centro universitário FEI

CURSO DE ENGENHARIA ELÉTRICA

EL6720 / NE7720 – LABORATÓRIO DE SISTEMAS DIGITAIS II

PROJETO 2 - 2° SEMESTRE DE 2017

Projeto 2 – Semáforo Adaptativo

Nome: Renato Bednarski Ramos
RA: 12.214.549-3
Professor: Valter Avelino
Disciplina: Sistemas Digitais II (NE7720)
Turma de laboratório: 615 – Noturno
Data entrega: 17/11/2017

Descrição do Projeto

No projeto 2 de Sistemas Digitais II, será ser desenvolvido um sistema digital de controle de um semáforo adaptativo em um cruzamento de via de transporte em uma via primária e secundária, ou seja, o semáforo deverá se adaptar às condições do trânsito. Será necessário o desenvolvimento de uma estrutura RTL (com Unidade de Controle e Unidade de Fluxo de Dados) em um FPGA, utilizando a linguagem de descrição VHDL. Será utilizado o Kit de desenvolvimento didático Altera DE-115.

As situações que alteram o funcionamento dos semáforos são:

- caso haja carro na via secundária, o tempo em que a lâmpada verde do semáforo primário (TP) é reduzida (tempo de redução: TR) para haver melhor fluidez no trânsito;
- caso haja pedestre querendo atravessar a rua, ele aperta o botão de pedestre (BP) que, no momento certo do ciclo, irá deixar os semáforos das vias primária e secundária vermelhos e o verde de pedestre aceso pelo tempo pré-determinado (TT).

O tempo de redução TR é definido inicialmente por TR0, mas pode ser programado para outro valor, se desejado. A condição de TR é ser menor que TP e maior que zero.

Os valores de TR0, TT, TP e TS são definidos de acordo com os últimos 4 algarismos do RA, no meu caso: 12.214.**549-3**

Nome	Símbolo	Condição	Valor (segundos)
Tempo de Verde Principal	TP	O dígito de maior valor	**9**
Tempo de Verde Secundário	TS	O segundo dígito de maior valor	**5**
Tempo de Verde de Travessia de Pedestre	TT	O terceiro dígito de maior valor	**4**
Tempo de Redução de Verde Principal	TR0	O menor dígito (maior que zero)	**3**

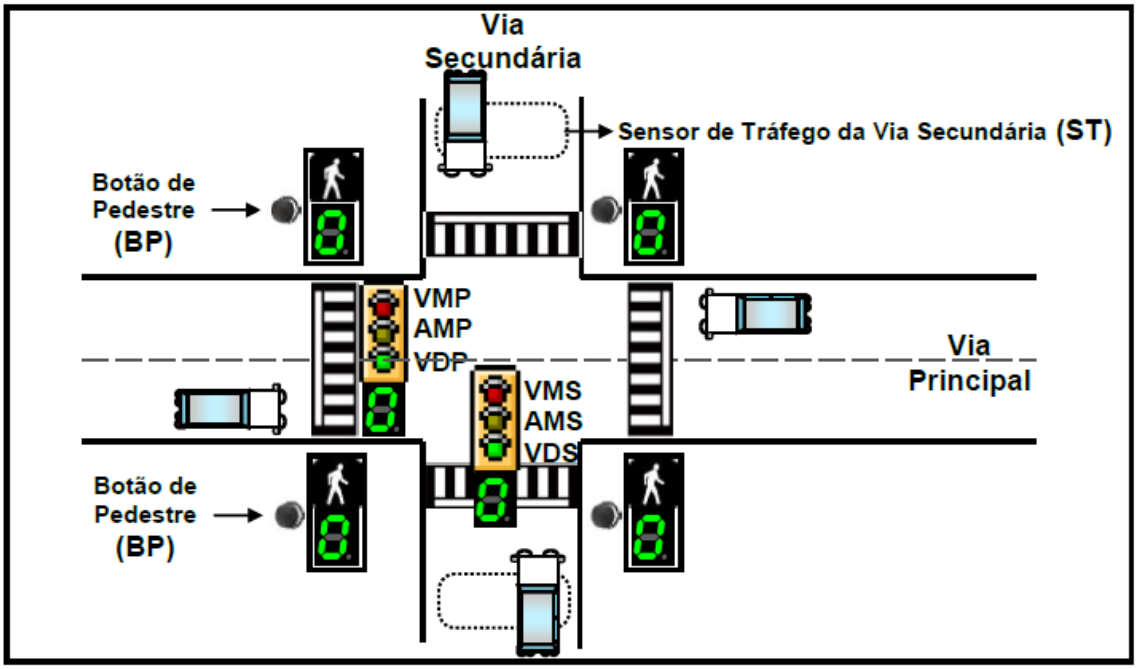


Figura 1: Diagrama esquemático do entroncamento das vias principal e secundária

Interface:

A interface com o usuário será feita por meio de:

- 1 chave para solicitar programação do TR (sinal **PGR**) – **SW17** (acionado em nível lógico 1);

- 1 chave para simular o botão de pedestre (sinal **BP**) – **SW16** (acionado em nível lógico 1);

- 1 chave para simular o sensor de tráfego na via secundária (sinal **ST**) – **SW15** (acionado em nível lógico 1);

- 1 botão de decrementa para programação do TR (sinal **DC**) – **KEY1** (acionado em nível lógico 1);

- 1 botão de incrementa para programação do TR (sinal **IC**) – **KEY2** (acionado em nível lógico 1);

- 1 botão de reset do sistema (sinal **INI**) – **KEY0** (acionado em nível lógico 0);

- 1 display de 7 segmentos de contagem decrescente do tempo faltante de verde do primário (sinal **TFP**) – **HEX4** (segmentos acionados em nível lógico 0);

- 1 display de 7 segmentos de contagem decrescente do tempo faltante de verde do secundário (sinal **TFS**) – **HEX5** (segmentos acionados em nível lógico 0);

- 1 display de 7 segmentos de contagem decrescente do tempo faltante de verde de travessia (sinal **TFT**) – **HEX6** (segmentos acionados em nível lógico 0);

- 1 display de 7 segmentos que mostra o valor de tempo de redução (sinal **TR**) – **HEX7** (segmentos acionados em nível lógico 0);

- 1 led para estado do reset – **LEDG0** (led acionado em nível lógico 1, inverso de **INI**);

- 1 led para estado da programação – **LEDR17** (led acionado em nível lógico 1);

- 1 led para estado do botão de pedestre – **LEDR16** (led acionado em nível lógico 1);

- 1 led para estado do sensor de trânsito – **LEDR17** (led acionado em nível lógico 1);

- 2 leds para estado do farol vermelho primário (**VMP**) – **LEDR3 e LEDR2** (leds acionados em nível lógico 1);

- 2 leds para estado do farol vermelho secundário (**VMS**) – **LEDR1 e LEDR0** (leds acionados em nível lógico 1);

- 1 led para estado do farol amarelo primário (**AMP**) – **LEDG7** (led acionado em nível lógico 1);

- 1 led para estado do farol amarelo secundário (**AMS**) – **LEDG4** (led acionado em nível lógico 1);

- 2 leds para estado do farol verde primário (**VDP**) – **LEDG6 e LEDG5** (leds acionados em nível lógico 1);

- 2 leds para estado do farol verde secundário (**VDS**) – **LEDG3 e LEDG2** (leds acionados em nível lógico 1);

- 1 led para estado do farol vermelho de travessia (**VMT**) – **LEDGR11** (led acionado em nível lógico 1);

- 1 led para estado do farol verde de travessia (**VDT**) – **LEDG8** (led acionado em nível lógico 1).

Sequenciamento:

A sequência de operação das lâmpadas semafóricas deve iniciar com o semáforo da via principal acendendo a lâmpada verde principal (**VDP** = **1**) por um intervalo de tempo principal (**TP**) enquanto todos os outros semáforos estão em vermelho (**VMS** e **VMT** = **1**). A seguir a lâmpada amarela do semáforo principal deve ser acesa (**AMP** = **1**) por uma unidade de intervalo de tempo. A seguir aciona-se a lâmpada verde da via secundária (**VDS** = **1**) enquanto todos os outros semáforos estão em vermelho (**VMP** e **VMT** = **1**) por um intervalo de tempo secundário (**TS**). Na sequência é acionada a lâmpada amarela secundária (**MAS** = **1**) por uma unidade de intervalo de tempo. Após o amarelo secundário reinicia-se a sequência com o verde principal (**VDP=1**).

Quando o sistema está na fase verde principal existe um display (**HEX4**) que mostra, em uma contagem decrescente, o tempo faltante de verde principal (**TFP**) para troca de fase. O mesmo é aplicável para o tempo faltante de verde secundário (**TFS**) (**HEX5**).

Para iniciar o sistema de controle deve existir um botão **INI** que, quando ativado (em nível lógico zero), deve levar o controle para o estado inicial de modo assíncrono. Nesse estado o sistema deve iniciar todos os registros do sistema e manter as lâmpadas vermelhas acesas;

Além do ciclo básico de sequenciamento, o controlador deve considerar os seguintes modos de operação:

1. Ativação do sensor de tráfego: quando o controle entrar na fase de verde da via principal se o sensor de tráfego estiver ativado (**ST=1**) então o intervalo de tempo principal (**TP**) deve ser diminuído de um valor de tempo de redução do verde principal (**TR**), de modo que o tempo de verde dessa via passa a ser um tempo principal reduzido (**TPR,** onde:**TPR = TP - TR**). O valor de **TR** é pré-definido (**TR0**), mas pode ser alterado pelo usuário autorizado (agente de trânsito) ao iniciar-se o sistema no modo de programação;

2. Modo de programação da temporização: quando o sistema for iniciado se a chave de programação estiver ativada (**PGR=1**) ativa-se o modo programação, que permite

ao usuário ajustar o parâmetro do tempo de redução do verde principal a partir de um valor inicial (**TR0**). No modo de programação a alteração do valor do parâmetro **TR** é executada ao pressionar o botão incrementa (**IC=1**) ou o botão decrementa (**DC=1**). A cada acionamento do botão **IC** o valor armazenado em **TR** é aumentado de uma unidade de tempo. Para cada acionamento do botão **DC** o valor armazenado em **TR** é decrementado de uma unidade de tempo. A lógica do sistema **deve garantir** que o valor ajustado de **TR** seja **menor** que **TP** e **maior** que zero (**TP** > **TR > 0)**. O acionamento simultâneo dos botões **IC** e **DC** não deve alterar o valor selecionado. O acionamento dos botões **IC** e **DC** fora do modo de programação não deve alterar a operação do sistema. Durante o modo programação as lâmpadas vermelhas de **ambos os semáforos ficam acesas** (**VMP=1** e **VMS=1**);

3. Solicitação de travessia de pedestre: normalmente a lâmpada vermelha de travessia do sinal de pedestre está acesa (**VMT=1**). A qualquer momento que ocorrer o acionamento de um dos botões de pedestre (**BP=1**) o sistema deve memorizar esse comando como uma solicitação de pedestre (**SP=1**) e atendê-la depois que o controlador entrar na fase de amarelo principal. Para atender à solicitação do pedestre as lâmpadas vermelhas de **ambos os semáforos devem ficar acesas** (**VMP=1** e **VMS=1**) e acende-se a lâmpada verde de travessia do sinal de pedestre (**VDT=1**) por um tempo de verde de travessia (**TT**). Um display ao lado do sinal de pedestre mostra a contagem decrescente do tempo faltante de verde de travessia (**TFT**). Uma vez encerrado o tempo de travessia, a memória de solicitação de pedestre **deve ser resetada** (**SP**= **0**) e o ciclo do semáforo retorna ao normal na fase de verde secundário.

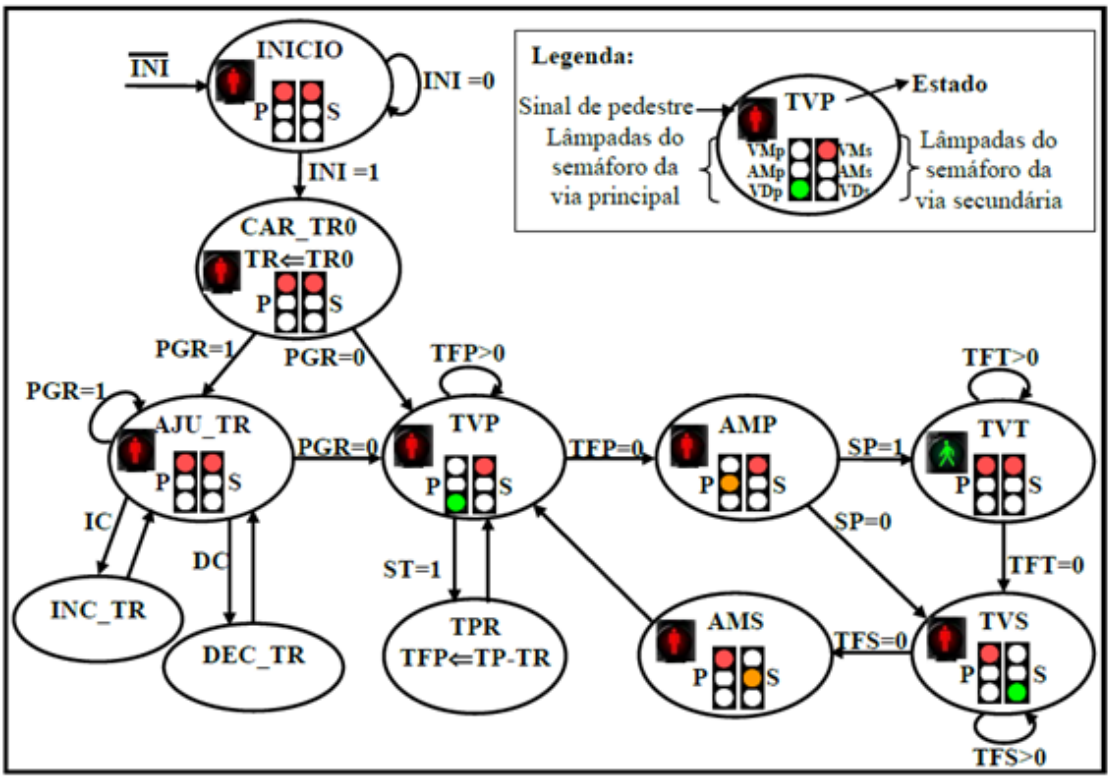


Figura 2: Diagrama de transição de estados do ciclo básico do controlador de semáforo adaptativo.

Descrição da Realização do Projeto

O projeto será desenvolvido com os seguintes componentes da figura:

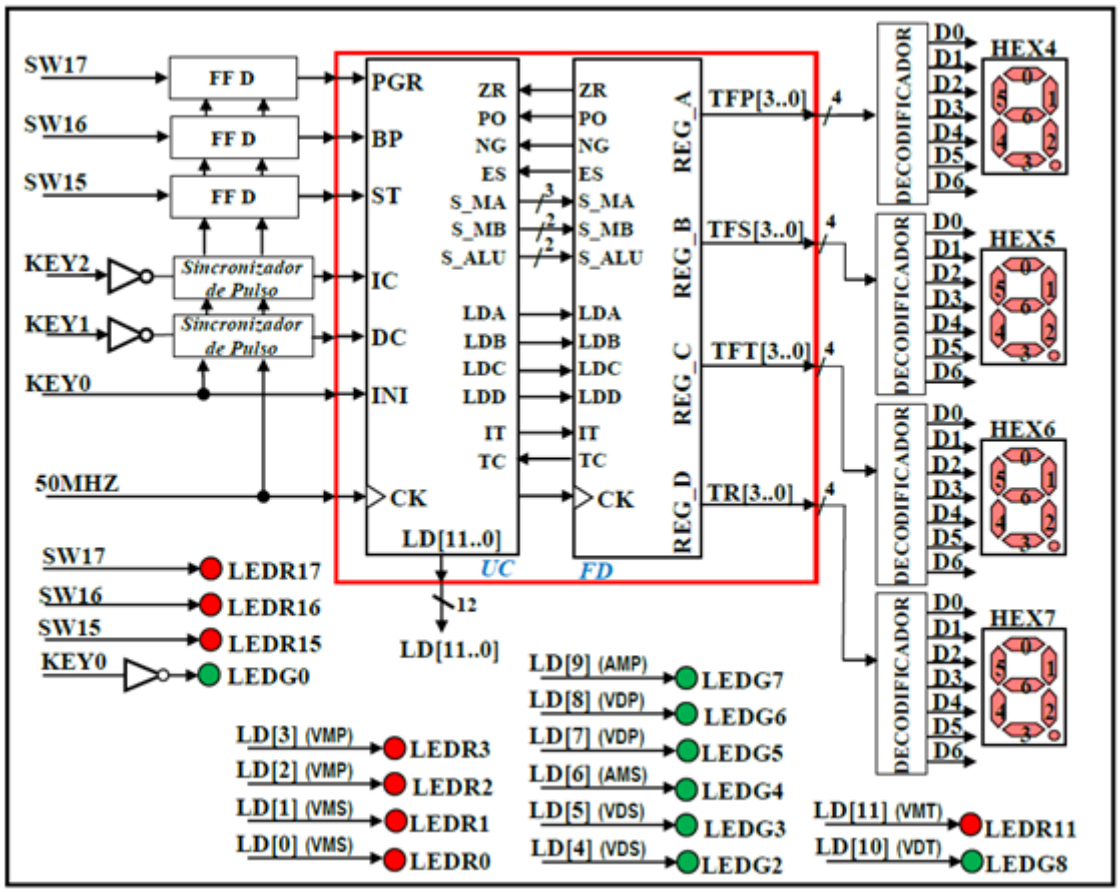


Figura 3 : Diagrama blocos do sistema controlador de semáforo adaptativo.

<u>Fluxo de Dados (FD):</u>

Sendo assim, o FD será composta por 2 multiplexadores (**MUX A** 8x1 de 4 bits e **MUX B** 4x1 de 4 bits), **ULA** de 2 canais de 4 bits e 4 operações, 4 registradores de 4 bits (**REG_A**, **REG_B**, **REG_C** e **REG_D** que registrarão, respectivamente, **TFP**, **TFS**, **TFT** e **TR**) e 1 **TIMER**.

Observar os blocos operacionais do FD na figura seguinte:

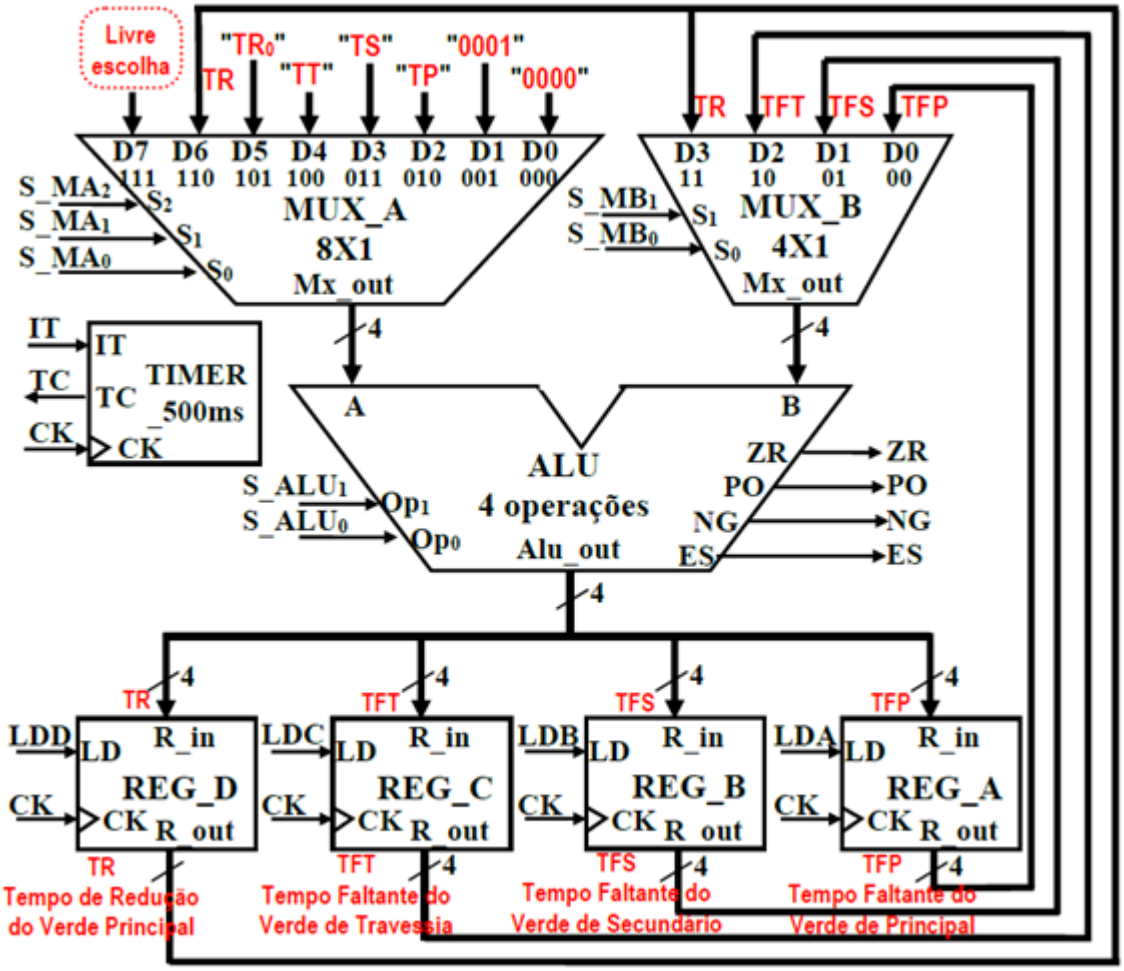
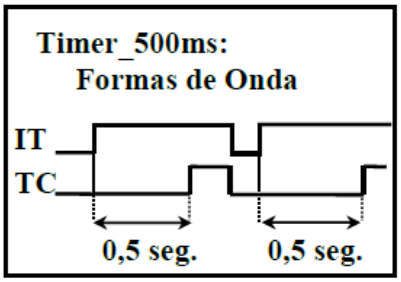


Figura 4: Diagrama dos blocos operacionais do sistema

Tabela de funções / sinalização de status da ALU e operação do Timer:

Código (Op_1 Op_0)	Operação da ALU
0 0	Passa A
0 1	NOT(A)
1 0	B + A
1 1	B – A

Sinal	Sinalização da ALU
ZR = 1	Resultado = zero
PO = 1	Resultado > zero
NG = 1	Resultado < zero
ES = 1	Resultado > 15



Registradores:

São a memória do sistema, será gravado o dado que sair da **ULA** se o registrador estiver habilitado (**LDD**, **LDC**, **LDB** e **LDA** são os sinais vindos da **Unidade de Controle** que habilitam, respectivamente, **REG_D**, **REG_C**, **REG_B** e **REG_A**, em nível lógico 1). As saídas dos registradores são ligados nas entradas do **MUX_B** para poder fazer operações na **ULA**. As saídas também são ligadas a um decodificador que é ligado a um display de 7 segmentos para mostrar o valor dos dados (no caso, um display para cada registrador).

O arquivo VHDL foi disponibilizado pronto pelo professor, pelo moodle.

ULA:

A ULA é o operador aritmético dos dados presentes nos multiplexadores **MUX_A** e **MUX_B**.

Os dados entram por **A** (vindos do **MUX_A**) e **B** (vindos do **MUX_B**) e os comandos sobre qual operação que a **ULA** deve fazer é feito pelas entradas **S_ALU1** e **S_ALU0**, cujos sinais vêm da **UC** (ativos em nível lógico 1).

O resultado da operação sai pela saída **Alu_out** que é ligado aos 4 registradores.

As saídas **ZR**, **PO**, **NG** e **ES** (nível lógico 1 significa que está ativo) são informações (verificar tabela da página anterior) oferecidas pela **ULA** sobre o status do resultado da operação atual. Essas informações vão para a **UC** para controle do sistema digital.

O arquivo VHDL foi disponibilizado pronto pelo professor, pelo moodle.

Timer:

Quando a entrada IT do timer é ativa pela UC (ativa em nível lógico 1) a saída TC se mantém em nível lógico zero até completar 25 milhões de ciclos de clock (500ms), quando TC muda de estado para nível lógico 1. Essa saída é ligada à UC para controle do sistema digital.

O arquivo VHDL foi disponibilizado pronto pelo professor, pelo moodle.

Multiplexadores:

- MUX_B

O **MUX_B** possui as informações presentes nos 4 registradores do bloco do **FD**, conforme a figura.

O canal é selecionado pelas entradas **S_MB1** e **S_MB0**, cujos sinais vêm da **UC** (ativos em nível lógico 1).

A saída dos dados **Mx_out** é ligada à entrada **B** da **ULA**.

O arquivo VHDL foi disponibilizado pronto pelo professor, pelo moodle.

- MUX_A

O **MUX_A**, conforme a figura, possui as informações de **TR**, **TR0**, **TT**, **TS**, **TP**, **0b0001** e **0b0001.**

O canal é selecionado pelas entradas **S_MA1** e **S_MA0**, cujos sinais vêm da **UC** (ativos em nível lógico 1).

A saída dos dados **Mx_out** é ligada à entrada **A** da **ULA**.

O arquivo VHDL foi disponibilizado pronto pelo professor, pelo moodle, porém foi necessário editá-lo, para a inclusão dos dados customizados de meu RA (**TR0** = 3, **TT** = 4, **TS** = 5 e **TP** = 9) e dos dados **0b0000** e **0b0001**.

VHDL:

```
library ieee;
use ieee.std_logic_1164.all;
entity MUX_A is
   port(TR     : in std_logic_vector(3 downto 0);
        S      : in std_logic_vector(2 downto 0);
        Mx_out : out std_logic_vector(3 downto 0)
);
end MUX_A;
architecture comportamental of MUX_A is
begin
process(TR, S)
begin
case S is
   when "000" => Mx_out <= "0000";
   when "001" => Mx_out <= "0001";
   when "010" => Mx_out <= "1001";
   when "011" => Mx_out <= "0101";
   when "100" => Mx_out <= "0100";
   when "101" => Mx_out <= "0011";
   when "110" => Mx_out <= TR;
   when "111" => Mx_out <= "0000";
   when others => null;
end case;
end process;
end comportamental;
```

Unidade de Controle (UC):

A **UC** é a responsável por controlar todos os acionamentos dos enables dos **registradores**, operação que a **ULA** deve fazer, interpretar os sinais **ZR**, **PO**, **NG** e **ES** que saem da **ULA**, acionar e verificar os pulsos do **Timer**, interpretar todos os **botões** e **chaves** de entrada, controlar os canais dos **multiplexadores** e controlar os acionamentos de todos os **leds** (lâmpadas dos semáforos).

Diagrama de Transição de Estados

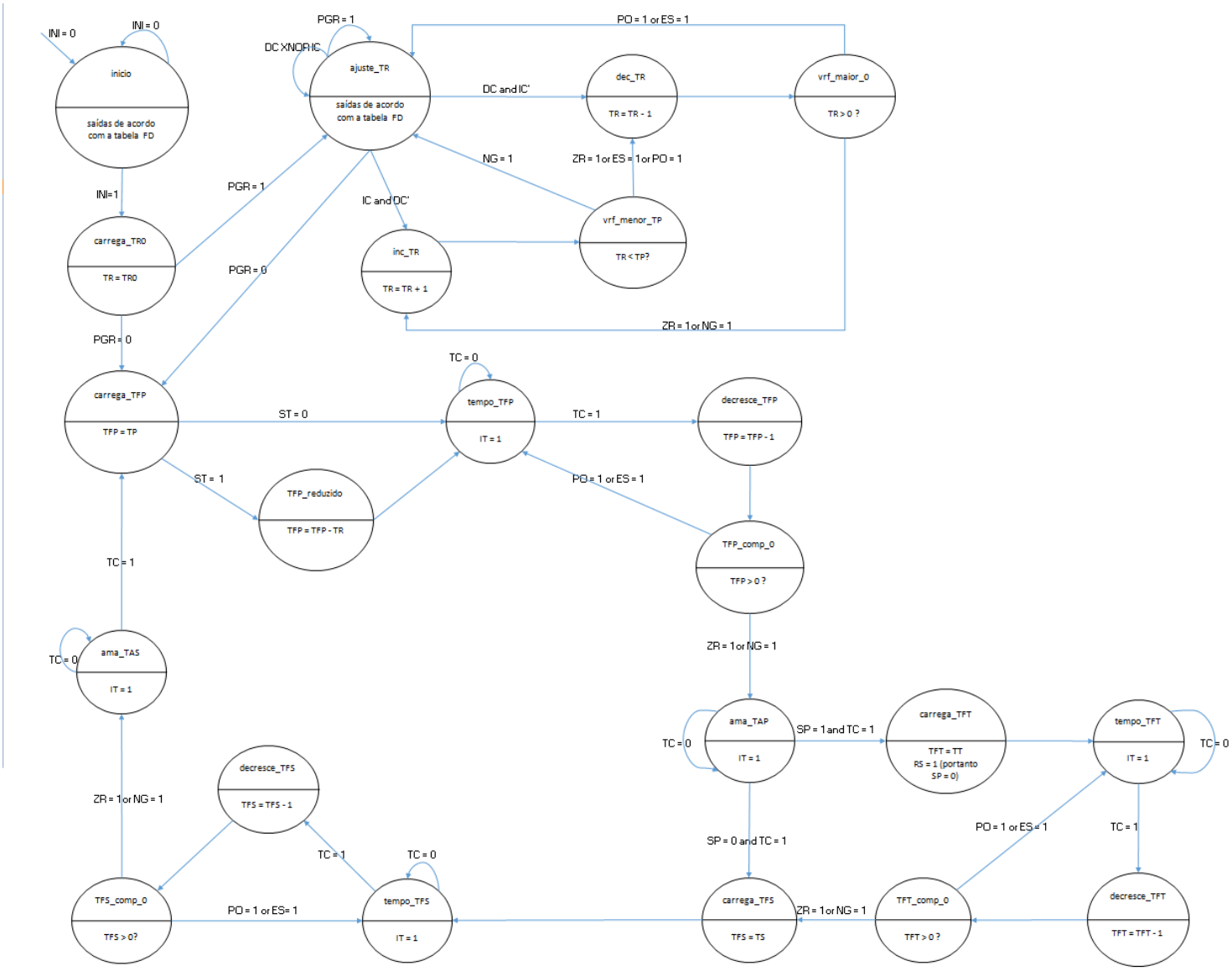


Tabela do FD originado pelo Diagrama de Estados

ESTADO	O QUE FAZ	S_MA	S_MB	S_ULA	LDD	LDC	LDB	LDA	IT	led[11..0]
inicio	RESETA	**000**	**XX**	**00**	**1**	**1**	**1**	**1**	**0**	**100000001111**
carrega_TR0	Carrega TR0 em TR	**101**	**XX**	**00**	**1**	**0**	**0**	**0**	**0**	**100000001111**
ajuste_TR	Ajusta o valor de TR	**XXX**	**XX**	**XX**	**0**	**0**	**0**	**0**	**0**	**100000001111**
inc_TR	Incrementa 1 no valor de TR	**001**	**11**	**10**	**1**	**0**	**0**	**0**	**0**	**100000001111**
vrf_menor_TP	Verifica se TR < TP	**010**	**11**	**11**	**0**	**0**	**0**	**0**	**0**	**100000001111**
dec_TR	Decrementa 1 no valor de TR	**001**	**11**	**11**	**1**	**0**	**0**	**0**	**0**	**100000001111**
vrf_maior_0	Verifica se TR > 0	**000**	**11**	**11**	**0**	**0**	**0**	**0**	**0**	**100000001111**
carrega_TFP	Carrega TP em TFP	**010**	**XX**	**00**	**0**	**0**	**0**	**1**	**0**	**100110000011**
TFP_reduzido	Carrega TP - TR em TFP	**110**	**00**	**11**	**0**	**0**	**0**	**1**	**0**	**100110000011**
decresce_TFP	Decresce em 1 o valor de TFP	**001**	**00**	**11**	**0**	**0**	**0**	**1**	**0**	**100110000011**
tempo_TFP	timer de TFP	**XXX**	**XXX**	**XX**	**0**	**0**	**0**	**0**	**1**	**100110000011**
TFP_comp_0	Verifica se TFP > 0	**000**	**00**	**11**	**0**	**0**	**0**	**0**	**0**	**100110000011**
ama_TAP	Amarelo Primário	**XXX**	**XX**	**XX**	**0**	**0**	**0**	**0**	**1**	**101000000011**
carrega_TFT	Carrega TT em TFT	**100**	**XX**	**00**	**0**	**1**	**0**	**0**	**0**	**010000001111**
decresce_TFT	Decresce em 1 o valor de TFT	**001**	**10**	**11**	**0**	**1**	**0**	**0**	**0**	**010000001111**
tempo_TFT	timer de TFT	**XXX**	**XXX**	**XX**	**0**	**0**	**0**	**0**	**1**	**010000001111**
TFT_comp_0	Verifica se TFT > 0	**000**	**10**	**11**	**0**	**0**	**0**	**0**	**0**	**010000001111**
carrega_TFS	Carrega TS em TFS	**011**	**XX**	**00**	**0**	**0**	**1**	**0**	**0**	**100000111100**
decresce_TFS	Decresce em 1 o valor de TFS	**001**	**01**	**11**	**0**	**0**	**1**	**0**	**0**	**100000111100**
tempo_TFS	timer de TFS	**XXX**	**XXX**	**XX**	**0**	**0**	**0**	**0**	**1**	**100000111100**
TFS_comp_0	Verifica se TFS > 0	**000**	**01**	**11**	**0**	**0**	**0**	**0**	**0**	**100000111100**
ama_TAS	Amarelo Secundário	**XXX**	**XX**	**XX**	**0**	**0**	**0**	**0**	**1**	**100001001100**

Obs: nível lógico alto = 1 e nível lógico baixo = 0;

LED[0] e **LED[1]** = sinal **VMS**;

LED[2] e **LED[3]** = sinal **VMP**;

LED[4] e **LED[5]** = sinal **VDS**;

LED[6] = sinal **AMS**;

LED[7] e **LED[8]** = sinal **VDP**;

LED[9] = sinal **AMP**;

LED[10] = sinal **VDT**;

LED[11] = sinal **VMT**.

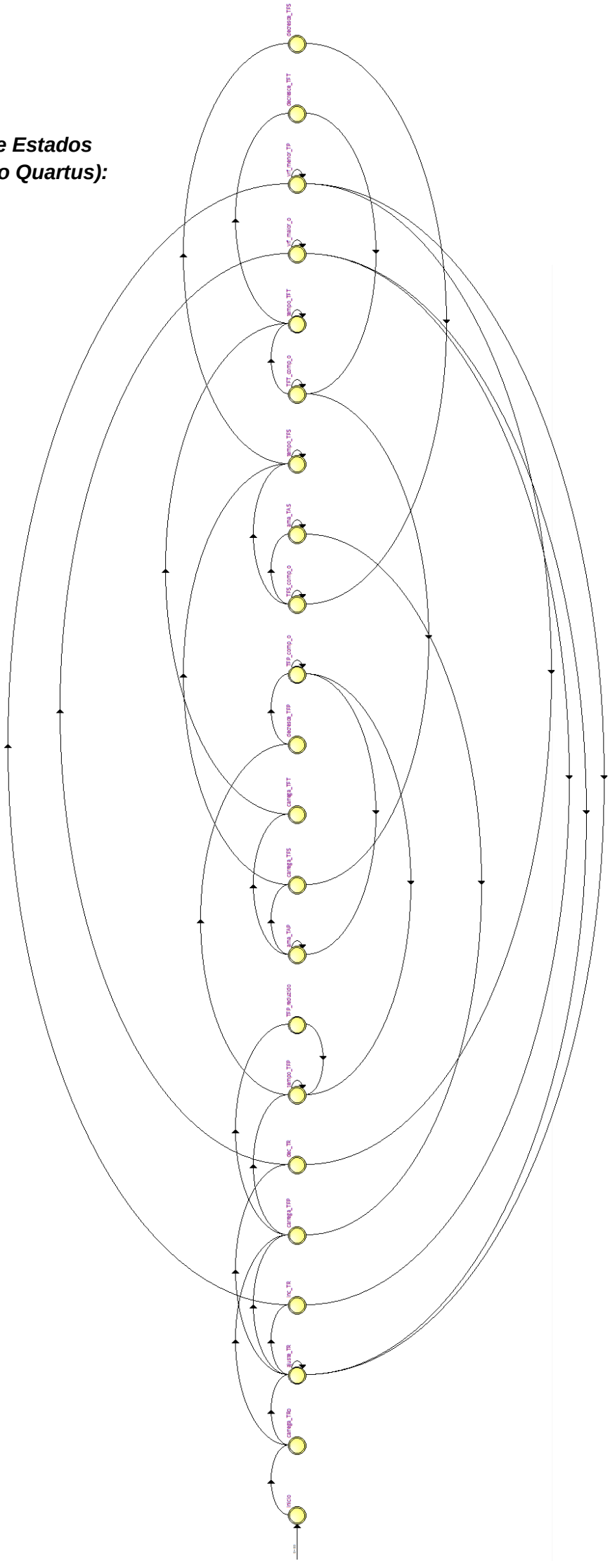
Código VHDL da Unidade de Controle

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

entity UC_SEMAFORO is
port( CK : in std_logic;            -- clock de 50MHz
-- Entradas Externas:
      INI: in std_logic;          -- sinal de reinício (ativo em '0')
      DC : in std_logic;          -- sinal para decremento  de TR (ativo em '1')
      IC : in std_logic;          -- sinal para  incremento de TR (ativo em '1')
      ST : in std_logic;          -- sinal de tráfego secundário (ativo em '1')
      BP : in std_logic;          -- sinal de solicitação de pedestre (ativo em '1')
      PGR: in std_logic;          -- sinal de modo (0=Normal , 1=Programação de TR)
-- Sinais de Estado da ALU:
      ZR : in std_logic;             -- resultado zero na operação (ativo em '1')
      PO : in std_logic;             -- resultado positivo na operação (ativo em '1')
      NG : in std_logic;             -- resultado negativo na operação (ativo em '1')
      ES : in std_logic;             -- resultado de estouro na operação (>15)
-- Sinais de Estado dos TIMER's:
      TC : in std_logic;             -- término de contagem do Timer_500ms (ativo em '1')
-- Sinais de Saída para MUX:
      Sel_mxa : out std_logic_vector(2 downto 0); -- seleciona entrada de MUX_A
      Sel_mxb : out std_logic_vector(1 downto 0);  -- seleciona entrada de MUX_B
-- Sinais de Saída para ALU:
      Sel_alu : out std_logic_vector(1 downto 0);  -- seleciona operação da ALU
-- Sinais de Saída para Registradores:
      LDA   : out std_logic;      -- carrega TFP (Tempo Faltante da Fase Verde Principal)
      LDB   : out std_logic;      -- carrega TFS (Tempo Faltante da Fase Verde Secundária)
      LDC   : out std_logic;      -- carrega TFT (Tempo Faltante de Transito de Pedestre)
      LDD   : out std_logic;      -- carrega TR  (Tempo de Redução da Fase Verde Principal)
-- Sinais de Saída para TIMER's:
      IT    : out std_logic;      -- inicia temporização do Timer_500ms (ativo em '0'->'1')
-- Sinais de Saída Externos:
      LD    :  out std_logic_vector(11 downto 0)); -- estado das lâmpadas (ativos em '1')
end UC_SEMAFORO;
--------------------------------------------------------------------------------
architecture FSM of UC_SEMAFORO is
type state_type is (inicio, carrega_TR0, ajuste_TR, inc_TR, vrf_menor_TP, dec_TR, vrf_maior_0,
carrega_TFP, TFP_reduzido, decresce_TFP, TFP_comp_0, ama_TAP, carrega_TFT, decresce_TFT,
TFT_comp_0, carrega_TFS, decresce_TFS, TFS_comp_0, ama_TAS, tempo_TFP, tempo_TFS, tempo_TFT);

signal E: state_type;
signal SP: std_logic;                              -- SP: solicitação de pedestre
signal RS: std_logic;                              -- RS: reseta solicitação de pedestre

begin
-- Armazena valor de BP até atendimento da solicitação:
Armazena_BP: process(CK, INI, BP, RS)
begin
if INI='0' then SP<='0';                           -- inicia com SP='0'
elsif rising_edge(CK) then
   if BP='1' then SP<= '1';                        -- atualiza SP quando BP='1'
   elsif RS='1' then SP<= '0';                     -- reseta SP quando RS='1'
   else SP<=SP;
   end if;
end if;
end process Armazena_BP;

-- Máquina de estados principal (transição de estados):
Logica_de_Estados: process(CK, INI)
begin
if INI='0' then E <= INICIO;
elsif rising_edge(CK) then
 case E is
   when inicio => E <= carrega_TR0;
   when carrega_TR0 =>
            if PGR = '0' then E <= carrega_TFP;
            else E <= ajuste_TR;
            end if;
   when carrega_TFP  =>
            if ST = '1' then E <= TFP_reduzido;
            else E <= tempo_TFP;
            end if;
   when TFP_reduzido  => E <= tempo_TFP;
   when decresce_TFP => E <= TFP_comp_0;
   when tempo_TFP =>
            if TC = '1' then E <= decresce_TFP;
            else E <= tempo_TFP;
            end if;
   when TFP_comp_0 =>
            if (PO or ES) = '1' then E <= tempo_TFP;
            elsif (ZR or NG) = '1' then E <= ama_TAP;
            end if;
   when ama_TAP =>
            if (SP and TC) = '1' then E <= carrega_TFT;
            elsif (not(SP) and TC) = '1' then E <= carrega_TFS;
            else E <= ama_TAP;
            end if;
   when carrega_TFT => E <= tempo_TFT;
   when decresce_TFT => E <= TFT_comp_0;
   when tempo_TFT =>
            if TC ='1' then E <= decresce_TFT;
            else E <= tempo_TFT;
            end if;
```

```
    when TFT_comp_0 =>
            if (PO or ES) = '1' then E <= tempo_TFT;
            elsif (ZR or NG) = '1' then E <= carrega_TFS;
            end if;
    when carrega_TFS => E <= tempo_TFS;
    when decresce_TFS => E <= TFS_comp_0;
    when tempo_TFS =>
            if TC = '1' then E <= decresce_TFS;
            else E <= tempo_TFS;
            end if;
    when TFS_comp_0 =>
            if (PO or ES) = '1' then E <= tempo_TFS;
            elsif (ZR or NG) = '1' then E <= ama_TAS;
            end if;
    when ama_TAS =>
            if TC = '1' then E <= carrega_TFP;
            else E <= ama_TAS;
            end if;
    when ajuste_TR =>
            if (DC and not(IC)) = '1' then E <= dec_TR;
            elsif (IC and not(DC)) = '1' then E <= inc_TR;
            elsif PGR = '0' then E <= carrega_TFP;
            else E <= ajuste_TR;
            end if;
    when inc_TR => E <= vrf_menor_TP;
    when vrf_menor_TP =>
            if (NG and (not(IC) and not(DC))) = '1' then E <= ajuste_TR;
            elsif  ((PO or ES or ZR) and (not(IC) and not(DC))) = '1' then E <= dec_TR;
            else E <= vrf_menor_TP;
            end if;
    when dec_TR => E <= vrf_maior_0;
    when vrf_maior_0 =>
            if ((PO or ES) and (not(IC) and not(DC))) = '1' then E <= ajuste_TR;
            elsif ((ZR or NG) and (not(IC) and not(DC))) = '1' then E <= inc_TR;
            else E <= vrf_maior_0;
            end if;

    when others => Null;
 end case;
end if;
end process Logica_de_Estados;

-- Atualização das saídas:
Logica_de_Saidas: process(E)
begin
 case E is
    when inicio =>
            Sel_mxa <= "000"; Sel_mxb <= "XX"; Sel_alu <= "00";
            LDD <= '1'; LDC <= '1'; LDB <= '1'; LDA <= '1'; IT <= '0'; RS <= '0';
            LD <= "100000001111";
    when carrega_TR0 =>
            Sel_mxa <= "101"; Sel_mxb <= "XX"; Sel_alu <= "00";
            LDD <= '1'; LDC <= '0'; LDB <= '0'; LDA <= '0'; IT <= '0';
            LD <= "100000001111";
    when ajuste_TR =>
            Sel_mxa <= "XXX"; Sel_mxb <= "XX"; Sel_alu <= "XX";
            LDD <= '0'; LDC <= '0'; LDB <= '0'; LDA <= '0'; IT <= '0';
            LD <= "100000001111";
    when inc_TR =>
            Sel_mxa <= "001"; Sel_mxb <= "11"; Sel_alu <= "10";
            LDD <= '1'; LDC <= '0'; LDB <= '0'; LDA <= '0'; IT <= '0';
            LD <= "100000001111";
    when vrf_menor_TP =>
            Sel_mxa <= "010"; Sel_mxb <= "11"; Sel_alu <= "11";
            LDD <= '0'; LDC <= '0'; LDB <= '0'; LDA <= '0'; IT <= '0';
            LD <= "100000001111";
    when dec_TR =>
            Sel_mxa <= "001"; Sel_mxb <= "11"; Sel_alu <= "11";
            LDD <= '1'; LDC <= '0'; LDB <= '0'; LDA <= '0'; IT <= '0';
            LD <= "100000001111";
    when vrf_maior_0 =>
            Sel_mxa <= "000"; Sel_mxb <= "11"; Sel_alu <= "11";
            LDD <= '0'; LDC <= '0'; LDB <= '0'; LDA <= '0'; IT <= '0';
            LD <= "100000001111";
    when carrega_TFP =>
            Sel_mxa <= "010"; Sel_mxb <= "XX"; Sel_alu <= "00";
            LDD <= '0'; LDC <= '0'; LDB <= '0'; LDA <= '1'; IT <= '0';
            LD <= "100110000011";
    when TFP_reduzido  =>
            Sel_mxa <= "110"; Sel_mxb <= "00"; Sel_alu <= "11";
            LDD <= '0'; LDC <= '0'; LDB <= '0'; LDA <= '1'; IT <= '0';
            LD <= "100110000011";
    when decresce_TFP  =>
            Sel_mxa <= "001"; Sel_mxb <= "00"; Sel_alu <= "11";
            LDD <= '0'; LDC <= '0'; LDB <= '0'; LDA <= '1'; IT <= '0';
            LD <= "100110000011";
    when tempo_TFP =>
          Sel_mxa <= "XXX"; Sel_mxb <= "XX"; Sel_alu <= "XX";
            LDD <= '0'; LDC <= '0'; LDB <= '0'; LDA <= '0'; IT <= '1';
            LD <= "100110000011";
    when TFP_comp_0  =>
            Sel_mxa <= "000"; Sel_mxb <= "00"; Sel_alu <= "11";
            LDD <= '0'; LDC <= '0'; LDB <= '0'; LDA <= '0'; IT <= '0';
```

```
            LD <= "100110000011";
   when ama_TAP  =>
            Sel_mxa <= "XXX"; Sel_mxb <= "XX"; Sel_alu <= "XX";
            LDD <= '0'; LDC <= '0'; LDB <= '0'; LDA <= '0'; IT <= '1';
            LD <= "101000000011";
   when carrega_TFT  =>
            Sel_mxa <= "100"; Sel_mxb <= "XX"; Sel_alu <= "00";
            LDD <= '0'; LDC <= '1'; LDB <= '0'; LDA <= '0'; IT <= '0'; RS <= '1';
            LD <= "010000001111";
   when decresce_TFT  =>
            Sel_mxa <= "001"; Sel_mxb <= "10"; Sel_alu <= "11";
            LDD <= '0'; LDC <= '1'; LDB <= '0'; LDA <= '0'; IT <= '0';
            LD <= "010000001111";
   when tempo_TFT =>
         Sel_mxa <= "XXX"; Sel_mxb <= "XX"; Sel_alu <= "XX";
            LDD <= '0'; LDC <= '0'; LDB <= '0'; LDA <= '0'; IT <= '1';
            LD <= "010000001111";
   when TFT_comp_0  =>
            Sel_mxa <= "000"; Sel_mxb <= "10"; Sel_alu <= "11";
            LDD <= '0'; LDC <= '0'; LDB <= '0'; LDA <= '0'; IT <= '0'; RS <= '0';
            LD <= "010000001111";
   when carrega_TFS  =>
            Sel_mxa <= "011"; Sel_mxb <= "XX"; Sel_alu <= "00";
            LDD <= '0'; LDC <= '0'; LDB <= '1'; LDA <= '0'; IT <= '0';
            LD <= "100000111100";
   when decresce_TFS  =>
            Sel_mxa <= "001"; Sel_mxb <= "01"; Sel_alu <= "11";
            LDD <= '0'; LDC <= '0'; LDB <= '1'; LDA <= '0'; IT <= '0';
            LD <= "100000111100";
   when tempo_TFS =>
         Sel_mxa <= "XXX"; Sel_mxb <= "XX"; Sel_alu <= "XX";
            LDD <= '0'; LDC <= '0'; LDB <= '0'; LDA <= '0'; IT <= '1';
            LD <= "100000111100";
   when TFS_comp_0  =>
            Sel_mxa <= "000"; Sel_mxb <= "01"; Sel_alu <= "11";
            LDD <= '0'; LDC <= '0'; LDB <= '0'; LDA <= '0'; IT <= '0';
            LD <= "100000111100";
   when ama_TAS  =>
            Sel_mxa <= "XXX"; Sel_mxb <= "XX"; Sel_alu <= "XX";
            LDD <= '0'; LDC <= '0'; LDB <= '0'; LDA <= '0'; IT <= '1';
            LD <= "100001001100";

   when others => Null;
 end case;
end process Logica_de_Saidas;
end FSM;
```

Diagrama de Estados
(Gerado pelo Quartus):



Decodificadores:

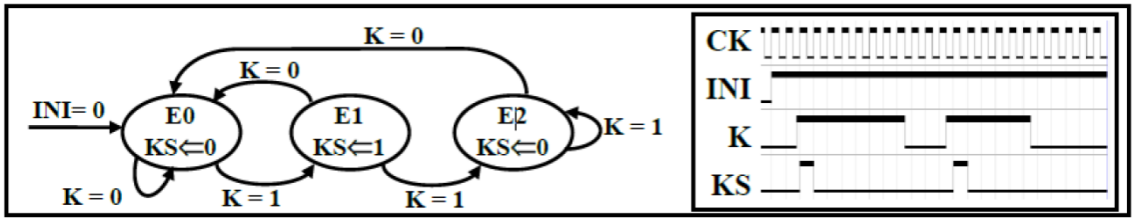
São utilizados 4 decodificadores para decodificar a saída dos 4 registradores e mostrar a informação em um display de 7 segmentos.

Observando o fato de que os segmentos dos displays acionam com nível lógico zero, então a saída do decodificador deverá ser zero aonde o segmento for aceso e nível lógico 1 aonde o segmento for apagado.

O arquivo VHDL foi disponibilizado pronto pelo professor, pelo moodle.

Sincronizador:

A ativação do sinal **K (**em nível lógico 1) deve gerar um sinal (**KS**) em nível lógico 1 sincronizado com o sinal de clock **(CK)** na borda de subida. Clock do sistema (50 MHz). O sinal **KS** deve ter a duração exata de um período de clock, independente do tempo em que a chave permanecer acionada.



O arquivo VHDL foi disponibilizado pronto pelo professor, pelo moodle.

Flip-Flop tipo D:

O flip-flop D tem o objetivo de memorizar o estado de um botão acionado.

No projeto são utilizados 3 flip-flops, um para cada botão **SW17**, **SW16** e **SW15** que geram os sinais **PGR**, **BP** e **ST** (ativos em nível lógico 1).

O arquivo VHDL foi disponibilizado pronto pelo professor, pelo moodle.

Simulação Funcional

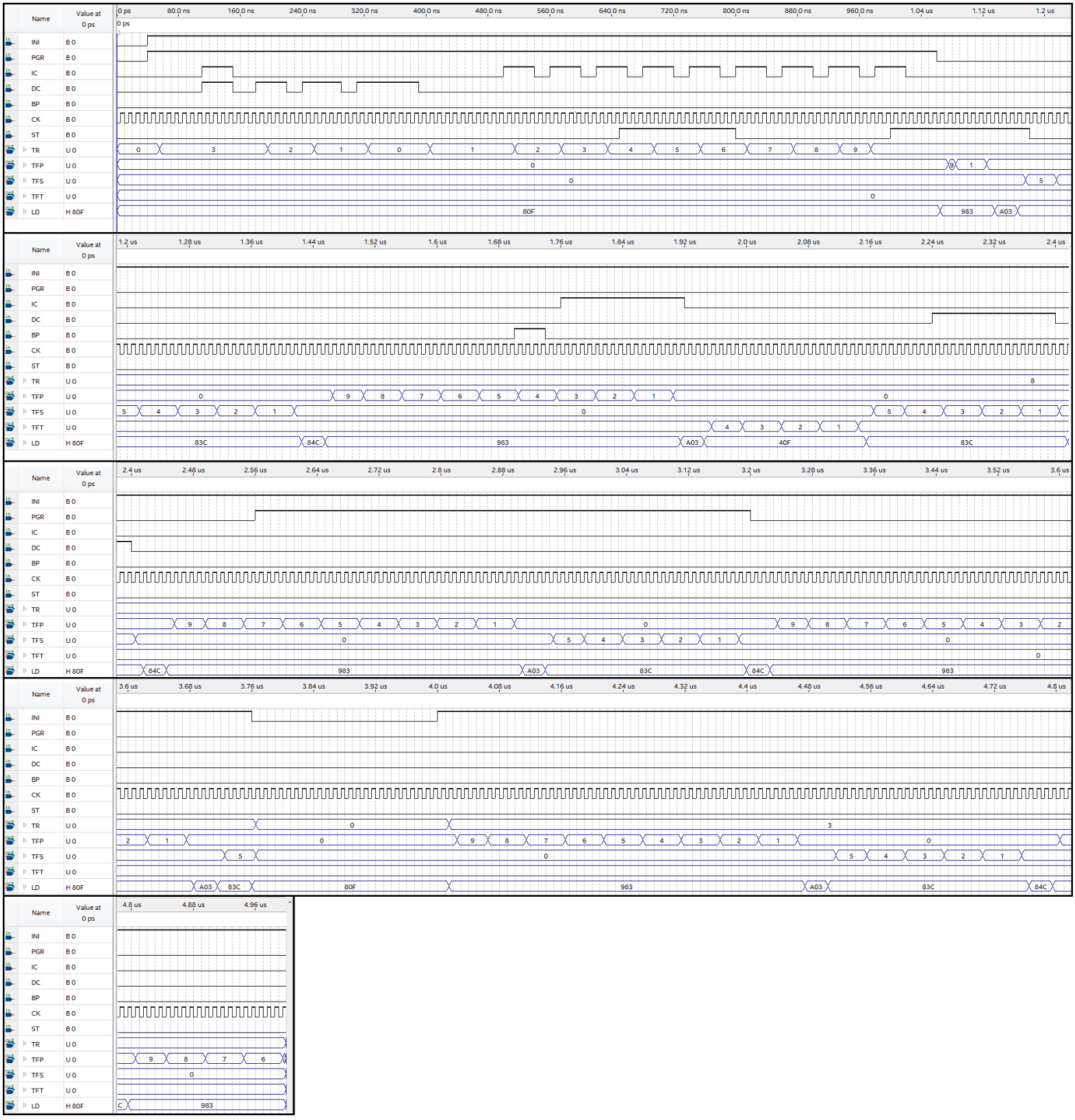
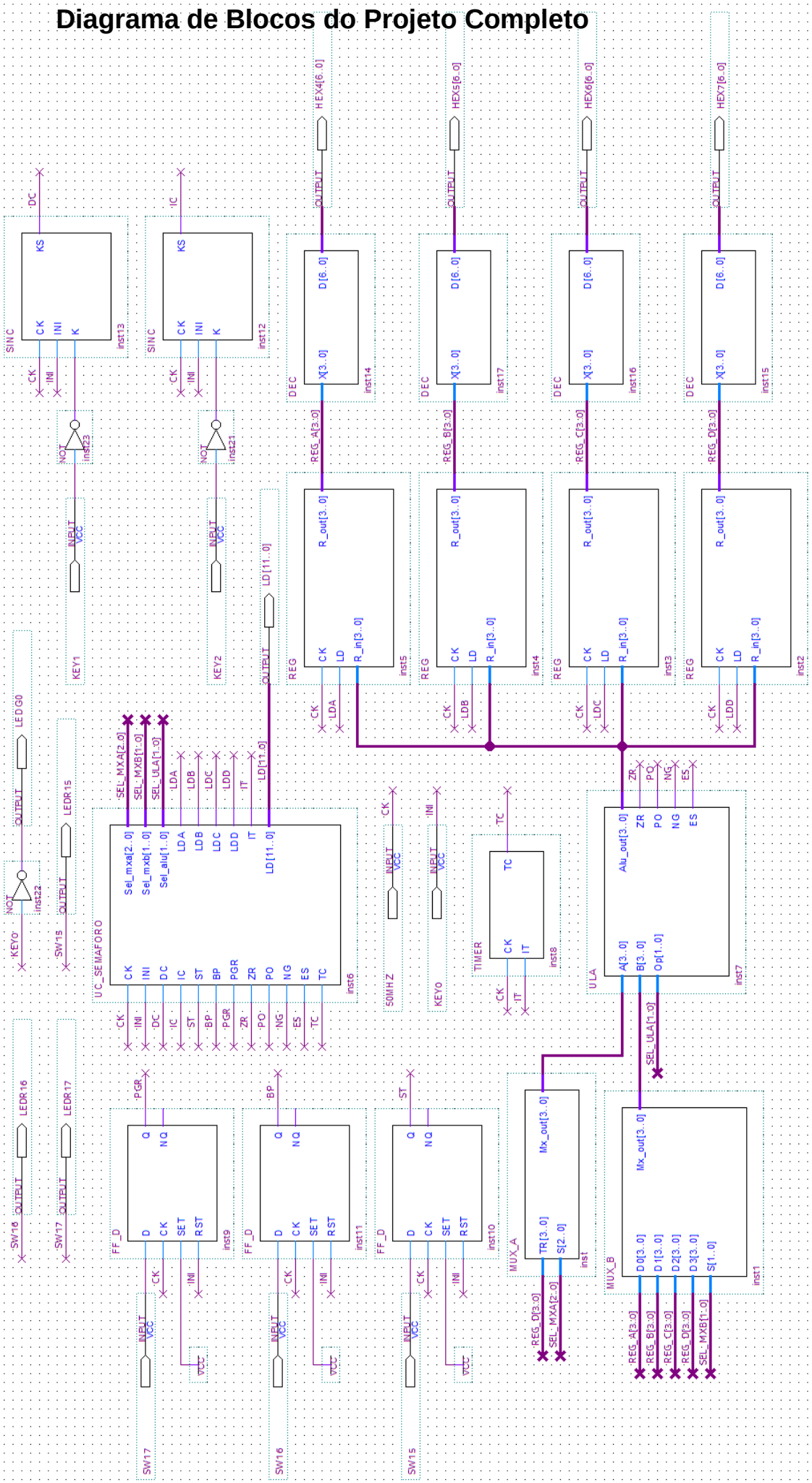


Diagrama de Blocos do Projeto Completo



Conclusão

Este projeto deixou claro para mim o funcionamento de uma Unidade de Controle trabalhando com um Bloco de Fluxo de Dados, onde na prática consegui entender o que na teoria estava complicado (sendo que agora entendo também a teoria).

Durante o projeto eu cometi alguns erros, o mais notável foi que na contagem regressiva do Verde Primário ele não começava do número correto (TP = 9), ele saía do 9 e um clock depois já passava para o número 8 e aí sim, a partir daí fazia a temporização corretamente. Isso por conta do estado de decremento estar antes do temporizador. Quando inverti os dois estados, o decremento começou a funcionar da maneira correta.

Uma observação é que no desenvolvimento da parte da programação para a apresentação na aula 2, na simulação se os botões de incremento ou decremento ficavam acionados por um período maior que de 1 período de clock, incrementava ou decrementava várias vezes antes de soltar o botão. Como o que controla isso na placa da Altera é o sincronizador, e até então não havia isso no projeto, eu projetei que só avança para o estado seguinte (**inc_TR** para **vrf_menor_TP** ou **dec_TR** para **vrf_maior_0**) caso o botão **IC** ou **DC** sejam desacionados (nível lógico 0). Para o projeto na placa Altera eu retirei essa programação por conta da inclusão do sincronizador.

Um ponto importante do projeto foi o fato de desenvolver mais ainda a técnica de programação VHDL, que é uma programação de hardware, não software.

O circuito de debouncing foi de extrema importância para o correto funcionamento do projeto. Sem ele, a trepidação ocorrida na mudança de estado de uma mesma chave poderia acionar várias vezes o sinal que entra na UC e causar algum tipo de transtorno para o sistema.