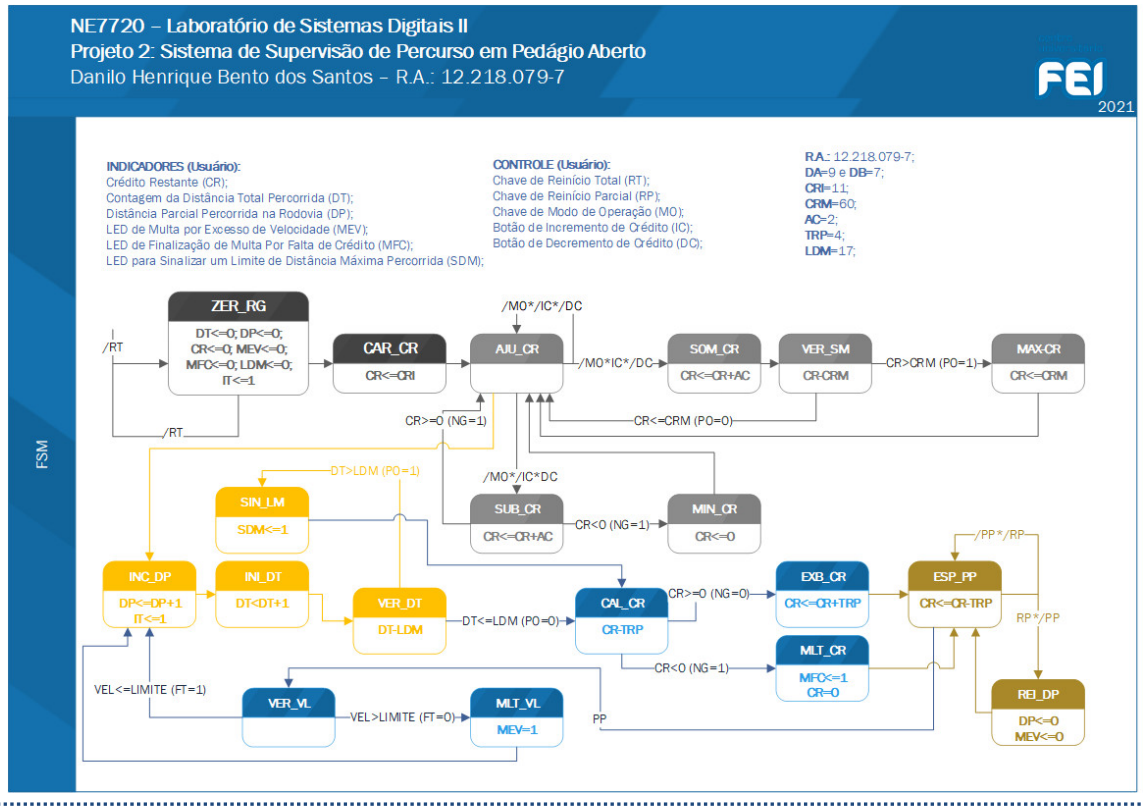
Figura 3: Interfaces do sistema de Supervisão de Percurso de Pedágio Aberto na placa DE10-lite.
(Fonte: TERASIC, 2020 (adaptado).)



2. MÁQUINA DE ESTADO DE ALTO NÍVEL PARA O SISTEMA (FSMD)



3. DIAGRAMA DE ESTADOS DA UNIDADE DE CONTROLE (FSM)



4. TABELA DE SINAIS DE CONTROLE DO FLUXO DE DADOS (FD)

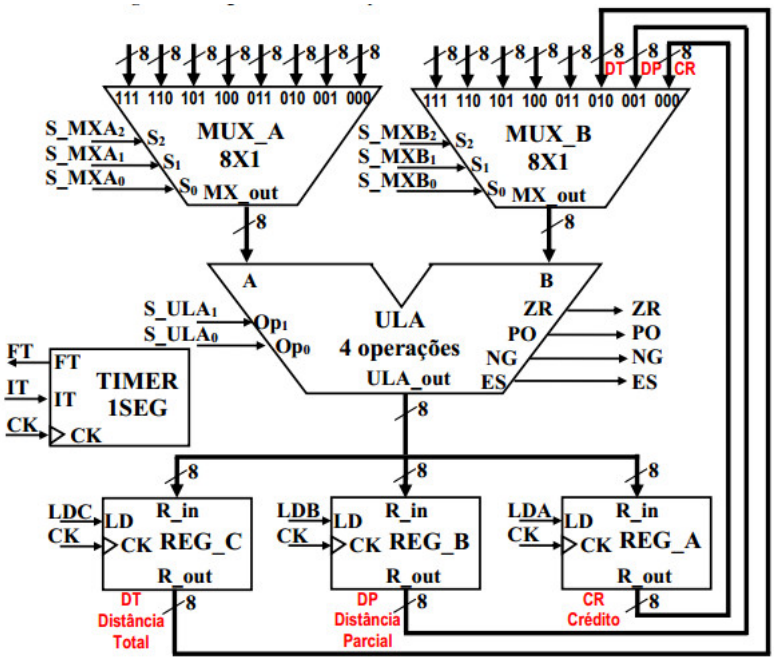
TABELA DE SINAIS

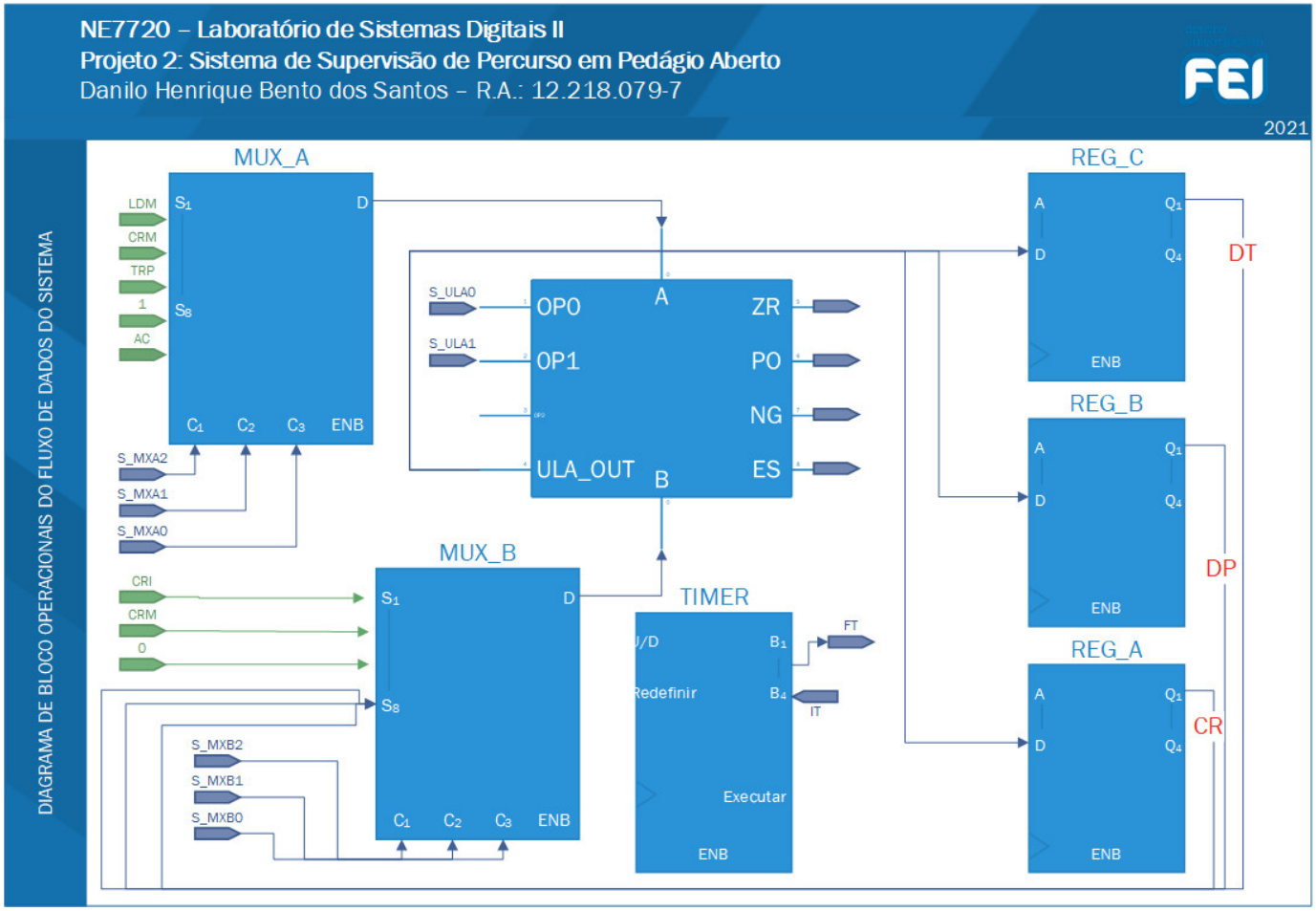
VARIÁVEIS	TIPO	ATIVAÇÃO	DESCRIÇÃO
RT	LOGIC	0	BOTÃO DE ACIONAMENTO DO RESET GERAL (INICIA OS REG) - SW9
MO	LOGIC	1	BOTÃO DE MODO DE OPERAÇÃO (OP. NORMAL OU OP. DE CARGA - SW1
RP	LOGIC	1	BOTÃO ACIONAMENTO DE RESET DA DISTÂNCIA PARCIAL E MEV - SW0
PP	LOGIC	PULSE	SINAL DE INDICAÇÃO DE PASSAGEM PELO PÓRTICO - PP
IC	LOGIC	PULSE	BOTÃO DE ACIONAMENTO DE INCREMENTO DO CRÉDITO - KEY0
DC	LOGIC	PULSE	BOTÃO DE ACIONAMENTO DO DECREMENTO DO CRÉDITO - KEY1
MFC	LOGIC	1	SINAL DE INDICAÇÃO DE MULTA POR FALTA DE CRÉDITO - MFC=1
MEV	LOGIC	1	SINAL DE INDICAÇÃO DE MULTA POR EXCESSO DE VELOCIDADE - MEV=1
SDM	LOGIC	1	SINAL DE INDICAÇÃO DE LIMITE DE DISTÂNCIA MÁXIMA ATINGIDO - SDM=1
CR	VECTOR	8 BITS	VALOR DE CRÉDITO RESTANTE (HEX1 E HEX0)
DP	VECTOR	8 BITS	VALOR DA DISTÂNCIA PARCIAL (HEX3 E HEX2)
DT	VECTOR	8 BITS	VALOR DA DISTÂNCIA TOTAL (HEX5 E HEX4)

ATRIBUIÇÃO DAS VARIÁVEIS NO FLUXO DE DADOS

ESTADO	S_MXA[2..0]	S_MXB[2..0]	S_ULA[1..0]	LDC	LDB	LDA	IT
ZER_RG	DC	101	00	1	1	1	1
CAR_CR	DC	111	00	0	0	1	DC
AJU_CR	DC	DC	DC	0	0	0	DC
SOM_CR	011	000	10	0	0	1	DC
VER_SM	110	000	11	0	0	0	DC
MAX_CR	DC	110	00	0	0	1	DC
SUB_CR	011	000	11	0	0	1	DC
MIN_CR	DC	101	00	0	0	1	DC
INC_DP	100	001	10	0	1	0	1
INC_DT	100	010	10	1	0	0	0
VER_DT	111	010	11	0	0	0	DC
SIN_LM	DC	DC	DC	0	0	0	DC
CAL_CR	101	000	11	0	0	0	DC
EXB_CR	101	000	11	0	0	1	DC
MLT_CR	DC	101	00	0	0	1	DC
ESP_PP	DC	DC	DC	0	0	0	DC
REI_DP	DC	101	00	0	1	0	DC
VER_VL	DC	DC	DC	0	0	0	DC
MLT_VL	DC	DC	DC	0	0	0	DC

DA	CRI [R$]	CRM [R$]	DB	AC [R$]	TRP [R$/KM]	LDM [KM]
9	11	60	7	2	4	17


111 110 101 100 011 010 001 000
MUX_A
8X1
MUX_B
8X1
S_MXA2
S_MXA1
S_MXA0
S_MXB2
S_MXB1
S_MXB0
MX_out
DT
DP
CR
A
B
ULA
4 operações
ULA_out
S_ULA1
S_ULA0
Op1
Op0
ZR
PO
NG
ES
FT
IT
CK
TIMER
1SEG
LDC
LDB
LDA
LD
R_in
R_out
REG_C
REG_B
REG_A
DT
Distância
Total
DP
Distância
Parcial
CR
Crédito


NE7720 – Laboratório de Sistemas Digitais II
Projeto 2: Sistema de Supervisão de Percurso em Pedágio Aberto
Danilo Henrique Bento dos Santos – R.A.: 12.218.079-7
2021
DIAGRAMA DE BLOCO OPERACIONAIS DO FLUXO DE DADOS DO SISTEMA
MUX_A
LDM
CRM
TRP
1
AC
S_MXA2
S_MXA1
S_MXA0
C1
C2
C3
ENB
S_ULA0
S_ULA1
OP0
OP1
ULA_OUT
A
B
ZR
PO
NG
ES
REG_C
REG_B
REG_A
DT
DP
CR
MUX_B
CRI
CRM
0
S_MXB2
S_MXB1
S_MXB0
TIMER
Redefinir
Executar
FT
IT

5. DIAGRAMA DE INTERCONEXÃO DO FLUXO DE DADOS (FD)

```
--Quartus Prime VHDL FEI DIGITAL REPORT: UNIDADE DE CONTROLE (UC)

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity UC is
port( CK : in std_logic;              -- clock de 50MHz
      RT : in std_logic;              -- reinicio total -> ativo em zero
-- Entradas Externas:
      IC : in std_logic;          -- incrementa credito -> ativo em um
      DC : in std_logic;          -- decrementa credito -> ativo em um
      RP : in std_logic;          -- reinicio parcial -> ativo em um
      PP : in std_logic;          -- passagem pelo portico -> ativo em um
      MO : in std_logic;          -- modo de operaÃ§Ã£o -> 1= Normal, 0= Ajusta Valor
-- Sinais de Estado da ULA:
      ZR : in std_logic;          -- resultado zero na operaÃ§Ã£o da ULA
      PO : in std_logic;          -- resultado positivo na operaÃ§Ã£o da ULA
      NG : in std_logic;          -- resultado negativo na operaÃ§Ã£o da ULA
      ES : in std_logic;          -- resultado da operaÃ§Ã£o da ULA maior que 255
-- Sinais de Estado do TIMER:
      FT : in std_logic;          -- fim da temporizaÃ§Ã£o de 1 segundo
-- Sinais de Saida para MUX:
      Sel_mxa : out std_logic_vector(2 downto 0);  -- seleciona entrada de MUX_A
      Sel_mxb : out std_logic_vector(2 downto 0);  -- seleciona entrada de MUX_B
-- Sinais de Saida para ULA:
      Sel_ula : out std_logic_vector(1 downto 0);  -- seleciona operaÃ§Ã£o da ULA
-- Sinais de Saida para Registradores:
      Lda   : out std_logic;      -- carrega RA
      Ldb   : out std_logic;      -- carrega RB
      Ldc   : out std_logic;      -- carrega RC
-- Sinais de Saida para TIMER:
      IT    : out std_logic;      -- inicia temporizaÃ§Ã£o
-- Sinais de Saida Externos:
      MEV   :  out std_logic;     -- sinaliza multa por excesso de velocidade
      MFC   :  out std_logic;     -- sinaliza multa por falta de crÃ©dito
      SDM   :  out std_logic      -- sinaliza limite de distÂ¢ncia total percorrida
   );
end UC;
-------------------------------------------------------------------------------
architecture FSM of UC is
type ESTADOS_ME is (ZER_RG, CAR_CR, AJU_CR, SOM_CR, VER_SM, MAX_CR, SUB_CR, MIN_CR, INC_DP,
                    INC_DT, VER_DT, SIN_LM, CAL_CR, MLT_CR, ESP_PP, REI_DP, VER_VL, MLT_VL,
                    EXB_CR
                    );
signal E: ESTADOS_ME;
begin
process(CK, RT)
begin
if RT='0' then E <= ZER_RG;                          -- zera registros
            MFC <= '0'; MEV <= '0'; SDM<='0';IT <= '1';  -- zera multas e sinalizaÃ§Ã£o
elsif (CK'event and CK='1') then
 case E is
    when ZER_RG =>
                E <= CAR_CR;          -- carrega CR com credito inicial
                IT <= '1';
    when CAR_CR =>
                E <= AJU_CR;          --- espera IC e DC com MO=0 para ajustar CR
    when AJU_CR =>
             if MO = '1' and PP='1' then
                E <= INC_DP;          -- incrementa distancia parcial
             elsif MO = '0' and IC = '1' and DC = '0' then
                E <= SOM_CR;
             elsif MO = '0' and IC = '0' and DC = '1' then
                E <= SUB_CR;
             else
                E <= AJU_CR;
             end if;
    when SOM_CR =>
                E <= VER_SM;
    when VER_SM =>
             if PO = '1' then
                E <= MAX_CR;
             elsif PO = '0' then
                E <= AJU_CR;
             end if;
    when MAX_CR =>
                E <= AJU_CR;
```

```
80      when SUB_CR =>
81              if NG = '1' then
82                 E <= MIN_CR;
83              elsif NG = '0' then
84                 E <= AJU_CR;
85              end if;
86      when MIN_CR =>
87                 E <= AJU_CR;
88      when INC_DP =>
89                 E <= INC_DT;
90                 IT <= '1';
91      when INC_DT =>
92                 E <= VER_DT;
93                 IT <= '0';
94      when VER_DT =>
95              if PO = '1' then
96                 E <= SIN_LM;
97              elsif PO = '0' then
98                 E <= CAL_CR;
99              end if;
100     when SIN_LM =>
101                E <= CAL_CR;
102                SDM <= '1';
103     when CAL_CR =>
104             if NG = '0' then
105                E <= EXB_CR;
106             elsif NG = '1' then
107                E <= MLT_CR;
108             end if;
109     when EXB_CR =>
110                E <= ESP_PP;
111     when MLT_CR =>
112                E <= ESP_PP;
113                MFC <= '1';
114     when ESP_PP =>
115             if PP = '1' then
116                E <= VER_VL;
117             elsif RP = '1' and PP = '0' then
118                E <= REI_DP;
119             else E <= ESP_PP;
120             end if;
121     when REI_DP =>
122                E <= ESP_PP;
123                MEV <= '0';
124     when VER_VL =>
125             if FT = '0' then
126                E <= MLT_VL;
127             elsif FT = '1' then
128                E <= INC_DP;
129             end if;
130     when MLT_VL =>
131                E <= INC_DP;
132                MEV <= '1';
133     when others => Null;
134  end case;
135 end if;
136  end process;
137
138  -- AtualizaÃ§Ã£o das SaÃdas para Fluxo de Dados (Multiplexadores, Registradores, ULA)
139 process(E)
140  begin
141   case E is
142     when ZER_RG =>
143             Sel_mxa <= "XXX"; Sel_mxb <= "101"; Sel_ula <= "00";
144             Ldc <= '1'; Ldb <= '1'; Lda <='1';
145     when CAR_CR =>
146             Sel_mxa <= "XXX"; Sel_mxb <= "111"; Sel_ula  <= "00";
147             Ldc <= '0'; Ldb <= '0'; Lda <='1';
148     when AJU_CR =>
149             Sel_mxa <= "XXX"; Sel_mxb <= "XXX"; Sel_ula  <= "XX";
150             Ldc <= '0'; Ldb <= '0'; Lda <='0';
151     when SOM_CR =>
152             Sel_mxa <= "011"; Sel_mxb <= "000"; Sel_ula  <= "10";
153             Ldc <= '0'; Ldb <= '0'; Lda <='1';
154     when VER_SM =>
155             Sel_mxa <= "110"; Sel_mxb <= "000"; Sel_ula  <= "11";
156             Ldc <= '0'; Ldb <= '0'; Lda <='0';
157     when MAX_CR =>
158             Sel_mxa <= "XXX"; Sel_mxb <= "110"; Sel_ula  <= "00";
159             Ldc <= '0'; Ldb <= '0'; Lda <='1';
160     when SUB_CR =>
161             Sel_mxa <= "011"; Sel_mxb <= "000"; Sel_ula  <= "11";
```

```
161             Sel_mxa <= "011"; Sel_mxb <= "000"; Sel_ula  <= "11";
162             Ldc <= '0'; Ldb <= '0'; Lda <='1';
163         when MIN_CR =>
164             Sel_mxa <= "XXX"; Sel_mxb <= "101"; Sel_ula  <= "00";
165             Ldc <= '0'; Ldb <= '0'; Lda <='1';
166         when INC_DP =>
167             Sel_mxa <= "100"; Sel_mxb <= "001"; Sel_ula  <= "10";
168             Ldc <= '0'; Ldb <= '1'; Lda <='0';
169         when INC_DT =>
170             Sel_mxa <= "100"; Sel_mxb <= "010"; Sel_ula  <= "10";
171             Ldc <= '1'; Ldb <= '0'; Lda <='0';
172         when VER_DT =>
173             Sel_mxa <= "111"; Sel_mxb <= "010"; Sel_ula  <= "11";
174             Ldc <= '0'; Ldb <= '0'; Lda <='0';
175         when SIN_LM =>
176             Sel_mxa <= "XXX"; Sel_mxb <= "XXX"; Sel_ula  <= "XX";
177             Ldc <= '0'; Ldb <= '0'; Lda <='0';
178         when CAL_CR =>
179             Sel_mxa <= "101"; Sel_mxb <= "000"; Sel_ula  <= "11";
180             Ldc <= '0'; Ldb <= '0'; Lda <='0';
181         when EXB_CR =>
182             Sel_mxa <= "101"; Sel_mxb <= "000"; Sel_ula  <= "11";
183             Ldc <= '0'; Ldb <= '0'; Lda <='1';
184         when MLT_CR =>
185             Sel_mxa <= "XXX"; Sel_mxb <= "101"; Sel_ula  <= "00";
186             Ldc <= '0'; Ldb <= '0'; Lda <='1';
187         when ESP_PP =>
188             Sel_mxa <= "XXX"; Sel_mxb <= "XXX"; Sel_ula  <= "XX";
189             Ldc <= '0'; Ldb <= '0'; Lda <='0';
190         when REI_DP =>
191             Sel_mxa <= "XXX"; Sel_mxb <= "101"; Sel_ula  <= "00";
192             Ldc <= '0'; Ldb <= '1'; Lda <='0';
193         when VER_VL =>
194             Sel_mxa <= "XXX"; Sel_mxb <= "XXX"; Sel_ula  <= "XX";
195             Ldc <= '0'; Ldb <= '0'; Lda <='0';
196         when MLT_VL =>
197             Sel_mxa <= "XXX"; Sel_mxb <= "XXX"; Sel_ula  <= "XX";
198             Ldc <= '0'; Ldb <= '0'; Lda <='0';
199
200         when others => Null;
201     end case;
202   end process;
203   end FSM;
204
```

```
1   -- Quartus Prime VHDL FEI DIGITAL REPORT: FLUXO DE DADOS (FD)
2
3   LIBRARY IEEE;
4   USE IEEE.STD_LOGIC_1164.ALL;
5
6   ENTITY FD IS
7       PORT(
8           LDA, LDB, LDC, IT, CK    : IN STD_LOGIC;
9           S_MXA, S_MXB             : IN STD_LOGIC_VECTOR (2 DOWNTO 0);
10          S_ULA                    : IN STD_LOGIC_VECTOR (1 DOWNTO 0);
11          ZR, PO, NG, ES, FT       : OUT STD_LOGIC;
12          CR, DP, DT               : BUFFER STD_LOGIC_VECTOR (7 DOWNTO 0)
13      );
14
15  END FD;
16
17  ARCHITECTURE ESTRUTURAL OF FD IS
18
19  COMPONENT MUX8
20      PORT(
21          Da     : in std_logic_vector(7 downto 0);
22          Db     : in std_logic_vector(7 downto 0);
23          Dc     : in std_logic_vector(7 downto 0);
24          Dd     : in std_logic_vector(7 downto 0);
25          De     : in std_logic_vector(7 downto 0);
26          Df     : in std_logic_vector(7 downto 0);
27          Dg     : in std_logic_vector(7 downto 0);
28          Dh     : in std_logic_vector(7 downto 0);
29          S      : in std_logic_vector(2 downto 0);
30          MX_out    : out std_logic_vector(7 downto 0)
31      );
32  END COMPONENT;
33
```

```
34  COMPONENT ULA8
35      PORT(
36          A, B      : in std_logic_vector(7 downto 0);
37          Op        : in std_logic_vector(1 downto 0);
38          ULA_out   : out std_logic_vector(7 downto 0);
39          ZR        : out std_logic;
40          PO        : out std_logic;
41          NG        : out std_logic;
42          ES        : out std_logic
43      );
44  END COMPONENT;
45
46  COMPONENT REG8
47      PORT(
48            CK : in std_logic;
49            LD : in std_logic;
50            R_in  : in std_logic_vector(7 downto 0);
51            R_out : out std_logic_vector(7 downto 0)
52      );
53  END COMPONENT;
54
55  COMPONENT TIMER_1S
56      PORT(
57            CK    : IN std_logic;
58            IT    : IN std_logic;
59            FT    : OUT std_logic
60      );
61  END COMPONENT;
62
63  SIGNAL   MXA_OUT        : std_logic_vector(7 downto 0);
64  SIGNAL   MXB_OUT        : std_logic_vector(7 downto 0);
65  SIGNAL   ULA_OUT        : STD_LOGIC_VECTOR (7 DOWNTO 0);
66
```

```
67  BEGIN
68
69      MUX_A    : MUX8      PORT MAP (Da=>(OTHERS=>'0'), Db=>(OTHERS=>'0'), Dc=>(OTHERS=>'0'), Dd=>"000
70      MUX_B    : MUX8      PORT MAP (Da=>CR, Db=>DP, Dc=>DT, Dd=>(OTHERS=>'0'), De=>(OTHERS=>'0'), Df=
71      ULA      : ULA8      PORT MAP (A=>MXA_OUT,B=>MXB_OUT,Op=>S_ULA,ULA_out=>ULA_OUT,ZR=> ZR,PO=>PO,N
72      REG_A    : REG8      PORT MAP (CK=>CK ,LD=>LDA ,R_in=>ULA_OUT,R_out=>CR);
73      REG_B    : REG8      PORT MAP (CK=>CK ,LD=>LDB ,R_in=>ULA_OUT,R_out=>DP);
74      REG_C    : REG8      PORT MAP (CK=>CK ,LD=>LDC ,R_in=>ULA_OUT,R_out=>DT);
75      TIMER    : TIMER_1S  PORT MAP (CK=>CK ,IT=>IT  ,FT=>FT );
76
77  END ESTRUTURAL;
78
```

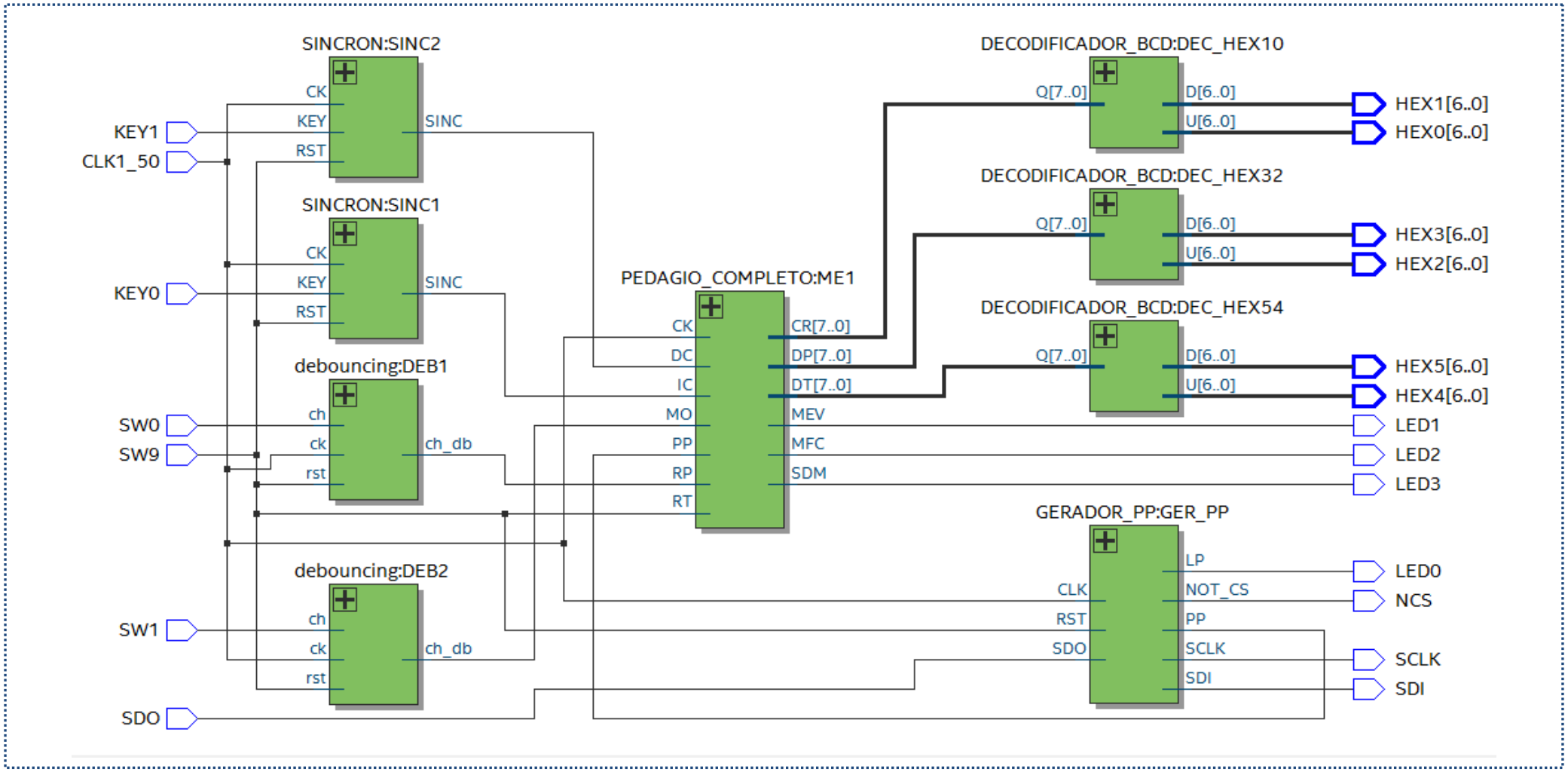
```
67
68
69  )010", De=>"00000001", Df=>"00000100", Dg=>"00111100", Dh=>"00010001",S=>S_MXA, MX_out=>MXA_OUT);
70  (OTHERS=>'0'), Dg=>"00111100", Dh=>"00001011", S=>S_MXB, MX_out=>MXB_OUT);
71  =>NG,ES=>ES);
72
73
74
75
76
77
78
```

6. CÓDIGO VHDL DO PROJETO COMPLETO

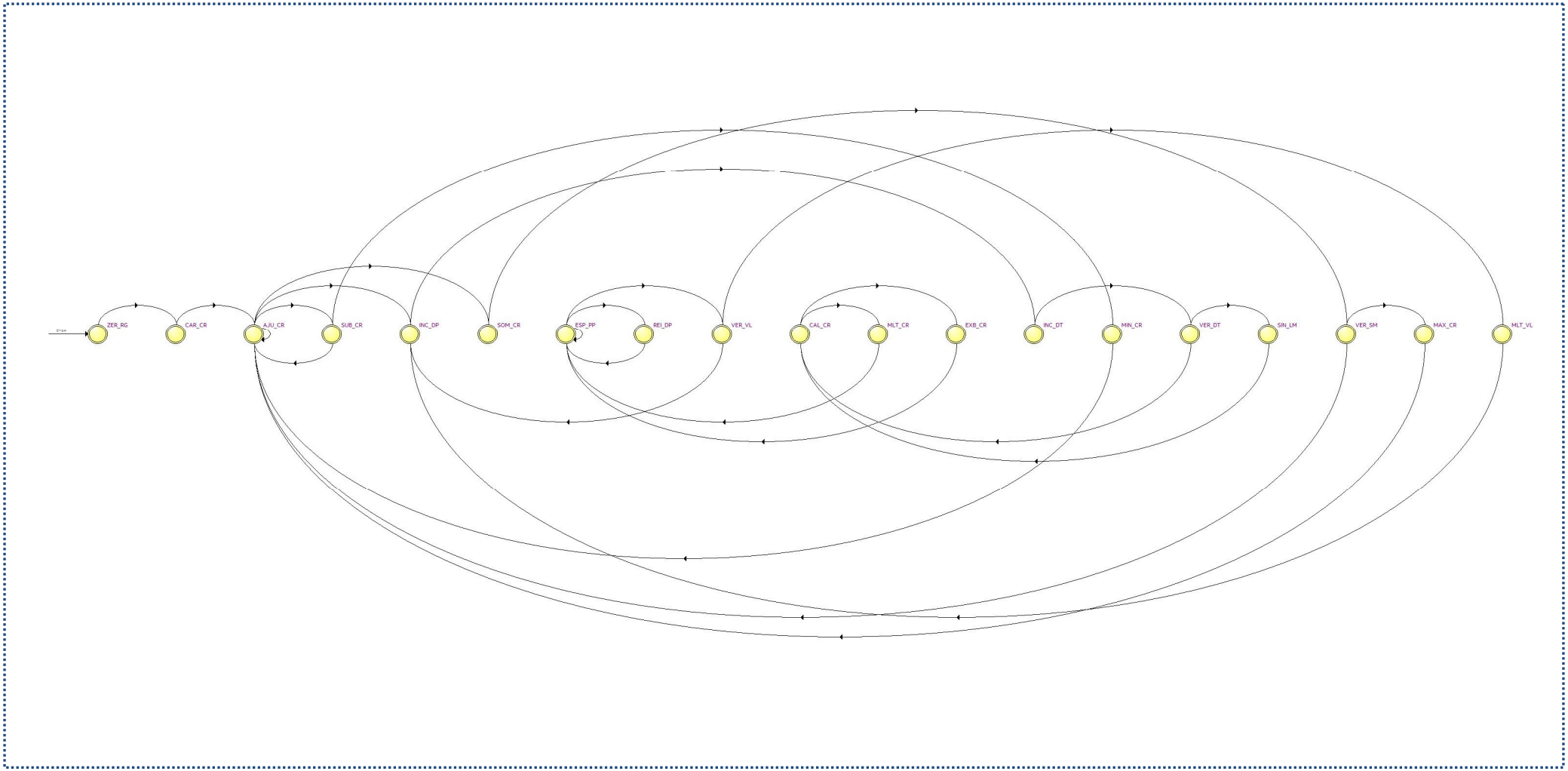
```
1   -- Quartus Prime VHDL FEI DIGITAL REPORT: Projeto Completo do Pedágio
2
3   LIBRARY IEEE;
4   USE IEEE.STD_LOGIC_1164.ALL;
5
6   ENTITY PEDAGIO_COMPLETO IS
7      PORT(
8         PP, IC, DC, CK, RP, MO, RT : IN STD_LOGIC;
9         IT_OUT, FT_OUT             : OUT STD_logic;
10        MEV, MFC, SDM              : OUT STD_LOGIC;
11        CR, DP, DT                 : OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
12     );
13  END PEDAGIO_COMPLETO;
14
15  ARCHITECTURE ESTRUTURAL OF PEDAGIO_COMPLETO IS
16
17  COMPONENT FD
18     PORT(
19        LDA, LDB, LDC, IT, CK   : IN STD_LOGIC;
20        S_MXA, S_MXB            : IN STD_LOGIC_VECTOR (2 DOWNTO 0);
21        S_ULA                   : IN STD_LOGIC_VECTOR (1 DOWNTO 0);
22        ZR, PO, NG, ES, FT      : OUT STD_LOGIC;
23        CR, DP, DT              : BUFFER STD_LOGIC_VECTOR (7 DOWNTO 0)
24     );
25  END COMPONENT;
26
27  COMPONENT UC
28     PORT(
29        CK : in std_logic;            -- clock de 50MHz
30        RT : in std_logic;            -- reinicio total -> ativo em zero
31  -- Entradas Externas:
32        IC : in std_logic;         -- incrementa credito -> ativo em um
33        DC : in std_logic;         -- decrementa credito -> ativo em um
34        RP : in std_logic;         -- reinicio parcial -> ativo em um
35        PP : in std_logic;         -- passagem pelo portico -> ativo em um
36        MO : in std_logic;         -- modo de operaÃ§Ã£o -> 1= Normal, 0= Ajusta Valor
37  -- Sinais de Estado da ULA:
38        ZR : in std_logic;         -- resultado zero na operaÃ§Ã£o da ULA
39        PO : in std_logic;         -- resultado positivo na operaÃ§Ã£o da ULA
40        NG : in std_logic;         -- resultado negativo na operaÃ§Ã£o da ULA
41        ES : in std_logic;         -- resultado da operaÃ§Ã£o da ULA maior que 255
42  -- Sinais de Estado do TIMER:
43        FT : in std_logic;         -- fim da temporizaÃ§Ã£o de 1 segundo
44  -- Sinais de Saida para MUX:
45        Sel_mxa : out std_logic_vector(2 downto 0);  -- seleciona entrada de MUX_A
46        Sel_mxb : out std_logic_vector(2 downto 0);  -- seleciona entrada de MUX_B
47  -- Sinais de Saida para ULA:
48        Sel_ula : out std_logic_vector(1 downto 0);  -- seleciona operaÃ§Ã£o da ULA
49  -- Sinais de Saida para Registradores:
50        Lda   : out std_logic;     -- carrega RA
51        Ldb   : out std_logic;     -- carrega RB
52        Ldc   : out std_logic;     -- carrega RC
53  -- Sinais de Saida para TIMER:
54        IT    : out std_logic;     -- inicia temporizaÃ§Ã£o
55  -- Sinais de Saida Externos:
56        MEV   :  out std_logic;    -- sinaliza multa por excesso de velocidade
57        MFC   :  out std_logic;    -- sinaliza multa por falta de credito
58        SDM   :  out std_logic     -- sinaliza limite de distÃ¢ncia total percorrida
59     );
60  END COMPONENT;
61
62  SIGNAL   ZR             : std_logic;
63  SIGNAL   PO             : std_logic;
64  SIGNAL   NG             : std_logic;
65  SIGNAL   ES             : std_logic;
66  SIGNAL   LDA            : std_logic;
67  SIGNAL   LDB            : std_logic;
68  SIGNAL   LDC            : std_logic;
69  SIGNAL   IT             : std_logic;
70  SIGNAL   FT             : std_logic;
71  SIGNAL   smxa           : std_logic_vector(2 downto 0);
72  SIGNAL   smxb           : std_logic_vector(2 downto 0);
73  SIGNAL   sula           : STD_LOGIC_VECTOR (1 DOWNTO 0);
74
75  BEGIN
76     UC1      : UC        PORT MAP (CK=>CK, RT=>RT, IC=>IC, DC=>DC, RP=>RP ,PP=>PP ,MO=>MO ,ZR=>ZR,PO
77     FD1      : FD        PORT MAP (LDA=>LDA ,LDB=>LDB ,LDC=>LDC ,IT=>IT ,CK=>CK ,S_MXA=>smxa ,S_MXB=
78     IT_OUT<= IT;
79     FT_OUT<= FT;
80  END ESTRUTURAL;
81
```

```
-- Quartus Prime VHDL FEI DIGITAL REPORT: Projeto Completo do Pedágio e Periféricos

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY PEDAGIO_TOTAL IS
    PORT(

        KEY0, KEY1, CLK1_50, SW0, SW1, SW9        : IN STD_LOGIC;
        LED0, LED1, LED2, LED3                    : OUT STD_LOGIC;
        NCS, SDI, SCLK                            : OUT STD_LOGIC;
        SDO                                       : IN STD_LOGIC;
        HEX0, HEX1, HEX2, HEX3, HEX4, HEX5        : OUT STD_LOGIC_VECTOR (6 DOWNTO 0)
    );
END PEDAGIO_TOTAL;

ARCHITECTURE ESTRUTURAL OF PEDAGIO_TOTAL IS

COMPONENT GERADOR_PP
    PORT(
        CLK, RST: IN STD_LOGIC;         -- sinais de controle
        PP : BUFFER STD_LOGIC;          -- puso de pórtico
        LP : BUFFER STD_LOGIC;          -- led de sinalização de frequência
        SDO: IN STD_LOGIC;              -- entrada de dados do sensor ADXL345
        SCLK,SDI,NOT_CS: OUT STD_LOGIC--sinais de comunicação com sensor ADXL345
        );
END COMPONENT;

COMPONENT SINCRON
    PORT(
            CK        : IN STD_LOGIC;    -- clock de 50MHz
            RST       : IN STD_LOGIC;    -- reset (ativo em zero)
            KEY       : IN STD_LOGIC;    -- botao de entrada (ativo em zero)
            SINC      : OUT STD_LOGIC    -- pulso de sinal de saida (ativo em um)
        );
END COMPONENT;

COMPONENT debouncing
    PORT(
        ck        : in std_logic;              -- periodo de referencia
        rst       : in std_logic;              -- iniciacao da contagem
        ch        : in std_logic;              -- sinal com oscilacao
        ch        : in std_logic;              -- sinal com oscilacao
        ch_db     : out std_logic              -- sinal sem oscilacao
        );
END COMPONENT;

COMPONENT DECODIFICADOR_BCD
    PORT(
        Q  : IN STD_LOGIC_VECTOR(7 DOWNTO 0); -- vetor de 8 bits de entrada
        D, U : OUT STD_LOGIC_VECTOR(6 DOWNTO 0) -- vetores dezena e unidade
        );
END COMPONENT;

COMPONENT PEDAGIO_COMPLETO
    PORT(
        PP, IC, DC, CK, RP, MO, RT : IN STD_LOGIC;
        IT_OUT, FT_OUT             : OUT STD_logic;
        MEV, MFC, SDM              : OUT STD_LOGIC;
        CR, DP, DT                 : OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
        );
END COMPONENT;

SIGNAL   PP             : std_logic;
SIGNAL   IC             : std_logic;
SIGNAL   DC             : std_logic;
SIGNAL   CK             : std_logic;
SIGNAL   RP             : std_logic;
SIGNAL   MO             : std_logic;
SIGNAL   FT             : std_logic;
SIGNAL   IT             : std_logic;
SIGNAL   CR             : std_logic_vector(7 downto 0);
SIGNAL   DP             : std_logic_vector(7 downto 0);
SIGNAL   DT             : STD_LOGIC_VECTOR(7 DOWNTO 0);

BEGIN
    GER_PP      : GERADOR_PP          PORT MAP (CLK=>CLK1_50 ,RST=>SW9 ,PP=>PP ,LP=>LED0 ,SDO=>SDO ,S
    SINC1       : SINCRON             PORT MAP (CK=>CLK1_50 ,RST=>SW9 ,KEY=>KEY0 ,SINC=>IC );
    SINC2       : SINCRON             PORT MAP (CK=>CLK1_50 ,RST=>SW9 ,KEY=>KEY1 ,SINC=>DC );
    DEB1        : debouncing          PORT MAP (ck=>CLK1_50 ,rst=>SW9 ,ch=>SW0 ,ch_db=>RP );
    DEB2        : debouncing          PORT MAP (ck=>CLK1_50 ,rst=>SW9 ,ch=>SW1 ,ch_db=>MO );
    DEC_HEX10   : DECODIFICADOR_BCD   PORT MAP (Q=>CR ,D=>HEX1 ,U=>HEX0 );
    DEC_HEX32   : DECODIFICADOR_BCD   PORT MAP (Q=>DP ,D=>HEX3 ,U=>HEX2 );
    DEC_HEX54   : DECODIFICADOR_BCD   PORT MAP (Q=>DT ,D=>HEX5 ,U=>HEX4 );
    ME1         : PEDAGIO_COMPLETO    PORT MAP (PP=>PP ,IC=>IC ,DC=>DC ,CK=>CLK1_50 ,RP=>RP ,MO=>MO ,

END ESTRUTURAL;
```

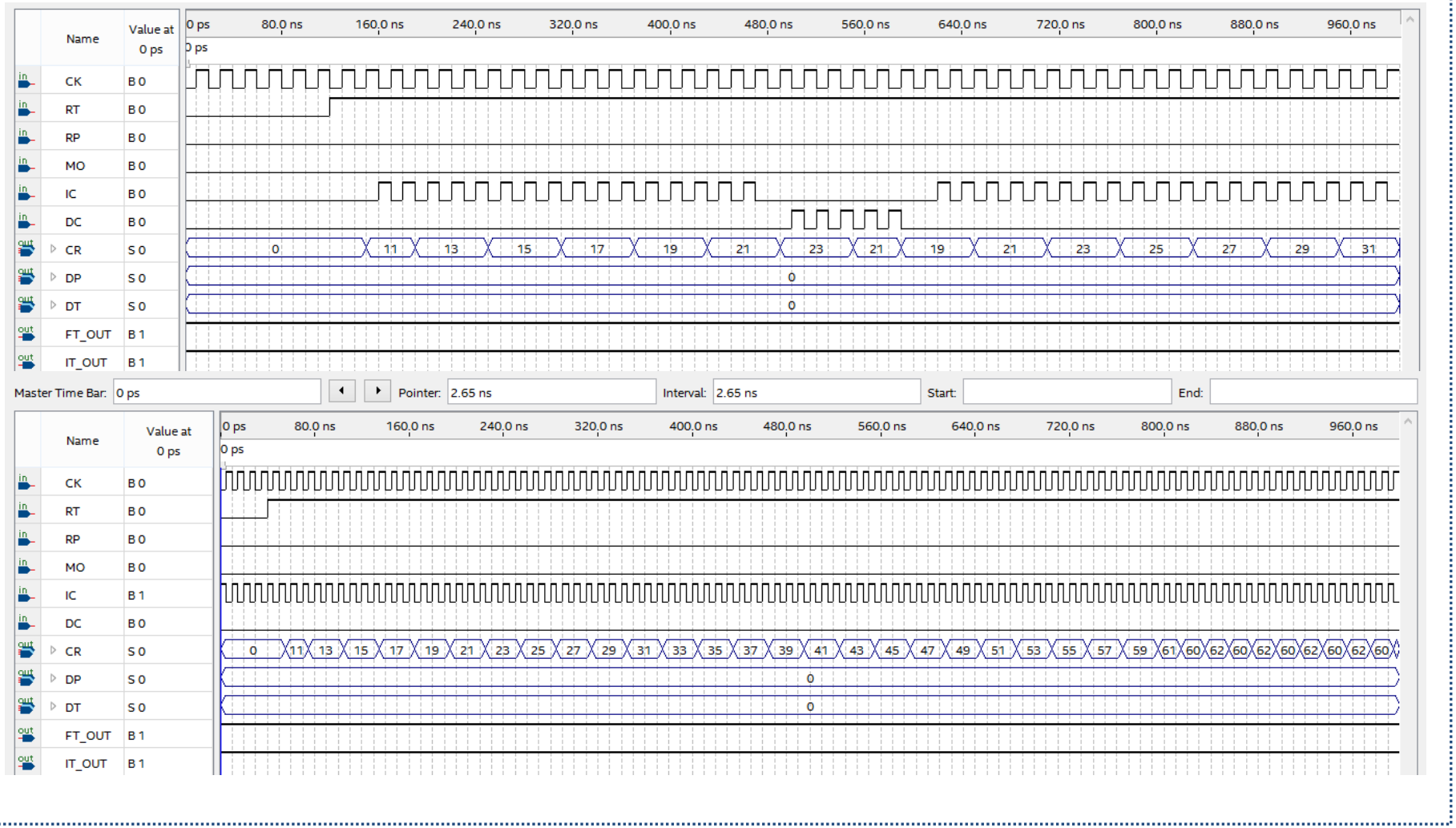
7. DIAGRAMA RTL DO PROJETO COMPLETO E PERIFÉRICOS



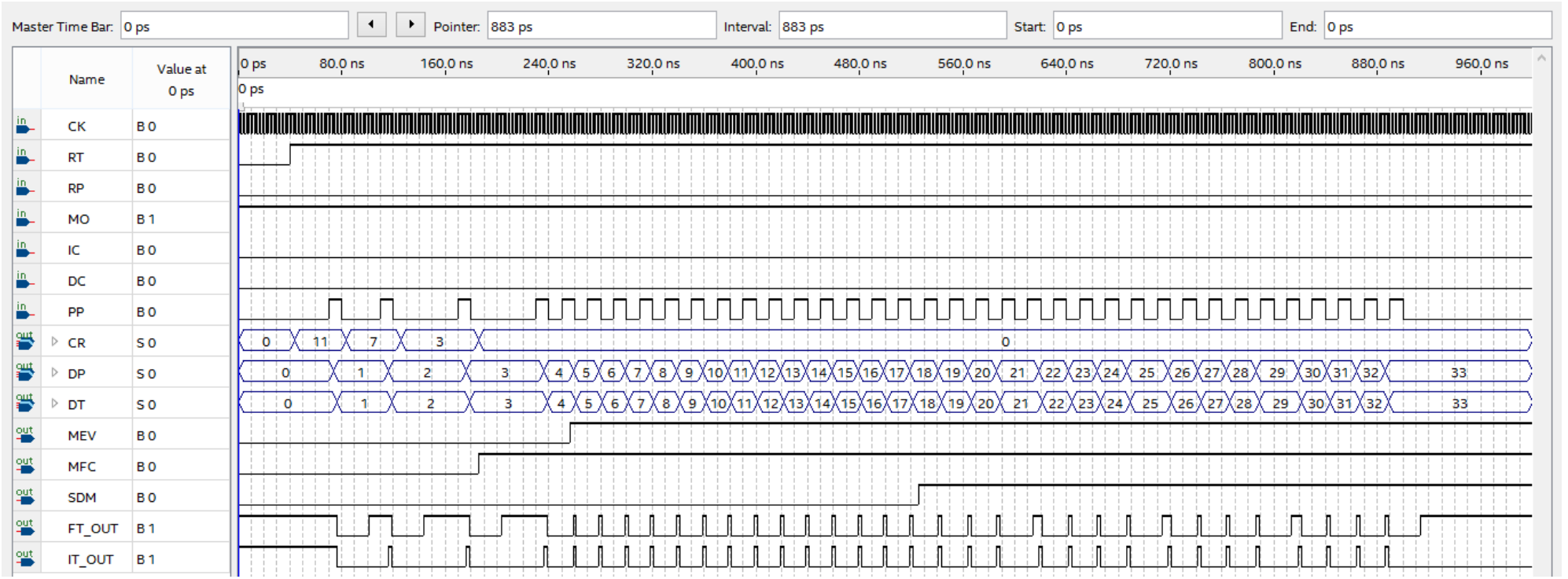
8. DIAGRAMA DE ESTADOS DA UC GERADO PELO QUARTUS PRIME



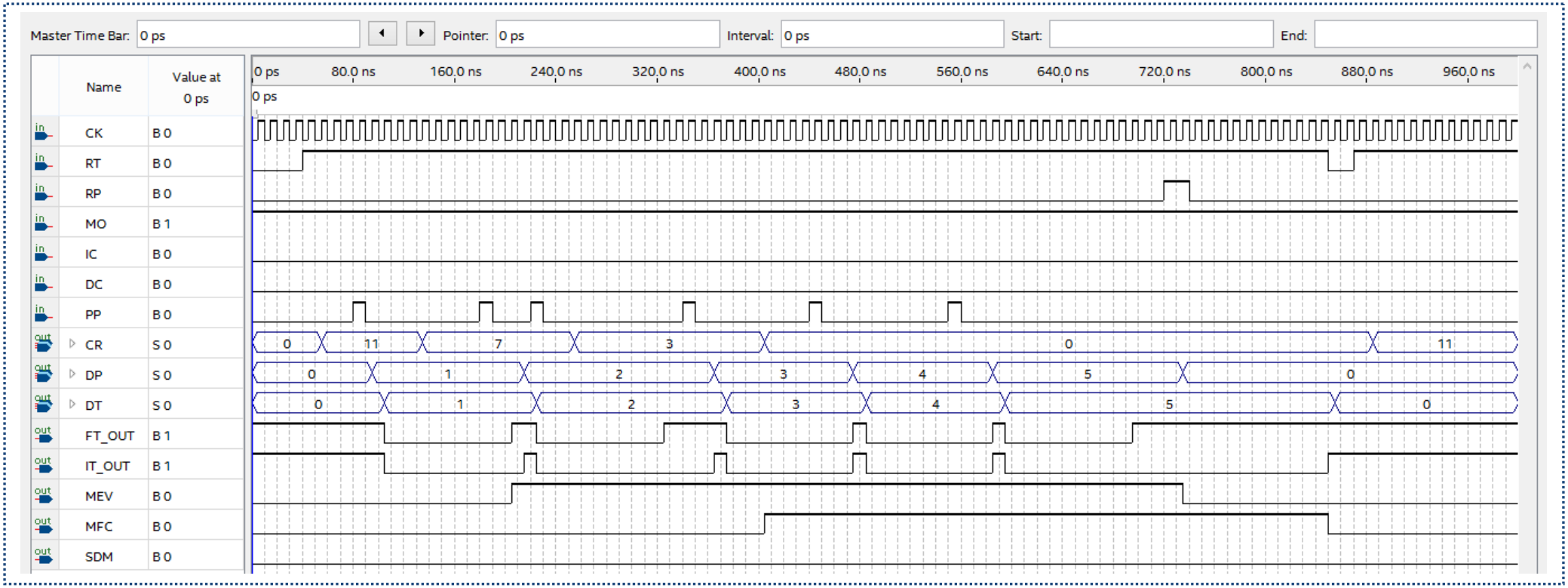
9. SIMULAÇÃO FUNCIONAL DO INCREMENTO DE CRÉDITO



10. SIMULAÇÃO FUNCIONAL DA SINALIZAÇÃO DA DISTÂNCIA MÁXIMA PERCORRIDA



11. SIMULAÇÃO FUNCIONAL DA UC + FD (MULTAS E SETUP DO SISTEMA POR REINÍCIO)



12.Conclusão

O objetivo geral deste projeto 1 do Laboratório de Sistemas Digitais II foi alcançado, no qual desenvolvemos um SCADA de Percurso de Pedágio Aberto. Para tanto, foi iniciado o desenvolvimento pela uma máquina de estado finita (FSM) e a lógica necessária para comando do sinal de abertura do SCADA em um FPGA, utilizando a linguagem de descrição VHDL implementada no software Quartus Prime Lite Edition 16.1 e o método de projeto RTL, bem como o projeto de uma Unidade de Controle e Fluxo de Dados. Os parâmetros de para as Multas, Limite de Distância Percorrida, bem como outros parâmetros para a condição estabelecida para o projeto (Requisitos) é baseada no número de matrícula (R.A) descrito no cabeçalho deste documento.

Esse projeto de implementação também alcançou objetivos exercitar a metodologia de desenvolvimento de projetos de engenharia apoiada em computador (CAE) onde todo o desenvolvimento do software embarcado e modelo final será feito no software Quartus Prime Lite Edition 16.1 onde será feita a simulação funcional e interface final do sistema de controle digital totalmente implementada utilizando o FPGA da família MAX 10 (modelo: 10M50DAF484C7G) existente na placa de desenvolvimento Altera DE10-Lite. Esse processo envolve a compreensão do problema, seu planejamento, desenvolvimento da solução lógica, integração dos subsistemas, implementação no ambiente computacional, simulação, testes, depuração do projeto, implementação física e registro dos resultados.

Através dos itens que correspondem à visão RTL dos principais circuitos, foi comprovada a complexidade da implementação e como a metodologia de Fluxo de Dados, uso de Registradores, bem como a ULA e o MUXl são facilitadores para a visão de implementação do projeto. A conformidade do Sistema Digital para o Cofre de Segurança foi garantida através da simulação via University UWF (Wave Form) realizada no software.