Simeon Thomas Bomfim de Faria,
John Wesley Bomfim de Faria

Contador eletrônico regressivo

Brasil
23 de novembro de 2017

Simeon Thomas Bomfim de Faria,
John Wesley Bomfim de Faria

Contador eletrônico regressivo

Relátorio referente ao projeto aplicativo da disciplina Microeletrônica I do curso de Engenharia Elétrica da Universidade Federal do Paraná apresentado a professora Sibilla França

Universidade Federal do Paraná

Setor de Tecnologia

Engenharia Elétrica

Brasil

23 de novembro de 2017

Sumário

1 Introdução

Na displina de Microeletrônica, como projeto final, foi proposto que fosse implementado no kit Nexys2 um contador regressivo com tempos pré programados ([10 – 15 – 20 – 30 – 40 – 50 – 60] minutos). Esse contador deverá ser implementado usando o código VHDL com auxílio dos conhecimentos obtidos durante a disciplina e terá como desafio a implementação do programa usando a saída VGA presente no kit Nexys2.

2 Especificação

Projetar, utilizando o kit NEXYS2, um dispositivo eletrônico capaz de efetuar uma contagem regressiva com passos definidos de configuração ([10 – 15 – 20 – 30 – 40 – 50 – 60] minutos). A aplicação lúdica para este projeto será o "contador eletrônico regressivo"

O dispositivo deverá obrigatoriamente implementar as seguintes funcionalidades:

1. Ao ligar o dispositivo, uma mensagem de inicialização deverá ser apresentada nos displays de 7 segmentos: "SELECIONE O TEMPO XY". Um rolamento das letras através dos 4 displays deverá garantir a visualização da mensagem inicial;

2. Através do acionamento de um botão do kit, o usuário deverá ser capaz de escolher entre uma das opções possíveis ([10 – 15 – 20 – 30 – 40 – 50 – 60] minutos);

3. Após a escolha do tempo, a contagem deverá ser iniciada através do acionamento de outro botão do kit. Uma função de pausa deverá ser prevista;

4. Uma vez iniciada a contagem regressiva, a mesma deverá ser mostrada nos displays de 7 segmentos e em um monitor com resolução de 800x600 (usar saída VGA do kit). Tanto nos displays quanto no monitor, deverão ser exibidos 4 dígitos (2 para os minutos e 2 para os segundos), **sendo que no monitor os mesmos devem estar separados por ":"**. No monitor, pelo menos 80 % do espaço útil deverá ser utilizado;

5. No monitor, os últimos 2 minutos de contagem deverão ser exibidos em outra cor, evidenciando que o final da contagem está próximo;

6. Após o final da contagem, uma mensagem deve aparecer nos displays de 7 segmentos: "TEMPO ESGOTADO XY". Um rolamento das letras através dos 4 displays deverá garantir a visualização da mensagem final. No monitor, os quatro dígitos assumirão o valor "00.00" e deverão permanecer "piscando" até a reinicialização;

7. O acionamento de um mecanismo mecânico do kit deverá permitir a reinicialização do dispositivo de forma a iniciar-se nova contagem (retorno a mensagem inicial e reset do monitor);

   XY – Primeira letra do nome dos membros da equipe de desenvolvimento. (Ex.: XY Xavier e Yann)

3 Sobre o Código

3.1 Entidade

Definição das bibliotecas e portas de entrada e saída presentes no projeto. A entidade presente no código pode ser representada pela figura 1:

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  entity projetoaplicativo is port(
7
8  reset, clk, start ,pause : in std_logic; --sinais de entrada
9  sel: out std_logic_vector(3 downto 0);--seleçao dos displays
10 seteseg: out std_logic_vector(6 downto 0);--vetor que o display recebe
11 botao_estado: in std_logic  :='0';--botão que altera o valor de estado
12 hsinc       : out std_logic; --saída sincronização horizontal
13 vsinc       : out std_logic; --saída sincronização vertical
14 corestovga      : out std_logic_vector(7 downto 0) --vetor que define as cores
15
16 );
17 end projetoaplicativo;
```

Figura 1: Entidades do Projeto Aplicativo.

3.2 Sinais

Os sinais são componentes usados para a lógica do projeto que não são explícitos na saída, porém de grande importancia. Para a elaboração do projeto foram declarados sinais para troca de estados, contagem regressiva, divisor de clock, definição dos estados, debounce dos botões, decoder, e para os contadores dos displays. Todos esse sinais elaboraram a lógica necessária para se chegar ao resultado esperado. A declaração dos sinais do código pode ser vista na figura 2.

```
20  architecture hardware of projetoaplicativo is
21  ----------------------------INTEIROS UTILIZADOS NA TROCA DE ESTADO--------------------------------
22  signal d0:  integer range 0 to 9 := 0;--dezena de minuto
23  signal d1:  integer range 0 to 9 := 0;--unidade de minuto
24  signal d3 : integer range 0 to 9 := 0;--unidade de segundos
25  signal d2 : integer range 0 to 5:= 0; --dezena de segundos
26  ----------------------------INTEIROS UTILIZADOS NA CONTAGEM----------------------------------------------------------
27  signal q0:  integer range 0 to 9 := 0;--dezena de minuto
28  signal q1:  integer range 0 to 9 := 0;--unidade de minuto
29  signal q3 : integer range 0 to 9 := 0;--unidade de segundos
30  signal q2 : integer range 0 to 5:= 0; --dezena de segundos
31  ----------------------------SINAIS PARA DIVISOR DE CLOCK--------------------------------
32  signal      clk1hz:     std_logic := '0'; --clock de 1hz
33  signal      clk100hz:   std_logic := '0'; --clock de 100hz
34  constant    gen1hz:     integer := 25_000_000; --usado no clock de 1hz
35  constant    gen100hz:   integer := 5000; --usado no clock de 100hz
36  ----------------------------SINAIS PARA DEFINIR ESTADOS--------------------------------
37  signal      flag :      std_logic:='1' ;--utilizado para saber o momento da contagem
38  signal      flag_fim :  std_logic:='0' ;--utilizado para saber o fim da contagem
39  signal      estado:     integer range 0 to 8 := 0; --utilizado para trocar os estados
40  ----------------------------SINAIS PARA DEBOUNCE--------------------------------
41  signal      flipflops:          std_logic_vector(1 DOWNTO 0); -- sinal criado para criar condição do debouncer
42  signal      counter_set:        std_logic; --sinal utilizado no debouncer
43  signal      counter_out:        std_logic_vector(2 DOWNTO 0) := (OTHERS => '0'); --sinal utilizado no debouncer
44  signal      out_debounce:       std_logic := '0'; --sinal da saida do debouncer
45  ----------------------------SINAIS PARA DECODER----------------------------------------------------------
46  signal      seven_seg_letra:        std_logic_vector(6 downto 0);--envia letra para o display de 7 segmentos
47  signal      seven_seg_numero:       std_logic_vector(6 downto 0);--envia numero para o display de 7 segmentos
48  constant    mensagem_ihardwareial:  string(1 to 21) := " SELECIONA O TEMPO JS";--string mensagem inicial
49  constant    mensagem_final:         string(1 to 18) := " TEMPO ESGOTADO JS";--string mensagem final
50  signal      letra:                  character; --utilizado para envio do letra para o decoder
51  signal      numero:                 integer range 0 to 9;--Envio do nº para o decoder
52  signal      selectdisp:             integer range 0 to 3 :=0; --usado na seleçao dos displays
53  ------------------------CONTADORES PARA DISPLAY-------------------------------------------------
54  signal      contador_disp1:     integer range 0 to 20 := 1;
55  signal      contador_disp2:     integer range 0 to 20 := 0;
56  signal      contador_disp3:     integer range 0 to 20 := 0;
57  signal      contador_disp4:     integer range 0 to 20 := 0;
```

Figura 2: Declaração dos sinais presentes no código.

3.3 Components

Para a execução do projeto, vão ser feitas chamadas a components no codigo, a fração de código que indica como foram feitas as declaração dos components pode ser observada na figura 3:

```
59  --COMPONENT VGA
60  component VGA  is port (    clk         : in std_logic;
61                              flag        : in std_logic;
62                              flag_fim    : in std_logic;
63                              q0,q1,q3    : in integer range 0 to 9;
64                              q2          : in integer range 0 to 5;
65                              clk1hz      : in std_logic;
66                              estado      : in integer range 0 to 8;
67                              sync_h      : out std_logic;
68                              sync_v      : out std_logic;
69                              corestovga  : out std_logic_vector(7 downto 0) );
70  end component;
71  --COMPONENT DIVISOR DE CLOCK
72  component Divisor_clock port (  clk     : in std_logic;
73                                  freqin  : in integer range 0 to 25_000_000;
74                                  clk_out : out std_logic );
75  end component;
76  -- COMPONENT DECODER
77  component Decoder is port(  letra       : in character;
78                              numero      : in integer range 0 to 9;
79                              seven_seg_letra     : out std_logic_vector(6 downto 0);
80                              seven_seg_numero    : out std_logic_vector(6 downto 0) );
81  end component;
```

Figura 3: Declaração dos components presentes no código.

3.4 Port maps

Após isso foi mapeado os portmaps que irão estar presentes no circuito final do projeto. os port maps podem ser representados pela figura 4:

```
93  --PORT MAP DECODER
94  Decoder1: Decoder PORT MAP (          letra => letra,
95                                        numero => numero,
96                                        seven_seg_letra => seven_seg_letra,
97                                        seven_seg_numero => seven_seg_numero );
98  --PORT MAP CLOCK1hz
99  Clock1hz: Divisor_clock PORT MAP (    clk => clk,
100                                       freqin => gen1hz,
101                                       clk_out => clk1hz );
102 --PORT MAP CLOCK1hz
103 Clock100hz: Divisor_clock PORT MAP (  clk => clk,
104                                       freqin => gen100hz,
105                                       clk_out => clk100hz );
106 --PORTMAP VGA
107 VGA1: VGA PORT MAP (                 clk  => clk,
108                                       clk1hz  => clk1hz,
109                                       q0 => q0,
110                                       q1 => q1,
111                                       q3 => q3,
112                                       q2 => q2,
113                                       flag => flag,
114                                       flag_fim => flag_fim,
115                                       estado => estado,
116                                       sync_h    => hsinc,
117                                       sync_v    => vsinc,
118                                       corestovga   => corestovga );
```

Figura 4: Port maps usados no código.

3.5 Component Divisor de clock

Para a contagem do tempo foi usado como referencia os ciclos de clock do kit que é de 50MHz para o ajuste dessa frequência foi usada uma estrutura já pronta que foi feita nos desafios anteriores da disciplina que é chamada de divisor de clock foi usado um component divisor de clock que faz o ajuste entre o clock do kit Nexys2 e o tempo em segundos. A figura 5 representa o component divisor de clock usado:

```
1
2   LIBRARY IEEE;
3   USE IEEE.STD_LOGIC_1164.ALL;
4
5   ENTITY Divisor_clock IS
6       PORT(
7               clk: in std_logic;--clk de 50MHz
8               freqin: in integer range 0 to 25_000_000; --valor de entrada
9               clk_out: out std_logic--clock de saída
10              );
11  END Divisor_clock;
12
13  ARCHITECTURE hardware OF Divisor_clock IS
14  -- código para divisão do clock
15      signal somador: integer :=0;
16      signal oclk: std_logic :='0';
17  BEGIN
18
19      clk_out <= oclk;
20
21      process(clk)
22      begin
23          if rising_edge(clk) then
24              somador <= somador +1;
25              if(somador = freqin)then
26              oclk <= not oclk;
27              somador <= 0;
28              end if;
29          end if;
30
31      end process; --fim do processo de clock
32
33  END hardware;
```

Figura 5: Component divisor de clock.

3.6 Component Decodificador

Para que seja mostrado numeros e letras no display de 7 segmentos, foi usado um component decodificador, o qual tem a função de traduzir para os displays os valores de strings e números em vetores de tamanho 7. Com ele é possível que a mensagem inicial, mensagem final, tempo do cronometro e tempo decorrido sejam mostrados nos display de 7 segmentos. A figura 6 mostra como é o funcionamento dos displays de sete seguimentos que o decoder irá atuar, de acordo com o manual do kit (NEXYS2. . . ,).

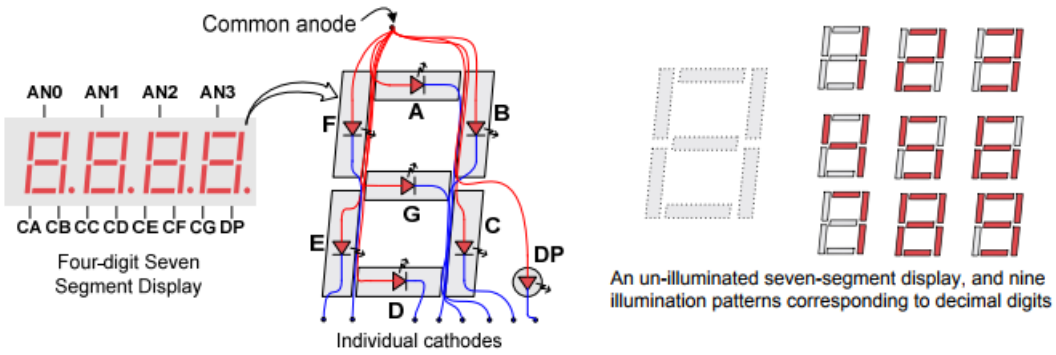

Figura 6: Estrutura do dispay de 7 segmentos do kit Nexys2.

No código cada representação no display de 7 segmentos é feita por meio de um vetor de tamanho 7 o qual segue a lógica [a b c d e e f g] e que quando é indicado valor 0, representa o segmento do display aceso, e 1 para apagado, apartir disso foi feito o mapeamento em cada letra e numero que é necessário no projeto para que as mensagens sejam interpretadas corretamente. A decodificação pode ser representada pela figura 7.

```vhdl
----------------------------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
----------------------------------------------------------------------------------------------------
entity Decoder is
    PORT(
            letra: in character;
            numero: in integer range 0 to 9:=0;
            seven_seg_letra: out std_logic_vector(6 downto 0);
            seven_seg_numero: out std_logic_vector(6 downto 0)
            );
end Decoder;
------------------------DECODER QUE ESCREVE OS TEXTOS-----------------------------
ARCHITECTURE Decoder of Decoder is
BEGIN
        WITH letra SELECT

                        seven_seg_letra <=  "1111011" WHEN 'I',
                                            "1101010" WHEN 'N',
                                            "1110010" WHEN 'C',
                                            "1100010" WHEN 'O',
                                            "1111111" WHEN ' ',
                                            "1110001" WHEN 'L',
                                            "0100100" WHEN 'S',
                                            "0011000" WHEN 'P',
                                            "1110000" WHEN 'T',
                                            "0110000" WHEN 'E',
                                            "1000011" WHEN 'J',
                                            "0001000" WHEN 'A',
                                            "0101011" WHEN 'M',
                                            "1000010" WHEN 'D',
                                            "0000100" WHEN 'G',
                                            "1111111" WHEN OTHERS;
---------------------DECODER QUE ESCREVE OS NÚMEROS -------------------------------
        WITH numero SELECT
            seven_seg_numero <=     "0000001" WHEN 0, --Escreve 0
                                    "1001111" WHEN 1, --Escreve 1
                                    "0010010" WHEN 2, --Escreve 2
                                    "0000110" WHEN 3, --Escreve 3
                                    "1001100" WHEN 4, --Escreve 4
                                    "0100100" WHEN 5, --Escreve 5
                                    "0100000" WHEN 6, --Escreve 6
                                    "0001111" WHEN 7, --Escreve 7
                                    "0000000" WHEN 8, --Escreve 8
                                    "0000100" WHEN 9, --Escreve 9
                                    "1111111" WHEN OTHERS; -- Espaço vazio
----------------------------------------------------------------------------------------------------
END Decoder;
----------------------------------------------------------------------------------------------------
```

Figura 7: Estrutura de decodificação para letras e números

3.7 Display de 7 segmentos

Para definir oque é mostrado nos displays de 7 segmentos do kit, foi feito um processo, o qual define as condições necessárias para cada fase do projeto, que são as mensagens inicial e final, seleção de tempo e contagem reversa. Na figura 8 ver a lógica do deslocamento das mensagens pelos 4 displays de 7 segmentos e na figura 9 é possível observar a fração do código que impõe as condições necessárias para a mensagem inicial. Esse processo é auxiliado pelos components divisor de clock e decoder, sendo o divisor usado para a passagem da mensagem inicial que no caso é "SELECIONE O TEMPO JS"pelos 4 displays e o decoder sendo usado como tradutor dessas strings para o vetor de tamanho 7 que é lido pelos displays de 7 segmentos presentes no kit.

```
278 process( clk1hz, out_debounce )
279 begin
280     if reset = '1' then --reset do estado e dos contadores do display
281         estado <=0;
282         contador_disp1 <=1; contador_disp2 <=0;
283         contador_disp3 <=0; contador_disp4 <=0;
284     elsif reset = '0' and flag = '1' then
285         if rising_edge(out_debounce) and estado <= 8 then --borda de subida no sinal e estado menor que 8 executa:
286             estado <= estado + 1;
287             end if;
288         if estado = 8 then
289             estado <= 0;
290         end if;
291             if rising_edge(clk1hz) then -- lógica utilizada para a mensagem inicial e final ser toda exibida
292                     contador_disp1 <= contador_disp1 + 1;
293 
294                     if (contador_disp1 > 0) then
295                         contador_disp2 <= contador_disp2 + 1;
296                     end if;
297 
298                     if (contador_disp2 > 0) then
299                         contador_disp3 <= contador_disp3 + 1;
300                     end if;
301 
302                     if (contador_disp3 > 0) then
303                         contador_disp4 <= contador_disp4 + 1;
304                     end if;
305 
306                     if (contador_disp4 = 20 ) then
307                         contador_disp1 <=1; contador_disp2 <=0;
308                         contador_disp3 <=0; contador_disp4 <=0;
309                     end if;
310 
311             end if;
312     end if;
313 end process;
```

Figura 8: Lógica usada para as mensagens inicial e final se deslocarem pelos displays de 7 segmentos.

```
134 ------------------------------------------------------------------------------------------
135 --PROCESSO QUE DEFINE O QUE É MOSTRADO NO DISPLAY DE 7 SEGMENTOS
136     process (clk100hz)
137         begin
138         if rising_edge(clk100hz) then
139             --MOSTRA MENSAGEM INICIAL
140             if estado = 0 and flag = '1' and flag_fim='0' then --CONDIÇÕES NECESSÁRIAS
141                 case selectdisp is
142                     when 0 => sel <= "1110";
143                         selectdisp  <= selectdisp  + 1;
144                         letra <= mensagem_ihardwareial(contador_disp1);
145                         seteseg <= seven_seg_letra;
146                     when 1 => sel <= "1101";
147                         selectdisp  <= selectdisp  + 1;
148                         letra <= mensagem_ihardwareial(contador_disp2);
149                         seteseg <= seven_seg_letra;
150                     when 2 => sel <= "1011";
151                         selectdisp  <= selectdisp  + 1;
152                         letra <= mensagem_ihardwareial(contador_disp3);
153                         seteseg <= seven_seg_letra;
154                     when 3 => sel <= "0111";
155                         selectdisp <= 0;
156                         letra <= mensagem_ihardwareial(contador_disp4);
157                         seteseg <= seven_seg_letra;
158                     when others => sel <= "0000";
159                 end case;
160             end if;
```

Figura 9: Condições necessárias iniciais.

Após as condições necessárias para a mensagem inicial, foi feita a estrutura com as condições necessárias para a seleção do tempo desejado de contagem, que são 10, 15, 20, 30, 40, 50 e 60 minutos. Para mostrar esses números nos displays, também é usado a estrutura do component decodificador que traduz os tempos para os displays. Essa estrutura pode ser observada na figura 10:

```
161 ----------------------------------------------------------------------------------------------
162             --MOSTRA AS OPÇÕES DO CRONÔMETRO(10-15-20-30-40-50-60)
163             if (flag = '1' and estado /= 0 ) then --CONDIÇÕES NECESSÁRIAS
164             case selectdisp is
165                 when 0 => sel <= "1110";
166                     selectdisp <= selectdisp + 1;
167                     numero <= d3;
168                     seteseg <= seven_seg_numero;
169                 when 1 => sel <= "1101";
170                     selectdisp <= selectdisp + 1;
171                     numero <= d2;
172                     seteseg <= seven_seg_numero;
173                 when 2 => sel <= "1011";
174                     selectdisp <= selectdisp + 1;
175                     numero <= d1;
176                     seteseg <= seven_seg_numero;
177                 when 3 => sel <= "0111";
178                     selectdisp <= 0;
179                     numero <= d0;
180                     seteseg <= seven_seg_numero;
181                 when others => sel <= "0000";
182             end case;
183             end if;
184 ----------------------------------------------------------------------------------------------
```

Figura 10: Condições necessárias para seleção do tempo do cronômetro.

Após a seleção do tempo, é mostrada a contagem do tempo que foi selecionado anteriormente nos displays e as condições necessárias para que a contagem seja inciada. A figura 11 representa a fração de código que permite a contagem do tempo.

```
184 ----------------------------------------------------------------------------------------------
185             --MOSTRA A CONTAGEM DO VALOR SELECIONADO ANTERIORMENTE
186             if (flag = '0' and estado /= 0  ) then --CONDIÇÕES NECESSÁRIAS
187             case selectdisp is
188                 when 0 => sel <= "1110";
189                     selectdisp <= selectdisp + 1;
190                     numero <= q3;
191                     seteseg <= seven_seg_numero;
192                 when 1 => sel <= "1101";
193                     selectdisp <= selectdisp + 1;
194                     numero <= q2;
195                     seteseg <= seven_seg_numero;
196                 when 2 => sel <= "1011";
197                     selectdisp <= selectdisp + 1;
198                     numero <= q1;
199                     seteseg <= seven_seg_numero;
200                 when 3 => sel <= "0111";
201                     selectdisp <= 0;
202                     numero <= q0;
203                     seteseg <= seven_seg_numero;
204                 when others => sel <= "0000";
205             end case;
206             end if;
```

Figura 11: Condições necessárias para seleção do tempo do cronômetro.

Após o tempo ser esgotado, deve ser mostrado nos displays a mensagem final, que é "TEMPO ESGOTADO JS". Para que isso aconteça foram impostas as condições necessãrias para que seja mostrada a mensagem final com auxilio do component decoder. A figura 12 representa a fração do código para a mensagem final.

```
207 ------------------------------------------------------------------------------------------
208             --MOSTRA A MENSAGEM FINAL SE A CONTAGEM FOR FINALIZADA
209             if  (flag='1' and flag_fim='1' and estado /= 0)then --CONDIÇÕES NECESSÁRIAS
210                 case selectdisp is
211                     when 0 => sel <= "1110";
212                         selectdisp  <= selectdisp  + 1;
213                         letra <= mensagem_final(contador_disp1);
214                         seteseg <= seven_seg_letra;
215                     when 1 => sel <= "1101";
216                         selectdisp  <= selectdisp  + 1;
217                         letra <= mensagem_final(contador_disp2);
218                         seteseg <= seven_seg_letra;
219                     when 2 => sel <= "1011";
220                         selectdisp  <= selectdisp  + 1;
221                         letra <= mensagem_final(contador_disp3);
222                         seteseg <= seven_seg_letra;
223                     when 3 => sel <= "0111";
224                         selectdisp <= 0;
225                         letra <= mensagem_final(contador_disp4);
226                         seteseg <= seven_seg_letra;
227                     when others => sel <= "0000";
228                 end case;
229             end if;
230         end if ;
231 end process;
```

Figura 12: Condições necessárias para a mensagem final.

3.8 Contagem regressiva do tempo

Para que a contagem seja iniciada, a variável startcount precisa receber o valor '0', e para a implementação da função reset é feito alteração no valor dessa variável, ou seja, quando reset recebe 1, o startcout também recebe 1 para que a contagem seja parada e o contador volte ao estado inicial. Já a função de pausa é implementada juntamente com a lógica para a contagem regressiva que não inicia não continua a contagem se a variável pause for igual ao valor 1, porém, diferentemente da função reset, a função pause somente interrompe a contagem e não leva o cronometro ao estado inicial. Após o fim da contagem o startcount recebe o valor 1 novamente. As figura 13 e 14 representam a fração de código responsável pelo inicio da contagem, e funções reset e pausa.

```
234 ------------------------------------------------------------------------------------------
235 process(clk1hz,reset,start,estado,flag)
236     variable startcount: bit:='1'; --variável que ao receber '0' inicia a contagem
237 begin
238 if reset = '1' then --volta as condições iniciais se reset ='1'
239     q0       <= d0;
240     q1       <= d1;
241     q2       <= d2;
242     q3       <= d3;
243     startcount   := '1';
244     flag     <= '1';
245     flag_fim <= '0';
246 elsif reset = '0' then --tarefas executadas se reset = '0'
247 --faz com que os valores de q0, q1, q2 e q3 recebam o valor de d0, d1, d2 e d3
248 --em qualquer momento
249 if estado > 0 and flag = '1' and flag_fim = '0' and startcount = '1' then
250         q0 <= d0;
251         q1 <= d1;
252         q2 <= d2;
253         q3 <= d3;
254     elsif (start = '1' ) then --recebe as condições para o início da contagem
255         q0 <= d0;
256         q1 <= d1;
257         q2 <= d2;
258         q3 <= d3;
259         startcount := '0';
260         flag <= '0';
```

Figura 13: Estrutura de inicio da contagem e reset.

```
261  ------------------------------------------------------------------------------------------
262      elsif (start='0') then
263      --lógica utilizada para fazer a contagem regressiva
264          if (rising_edge(clk1hz) and pause = '0' )then
265              if (q0 /= 0 or q1 /= 0 or q2 /= 0 or q3 /= 0) and(startcount = '0') then
266                  if q3 = 0 then
267                      q3 <= 9;
268                      if q2 = 0 then
269                          q2 <= 5;
270                          if q1 = 0 then
271                              q1 <= 9;
272                              if q0 = 0 then
273                                  q0 <= 9;
274                              else
275                                  q0 <= q0 - 1;
276                              end if;
277                          else
278                              q1 <= q1 - 1;
279                          end if;
280                      else
281                          q2 <= q2 - 1;
282                      end if;
283                  else
284                      q3 <= q3 - 1;
285                  end if;
286      --indica o fim da contagem
287              elsif (q0 = 0 and q1 = 0 and q2 = 0 and q3 =0) and (startcount= '0') then
288                  q0 <= 0;
289                  q1 <= 0;
290                  q2 <= 0;
291                  q3 <= 0;
292                  startcount := '1';
293                  flag <= '1';
294                  flag_fim <= '1';
295              end if;
296          end if;
297      end if;
298  end if;
299  end process;
```

Figura 14: Estrutura de inicio da contagem e reset.

3.9 Seleção do tempo

Para que seja feita a seleção do tempo, foi usado uma estrutura que apartir do valor do estado, que possui 9 estados para os 7 tempos, sendo o estado 8 semelhante ao estado 0. As figuras 15 e 16 representam a fração do código da seleção do tempo.

```
357  ----------------------ATRIBUÍ VALORES A d0, d1, d2 E d3 BASEADO NO VALOR DE ESTADO----------------------------
358  process( estado )
359  begin
360  if estado=0 then --valor 00:00
361      d0<=0;
362      d1<=0;
363      d2<=0;
364      d3<=0;
365  end if ;
366  if estado=1 then --valor 10:00
367      d0<=1;
368      d1<=0;
369      d2<=0;
370      d3<=0;
371  end if ;
372  if estado=2 then --valor 15:00
373      d0<=1;
374      d1<=5;
375      d2<=0;
376      d3<=0;
377  end if ;
378  if estado=3 then --valor 20:00
379      d0<=2;
380      d1<=0;
381      d2<=0;
382      d3<=0;
383  end if ;
384  if estado=4 then --valor 30:00
385      d0<=3;
386      d1<=0;
387      d2<=0;
388      d3<=0;
389  end if ;
390  if estado=5 then --valor 40:00
391      d0<=4;
392      d1<=0;
393      d2<=0;
394      d3<=0;
395  end if ;
```

Figura 15: Fração do código contendo os estados.

```
397  if estado=6 then --valor 50:00
398      d0<=5;
399      d1<=0;
400      d2<=0;
401      d3<=0;
402  end if ;
403  if estado=7 then --valor 60:00
404      d0<=6;
405      d1<=0;
406      d2<=0;
407      d3<=0;
408  end if ;
409  if estado=8 then --valor 00:00
410      d0<=0;
411      d1<=0;
412      d2<=0;
413      d3<=0;
414  end if ;
415  end process ;
416  end hardware;
```

Figura 16: Fração do código contendo os estados.

3.10 VGA

Para implementação do Video Graphics Array (VGA) foi necessário um component. Com o VGA é possível gerar imagens de 640x480 pixels ou 320x240 pixels e são usados 8 bits para cada paleta RGB possuindo assim 262.144 possibilidades de cores.Atualmente o VGA é capaz de atingir resoluções muito maiores com o padrão SVGA que possível ter 800x600 pixels ou com o XGA 1024x768. No padrão VGA são utilizados 5 sinais de controle no conector, o modelo do conector é definido padrão VGA independente de ser VGA, SVGA ou XGA. A pinagem do conector presente no kit segundo o manual é representada pela figura 17 (NEXYS2. . . ,):

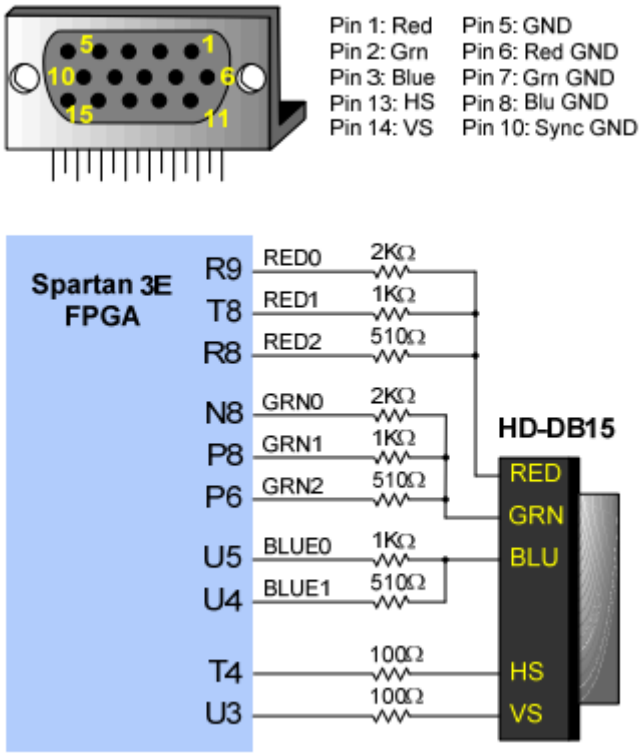

Figura 17: Fração do código contendo os estados.

Para geração das cores foram usados vetores de 8 bits que representa a paleta de cores, que combinadas podem gerar cores intermediárias. Para mostrar os números na tela, foram feitas matrizes de '0's e '1's de acordo com cada valor de 0 a 9, e também um ponto para separar horas e minutos.Após isso foi feito uma estrutura de seleção de cada matriz dessa de acordo com os números presentes na contagem, sincronizado com o código principal, fazendo assim que a mesma contagem mostrada nos displays de 7 segmentos seja visualizada no VGA.

Para o código do VGA foram declaradas as bibliotecas e portas de entrada e saída representadas pela figura 18

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity VGA is port (
5
6              clk          : in std_logic;
7              clk1hz           : in std_logic;
8              q0,q1,q3,d0,d1,d3     : in integer range 0 to 9;
9              q2,d2    : in integer range 0 to 5;
10             estado : in integer range 0 to 8;
11             flag         : in std_logic;
12             flag_fim         : in std_logic;
13             sync_h       : out std_logic;
14             sync_v       : out std_logic;
15             corestovga       : out std_logic_vector(7 downto 0)
16             );
17 end VGA;
18
19 architecture hardware of VGA is
20 ------------------------DEFINIÇÃO DAS MATRIZES-----------------------------------------
21 type digito is array (0 to 97) of std_logic_vector (0 to 63);
22 type ponto is array (0 to 97) of std_logic_vector (0 to 63);
```

Figura 18: Exemplo de matriz para representação dos números.

A figura 19 representa como foram estruturadas as matrizes de '0's e '1's que foram usadas para representar cada numero ou divisão na saída VGA.

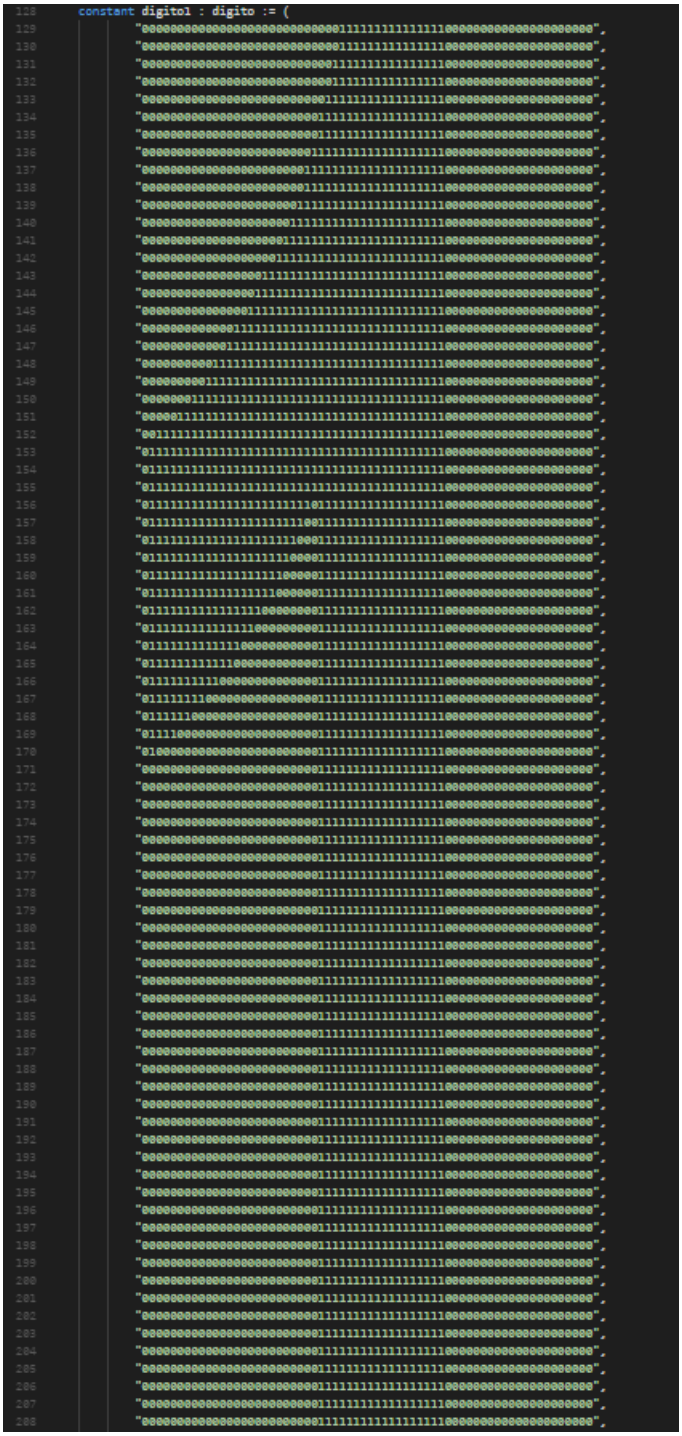

Figura 19: Exemplo de matriz para representação dos números.

Após isso foram declarados os sinais e variáveis necessárias para que seja feita a contagem dos números no display VGA. A figura 20 representa a fração do código que contem a declaração das variáveis e sinais.

```
1137  ------------------------SINAIS DE CORES UTILIZADOS-----------------------------------------
1138  signal color_red            : std_logic_vector (7 downto 0) :=("11100000");
1139  constant color_white            : std_logic_vector (7 downto 0) :=("11111111");
1140  ------------------------CONFIGURAÇÃO 800 X 600---------------------------------------------
1141  ------------------------CONFIGURAÇÃO HORIZONTAL---------------------------------------------
1142  constant h_area         : integer :=800;
1143  constant h_frontporch   : integer :=40;
1144  constant h_syncpulse        : integer :=128;
1145  constant h_backporch        : integer :=88;
1146  constant h_wholeline        : integer :=1056;
1147  ------------------------CONFIGURAÇÃO VERTICAL-----------------------------------------------
1148  constant v_area             : integer :=600;
1149  constant v_frontporch   : integer :=1;
1150  constant v_syncpulse    : integer :=4;
1151  constant v_backporch        : integer :=23;
1152  constant v_wholeline        : integer :=628;
1153  constant h_ajuste       : integer:= 216;
1154  constant v_ajuste           : integer:= 27;
1155
1156  begin
1157
1158      process (clk)
1159  ------------------------VARIÁVEIS DE CONTAGEM-----------------------------------------------
1160      variable horizontal : integer :=0;
1161      variable vertical   : integer :=0;
1162  ------------------------VARIÁVEIS PARA MOSTRAR OS NÚMEROS-----------------------------------
1163      variable matriz_digitoq0 : digito;
1164      variable matriz_digitoq1 : digito;
1165      variable matriz_digitoq2 : digito;
1166      variable matriz_digitoq3 : digito;
```

Figura 20: Declaração dos sinais e variáveis do VGA.

Após isso foi implementado uma estrutura para mostrar os números na tela, a qual define também a localização dos números na tela, e juntamente com isso foi implementado que se o tempo for menor que dois minutos(q0=0 e q1<2), a cor atribuída aos números é vermelha como sinal de alerta para o tempo acabar. essa estrutura pode ser representada pela figura 21:

```
1156  begin
1157      process (clk)
1158  ------------------------VARIÁVEIS DE CONTAGEM-----------------------------------------------
1159      variable horizontal : integer :=0;
1160      variable vertical   : integer :=0;
1161  ------------------------VARIÁVEIS PARA MOSTRAR OS NÚMEROS----------------------------------
1162      variable matriz_digitoq0 : digito;
1163      variable matriz_digitoq1 : digito;
1164      variable matriz_digitoq2 : digito;
1165      variable matriz_digitoq3 : digito;
1166
1167  begin
1168  if rising_edge(clk) then
1169      if (horizontal >= h_syncpulse+h_backporch ) and (horizontal < h_syncpulse+h_backporch+h_area )
1170      and (vertical >= v_syncpulse+v_backporch )and (vertical < v_syncpulse+v_backporch+v_area )then
1171          if (estado < 8) then
1172  ------------------------MOSTRA NUMEROS Q0 Q1 Q2 Q3 EM POSIÇÕES DIFERENTES-------------------------------------
1173  ------------------------VARREDURA DE TODO O MONITOR NA RESOLUÇÃO 800X600--------------------------------------------
1174              if ((vertical>=260+v_ajuste)and(vertical<=460+v_ajuste)and(horizontal>=200+h_ajuste)and(horizontal<=300+h_ajuste)) then
1175  ------------------------ESCREVE DIGITO Q0-----------------------------------------------------------------------
1176                  if(matriz_digitoq0(vertical-(310+v_ajuste))(horizontal-(215+h_ajuste))='1') then
1177                      if (q0 = 0 and q1 < 2) then
1178                      corestovga <= color_red;
1179                      else
1180                      corestovga <= color_white;
1181                      end if;
1182                  else
1183                      corestovga<="00000000"; --COR DO FUNDO GERAL DO BLOCO (PRETO)
1184                  end if;
1185  ------------------------ESCREVE DIGITO Q1-----------------------------------------------------------------------
1186              elsif ((vertical>=260+v_ajuste)and(vertical<=460+v_ajuste)and(horizontal>=320+h_ajuste)and(horizontal<=420+h_ajuste)) then
1187                  if(matriz_digitoq1(vertical-(310+v_ajuste))(horizontal-(335+h_ajuste))='1') then
1188                      if (q0 = 0 and q1 < 2) then
1189                          corestovga <= color_red;
1190                      else
1191                          corestovga <= color_white;
1192                      end if;
1193                  else
1194                      corestovga<="00000000"; --COR DO FUNDO GERAL DO BLOCO (PRETO)
1195                  end if;
```

Figura 21: Estrutura para exibição dos números no display.

Para que seja feita a atualização dos valores e varredura do monitor foi criada uma estrutura que além disso, faz a ligação do contador com os números que são mostrados nos displays do kit. Isso foi feito por uma estrutura "case"a qual associa cada numero da contagem à uma das matrizes. Essa estrutura pode ser representada pela fração de código da figura 22.

```
1237    ------------------------ATUALIZAÇÃO DOS VALORES PARA VARREDURA DO MONITOR-----------------
1238        if (horizontal > 0 )and (horizontal < h_syncpulse+1 )   then
1239            sync_h <= '0';
1240        else
1241            sync_h <= '1';
1242        end if;
1243
1244        if (vertical > 0 ) and (vertical < v_syncpulse+1 )then
1245            sync_v <= '0';
1246        else
1247            sync_v <= '1';
1248        end if;
1249
1250        horizontal := horizontal+1;
1251
1252        if (horizontal=h_wholeline) then
1253            vertical := vertical+1;
1254            horizontal := 0;
1255        end if;
1256
1257        if (vertical=v_wholeline) then
1258            vertical := 0;
1259        end if;
1260    end if;
1261
1262        case q0 is
1263            when 0 =>matriz_digitoq0:=digito0;
1264            when 1 =>matriz_digitoq0:=digito1;
1265            when 2 =>matriz_digitoq0:=digito2;
1266            when 3 =>matriz_digitoq0:=digito3;
1267            when 4 =>matriz_digitoq0:=digito4;
1268            when 5 =>matriz_digitoq0:=digito5;
1269            when 6 =>matriz_digitoq0:=digito6;
1270            when 7 =>matriz_digitoq0:=digito7;
1271            when 8 =>matriz_digitoq0:=digito8;
1272            when 9 =>matriz_digitoq0:=digito9;
1273            when others=>matriz_digitoq0:=digito0;
1274        end case;
```

Figura 22: Estrutura de atualização e seleção de matrizes para cada número.

Após o termino do tempo outro processo entra em execução fazendo com que a cor atribuída aos números seja trocada a cada 1 segundo, alterando o valor do vetor de 8 bits entre a configuração de vermelho e qualquer outra cor. No caso desse projeto foi escolhido o branco. Essa estrutura pode ser representada pela figura 23:

```
1318  ------------------------PROCESSO PARA PISCAR NUMEROS QUANDO ESGOTA O TEMPO----------------------------
1319  process(clk1hz)
1320      variable cont: integer:=0;
1321  begin
1322     if rising_edge(clk1hz) then
1323          if cont = 0 then     --CONT VARIANDO DE 0 PARA 1 A CADA PULSO DE CLOCK
1324          cont := cont+1;
1325          elsif cont > 0 then
1326              cont := 0;
1327        end if;
1328      end if;
1329  -------------------SE O TEMPO ESGOTAR O VALOR DE COLOR RED IRA MUDAR DE ACORDO COM CONT----------------
1330      if (q0 = 0 and q1 = 0 and q2 = 0 and q3 = 0) and flag_fim ='1' then
1331          if cont = 0 then
1332              color_red <= "11111111";
1333          elsif cont = 1 then
1334              color_red <= "11100000";
1335          end if;
1336      end if;
1337  end process;
1338
1339  end hardware;
```

Figura 23: Estrutura para função de alternar as cores dos numeros mostrados na tela ao término da contagem/.

3.11 Debounce

Para fazer o debounce do botão foi usado um process que foi baseado em um material da web (DEBOUNCE. . . ,), e seu código pode ser representado pela figura 24:

```
309  ------------------------PROCESSO DE DEBOUNCE-----------------------------------------------------------------------
310  counter_set <= flipflops(0) xor flipflops(1); --inicia contador
311  process(clk)
312      BEGIN
313          if rising_edge(clk) then
314              flipflops(0) <= botao_estado; --atribui o valor do botão "0" ou "1" a posiçaoo 0 do vetor flipflop
315              flipflops(1) <= flipflops(0); --atribui o valor da posiçaoo 00 do flipflop á posição 1
316              if(counter_set = '1') then --reset se a entrada for alterada
317                  counter_out <= (others => '0');
318              elsif(counter_out(2) = '0') then -- condição da entrada não foi cumprida
319                  counter_out <= counter_out + 1;
320          else --condição  entrada foi cumprida
321              out_debounce <= flipflops(1);
322              end if;
323          end if;
324  end process;
```

Figura 24: Fração do código contendo o process do debouce do botão.

4 Mapeamento I/O

Ao final do projeto, o mapeamento feito para a implementação na placa do circuito pode ser representado pela figura 25:

```
--SINAL DE CLOCK
NET clk LOC = "B8";
--SELECIONA ANODO DOS DISPLAYS
NET sel(1) LOC = "F17";
NET sel(2) LOC = "H17";
NET sel(3) LOC = "C18";
NET sel(0) LOC = "F15";
--LEDS DOS DISPLAYS
NET seteseg(6) LOC = "L18";
NET seteseg(5) LOC = "F18";
NET seteseg(4) LOC = "D17";
NET seteseg(3) LOC = "D16";
NET seteseg(2) LOC = "G14";
NET seteseg(1) LOC = "J17";
NET seteseg(0) LOC = "H14";
--BOTÕES E CHAVES
NET reset           LOC = "B18";
NET pause           LOC = "G18";
NET start           LOC = "E18";
NET botao_estado    LOC = "D18";
--DECLARAÇÃO DOS PINOS DO VGA
NET corestovga(7)   LOC = "R8";
NET corestovga(6)   LOC = "T8";
NET corestovga(5)   LOC = "R9";
NET corestovga(4)   LOC = "N8";
NET corestovga(3)   LOC = "P8";
NET corestovga(2)   LOC = "P6";
NET corestovga(1)   LOC = "U5";
NET corestovga(0)   LOC = "U4";
NET hsinc           LOC = "T4";
NET vsinc           LOC = "U3";
```

Figura 25: Mapeamento I/O usado para implementação do programa na placa.

5 Resultados e Simulação

5.1 Simulação

A simulação representando a troca de estados (10-15-20-30-40-50-60) pode ser vista na figura 26:

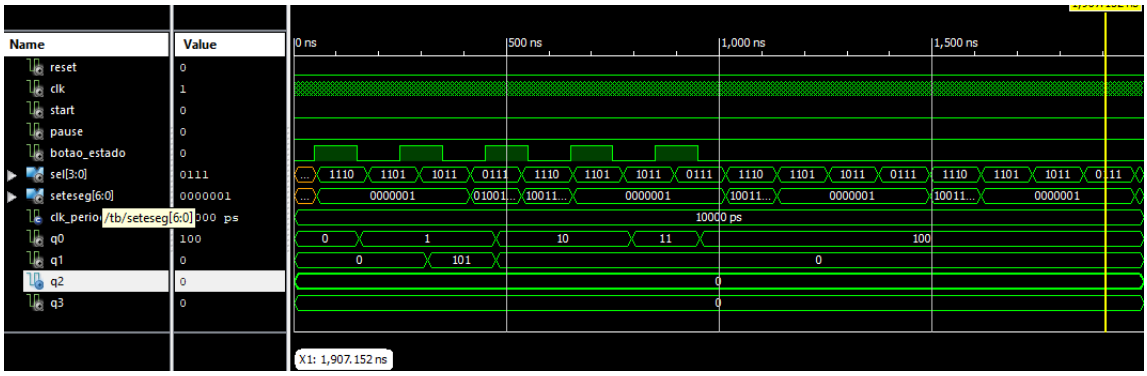

Figura 26: Simulação representando a passagem dos estados.

A figura 27 a seguir demonstra o não funcionamento do botão start apenas no testbench. Era esperado que os valores de q0, q1, q2 e q3 começassem a ser decrementados após o botão start.

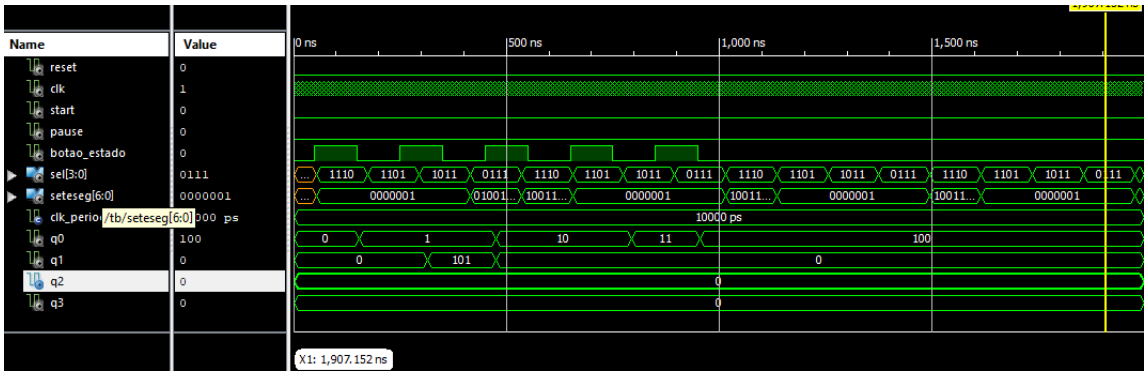

Figura 27: Simulação representando o não funcionamento do botão start somente no testbench.

A figura 28 representa a simulação do não funcionamento do botão reset do contador somente no testbench.

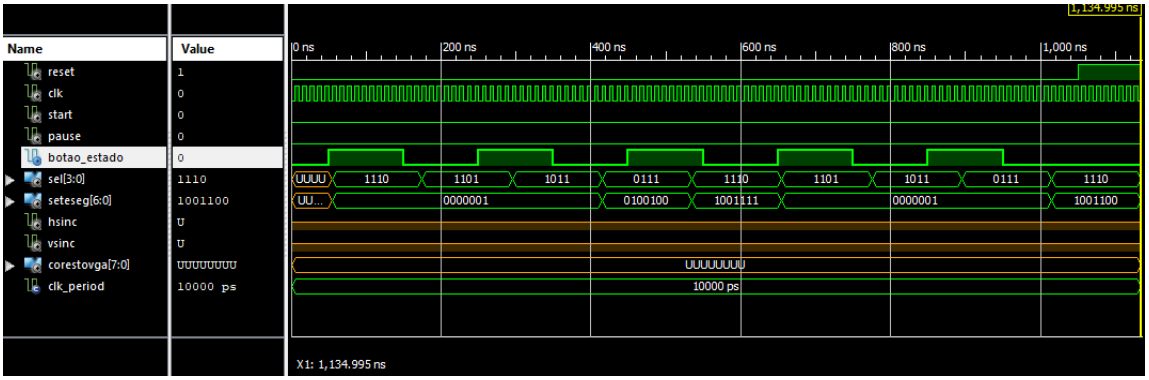

Figura 28: Simulação representando o não funcionamento do botão de reset somente no testbench.

Quando é simulado o reset a simulação é interrompida por uma erro, e retorna a seguinte mensagem: (ERROR: In process main.vhd:94) e foi visto que neste processo no codigo principal está a seleção do que é mostrado nos displays de 7 segmentos do kit.

5.2 Funcionamento

As figura 29 representa o programa em funcionamento em um monitor VGA, na fase de seleção de tempo.

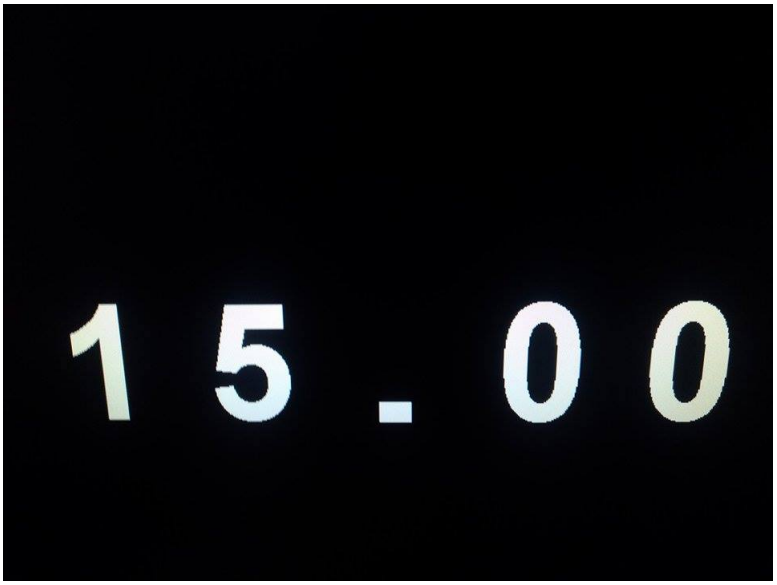

Figura 29: Monitor VGA na fase de seleção de tempo, na figura aos 15 minutos

A figura 30 representa o monitor durante uma contagem, no tempo de 19 minutos e 54 segundos.

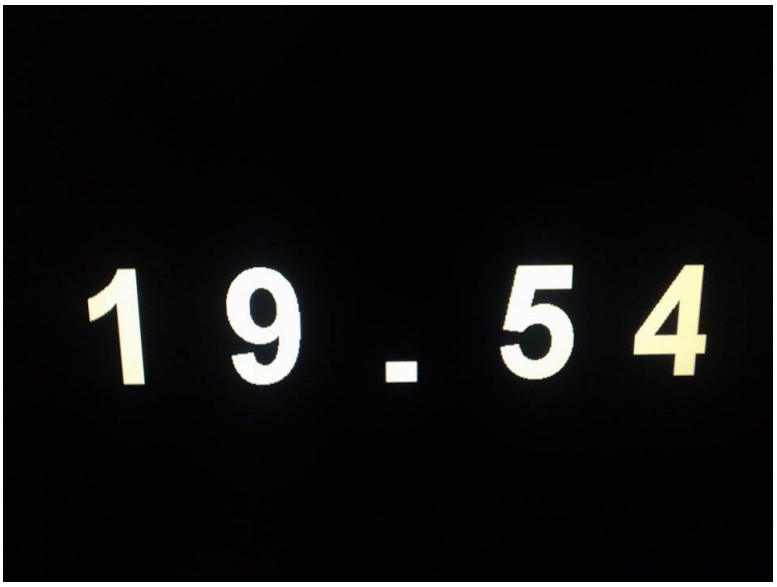

Figura 30: Monitor VGA durante a contagem de tempo, na figura aos 19 minutos e 54 segundos.

A figura 31 representa a contagem em tempo abaixo de 2 minutos, com a cor vermelha demonstrando estado de tempo acabando.

Figura 31: Monitor VGA durante a contagem de tempo, na figura à 1 minuto e 19 segundos, mostrando os números já em vermelho.

As figuras 32 e 33 representam a cor dos números após o fim da contagem alternando as cores entre vermelho e branco.

Figura 32: Monitor VGA ao final da contagem alternando as cores dos números, no momento em vermelho.

Figura 33: Monitor VGA ao final da contagem alternando as cores dos números, no momento em branco.

6 Desafios de execução

No decorrer do desenvolvimento desta atividade foram superadas várias dificuldades para se chegar a conclusão do mesmo, primeiramente foi desenvolvida uma lógica para fazer a troca dos displays o mesmo foi resolvidos com a utilização de contadores que são incrementados a cada um segundo com o clock de 1hz m após esse primeiro processo era necessário implementar na linguagem VHDL um contador regressivo, esse processo foi desafiador e para sua conclusão foi utilizado outra linguagem de programação, o python. O código do contador regressivo em python pode ser representado pela figura 34

```python
import time
#contador regressivo
mind=6
minu=0
segd=0
segu=0
final=False

for i in range (0,3600):
  time.sleep(1) #delay de 1 segundo

  if (mind != 0 or minu != 0 or segd != 0 or segu != 0) and (final==False) :
    if segu == 0:
      segu = 9
      if segd == 0:
        segd = 5
        if minu == 0:
          minu = 9
          if mind == 0:
            mind = 5
          else:
            mind=mind-1
        else:
          minu=minu-1
      else:
        segd=segd-1
    else:
      segu=segu-1
  print(mind,minu,segd,segu)
```

Figura 34: Código em python do contador regressivo

Após o funcionamento do contador regressivo nessa linguagem se teve que implementar o mesmo algoritmo utilizando a sintaxe da linguagem VHDL, se notou então que era necessário outros sinais para representar os dígitos do contador para que se pudesse fazer mudança das opções de tempo pois não era possível trabalhar com os mesmos sinais

utilizados na contagem , isso fez com que fosse necessário a criação de 4 momentos de decodificação dos displays de sete segmentos (mensagem inicial , troca de tempos , contagem regressiva e mensagem final). Como não foi utilizado máquina de estados foram criadas várias flags que determinavam qual tarefa estava acontecendo, assim determinando o que era mostrado nos displays. Chegamos então no momento de enviar todos os valores que estavam no código principal para o componente VGA a princípio era necessário enviar oito valores para o vga se notou que poderíamos com a alteração do código principal enviar apenas 4 , após esse problema resolvido o tempo de compilação diminuiu fazendo com que a parte de ajuste de cores e posicionamento dos valores fosse mais produtivo.

Conclusão

O projeto final, tem como referencia aplicar os conhecimentos obtidos durante a disciplina, e essa referencia foi usada significativamente no desenvolvimento desse projeto que aborda desde os conceitos mais básicos até os mais complexos de FPGA's e VHDL abordados em sala. Para a estruturação do trabalho foi necessário consultar materiais dos laboratórios anteriores, como por exemplo components de clock e traze-los à esse projeto adaptando à essa aplicação, embora sejam semelhantes, e livros do tema como Circuit Design With VHDL escrito por PEDRONI.

Além dos conhecimentos obtidos, foi proposto como desafio a implementação do VGA, que embora não tenha sido abordado em sala, só foi possível com o embasamento obtido na linguagem de programação VHDL já adquirido sobre outros temas. O Projeto se mostrou importante também no sentido de poder extrair mais os recursos presentes no kit Nexys2 ou qualquer outra interface para uma eventual aplicação que pode ser aprimorada, se torne um produto que chegue a um consumidor final.

Referências

DEBOUNCE Logic Circuit (with VHDL example). https://eewiki.net/pages/viewpage.action?pageId=4980758. Citado na página 26.

NEXYS2 Reference Manual. https://reference.digilentinc.com/reference/programmable-logic/nexys-2/reference-manual. Citado 2 vezes nas páginas 10 e 20.

PEDRONI, V. A. *Circuit Design With VHDL*. 1st. ed. [S.l.]: Massachusetts Institute of Technology, 2004. ISBN 0-262-16224-5. Citado na página 35.