Universidade Federal de Santa Catarina
Centro Tecnológico - CTC
Departamento de Engenharia Elétrica - EEL

Everton Sperfeld Coan (16100723)
Lucas Henrique Wodtke (16202258)

TURMA 01208D

<everton_coan@hotmail.com>
<lhwodtke@gmail.com>

Relatório de Projeto Final EEL5105
2016.2

Florianópolis, 26 de Novembro de 2016.

Conteúdo

1. Introdução

A ideia do projeto é implementar uma simulação de um sistema de micro-ondas, a fim de trazer mais experiência prática para o aluno desafiando-o, e fazendo que tenha o domínio de diversas opções de desenvolvimento em VHDL estudados nas aulas teóricas e principalmente na prática, como contadores, comparadores, seletores, conversor de Clock, registradores, LEDs, Decod7Seg, memória Rom entre outros.

No projeto temos o bloco dos seletores composto por 4 Mux um de Estados, Tempo, Potencia e de Display, que devem selecionar corretamente as variáveis em cada estado.

Temos o bloco de registrados composto de Flip-Flops que tem como objetivo emitir sinais de luzes nos LEDs piscar nos primeiros estados e no último estado realizar a sequência do grupo e o último registrador exibir a potência nos LEDs.

Temos o bloco Comparador que verifica se a Contagem chegou a 0 transformando READY 0 ou 1.

Temos o bloco dos contadores, um responsável por converter o clock de 50 MHz para 1Hz e 13Hz para nosso grupo, e o outro responsável pela lógica de decrementar o tempo e pausar.

Temos a predefinição responsável por armazenar em uma memória tempos e potências os quais o usuário pode escolher.

Tempo o Debouncer, que evita uma falha de sincronismo ao clicar os botões.

E por último o controle a máquina de estados geral responsável por controlar e organizar todas as funções.

No topo conectamos todos os blocos através de sinais.

1.1 Predefinição

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

entity ROM1 is
  port ( SW : in std_logic_vector(2 downto 0);--A entrada são os 3 switches onde o usuario ira selecionar qual a pré-definição,assim temos 2^3 posibilidades
         MODE : out std_logic_vector(39 downto 0) );--A saida será o MODE os bits 39 downto 20 será o nome exibido nos 4 Displays7Seg, os bits 19 downto 10 serão
end ROM1;-- para o tempo sendo que 4 para os minutos e 6 para os segundos, o tempo será exibidos nos Displays7Seg, os bits 9 downto 0 irao representar a potencia
 --que será exibida nos leds.
architecture behavioral of ROM1 is
  type mem is array ( 7 downto 0 ) of std_logic_vector(39 downto 0);
  constant my_Rom : mem := (
    0  => "1100010010110001100100100000000000010000", -- PIPO 2 MINUTOS POTENCIA 5                 Aqui é registrado na memoria em 7 downto to 0 os
    1  => "1111101100100010101000100111100000001000", --  ChA 2 MUNITOS E 30 SEGUNDOS POTENCIA 4     dados(As pre difiniçoes)
    2  => "1010101110100001111001100000000001000000", -- LEgU 6 MINUTOS POTENCIA 7
    3  => "0110001010011110111000100000000000100000", -- CAFE 2 MINUTOS POTENCIA 6
    4  => "1110011001110000101001110000000000100000", -- SOPA 7 MINUTOS POTENCIA 6
    5  => "0101011011110111100101010000000010000000", -- ARRO 5 MINUTOS POTENCIA 8
    6  => "0110001010110111011110101010001000000000", -- CARN 10 MINUTOS E 40 SEGUNDOS POTENCIA 10
    7  => "1100011110011011001010001100100100000000" -- PUDI 8 MINUTOS E 50 SEGUNDOS POTENCIA 9
    );

begin
   process (SW)
   begin
     case SW is
       when "000" => MODE <= my_rom(0);          --Aqui quando o usuario selecionar os switches em binario zero pegara o dado guardado na memoria 0(pipo)
       when "001" => MODE <= my_rom(1);          --Se for 1 dado na memoria 1(CHA) e assim por diante.
       when "010" => MODE <= my_rom(2);
       when "011" => MODE <= my_rom(3);
       when "100" => MODE <= my_rom(4);
       when "101" => MODE <= my_rom(5);
       when "110" => MODE <= my_rom(6);
       when "111" => MODE <= my_rom(7);
       when others => MODE <= "0000000000000000000000000000000000000000"; -- Caso ele tente algo inválido o modo selecionado sera 0 em 39 downto 0 exibindo nada.
    end case;
  end process;
end architecture behavioral;
```

Simulações:

Como já dito anteriormente os switches indicam a posição em que o dado(MODE) está na memória, ao selecionar 000 ele pega my_rom(0) onde está localizado MODE “1100010010110001100100100000000000010000” nesse exemplo:

“11000” - P - exibido no decod7Seg 0.

“10010” -I - exibido no decod7Seg 1.

“11000” - P- exibido no decod7Seg 2.

“11001” - O - exibido no decod7Seg 3.

“0010” – Minutos (2) - exibido no decod7Seg 0 e decod7Seg1.

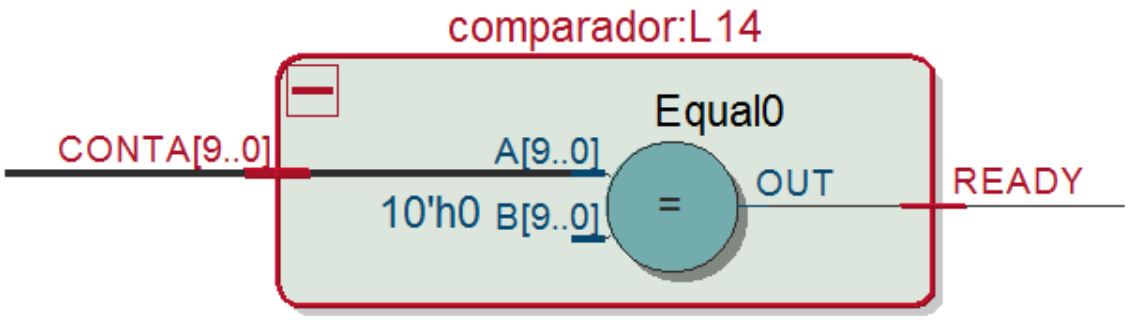
“000000” – Segundos (0) - exibido no decod7Seg 2 e decod7Seg3.

“0000010000” - Potencia 5 - Exibido nos LEDs.

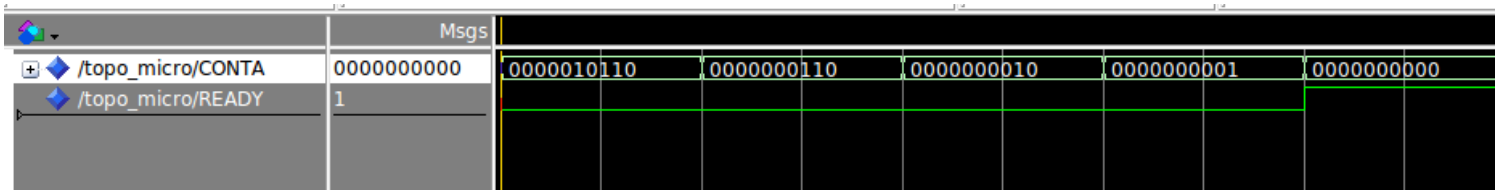
Cada predefinição tem suas características comentadas no código.

1.2 Comparador

A abordagem utilizada foi a comportamental a que não utiliza apenas funções booleanas, pois otimiza a lógica do comparador utilizando menos linhas de códigos e uma lógica mais palpável ficando claro a visualização e interpretação.




```
architecture circuito_comparador of comparador is
begin
READY <= '1' when (CONTA="0000000000")else '0';-- Como podemos ver no RTL, o comparador recebe Conta(Contagem do tempo "A"),
end circuito_comparador;                        -- se ele for igual a "0000000000" ("B") significa que a contagem chegou ao fim,
                                                -- e a saida Ready será 1 já que está pronto,senão a saída recebe 0.
```



Ao testar com variáveis aleatórias percebemos que realmente o comparador funciona, o READY só receberá 1 quando o Conta chegar a “0000000000”.

1.3 Contador

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

entity FSM_clock is
   port( clock_50: in std_logic;
         CLK1: out std_logic;
         CLK2: out std_logic
       );
end FSM_clock;

architecture bhv of FSM_clock is
   signal contador1: std_logic_vector(27 downto 0):= x"0000000";
   signal contador2: std_logic_vector(23 downto 0):= x"000000";

begin
   P1: process(clock_50)
   begin
      if clock_50'event and clock_50 = '1' then  --A fim de gerar um frequencia de 1 HZ recebemeos um clock_50 e fazemos ele contar
         contador1 <= contador1 + 1;             -- Até 49999999(hEXADECIMAL),alcançado esse valor ele gera uma subida do clock e reinicia a
            if contador1 = x"2FAF07F" then       --contagem.
            contador1 <= x"0000000";
            CLK1 <= '1';
         else
            CLK1 <= '0';
         end if;
      end if;

   end process;

      P2: process(clock_50)
   begin
      if clock_50'event and clock_50 = '1' then --Usando o mesmo principio de gerar 1 HZ agora queremos gerar 13 HZ
         contador2 <= contador2 +1;             -- para isso ele feve contar até 3846153,alcançado esse valor o clk2 recebe 1
            if contador2 = x"3AB009" then       --e a contagem é reiniciada.
            contador2 <= x"000000";
            CLK2 <= '1';
         else
            CLK2 <= '0';
         end if;
      end if;
   end process;
end bhv;
```

```
library IEEE;
use IEEE.Std_Logic_1164.all;

entity combContaRst is
port (RST:       in std_logic;
      RESET_TIME:    in std_logic;
      RST_CONT: out std_logic
      );
end combContaRst;

architecture circuito_combContaRst of combContaRst is
begin
          RST_CONT <= (not RST) or RESET_TIME;--Através da tabela verdade percebemos que o RST_CONT só será 1 se
end circuito_combContaRst;                    -- o RST negado(pois ele é o botão ou seja invertido) ou REST_TIME,
                                              --Assim será resetado o contador quando o botão precionado ou RESeT_TIME
                                              --for necessário.
```

Logica de reset

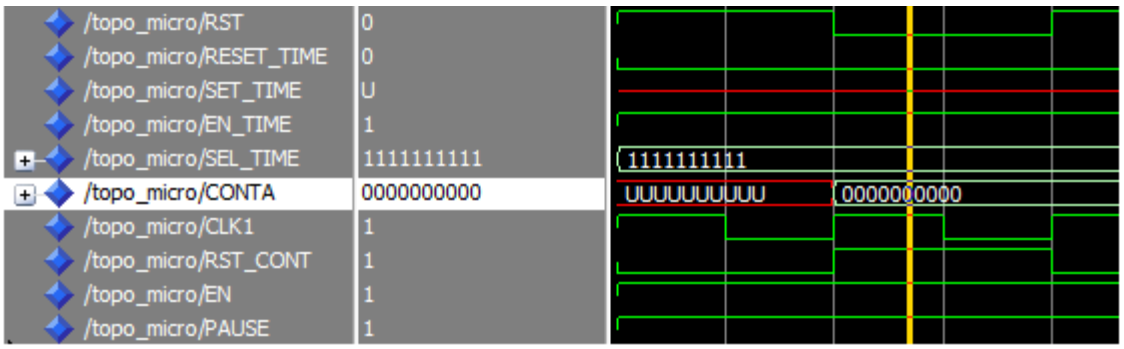
Lógica de Enable

```
library IEEE;
use IEEE.Std_Logic_1164.all;

entity combConta is
port (PAUSE:      in std_logic;
      EN_TIME:    in std_logic;
      EN: out std_logic
      );
end combConta;

architecture circuito_combConta of combConta is
begin
EN <= PAUSE and EN_TIME;   --Através da tabela verdade obtemos essa logica combinacional
end circuito_combConta;    --Levando em conta que Pause é um botao(Inversão) o EN só será
                           -- ativado quando o pause(botao) nao estiver sendo precionado e o EN_TIME for 1.
```

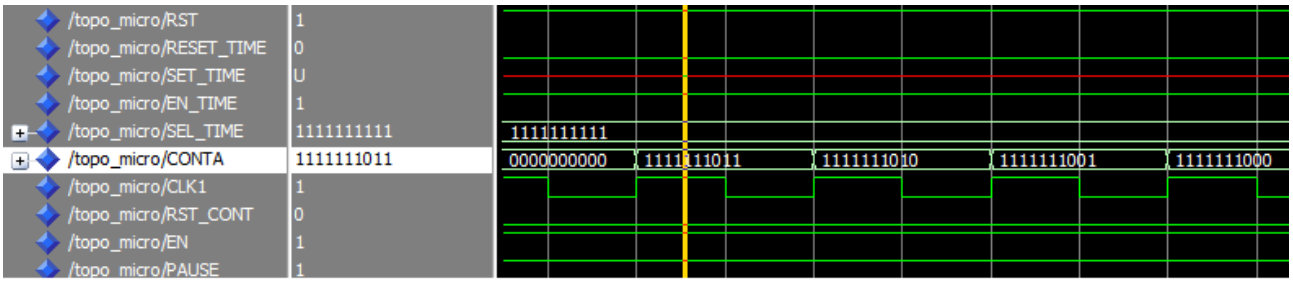
Logica Contador

```
entity CONTA_DESC is
port(
      CLK1: in std_logic;
      SEL_TIME: in std_logic_vector(9 downto 0);
      SET_TIME: in std_logic;
      EN: in std_logic;
      RST_CONT: in std_logic;
      CONTA: out std_logic_vector(9 downto 0)
   );
end CONTA_DESC;

architecture bhv of CONTA_DESC is

signal contador: std_logic_vector(5 downto 0);
signal contadorMin: std_logic_vector(3 downto 0);

begin

P1: process(CLK1,RST_CONT,sel_time,contador,contadorMin)
   begin

   if RST_CONT = '1' then  -- Se o RST_CONT é ativado os valores do contador(segundos) e contadorMin(Minutos)serão zerados
      contador<= "000000";
      contadorMin<="0000";
   elsif CLK1'event and CLK1= '1' then    -- Se ocorrer um ciclo de clock e o sel_time for 1
      if set_time = '1' then               -- então o contador recebe o tempo que o usuario colocou ou do modulo pre-definido
         contador<= sel_time (5 downto 0); -- (Mux tem a função de escole=her o correto) entao os contador recebe os segundos
         contadorMin<= sel_time (9 downto 6); -- e o contadorMin recebe os minutos
         elsif EN='1' then                   --Aqui funciona o metodo de Pause se o enable for 1 continua decrescendo senão pausa
            contador <= contador -'1';
            if contador = "000000" then
               contador <= "111011";            -- Nota que 111011 representa 59 segundos valor máximo dos segundos
               contadorMin <= contadorMin -'1';  -- Aqui ocorre a logica de decrementação do tempo quando se esgota os segundos
                                                 -- ele decrementa 1 do minuto
         end if;
      end if;
   end if;
end process;
      CONTA(9 downto 6) <=contadorMin;     --Aqui o Conta recebe os minutos nos bits 9 downto 6 e segundos nos bits 5 downto 0
      CONTA(5 downto 0)<=contador;         --e assim o Conta está preenchido e segue seu caminho para outro bloco.
end bhv;
```

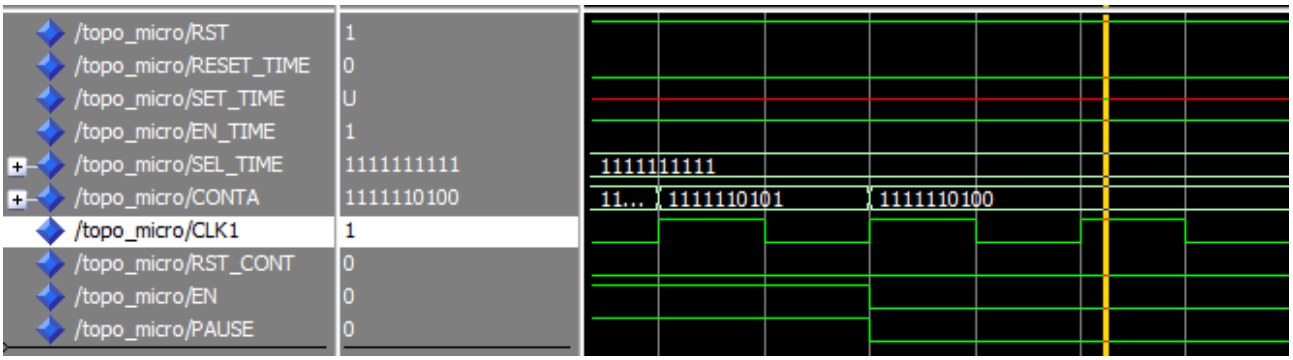
Simulação:



Como já descrito no código primeiro selecionamos o valor depois resetamos ao clicar o botão RST fazendo que o conta deixasse de ser indefinido para 0.



Aqui vemos o decremento como esperado.



Aqui vemos que a lógica do pause funciona.

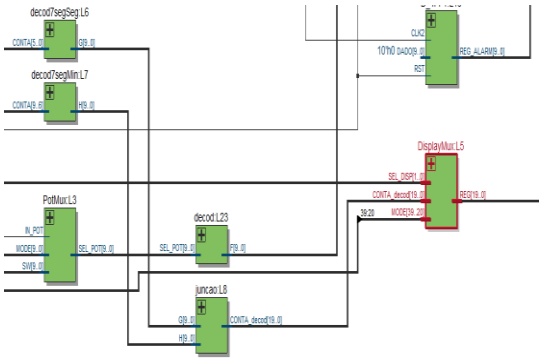
Descrição da Máquina de estados:

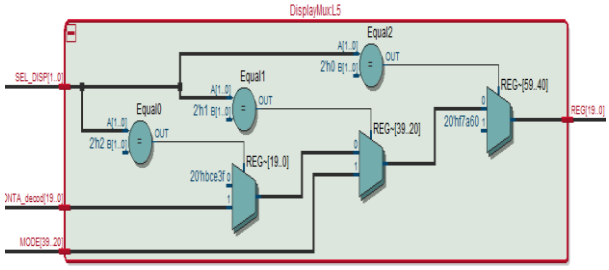
Primeiro estado é quando o RST_CONT é acionado, ou seja, Conta é zero o usuário então insere o tempo no SET_TIME se não aguarda no primeiro estado. Se EN for 1 ele avança para o 2 estado onde conta é decrementado se EN for 1 continua a decrementar senão vai para o terceiro estado onde o Conta fica parado se EN continuar a 0 fica no estado 3 se for 1 vai para o estado 2, quando a contagem chega a 0 no estado 2 ele volta para o estado 1.

1.4 Seletores

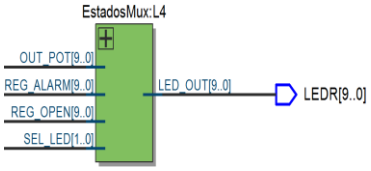
RTL:

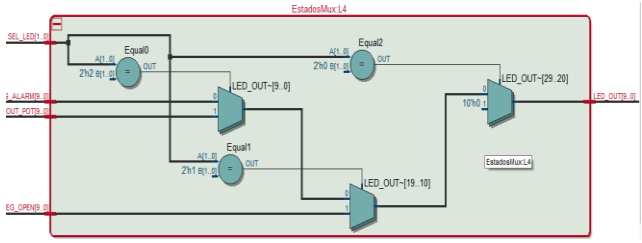
Mux do display:



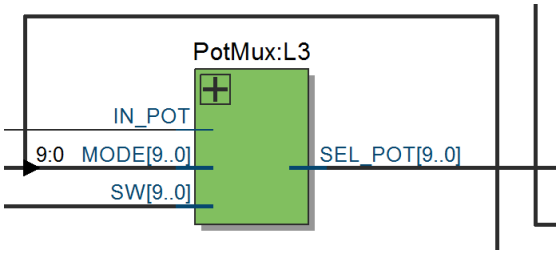


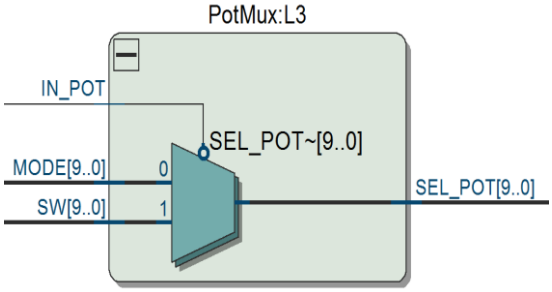
Mux de Estados:



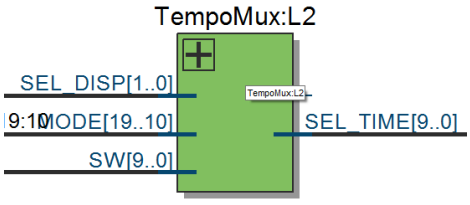


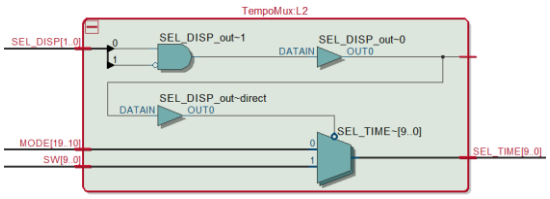
Mux de Potência:





Mux de Tempo:





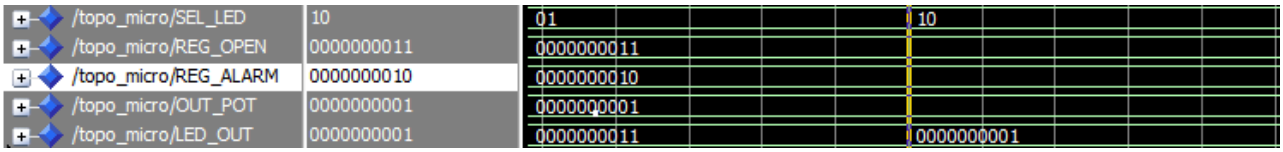

```
rchitecture circuito_Estado of EstadosMux is
egin
 LED_OUT <= "0000000000" when  SEL_LED= "00" else--Essa contantes exibe os leds apagados.
            REG_OPEN when SEL_LED = "01" else    --Aqui é a variavel que exibe a sequencia piscante nos leds no primeiro estado.
            OUT_POT when SEL_LED = "10" else     --Aqui é a variável que exibe as pontencias no led
            REG_ALARM;                           --Aqui é a variável que exibe a senquencia do nosso grupo.
 nd circuito_Estado;                             -- é função da maquina de estados selecionar o SEL_LED de forma correta
```

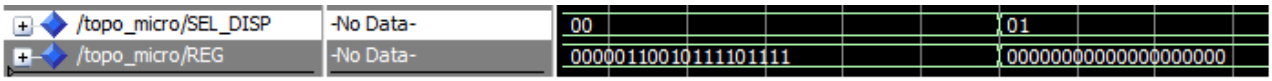
```
entity DisplayMux is
port (MODE:      in std_logic_vector (39 downto 20);
      CONTA_decod:    in std_logic_vector(19 downto 0);
      SEL_DISP: in std_logic_vector(1 downto 0);
      REG:      out std_logic_vector(19 downto 0)
      );
end DisplayMux;

architecture circuito_Display of DisplayMux is
begin
REG <= "00000110010111101111" when SEL_DISP = "00" else  --Aqui será exibido OFF no displays7Seg
       MODE when SEL_DISP = "01" else                     --Aqui será exibido os nomes(Pipo,Pudi...) da Pré-definição
       CONTA_decod when SEL_DISP = "10" else              --Aqui será exibido o CONTA(o tempo em minutos e segundos)
       "11111100011100111101";                           --Aqui será exibido HOT.
end circuito_Display;                                     --Cabe a maquina de estados escolher qual será usado.
```

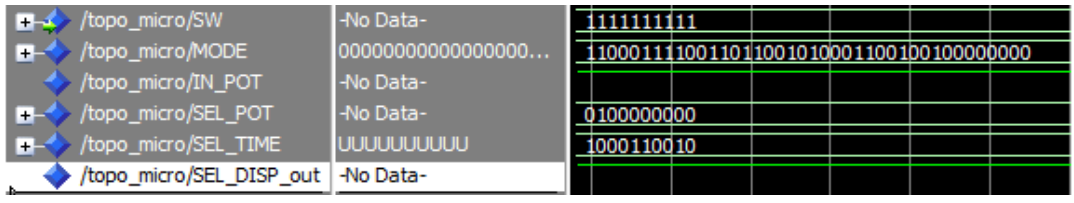
```
entity PotMux is
port (SW:      in std_logic_vector(9 downto 0);
      MODE:    in std_logic_vector(9 downto 0);
      IN_POT:  in std_logic;
      SEL_POT: out std_logic_vector(9 downto 0)
      );
end PotMux;

architecture circuito_Pot of PotMux is
begin
SEL_POT <= SW when IN_POT = '0' else -- Aqui a potencia selecionada pelo usário vai ser selecionada
           MODE;                     -- Aqui a potencia pré-definida será usada
end circuito_Pot;
```

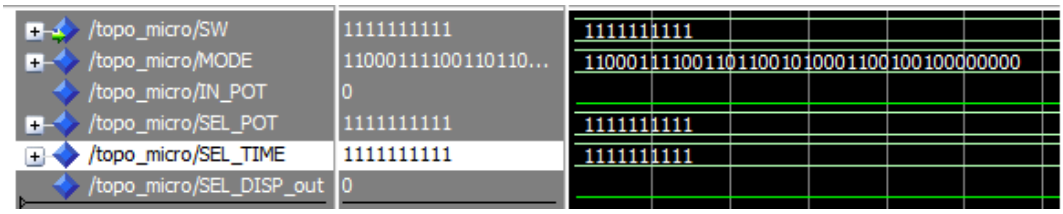
```
library IEEE;
use IEEE.Std_Logic_1164.all;

entity TempoMux is
port (SW:      in std_logic_vector (9 downto 0);
      MODE:    in std_logic_vector (19 downto 10);
      SEL_DISP: in std_logic_vector(1 downto 0);
      SEL_DISP_out: inout std_logic;
      SEL_TIME: out std_logic_vector(9 downto 0)
      );
end TempoMux;

architecture TempoMux of TempoMux is
begin
SEL_DISP_out <= not SEL_DISP(1) and SEL_DISP(0);-- lOGICA da Seleção

SEL_TIME <= SW when SEL_DISP_out = '0' else    --Aqui o tempo selecionado é definido pelo usuario através os swicthes
            MODE(19 downto 10);                -- Aqui o tempo selecionado é o da pré-definição
end TempoMux;
```

Como podemos ver nas simulações com algumas variáveis ocorreu exatamente como explicado.



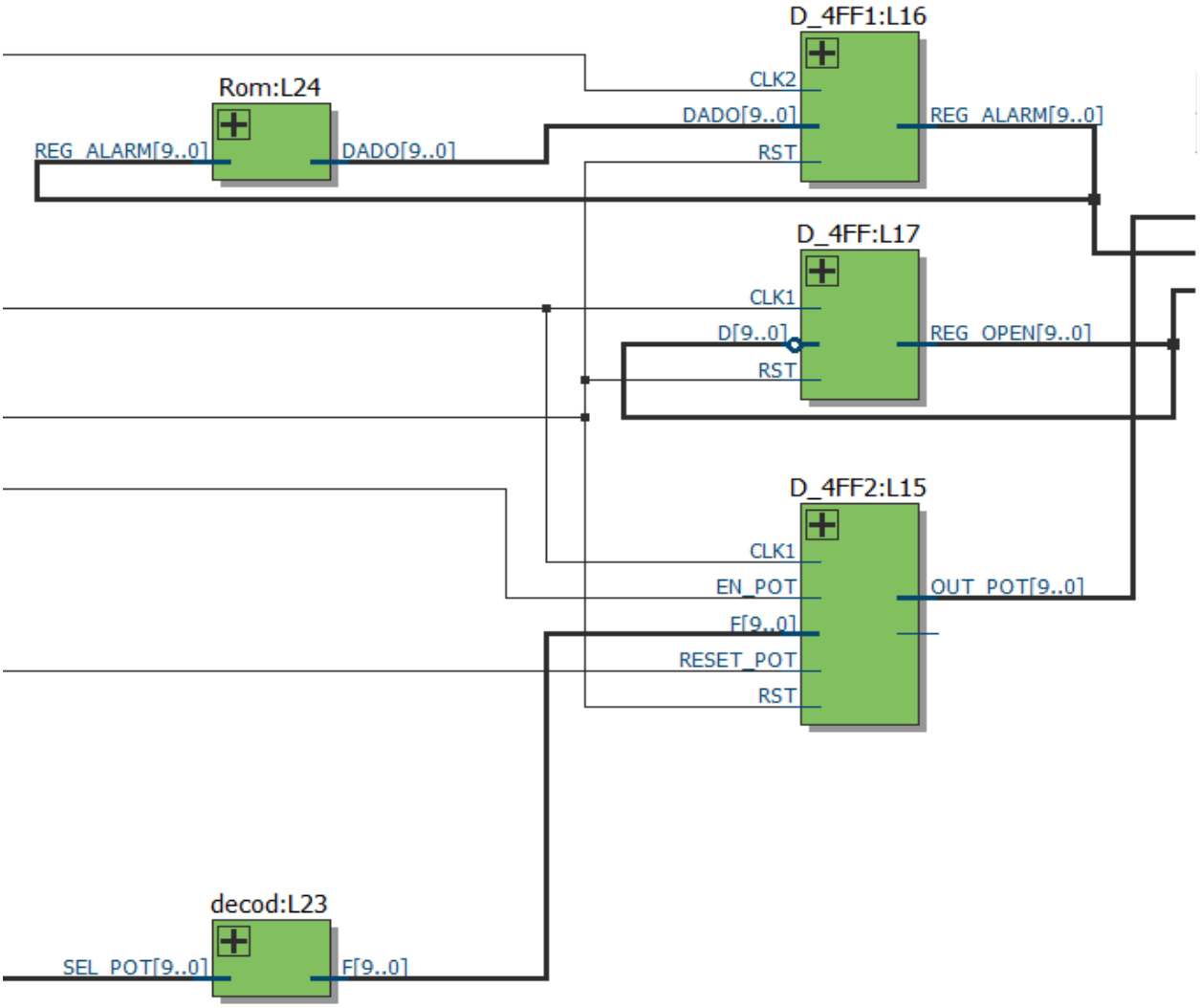






A abordagem utilizada foi a comportamental a fim de simplificar o código.

1.5 Registradores



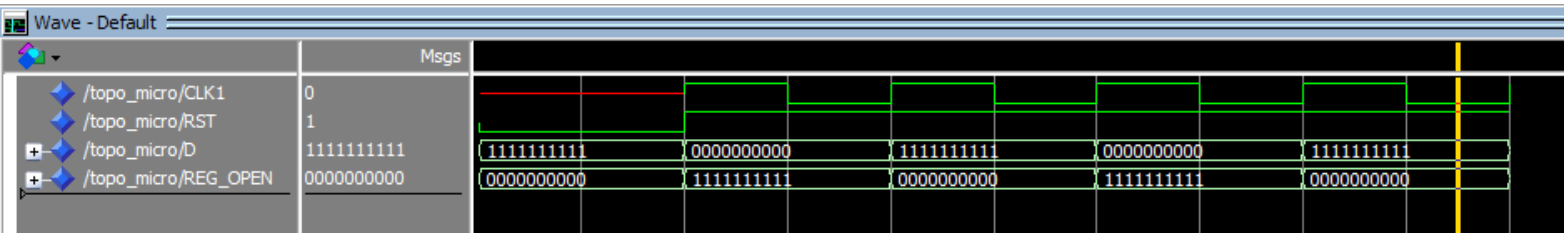
Decodificador de 10 para 10 bits (decod:L23) feito usando abordagem comportamental, recebe como entrada a saída do seletor de potência no formato “0001000000” e o decodifica para o formato “0001111111”.

O terceiro registrador (D_4FF2:L15) recebe o sinal *F* do decodificador, o armazena enquanto *EN_POT* estiver com o valor ‘1’, e o envia pelo sinal *OUT_POT*.

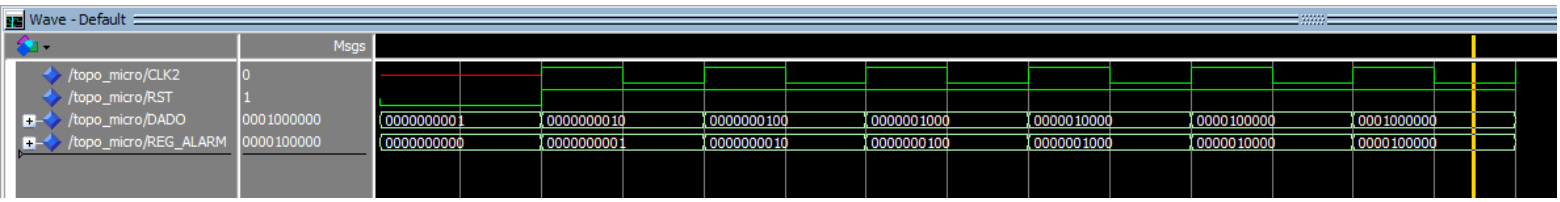
O segundo registrador (D_4FF1:L16) recebe o sinal *DADO* com o efeito luminoso da Rom:L24, o armazena durante um ciclo de *CLK1*, e o envia pelo sinal *REG_ALARM* para a Rom:L24, que ao recebe-lo, envia o sinal seguinte.

O primeiro registrador (D_4FF:L17) recebe como entrada o sinal *D* que é a sua saída REG_OPEN negada.

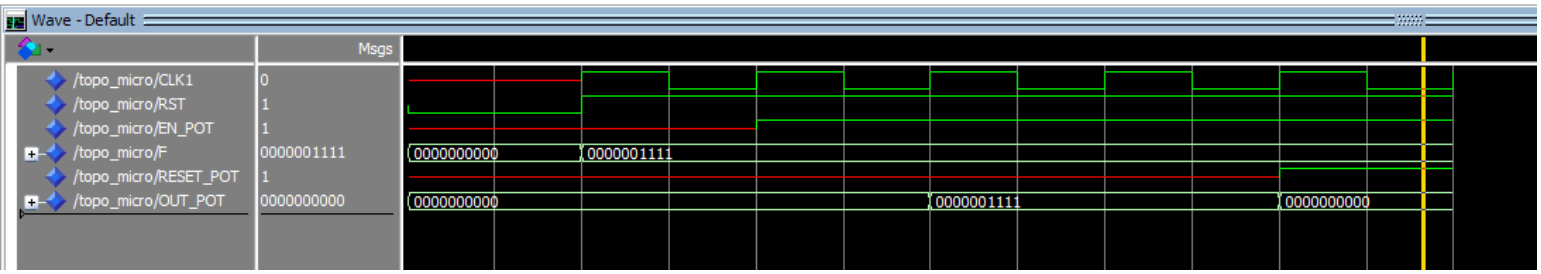
```
17  begin
18      RST_Ps <= (not RST) or RESET_POT; -- Lógica combinatória do Reset
19      process(CLK1, RST_Ps,EN_POT, F)
20  begin
21      if (RST_Ps = '1') then OUT_POT <= "0000000000"; -- RST assíncrono do registrador
22      elsif(CLK1'event and CLK1 = '1') then -- Clock sensível a borda de subida
23          if (EN_POT = '1') then OUT_POT <= F; -- Saída será igual a potência quando EN_POT for '1'
24          end if;
25      end if;
```



Primeiro registrador (D_4FF:L17), inicia com *RST* ativo que mantém a saída *REG_OPEN* em “0000000000” e a entrada *D* em “1111111111”, ao desativar o RST, a saída e a entrada passam a ser invertidas a cada evento de clock.

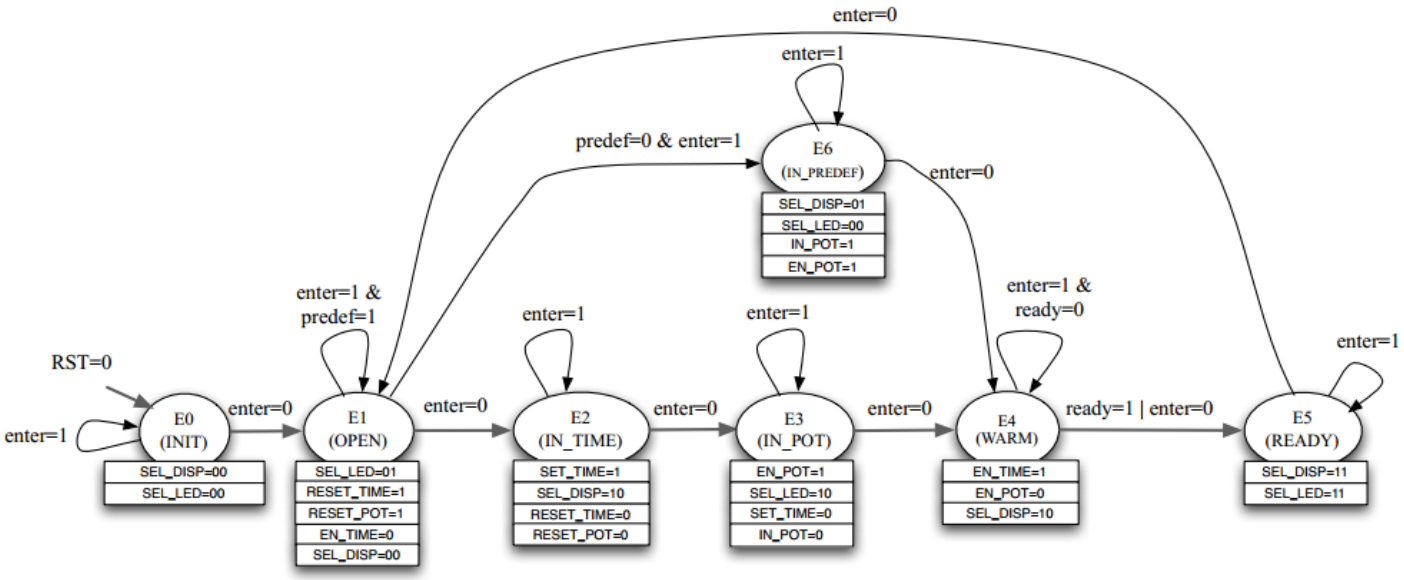


Segundo registrador (D_4FF:L16), inicia com *RST* ativo que mantém a saída *REG_ALARM* em “0000000000”, ao desativar o *RST* a saída passa a ser o sinal *DADO* que contém o efeito luminoso vindo da Rom, e segue para o próximo efeito a cada evento de *CLK2*.



O terceiro registrador (D_4FF2:L15), inicia com RST ativo que mantém a saída OUT_POT em “0000000000”, ao desativar o RST e forçar uma entrada em F, que é o sinal de saída do decodificador, a saída continua em “0000000000” por causa do sinal EN_POT que está sem valor, ao definir o valor ‘1’ para EN_POT a saída passa a ser o sinal F no próximo ciclo de CLK1, e continua assim até RESET_POT passar a ser ‘1’ e o sinal OUT_POT voltar a ser “0000000000”.

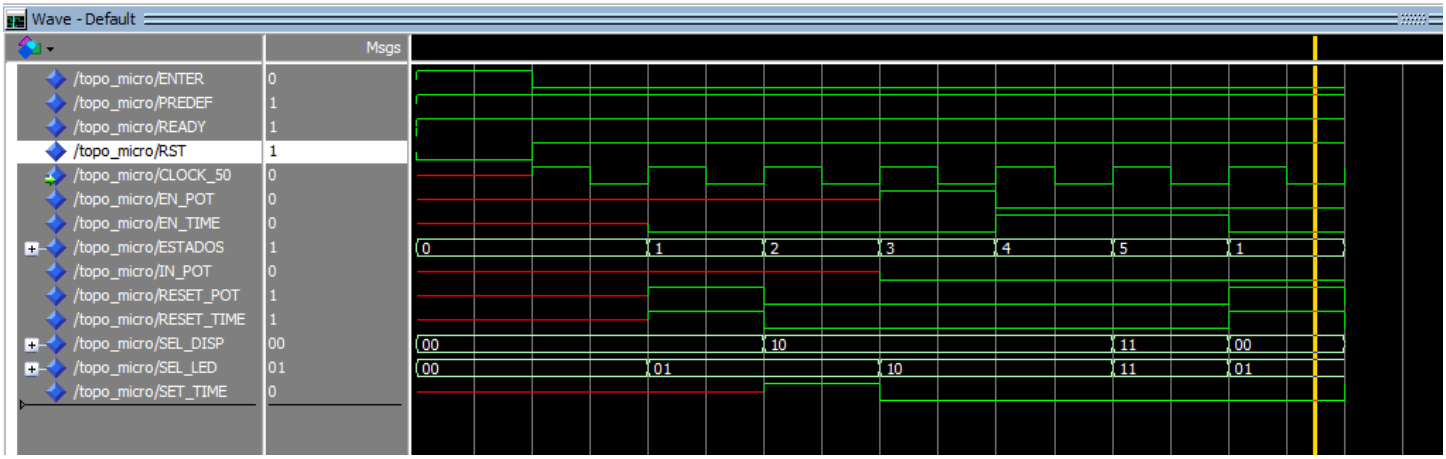
2 Controlador



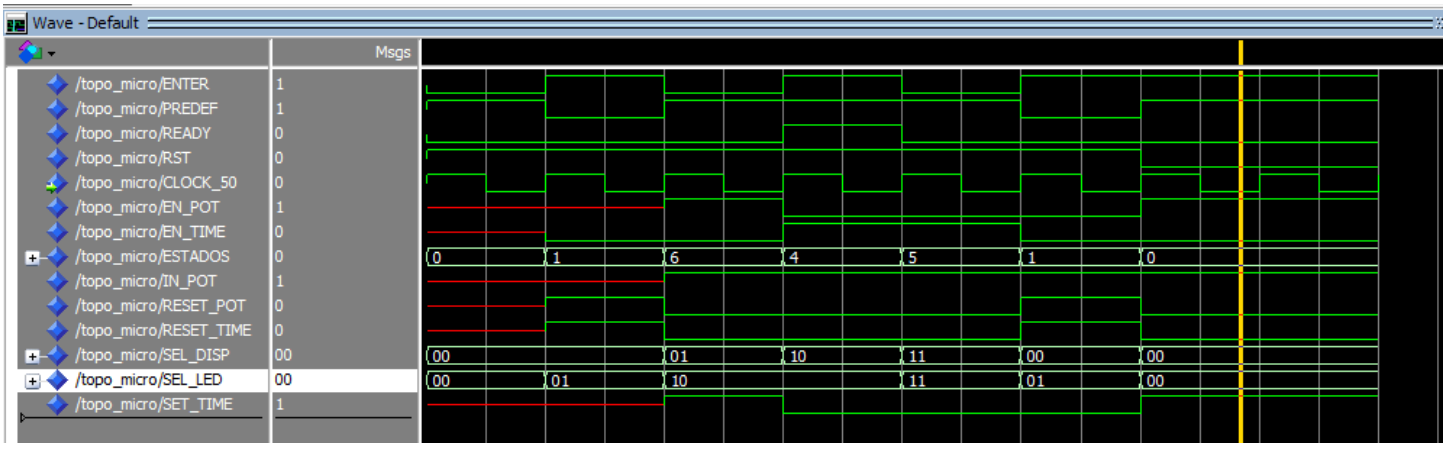
Inicia no estado E0 com as saídas zeradas, ao pressionar o botão KEY1 (enter) passa para o estado E1 onde o usuário deve escolher entre o modo manual ou predefinido, caso escolher manual e pressionar KEY1 segue para o estado E2, onde deve escolher o tempo e pressionar KEY1 novamente para passar para o estado E3, onde será selecionada a potência e seguirá para E4 quando enter for pressionado, nesse estado caso o sinal READY fique em '1' ou o botão enter for pressionado a máquina transitará para o estado E5 e voltará para o estado E1 se enter for pressionado. Caso o modo predefinido for escolhido e pressionar KEY2 (predef) passará para o estado E6, onde será escolhido o modo de preparo e seguira para E4 quando KEY1 for pressionado.

```
23  architecture bhv of FSM_Control is
24      type STATES is (E0, E1, E2, E3, E4, E5, E6); -- Estados da máquina
25      signal EA, PE: STATES;
26
27  begin
28
29  P1: process(EA, ENTER, PREDEF) -- Lógica de próximo estado
30  begin
31      case EA is
32          when E0 =>
33              if enter = '0' then -- Quando enter for pressionado transitará para E1
34                  PE <= E1;
35              else
36                  PE <= E0; -- Senão continará em E0
37              end if;
```

```
87   P2: process(clock_50, RST)
88   begin
89      if RST = '0'then -- RST assíncrono da máquina de estados
90         EA <= E0;
91      elsif clock_50'event and clock_50 = '1' then -- Clock sensível a borda de subida
92         EA <= PE;
93      end if;
94   end process;
95
96   P3: process(EA) -- Lógica de saída
97   begin
98      case EA is
99         when E0 => -- Valores das variáveis de saída quando estiver em E0
100           SEL_DISP <= "00";
101           SEL_LED <= "00";
102           ESTADOS <= "00000";
```



Inicia com RST ativo e valores zerados, ao desativar o RST e trocar o valor de ENTER para ‘0’ transita para o estado 1, depois 2, 3, 4, 5 e 1 novamente como esperado.



No estado 1, ENTER passa a ser ‘1’ e PREDEF a ser ‘0’, então a máquina transita para o estado 6 e depois para E4, quando READY passa a ser ‘1’ ela segue para E5 e volta para E1, quando RST passa a ser ’0’ e ela volta para o estado inicial.

3. Resultados e conclusões

Para a realização da conexão de todos os blocos utilizamos um único topo, usando como entrada os switches, os KEYS e o clock 50Mhz e como saída os LEDs e o HEX (5 downto 0). A interconexão é feita através dos sinais. Através do port map foi especificado as conexões.

As simulações funcionaram como esperado assim como os testes na placa.

Anexo A – Observações

Após as simulações descobrimos alguns problemas principalmente na lógica do pause do contador. Durante o teste na placa também persistiram, depois de algumas alterações e testes foi resolvido. Algumas alterações no controlador foram também realizadas a fim de evitar execuções fora dos padrões.