TRABAJO VHDL SED 2023/2024: CRONÓMETRO

Grupo 6

Realizado por:

Javier Robinat Cuiñas 55430

Fernando Marín Raez 54724

Álvaro Rico García 55425

Repositorio Github: https://github.com/Alvarorico/Crono/tree/main

Índice

Introducción

El objetivo del trabajo es realizar un cronómetro y un temporizador utilizando VHDL como lenguaje de descripción Hardware e implementarlo en una FPGA Nexys DDR4.

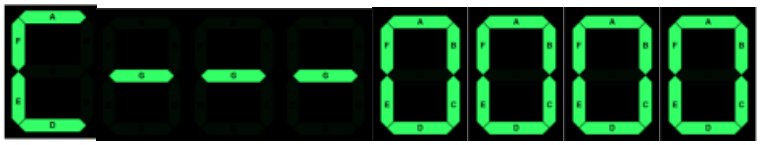
Las entradas del diseño serán los propios botones de la placa. Debido a la cantidad de ruido que pueden generar, se implementará un sincronizador para reducir el ruido y evitar los rebotes que se pudiesen generar.

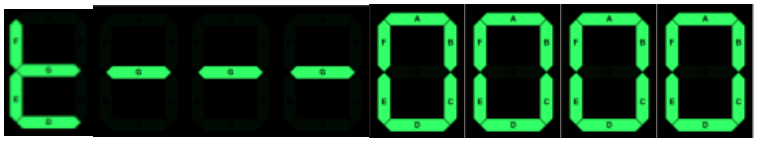
A través de 4 de los 8 dispalys de 7 segmentos se mostrará el tiempo transcurrido, de forma ascendente en el modo cronómetro y descendente en el modo temporizador, desde el momento en el que se haya pulsado el botón “Start”.

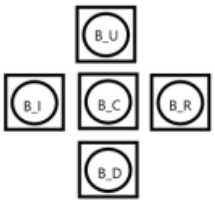
Por otro lado, en los 4 displays restantes se mostrará permanentemente “C---“o “t---“en función del modo en el que se encuentre.

Instrucciones generales

Interfaz:



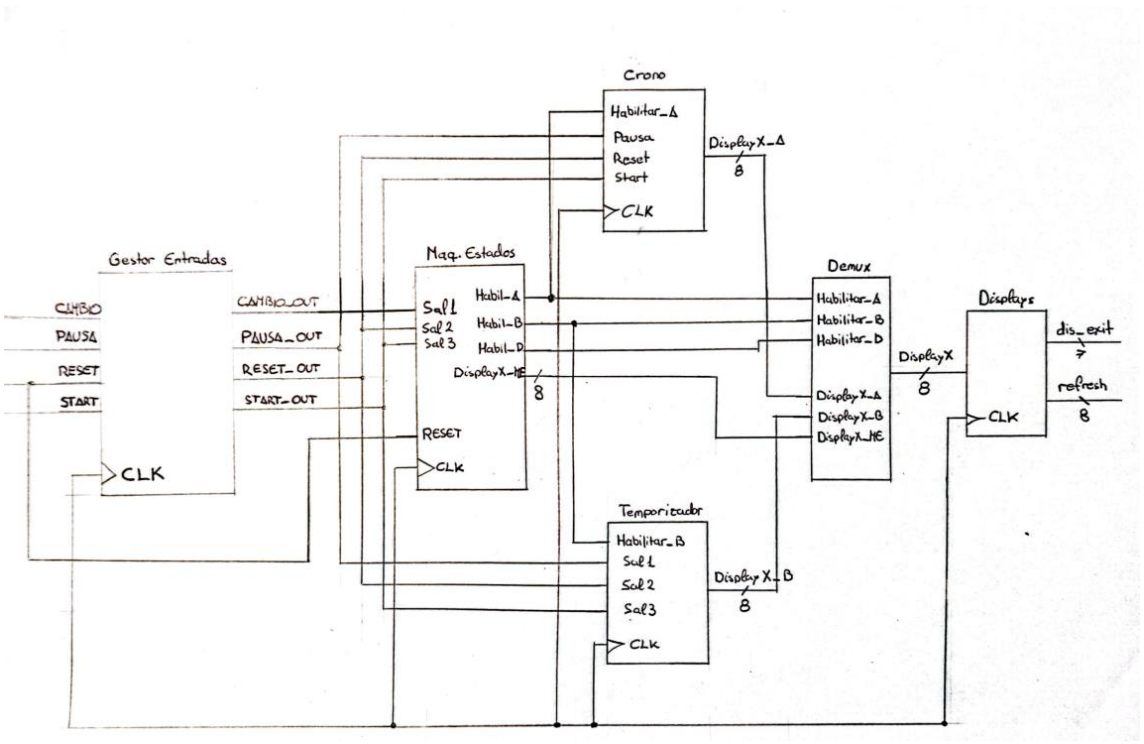




Displays	
1	Unidades de segundo
2	Decenas de segundo
3	Unidades de minuto
4	Decenas de minuto

Botonera	
B_U	RESET
B_D	CAMBIO DE MODO
B_R	PAUSA
B_I	START
B_C	-

Diagrama general



Estrategias y algoritmos

Para una correcta organización del proyecto, nos repartimos roles para que la forma de trabajar fuese más eficiente.

-Programadores: Álvaro Rico García y Javier Robinat Cuiñas

-Arquitecto y responsable del correcto funcionamiento: Fernando Marín Raez

Una vez asignados los roles, para el desarrollo del trabajo se plantea una metodología Top-Down que consiste en la división del proyecto en bloques hasta llegar a las partes más sencillas.

En primer lugar, se realizará el gestor de entradas, que a través de un detector de flancos y un sincronizador por cada entrada, dará como resultado una señal con mayor inmunidad al ruido.

Después, se realizarán el cronometro y el temporizador de forma separada, una vez comprobado su correcto funcionamiento por separado en la placa, se implementarán mediante una máquina de estados y usando un demultiplexor para combinarlos.

Por último, se creará el módulo Top que contendrá todo el trabajo con todo perfectamente sincronizado y agrupado.

Descripción VHDL de los bloques funcionales

Gestor de entradas:

Como se ha mencionado en el apartado anterior el gestor de entradas contará con un detector de flancos y un sincronizador por entrada.

Sincronizador:

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity SINCRONIZADOR is
5   port (
6      CLK : in std_logic;
7      ASYNC_IN : in std_logic;
8      SYNC_OUT : out std_logic
9   );
10 end SINCRONIZADOR;
11
12 architecture BEHAVIORAL of SINCRONIZADOR is
13     signal sreg : std_logic_vector(1 downto 0);
14 begin
15
16     process (CLK)
17     begin
18         if rising_edge(CLK) then
19             sync_out <= sreg(1);
20             sreg <= sreg(0) & async_in;
21         end if;
22     end process;
23 end BEHAVIORAL;
24
```

Detector de flancos:

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity FLANCO is
5   port (
6   CLK : in std_logic;
7   SYNC_IN : in std_logic;
8   EDGE : out std_logic
9   );
10 end FLANCO;
11
12 architecture BEHAVIORAL of FLANCO is
13  signal sreg : std_logic_vector(2 downto 0);
14 begin
15  process (CLK)
16  begin
17  if rising_edge(CLK) then
18  sreg <= sreg(1 downto 0) & SYNC_IN;
19  end if;
20  end process;
21  with sreg select
22  EDGE <= '1' when "100",
23  '0' when others;
24 end BEHAVIORAL;
25
```

Descripción VHDL del gestor de entradas:

```
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4
5  entity Entradas is
6   Port (
7          CLK       : in std_logic;
8          INICIO    : in std_logic;  --BTNL
9          STOP       : in std_logic;  --BTNR
10         ZERO       : in std_logic;  --BTNU
11         CAMBIO    : IN STD_LOGIC;   --BTND
12         reset     : in std_logic;
13         INICIO_out   : out std_logic;
14         STOP_out   : out std_logic;
15         ZERO_out   : out std_logic;
16         CAMBIO_out   : out std_logic;
17         reset_out : out std_logic
18     );
19 end Entradas;
20
21 architecture Behavioral of Entradas is
22
23     signal sync_auxINICIO : std_logic;
24     signal sync_auxSTOP: std_logic;
25     signal sync_auxZERO: std_logic;
26     signal sync_auxCAMBIO: std_logic;
27
28     signal sync_auxReset: std_logic;
31     COMPONENT SINCRONIZADOR
32        PORT (
33               async_in : IN std_logic;
34               clk: IN std_logic;
35               sync_out : OUT std_logic
36        );
37     END COMPONENT;
38
39     COMPONENT FLANCO
40        PORT (
41               sync_in : IN std_logic;
42               clk: IN std_logic;
43               edge : OUT std_logic
44        );
45     END COMPONENT;
46
47  begin
48     Sincronizador1: SINCRONIZADOR
49     PORT MAP(
50         ASYNC_IN=>INICIO,
51         CLK=>clk,
52         SYNC_OUT=>sync_auxINICIO
53     );
54
55     DetectorFlanco: FLANCO
56     PORT MAP(
57         clk=>clk,
58         SYNC_IN=>sync_auxINICIO,
59         EDGE=>INICIO_out
60     );
```

```
    Sincronizador2: SINCRONIZADOR
    PORT MAP(
        ASYNC_IN=>STOP,
        CLK=>clk,
        SYNC_OUT=>sync_auxSTOP
    );

    DetectorFlanco2: FLANCO
    PORT MAP(
        clk=>clk,
        SYNC_IN=>sync_auxSTOP,
        EDGE=>STOP_out
    );

    Sincronizador3: SINCRONIZADOR
    PORT MAP(
        ASYNC_IN=>ZERO,
        CLK=>clk,
        SYNC_OUT=>sync_auxZERO
    );

    DetectorFlanco3: FLANCO
    PORT MAP(
        clk=>clk,
        SYNC_IN=>sync_auxZERO,
        EDGE=>ZERO_out
    );

     Sincronizador4: SINCRONIZADOR
     PORT MAP(
         ASYNC_IN=>reset,
         CLK=>clk,
         SYNC_OUT=>sync_auxReset
     );

     DetectorFlanco4: FLANCO
     PORT MAP(
         clk=>clk,
         SYNC_IN=>sync_auxReset,
         EDGE=>reset_out
     );

      Sincronizador5: SINCRONIZADOR
     PORT MAP(
         ASYNC_IN=>CAMBIO,
         CLK=>clk,
         SYNC_OUT=>sync_auxCAMBIO
     );

     DetectorFlanco5: FLANCO
     PORT MAP(
         clk=>clk,
         SYNC_IN=>sync_auxCAMBIO,
         EDGE=>CAMBIO_out
     );

end Behavioral;
```

Modo cronómetro:

Preescaler:

Previo a realizar el modo cronometro, es necesario introducir el componente “clk1hz” que nos proporciona una señal de 1 Hz teniendo como entrada el reloj de la placa. La operación se realiza mediante un contador.

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity clk1Hz is
5      Port (
6          CLK: in  STD_LOGIC;
7          CLK_1hz : out STD_LOGIC
8      );
9  end clk1Hz;
10
11 architecture Behavioral of clk1Hz is
12     signal temporal: STD_LOGIC := '0';
13     signal contador: integer range 0 to 49999999 := 0;
14 begin
15     divisor_frecuencia: process (CLK) begin
16         if rising_edge(CLK) then
17             if (contador = 49999999) then
18                 temporal <= NOT(temporal);
19                 contador <= 0;
20             else
21                 contador <= contador+1;
22             end if;
23         end if;
24     end process;
25
26     CLK_1hz <= temporal;
27
28 end Behavioral;
29
```

Entidad modo cronómetro:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.all;

entity Crono is
generic(
        width:positive:=3
        );
    Port (
        CLK : in std_logic;
        display1 : out std_logic_vector(width downto 0);
        display2 : out std_logic_vector(width downto 0);
        display3 : out std_logic_vector(width downto 0);
        display4 : out std_logic_vector(width downto 0);
        display5 : out std_logic_vector(width downto 0);
        display6 : out std_logic_vector(width downto 0);
        display7 : out std_logic_vector(width downto 0);
        display8 : out std_logic_vector(width downto 0);
        Habilitar_A : in std_logic;
        Start : in std_logic;
        Pause : in std_logic;
        Reset : in std_logic
     );
end Crono;

architecture Behavioral of Crono is
    signal Start_i : std_logic :='0';
    signal Reset_i : std_logic :='0';

    signal clk_1hz : std_logic;

    COMPONENT clk1hz
       PORT (
              CLK: in  STD_LOGIC;
              clk_1hz : out STD_LOGIC
            );
       END COMPONENT;

    begin
    Inst_clk1hz: clk1hz
    PORT MAP (
        CLK => CLK,
        CLK_1hz => clk_1hz
    );
```

Las señales Start_i y Rset_i son los estados que habilitan la cuenta o el reinicio, por otro lado, se utiliza la señal clk_1hz para contar segundo a segundo. También, se ha instanciado el preescaler “clk1hz”.

```
47      ME : process (Habilitar_A,Start,Pause,Reset)
48      begin
49          if Habilitar_A = '1' and Start = '1' and Pause = '0'then
50              Start_i<='1';
51              Reset_i<='0';
52          elsif Pause='1' then
53              Start_i<='0';
54              Reset_i<='0';
55          elsif Reset = '1'then
56              Reset_i<='1';
57          end if;
58      end process;
59
60      process (clk_1hz, Start_i, Reset_i)
61      subtype V is integer range 0 to 15;
62      variable unit_sec : V :=0;
63      variable unit_min : V :=0;
64      variable dec_sec  : V :=0;
65      variable dec_min  : V :=0;
66      begin
67
68          if Reset_i='1' then
69                  unit_sec:=0;
70                  dec_sec:=0;
71                  unit_min:=0;
72                  dec_min:=0;
73          elsif rising_edge(clk_1hz) and Start_i='1' then
74              unit_sec:=unit_sec+1;
75              if unit_sec=10 then
76              unit_sec:=0;
77              dec_sec:=dec_sec+1;
78                  if dec_sec=6 then
79                      dec_sec:=0;
80                      unit_min:=unit_min+1;
81                      if unit_min=10 then
82                          unit_min:=0;
83                          dec_min:=dec_min+1;
84                       if dec_min=9 then
85                          dec_min:=0;
86                          unit_min:=0;
87                          dec_sec:=0;
88                          unit_sec:=0;
89                       end if;
90                  end if;
91               end if;
92
93            end if;
94
95         end if;
96
97          display1 <= std_logic_vector(to_unsigned(unit_sec,display8'length));
98          display2 <= std_logic_vector(to_unsigned(dec_sec,display7'length));
99          display3 <= std_logic_vector(to_unsigned(unit_min,display6'length));
100         display4 <= std_logic_vector(to_unsigned(dec_min,display5'length));
101     end process;
102
103     Indicador : process (Habilitar_A)
104     begin
105     if Habilitar_A='1' then
106         display8<="1010";
107         display7<="1100";
108         display6<="1100";
109         display5<="1100";
110     end if;
111     end process;
112
113
114 end Behavioral;
```

En las imágenes anteriores, se muestra:

1- Máquina de estados:
   Máquina de estados asíncrona que tiene como condición necesaria para comenzar la cuenta las entradas de habilitación “Habilitar_A” y “Start”, además de que no se esté pulsando “Pausa”.
2- Process principal:
   En primer lugar, se ha definido un tipo de variable entero con rango de 0 a 15 (rango de los vectores de salida de 4 bits) para obtener un código más sencillo y fácil de ejecutar. Se crean cuatro variables correspondientes con las unidades y decenas de segundos y minutos.
   Teniendo la señal “Reset_i” prioritaria, después, en cada flanco de reloj se sumará un segundo, y se gestionarán los límites para que entre dentro de la normalidad.
   Para terminar, se convierten las variables utilizadas en vectores de 4 bits de salida que se mostrarán en los displays 1,2,3 y 4.
3- Process secundario:
   Se muestra de forma permanente “C---“en la salida mientras “Habilitar_A” esté activo.

Modo temporizador:

Está entidad sigue la misma estructura que la del modo cronómetro con la variación de:

1- La condición de habilitación de “Habilitar_A”, se sustituye por “Habilitar_B”, además de las ya explicadas en la entidad anterior.
2- En el process principal en vez de sumar uno, se resta.
3- En el process secundario se muestra “t---“y no “C---“.

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.all;

entity Inverso is
generic(
        width:positive:=3
        );
    Port (
        CLK : in std_logic;
        display1 : out std_logic_vector(width downto 0);
        display2 : out std_logic_vector(width downto 0);
        display3 : out std_logic_vector(width downto 0);
        display4 : out std_logic_vector(width downto 0);
        display5 : out std_logic_vector(width downto 0);
        display6 : out std_logic_vector(width downto 0);
        display7 : out std_logic_vector(width downto 0);
        display8 : out std_logic_vector(width downto 0);
        Habilitar_B : in std_logic;
        Start : in std_logic;
        Pause : in std_logic;
        Reset : in std_logic
     );
end Inverso;

architecture Behavioral of Inverso is
    signal Start_i : std_logic :='0';
    signal Reset_i : std_logic :='0';

    signal clk_1hz : std_logic;

    COMPONENT clk1hz
       PORT (
              CLK: in  STD_LOGIC;
              clk_1hz : out STD_LOGIC
            );
       END COMPONENT;

    begin
    Inst_clk1hz: clk1hz
    PORT MAP (
        CLK => CLK,
        CLK_1hz => clk_1hz
    );
```

```
    ME : process (Habilitar_B,Start,Pause,Reset)
    begin
        if Habilitar_B = '1' and Start = '1' and Pause = '0'then
            Start_i<='1';
            Reset_i<='0';
        elsif Pause='1' then
            Start_i<='0';
            Reset_i<='0';
        elsif Reset = '1'then
            Reset_i<='1';
        end if;
    end process;

    process (clk_1hz, Start_i, Reset_i)
    subtype V is integer range 0 to 15;
    variable unit_sec : V :=0;
    variable unit_min : V :=1;
    variable dec_sec  : V :=0;
    variable dec_min  : V :=0;
    begin

        if Reset_i='1' then --Reset prioritario
            unit_sec:=0;
            dec_sec:=0;
            unit_min:=1;
            dec_min:=0;

        elsif rising_edge(clk_1hz) and Start_i='1' then
            if unit_sec=0 then
                unit_sec:=9;
                if dec_sec=0 then
                    dec_sec:=5;
                    if unit_min=0 then
                        unit_min:=9;
                        if dec_min=0 then
                            dec_min:=0;
                        else
                            dec_min:=dec_min-1;
                        end if;
                    else
                        unit_min:=unit_min-1;
                end if;
              else
                  dec_sec:=dec_sec-1;
              end if;
          else
              unit_sec:=unit_sec-1;
          end if;
      end if;

        display1 <= std_logic_vector(to_unsigned(unit_sec,display8'length));
        display2 <= std_logic_vector(to_unsigned(dec_sec,display7'length));
        display3 <= std_logic_vector(to_unsigned(unit_min,display6'length));
        display4 <= std_logic_vector(to_unsigned(dec_min,display5'length));
    end process;

    Indicador : process (Habilitar_B)
    begin
    if Habilitar_B='1' then
        display8<="1011";
        display7<="1100";
        display6<="1100";
        display5<="1100";
    end if;
    end process;

end Behavioral;
```

Máquina de Estados:

Se trata de una máquina de Moore, ya que las salidas dependen solo del estado en el que se encuentre el sistema.

Descripción:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity MaquinaEstados is
 Port (
    CLK          : in std_logic;
    Boton1       : in std_logic;
    Boton2       : in std_logic;
    Boton3       : in std_logic;
    display1     : out std_logic_vector(3 downto 0);
    display2     : out std_logic_vector(3 downto 0);
    display3     : out std_logic_vector(3 downto 0);
    display4     : out std_logic_vector(3 downto 0);
    display5     : out std_logic_vector(3 downto 0);
    display6     : out std_logic_vector(3 downto 0);
    display7     : out std_logic_vector(3 downto 0);
    display8     : out std_logic_vector(3 downto 0);
    Habilitar_A : out std_logic :='0';
    Habilitar_B : out std_logic :='0';
    Habilitar_D : out std_logic :='1'
 );

end MaquinaEstados;
architecture Behavioral of MaquinaEstados is
  type estados is (crono_selec,tempo_selec,crono_func,tempo_func);
    signal actual_state,next_state:estados;
    signal clk_10khz : std_logic;

 component clk10khz
   PORT (
     CLK: in STD_LOGIC;
     clk_1hz : out STD_LOGIC
   );
 end component;
begin
 Inst_clk10khz: clk10khz
   PORT MAP (
     CLK => CLK,
     clk_1hz => clk_10khz
   );
 Salida:process(CLK)
   begin
     if rising_edge(CLK)then
       actual_state<=next_state;
     end if;
   end process;
```

```
 gestionmaquinaestados:process(actual_state,Boton1,Boton2,Boton3)
 begin
   next_state<=actual_state;

   case (actual_state)is
    when crono_selec=>
     if Boton2='1' then
       next_state<=tempo_selec;
      elsif Boton1='1' then
       next_state<=crono_func;
     end if;
     when crono_func=>
     if Boton3='1' then
       next_state<=crono_selec;
     end if;
    when tempo_selec=>
     if Boton2='1' then
       next_state<=crono_selec;
      elsif Boton1='1' then
       next_state<=tempo_func;
     end if;
    when tempo_func=>
     if Boton3='1' then
       next_state<=tempo_selec;
     end if;
    when others => next_state<=actual_state;
   end case;
 end process;
SalidasDisplays : process (actual_state)
 begin
   case(actual_state) is
    when crono_selec=>  --"C---0000"
      display1<="0000";
      display2<="0000";
      display3<="0000";
      display4<="0000";
      display5<="1100";
      display6<="1100";
      display7<="1100";
      display8<="1010";
      Habilitar_A<='0';
      Habilitar_B<='0';
      Habilitar_D<='1';
   when tempo_selec=>  --"t---0000"
      display1<="0000";
      display2<="0000";
      display3<="0000";
      display4<="0000";
      display5<="1100";
      display6<="1100";
      display7<="1100";
      display8<="1011";
      Habilitar_A<='0';
      Habilitar_B<='0';
      Habilitar_D<='1';
   when crono_func=>
      Habilitar_A<='1';
      Habilitar_B<='0';
      Habilitar_D<='0';
    when tempo_func=>
      Habilitar_A<='0';
      Habilitar_B<='1';
      Habilitar_D<='0';
   end case;
 end process;

end Behavioral;
```

Demux:

Es una entidad con un funcionamiento sencillo pese a su larga descripción, básicamente, en función de la entrada de habilitación que esté activada copia el valor de los vectores que corresponda en su salida.

Descripción:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity Demux is
  Port (
         Habilitar_A : in std_logic;
         Habilitar_B : in std_logic;
         Habilitar_D : in std_logic;

         Display1_ME : in std_logic_vector(3 downto 0);
         Display2_ME : in std_logic_vector(3 downto 0);
         Display3_ME : in std_logic_vector(3 downto 0);
         Display4_ME : in std_logic_vector(3 downto 0);
         Display5_ME : in std_logic_vector(3 downto 0);
         Display6_ME : in std_logic_vector(3 downto 0);
         Display7_ME : in std_logic_vector(3 downto 0);
         Display8_ME : in std_logic_vector(3 downto 0);

         Display1_C : in std_logic_vector(3 downto 0);
         Display2_C : in std_logic_vector(3 downto 0);
         Display3_C : in std_logic_vector(3 downto 0);
         Display4_C : in std_logic_vector(3 downto 0);
         Display5_C : in std_logic_vector(3 downto 0);
         Display6_C : in std_logic_vector(3 downto 0);
         Display7_C : in std_logic_vector(3 downto 0);
         Display8_C : in std_logic_vector(3 downto 0);

         Display1_T : in std_logic_vector(3 downto 0);
         Display2_T : in std_logic_vector(3 downto 0);
         Display3_T : in std_logic_vector(3 downto 0);
         Display4_T : in std_logic_vector(3 downto 0);
         Display5_T : in std_logic_vector(3 downto 0);
         Display6_T : in std_logic_vector(3 downto 0);
         Display7_T : in std_logic_vector(3 downto 0);
         Display8_T : in std_logic_vector(3 downto 0);

         Display1 : out std_logic_vector(3 downto 0);
         Display2 : out std_logic_vector(3 downto 0);
         Display3 : out std_logic_vector(3 downto 0);
         Display4 : out std_logic_vector(3 downto 0);
         Display5 : out std_logic_vector(3 downto 0);
         Display6 : out std_logic_vector(3 downto 0);
         Display7 : out std_logic_vector(3 downto 0);
         Display8 : out std_logic_vector(3 downto 0)
  );
end Demux;
architecture Behavioral of Demux is

begin
 process(Habilitar_A,Habilitar_B,Habilitar_D)
   begin
     if Habilitar_A='1' then
         Display1 <= Display1_C;
         Display2 <= Display2_C;
         Display3 <= Display3_C;
         Display4 <= Display4_C;
         Display5 <= Display5_C;
         Display6 <= Display6_C;
         Display7 <= Display7_C;
         Display8 <= Display8_C;
        elsif Habilitar_B='1' then
         Display1 <= Display1_T;
         Display2 <= Display2_T;
         Display3 <= Display3_T;
         Display4 <= Display4_T;
         Display5 <= Display5_T;
         Display6 <= Display6_T;
         Display7 <= Display7_T;
         Display8 <= Display8_T;
        elsif Habilitar_D='1' then
         Display1 <= Display1_ME;
         Display2 <= Display2_ME;
         Display3 <= Display3_ME;
         Display4 <= Display4_ME;
         Display5 <= Display5_ME;
         Display6 <= Display6_ME;
         Display7 <= Display7_ME;
         Display8 <= Display8_ME;
     end if;
 end process;
end Behavioral;
```

Displays:

Esta entidad contará con 8 vectores de 4 bits como entradas que se corresponderán con lo que debe mostrar cada display.

Como salidas se tiene refresh para poner a nivel alto o bajo los ánodos de cada display, y dis_exit que enviará el código que corresponda a los displays.

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Displays is
generic(
        width:positive:=3
        );
    port(
        CLK : in std_logic;
        display1 : in std_logic_vector(width downto 0);
        display2 : in std_logic_vector(width downto 0);
        display3 : in std_logic_vector(width downto 0);
        display4 : in std_logic_vector(width downto 0);
        display5 : in std_logic_vector(width downto 0);
        display6 : in std_logic_vector(width downto 0);
        display7 : in std_logic_vector(width downto 0);
        display8 : in std_logic_vector(width downto 0);
        refresh : out std_logic_vector(7 downto 0); --vector que pone a 1 el ánodo correspondiente para actualizar
        dis_exit : out std_logic_vector(6 downto 0) --salida de los displays
    );
end Displays;
```

A continuación, se muestran las instanciaciones de los decodificadores y de un preescaler a 10kHz análogo al de 1 Hz descrito antes.

```vhdl
architecture Behavioral of Displays is

    signal dis1 : std_logic_vector(6 downto 0);
    signal dis2 : std_logic_vector(6 downto 0);
    signal dis3 : std_logic_vector(6 downto 0);
    signal dis4 : std_logic_vector(6 downto 0);
    signal dis5 : std_logic_vector(6 downto 0);
    signal dis6 : std_logic_vector(6 downto 0);
    signal dis7 : std_logic_vector(6 downto 0);
    signal dis8 : std_logic_vector(6 downto 0);


    signal flag : integer := 1;

    signal clk_10khz : std_logic;

    COMPONENT clk10khz
       PORT (
              CLK: in  STD_LOGIC;
              clk_1hz : out STD_LOGIC
            );
     END COMPONENT;

    COMPONENT deco
       PORT (
              code : IN std_logic_vector(3 DOWNTO 0);
              led : OUT std_logic_vector(6 DOWNTO 0)
       );
    END COMPONENT;
```

```
54  begin
55
56  Inst_clk10khz: clk10khz
57      PORT MAP (
58          CLK => CLK,
59          CLK_1hz => clk_10khz
60      );
61
62  Inst_deco1: deco PORT MAP (
63          code => display1,
64          led => dis1
65  );
66  Inst_deco2: deco PORT MAP (
67          code => display2,
68          led => dis2
69  );
70  Inst_deco3: deco PORT MAP (
71          code => display3,
72          led => dis3
73  );
74  Inst_deco4: deco PORT MAP (
75          code => display4,
76          led => dis4
77  );
78  Inst_deco5: deco PORT MAP (
79          code => display5,
80          led => dis5
81  );
82  Inst_deco6: deco PORT MAP (
83          code => display6,
84          led => dis6
85  );
86  Inst_deco7: deco PORT MAP (
87          code => display7,
88          led => dis7
89  );
90  Inst_deco8: deco PORT MAP (
91          code => display8,
92          led => dis8
93  );
```

Se introduce una señal “flag” de tipo integer para realizar la cuenta correspondiente y actualizar los displays en orden a una frecuencia de 10 kHz.

```
    process (clk_10khz)
    begin
        if rising_edge(clk_10khz) then
            if flag=1 then
                refresh(0) <=  '0';
                refresh(7 downto 1) <=  "1111111";
                dis_exit <= dis1;
                flag<=2;
            end if;
            if flag=2 then
                refresh(1) <=  '0';
                refresh(0) <=  '1';
                refresh(7 downto 2) <=  "111111";
                dis_exit <= dis2;
                flag<=3;
            end if;
            if flag=3 then
                refresh(2) <=  '0';
                refresh(1 downto 0) <=  "11";
                refresh(7 downto 3) <=  "11111";
                dis_exit <= dis3;
                flag<=4;
            end if;
            if flag=4 then
                refresh(3) <=  '0';
                refresh(2 downto 0) <=  "111";
                refresh(7 downto 4) <=  "1111";
                dis_exit <= dis4;
                flag<=5;
            end if;
            if flag=5 then
                refresh(4) <=  '0';
                refresh(3 downto 0) <=  "1111";
                refresh(7 downto 5) <=  "111";
                dis_exit <= dis5;
                flag<=6;
            end if;
            if flag=6 then
                refresh(5) <=  '0';
                refresh(4 downto 0) <=  "11111";
                refresh(7 downto 6) <=  "11";
               dis_exit <= dis6;
               flag<=7;
            end if;
            if flag=7 then
                refresh(6) <=  '0';
                refresh(5 downto 0) <=  "111111";
                refresh(7) <=  '1';
                dis_exit <= dis7;
                flag<=8;
            end if;
            if flag=8 then
                refresh(7) <=  '0';
                refresh(6 downto 0) <=  "1111111";
                dis_exit <= dis8;
                flag<=1;
            end if;
        end if;

    end process;

end Behavioral;
```

Entidad Top:

En la entidad top se encuentran todas las entidades descritas anteriormente, además, se realizarán las respectivas conexiones entre ellas mediante diferentes señales.

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4
5  entity TOP is
6  Port (
7          CLK      : in std_logic;
8          INICIO   : in std_logic;   --BTNL
9          STOP     : in std_logic;   --BTNR
10         ZERO     : in std_logic;   --BTNU
11         CAMBIO   : in std_logic;   --BTND
12         reset    : in std_logic;
13         refresh  : out std_logic_vector(7 downto 0); --vector que pone a 1 el ánodo correspondiente para actualizar
14         dis_exit : out std_logic_vector(6 downto 0); --salida de los displays
15         led : out std_logic
16     );
17 end TOP;
19 architecture Behavioral of TOP is
20
21     signal sync_auxINICIO: std_logic;
22     signal sync_auxSTOP: std_logic;
23     signal sync_auxZERO: std_logic;
24
25     signal INICIO_aux: std_logic;
26     signal STOP_aux: std_logic;
27     signal ZERO_aux: std_logic;
28     signal Reset_aux: std_logic;
29     signal Cambio_aux: std_logic;
30
31     signal Habilitar_A : std_logic;
32     signal Habilitar_B : std_logic;
33     signal Habilitar_D : std_logic;
34
35     signal dis1_aux : std_logic_vector(3 downto 0);
36     signal dis2_aux : std_logic_vector(3 downto 0);
37     signal dis3_aux : std_logic_vector(3 downto 0);
38     signal dis4_aux : std_logic_vector(3 downto 0);
39     signal dis5_aux : std_logic_vector(3 downto 0);
40     signal dis6_aux : std_logic_vector(3 downto 0);
41     signal dis7_aux : std_logic_vector(3 downto 0);
42     signal dis8_aux : std_logic_vector(3 downto 0);
43
44     signal dis1_ME: std_logic_vector(3 downto 0);
45     signal dis2_ME : std_logic_vector(3 downto 0);
46     signal dis3_ME : std_logic_vector(3 downto 0);
47     signal dis4_ME : std_logic_vector(3 downto 0);
48     signal dis5_ME : std_logic_vector(3 downto 0);
49     signal dis6_ME : std_logic_vector(3 downto 0);
50     signal dis7_ME : std_logic_vector(3 downto 0);
51     signal dis8_ME : std_logic_vector(3 downto 0);
```

```
53      signal dis1_C: std_logic_vector(3 downto 0);
54      signal dis2_C : std_logic_vector(3 downto 0);
55      signal dis3_C : std_logic_vector(3 downto 0);
56      signal dis4_C : std_logic_vector(3 downto 0);
57      signal dis5_C : std_logic_vector(3 downto 0);
58      signal dis6_C : std_logic_vector(3 downto 0);
59      signal dis7_C : std_logic_vector(3 downto 0);
60      signal dis8_C : std_logic_vector(3 downto 0);
61
62      signal dis1_T: std_logic_vector(3 downto 0);
63      signal dis2_T : std_logic_vector(3 downto 0);
64      signal dis3_T : std_logic_vector(3 downto 0);
65      signal dis4_T : std_logic_vector(3 downto 0);
66      signal dis5_T : std_logic_vector(3 downto 0);
67      signal dis6_T : std_logic_vector(3 downto 0);
68      signal dis7_T : std_logic_vector(3 downto 0);
69      signal dis8_T : std_logic_vector(3 downto 0);
73      COMPONENT Entradas
74         PORT (
75              CLK : in std_logic;
76              INICIO : in std_logic;
77              STOP : in std_logic;
78              ZERO : in std_logic;
79              CAMBIO : IN STD_LOGIC;
80              reset : in std_logic;
81              INICIO_out : out std_logic;
82              STOP_out : out std_logic;
83              ZERO_out : out std_logic;
84              CAMBIO_out : out std_logic;
85              reset_out : out std_logic
86         );
87      END COMPONENT;
89      component MaquinaEstados
90      PORT (
91              CLK         : in std_logic;
92              Boton1        : in std_logic;
93              Boton2        : in std_logic;
94              Boton3        : in std_logic;
95              display1    : out std_logic_vector(3 downto 0);
96              display2    : out std_logic_vector(3 downto 0);
97              display3    : out std_logic_vector(3 downto 0);
98              display4    : out std_logic_vector(3 downto 0);
99              display5    : out std_logic_vector(3 downto 0);
100             display6    : out std_logic_vector(3 downto 0);
101             display7    : out std_logic_vector(3 downto 0);
102             display8    : out std_logic_vector(3 downto 0);
103             Habilitar_A : out std_logic :='0';
104             Habilitar_B : out std_logic :='0';
105             Habilitar_D : out std_logic :='1'
106     );
107 end component;
```

```
109     COMPONENT Crono
110        PORT (
111             CLK : in std_logic;
112             display1 : out std_logic_vector(3 downto 0);
113             display2 : out std_logic_vector(3 downto 0);
114             display3 : out std_logic_vector(3 downto 0);
115             display4 : out std_logic_vector(3 downto 0);
116             display5 : out std_logic_vector(3 downto 0);
117             display6 : out std_logic_vector(3 downto 0);
118             display7 : out std_logic_vector(3 downto 0);
119             display8 : out std_logic_vector(3 downto 0);
120             Habilitar_A : in std_logic;
121             Start : in std_logic;
122             Pause : in std_logic;
123             Reset : in std_logic
124        );
125     END COMPONENT;
127     COMPONENT Inverso
128        PORT (
129             CLK : in std_logic;
130             display1 : out std_logic_vector(3 downto 0);
131             display2 : out std_logic_vector(3 downto 0);
132             display3 : out std_logic_vector(3 downto 0);
133             display4 : out std_logic_vector(3 downto 0);
134             display5 : out std_logic_vector(3 downto 0);
135             display6 : out std_logic_vector(3 downto 0);
136             display7 : out std_logic_vector(3 downto 0);
137             display8 : out std_logic_vector(3 downto 0);
138             Habilitar_B : in std_logic;
139             Start : in std_logic;
140             Pause : in std_logic;
141             Reset : in std_logic
142        );
143     END COMPONENT;
```

```
COMPONENT Demux
  PORT (
        Habilitar_A : in std_logic;
        Habilitar_B : in std_logic;
        Habilitar_D : in std_logic;

        Display1_ME : in std_logic_vector(3 downto 0);
        Display2_ME : in std_logic_vector(3 downto 0);
        Display3_ME : in std_logic_vector(3 downto 0);
        Display4_ME : in std_logic_vector(3 downto 0);
        Display5_ME : in std_logic_vector(3 downto 0);
        Display6_ME : in std_logic_vector(3 downto 0);
        Display7_ME : in std_logic_vector(3 downto 0);
        Display8_ME : in std_logic_vector(3 downto 0);

        Display1_C : in std_logic_vector(3 downto 0);
        Display2_C : in std_logic_vector(3 downto 0);
        Display3_C : in std_logic_vector(3 downto 0);
        Display4_C : in std_logic_vector(3 downto 0);
        Display5_C : in std_logic_vector(3 downto 0);
        Display6_C : in std_logic_vector(3 downto 0);
        Display7_C : in std_logic_vector(3 downto 0);
        Display8_C : in std_logic_vector(3 downto 0);

        Display1_T : in std_logic_vector(3 downto 0);
        Display2_T : in std_logic_vector(3 downto 0);
        Display3_T : in std_logic_vector(3 downto 0);
        Display4_T : in std_logic_vector(3 downto 0);
        Display5_T : in std_logic_vector(3 downto 0);
        Display6_T : in std_logic_vector(3 downto 0);
        Display7_T : in std_logic_vector(3 downto 0);
        Display8_T : in std_logic_vector(3 downto 0);

        Display1 : out std_logic_vector(3 downto 0);
        Display2 : out std_logic_vector(3 downto 0);
        Display3 : out std_logic_vector(3 downto 0);
        Display4 : out std_logic_vector(3 downto 0);
        Display5 : out std_logic_vector(3 downto 0);
        Display6 : out std_logic_vector(3 downto 0);
        Display7 : out std_logic_vector(3 downto 0);
        Display8 : out std_logic_vector(3 downto 0)
  );
  END COMPONENT;
```

```
189      COMPONENT Displays
190         PORT (
191          CLK : in std_logic;
192          display1 : in std_logic_vector(3 downto 0);
193          display2 : in std_logic_vector(3 downto 0);
194          display3 : in std_logic_vector(3 downto 0);
195          display4 : in std_logic_vector(3 downto 0);
196          display5 : in std_logic_vector(3 downto 0);
197          display6 : in std_logic_vector(3 downto 0);
198          display7 : in std_logic_vector(3 downto 0);
199          display8 : in std_logic_vector(3 downto 0);
200          refresh : out std_logic_vector(7 downto 0);
201          dis_exit : out std_logic_vector(6 downto 0)
202         );
203      END COMPONENT;
205  begin
206
207          GestorEntradas1 : Entradas PORT MAP(
208          CLK => CLK,
209          INICIO => INICIO,
210          STOP => STOP,
211          ZERO => ZERO,
212          CAMBIO => CAMBIO,
213          reset => reset,
214          INICIO_out => INICIO_aux,
215          STOP_out => STOP_aux,
216          ZERO_out => ZERO_aux,
217          CAMBIO_OUT => CAMBIO_AUX,
218          reset_out => Reset_aux
219      );
220
221      MaquinaEstados1 : MaquinaEstados
222       PORT MAP(
223          CLK         => CLK,
224          Boton1      => INICIO_aux,
225          Boton2      => CAMBIO_aux,
226          Boton3      => ZERO_AUX,
227          display1    => dis1_ME,
228          display2    => dis2_ME,
229          display3    => dis3_ME,
230          display4    => dis4_ME,
231          display5    => dis5_ME,
232          display6    => dis6_ME,
233          display7    => dis7_ME,
234          display8    => dis8_ME,
235          Habilitar_A => Habilitar_A,
236          Habilitar_B => Habilitar_B,
237          Habilitar_D => Habilitar_D
238      );
```

```
240      Crono1 : Crono PORT MAP(
241          CLK=>clk,
242          display1=>dis1_C,
243          display2=>dis2_C,
244          display3=>dis3_C,
245          display4=>dis4_C,
246          display5=>dis5_C,
247          display6=>dis6_C,
248          display7=>dis7_C,
249          display8=>dis8_C,
250          Habilitar_A=>Habilitar_A,
251          Start=>INICIO_aux,
252          Pause=>STOP_aux,
253          Reset=>ZERO_aux
254      );
255
256          Inverso1 : Inverso PORT MAP(
257          CLK=>clk,
258          display1=>dis1_T,
259          display2=>dis2_T,
260          display3=>dis3_T,
261          display4=>dis4_T,
262          display5=>dis5_T,
263          display6=>dis6_T,
264          display7=>dis7_T,
265          display8=>dis8_T,
266          Habilitar_B=>Habilitar_B,
267          Start=>INICIO_aux,
268          Pause=>STOP_aux,
269          Reset=>ZERO_aux
270      );
```

```
272      DeMux1 : Demux
273     PORT MAP(
274           Habilitar_A => Habilitar_A,
275           Habilitar_B => Habilitar_B,
276           Habilitar_D => Habilitar_D,
277
278           display1_ME => dis1_ME,
279           display2_ME => dis2_ME,
280           display3_ME => dis3_ME,
281           display4_ME => dis4_ME,
282           display5_ME => dis5_ME,
283           display6_ME => dis6_ME,
284           display7_ME => dis7_ME,
285           display8_ME => dis8_ME,
286           display1_C  => dis1_C,
287           display2_C  => dis2_C,
288           display3_C  => dis3_C,
289           display4_C  => dis4_C,
290           display5_C  => dis5_C,
291           display6_C  => dis6_C,
292           display7_C  => dis7_C,
293           display8_C  => dis8_C,
294
295           display1_T  => dis1_T,
296           display2_T  => dis2_T,
297           display3_T  => dis3_T,
298           display4_T  => dis4_T,
299           display5_T  => dis5_T,
300           display6_T  => dis6_T,
301           display7_T  => dis7_T,
302           display8_T  => dis8_T,
303
304           display1    => dis1_aux,
305           display2    => dis2_aux,
306           display3    => dis3_aux,
307           display4    => dis4_aux,
308           display5    => dis5_aux,
309           display6    => dis6_aux,
310           display7    => dis7_aux,
311           display8    => dis8_aux
312 );
```

```
316         Dispalys1 : Displays PORT MAP(
317         CLK=>clk,
318         display1=>dis1_aux,
319         display2=>dis2_aux,
320         display3=>dis3_aux,
321         display4=>dis4_aux,
322         display5=>dis5_aux,
323         display6=>dis6_aux,
324         display7=>dis7_aux,
325         display8=>dis8_aux,
326         refresh=>refresh,
327         dis_exit=>dis_exit
328     );
329
330 end Behavioral;
```

Simulaciones

Se han realizado varios testbench, a continuación:

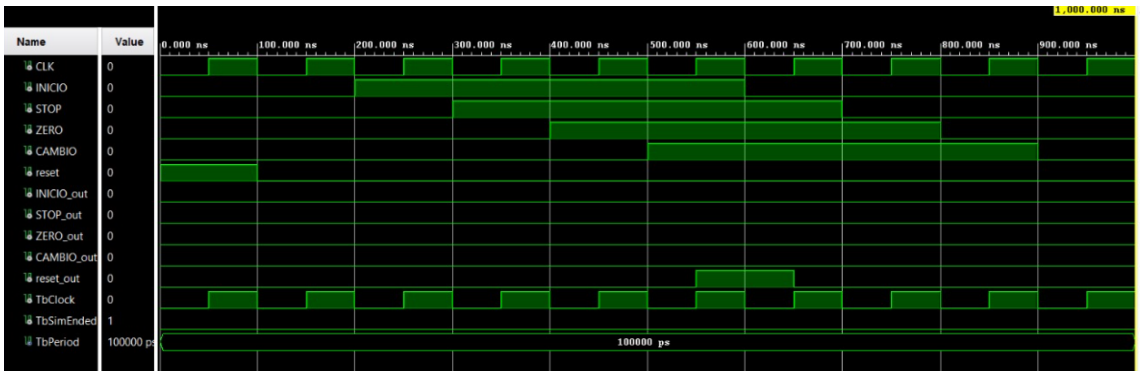
Simulación del gestor de entradas:

Testbench empleado:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Entradas_tb is
end Entradas_tb;

architecture tb of Entradas_tb is
  component Entradas
    Port (
        CLK       : in std_logic;
        INICIO    : in std_logic;  --BTNL
        STOP      : in std_logic;  --BTNR
        ZERO      : in std_logic;  --BTNU
        CAMBIO    : IN STD_LOGIC;  --BTND
        reset     : in std_logic;
        INICIO_out   : out std_logic;
        STOP_out   : out std_logic;
        ZERO_out   : out std_logic;
        CAMBIO_out   : out std_logic;
        reset_out : out std_logic
    );
  end component;

    signal CLK       : std_logic;
    signal INICIO      : std_logic;
    signal STOP      : std_logic;
    signal ZERO      : std_logic;
    signal CAMBIO      : std_logic;
    signal reset     : std_logic;
    signal INICIO_out   : std_logic;
    signal STOP_out   : std_logic;
    signal ZERO_out   : std_logic;
    signal CAMBIO_out   : std_logic;
    signal reset_out : std_logic;

    constant TbPeriod : time := 100 ns; -- EDIT Put right period here
    signal TbClock : std_logic := '0';
    signal TbSimEnded : std_logic := '0';

BEGIN

dut : Entradas
    port map (CLK        => CLK,
              INICIO     => INICIO,
              STOP       => STOP,
              ZERO       => ZERO,
              CAMBIO     => CAMBIO,
              reset      => reset,
              INICIO_out => INICIO_out,
              STOP_out   => STOP_out,
              ZERO_out   => ZERO_out,
              CAMBIO_out => CAMBIO_out,
              reset_out  => reset_out);

    -- Clock generation
    TbClock <= not TbClock after TbPeriod/2 when TbSimEnded /= '1' else '0';

    -- EDIT: Check that CLK is really your main clock signal
    CLK <= TbClock;
```

```
62      stimuli : process
63      begin
64          -- EDIT Adapt initialization as needed
65          INICIO <= '0';
66          STOP   <= '0';
67          ZERO   <= '0';
68          CAMBIO <= '0';
69
70          -- Reset generation
71          -- EDIT: Check that reset is really your re
72          reset <= '1';
73          wait for 100 ns;
74          reset <= '0';
75          wait for 100 ns;
76
77          -- EDIT Add stimuli here
78          INICIO <= '1';
79          --wait for 100 * TbPeriod
80          wait for 100ns;
81          STOP <= '1';
82          --wait for 100 * TbPeriod;
83          wait for 100ns;
84          ZERO <= '1';
85          --wait for 100 * TbPeriod;
86          wait for 100ns;
87          CAMBIO <= '1';
88         --wait for 100 * TbPeriod;
89         wait for 100ns;
90          INICIO <= '0';
91          --wait for 100 * TbPeriod;
92          wait for 100ns;
93          STOP <= '0';
94          --wait for 100 * TbPeriod;
95          wait for 100ns;
96          ZERO <= '0';
97          --wait for 100 * TbPeriod;
98          wait for 100ns;
99          CAMBIO <= '0';
100         --wait for 100 * TbPeriod;
101         wait for 100ns;
102
103         -- Stop the clock and hence terminate the simulation
104         TbSimEnded <= '1';
105         wait;
106     end process;
107
108 end tb;
```

Resultados obtenidos:



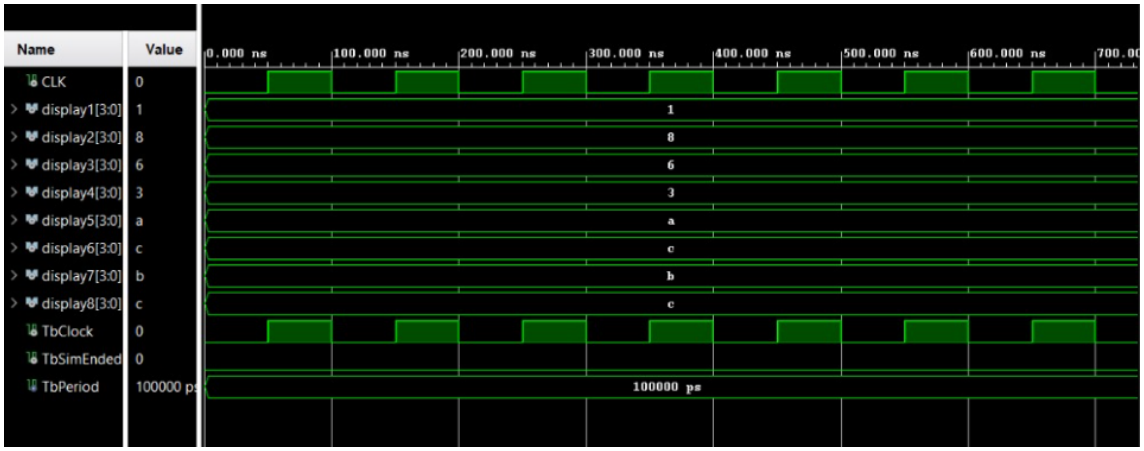
Simulación displays:

Testbench empleado:

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

entity Displays_tb is
end Displays_tb;

architecture tb of Displays_tb is

    component Displays
        port (CLK               : in std_logic;
              display1          : in std_logic_vector (3 downto 0);
              display2          : in std_logic_vector (3 downto 0);
              display3          : in std_logic_vector (3 downto 0);
              display4          : in std_logic_vector (3 downto 0);
              display5          : in std_logic_vector (3 downto 0);
              display6          : in std_logic_vector (3 downto 0);
              display7          : in std_logic_vector (3 downto 0);
              display8          : in std_logic_vector (3 downto 0));
             -- refresh           : out std_logic_vector (7 downto 0);
            --  dis_exit          : out std_logic_vector (6 downto 0));
    end component;

    signal CLK            : std_logic;
    signal display1       : std_logic_vector (3 downto 0);
    signal display2       : std_logic_vector (3 downto 0);
    signal display3       : std_logic_vector (3 downto 0);
    signal display4       : std_logic_vector (3 downto 0);
    signal display5       : std_logic_vector (3 downto 0);
    signal display6       : std_logic_vector (3 downto 0);
    signal display7       : std_logic_vector (3 downto 0);
    signal display8       : std_logic_vector (3 downto 0);
   -- signal refresh        : std_logic_vector (7 downto 0);
    --signal dis_exit      : std_logic_vector (6 downto 0);

    constant TbPeriod : time := 100 ns; -- EDIT Put right period here
    signal TbClock : std_logic := '0';
    signal TbSimEnded : std_logic := '0';

begin

    dut : Displays
    port map (CLK             => CLK,
              display1           => display1,
              display2           => display2,
              display3           => display3,
              display4           => display4,
              display5           => display5,
              display6           => display6,
              display7           => display7,
              display8           => display8
              --refresh            => refresh,
              --dis_exit           => dis_exit
              );

    -- Clock generation
    TbClock <= not TbClock after TbPeriod/2 when TbSimEnded /= '1' else '0';

    -- EDIT: Check that CLK is really your main clock signal
    CLK <= TbClock;

    stimuli : process
    begin
        -- EDIT Adapt initialization as needed
        display1 <= "0001";
        display2 <= "1000";
        display3 <= "0110";
        display4 <= "0011";
        display5 <= "1010";
        display6 <= "1100";
        display7 <= "1011";
        display8 <= "1100";


        wait for 8000 ns;
```

```
78          -- Stop the clock and hence terminate the simulation
79          TbSimEnded <= '1';
80          wait;
81      end process;
82
83  end tb;
```

Resultados obtenidos:



Constrains:

```
7   set_property -dict { PACKAGE_PIN E3    IOSTANDARD LVCMOS33 } [get_ports { CLK }]; #IO_L12P_T1_MRCC_35 Sch=clk100mhz

33  set_property -dict { PACKAGE_PIN H17   IOSTANDARD LVCMOS33 } [get_ports { led }]; #IO_L18P_T2_A24_15 Sch=led[0]
60  set_property -dict { PACKAGE_PIN T10   IOSTANDARD LVCMOS33 } [get_ports { dis_exit[6] }]; #IO_L24N_T3_A00_D16_14 Sch=ca
61  set_property -dict { PACKAGE_PIN R10   IOSTANDARD LVCMOS33 } [get_ports { dis_exit[5] }]; #IO_25_14 Sch=cb
62  set_property -dict { PACKAGE_PIN K16   IOSTANDARD LVCMOS33 } [get_ports { dis_exit[4] }]; #IO_25_15 Sch=cc
63  set_property -dict { PACKAGE_PIN K13   IOSTANDARD LVCMOS33 } [get_ports { dis_exit[3] }]; #IO_L17P_T2_A26_15 Sch=cd
64  set_property -dict { PACKAGE_PIN P15   IOSTANDARD LVCMOS33 } [get_ports { dis_exit[2] }]; #IO_L13P_T2_MRCC_14 Sch=ce
65  set_property -dict { PACKAGE_PIN T11   IOSTANDARD LVCMOS33 } [get_ports { dis_exit[1] }]; #IO_L19P_T3_A10_D26_14 Sch=cf
66  set_property -dict { PACKAGE_PIN L18   IOSTANDARD LVCMOS33 } [get_ports { dis_exit[0] }]; #IO_L4P_T0_D04_14 Sch=cg
67
68  #set_property -dict { PACKAGE_PIN H15   IOSTANDARD LVCMOS33 } [get_ports { DP }]; #IO_L19N_T3_A21_VREF_15 Sch=dp
69
70  set_property -dict { PACKAGE_PIN J17   IOSTANDARD LVCMOS33 } [get_ports { refresh[0] }]; #IO_L23P_T3_FOE_B_15 Sch=an[0]
71  set_property -dict { PACKAGE_PIN J18   IOSTANDARD LVCMOS33 } [get_ports { refresh[1] }]; #IO_L23N_T3_FWE_B_15 Sch=an[1]
72  set_property -dict { PACKAGE_PIN T9    IOSTANDARD LVCMOS33 } [get_ports { refresh[2] }]; #IO_L24P_T3_A01_D17_14 Sch=an[2]
73  set_property -dict { PACKAGE_PIN J14   IOSTANDARD LVCMOS33 } [get_ports { refresh[3] }]; #IO_L19P_T3_A22_15 Sch=an[3]
74  set_property -dict { PACKAGE_PIN P14   IOSTANDARD LVCMOS33 } [get_ports { refresh[4] }]; #IO_L8N_T1_D12_14 Sch=an[4]
75  set_property -dict { PACKAGE_PIN T14   IOSTANDARD LVCMOS33 } [get_ports { refresh[5] }]; #IO_L14P_T2_SRCC_14 Sch=an[5]
76  set_property -dict { PACKAGE_PIN K2    IOSTANDARD LVCMOS33 } [get_ports { refresh[6] }]; #IO_L23P_T3_35 Sch=an[6]
77  set_property -dict { PACKAGE_PIN U13   IOSTANDARD LVCMOS33 } [get_ports { refresh[7] }]; #IO_L23N_T3_A02_D18_14 Sch=an[7]
78
79
80
81  ##Buttons
82
83  #set_property -dict { PACKAGE_PIN C12   IOSTANDARD LVCMOS33 } [get_ports { CPU_RESETN }]; #IO_L3P_T0_DQS_AD1P_15 Sch=cpu_resetn
84
85  #set_property -dict { PACKAGE_PIN N17   IOSTANDARD LVCMOS33 } [get_ports { BTNC }]; #IO_L9P_T1_DQS_14 Sch=btnc
86  set_property -dict { PACKAGE_PIN M18   IOSTANDARD LVCMOS33 } [get_ports { ZERO }]; #IO_L4N_T0_D05_14 Sch=btnu
87  set_property -dict { PACKAGE_PIN P17   IOSTANDARD LVCMOS33 } [get_ports { INICIO }]; #IO_L12P_T1_MRCC_14 Sch=btnl
88  set_property -dict { PACKAGE_PIN M17   IOSTANDARD LVCMOS33 } [get_ports { STOP }]; #IO_L10N_T1_D15_14 Sch=btnr
89  set_property -dict { PACKAGE_PIN P18   IOSTANDARD LVCMOS33 } [get_ports { CAMBIO }]; #IO_L9N_T1_DQS_D13_14 Sch=btnd
```