ARQUITECTURA DEL COMPUTADOR 1

PROYECTO RECORDATORIO

"REMEMBRALL"

(ENTREGA FINAL)

PROFESOR: JOSÉ OLIDEN SANCHEZ

DIEGO FELIPE GALARZA CHAMORRO

ANDREA ESTEFANIA TIMARAN BUCHELY

ANTONIO YU CHEN

OCTUBRE 2017

DESCRIPCIÓN:

La alarma recordatorio “Remembrall” es un dispositivo portátil de bolsillo pequeño y esférico, en donde el usuario tendrá total interacción con la esfera y podrá revisar tanto la hora como la, o las alarmas que desee programar; Remembrall tiene un máximo de 5 alarmas programables, en donde el usuario puede asignarle un nombre distinto a cada una y que se ajuste a sus necesidades, cada alarma puede tener hasta un máximo de 10 recordatorios en el día por lo que en el momento en que se active un recordatorio la esfera se tornara roja, para que la persona pueda tener una referencia visible de que tiene algún evento, actividad o acción pendiente que debe hacer, mientras la persona no tenga ningún recordatorio la esfera se mantendrá transparente.

Para el prototipo nuestro Remembrall tiene su funcionamiento sobre el lenguaje de programación C en donde llevamos registro de todo el funcionamiento correcto del programa; como producto del proyecto Remembrall funcionará bajo el lenguaje VHDL en el dispositivo FPGA.

ELEMENTOS A UTILIZAR:

Para nuestro proyecto usaremos varios elementos que harán más fácil la comprensión y elaboración de nuestra alarma recordatorio “Remembrall”:

La aplicación Draw.io nos permitió realizar el diagrama UML de nuestro prototipo en C y producto final en VHDL para una mejor comprensión del esquema y manejo de funciones.

El lenguaje de programación VHDL es el lenguaje que se utilizó para nuestro producto final y poder realizar las pruebas necesarias en la FPGA.

La FPGA es el dispositivo que nos permitió tener un registro visual de los resultados de nuestro programa al ejecutarlo con el VHDL.

ESPECIFICACIONES Y FUNCIONAMIENTO:

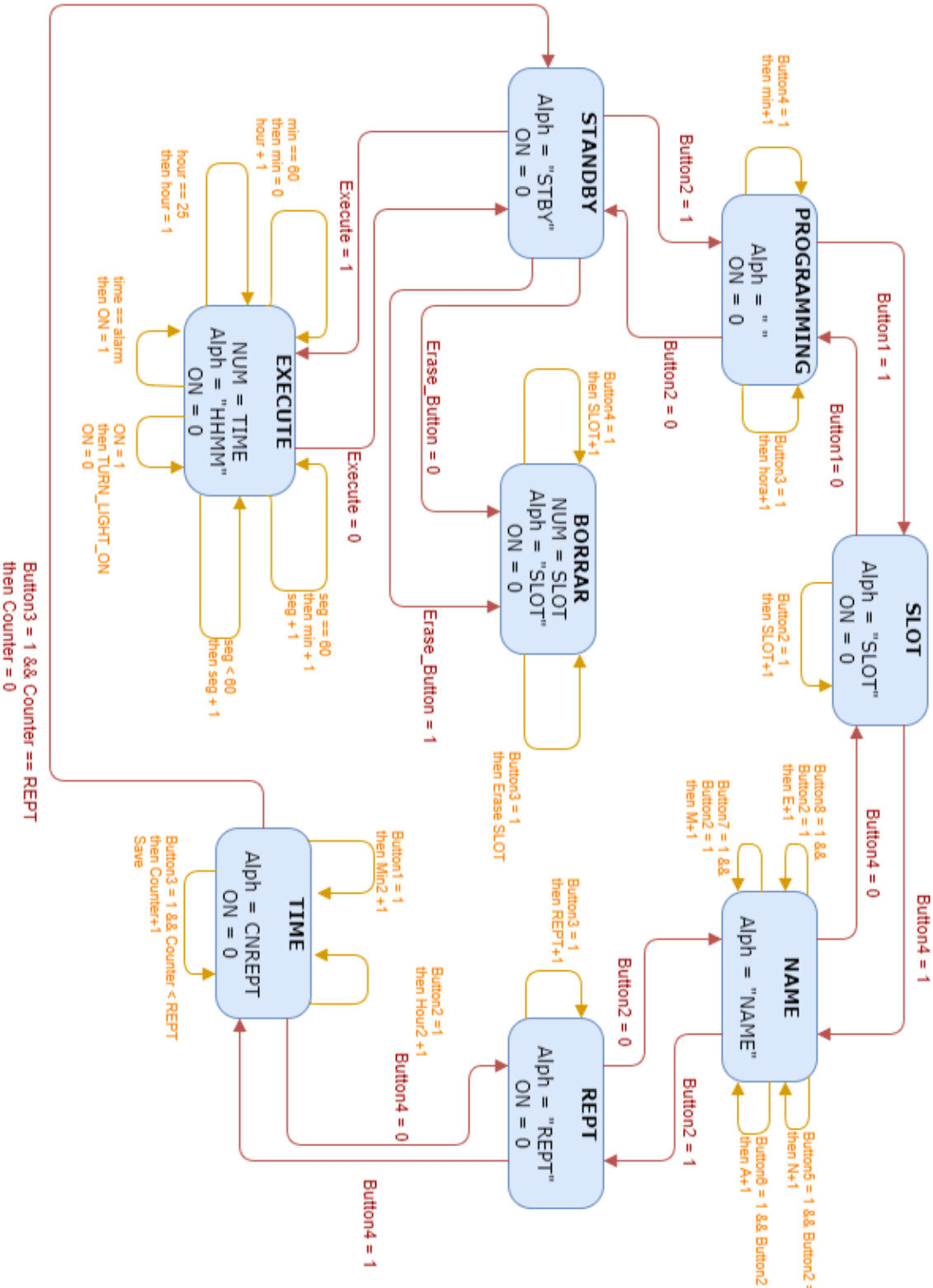
La alarma recordatorio “Remembrall” esta diseñada para darle al usuario la mejor experiencia a la hora de asignar sus respectivas tareas, deberes y/o actividades a realizar durante el transcurso del dia. Ese dispositivo esta compuesto por una bateria recargable integrada, un display esferico interactivo y de facil aprendizaje y adaptación en donde el usuario puede ingresar todos los datos referentes a cada recordatorio (Nombre del recordatorio, horas de activación de la alarma, cantidad de alarmas por recordatorio al día y días de repetición); además nuestro programa cuenta con un código estructurado y seguro el cual esta diseñado especificamente para facilitarle el ingreso de datos y configuracion personalizada del Remembrall; este codigo está implementado en VHDL para poder realizar nuestras pruebas del prototipo en el dispositivo FPGA, pero igualmente tenemos desarrollado el programa en el lenguaje C para poder adaptar nuestro trabajo a cualquier ambiente de desarrollo.

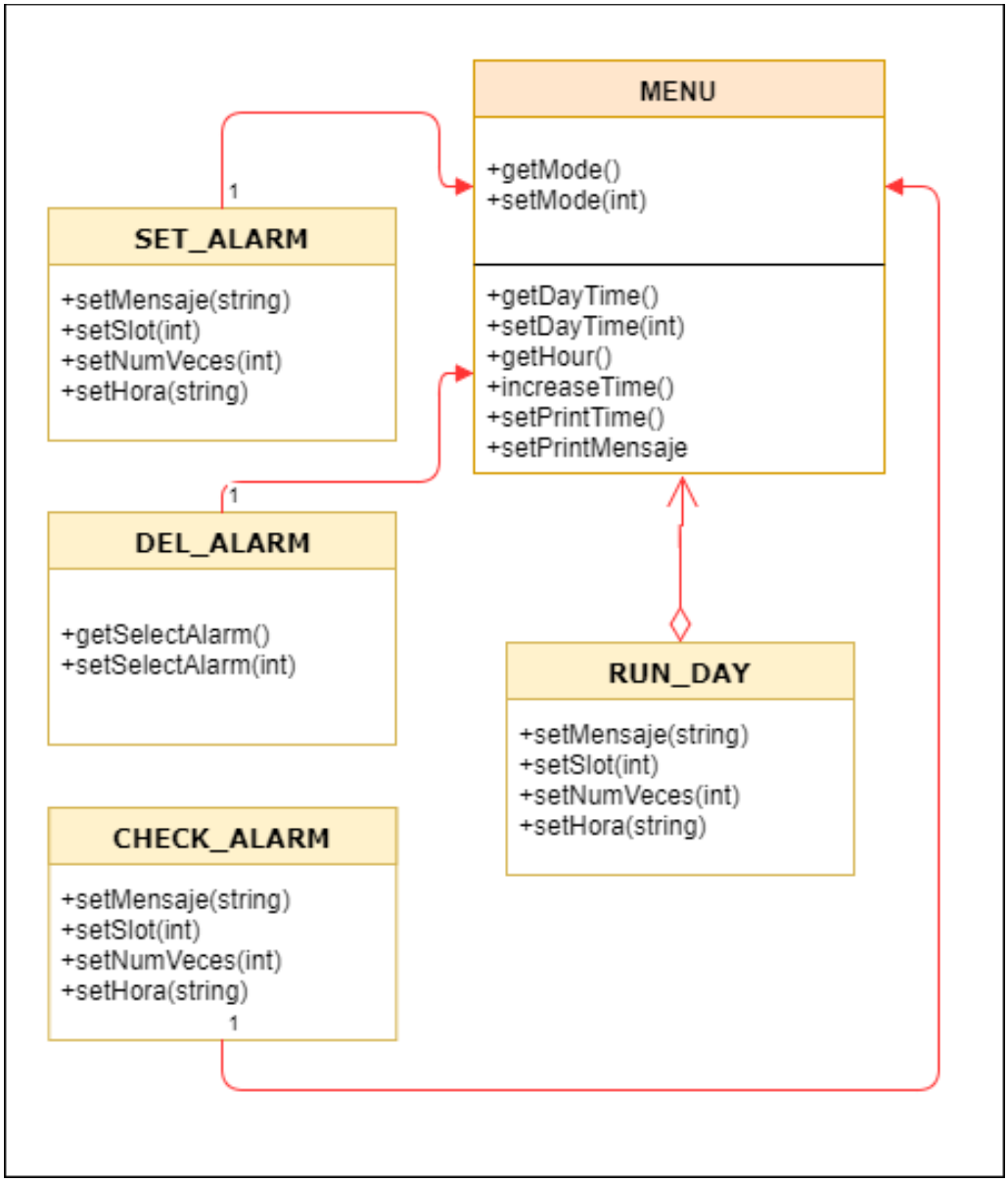
PROTOTIPO:

CONCLUSIONES:

Para nuestro proyecto Remembrall pudimos programar muchas de las funciones principales como lo son la programación de las horas y los minutos por parte del usuario, pero a la vez nos quedaron faltando otras instancias por programar como fueron el guardado y borrado de alarmas, entre otros. Nos encontramos con ciertas dificultades durante la programación del proyecto como lo fueron, principalmente el entendimiento de este nuevo lenguaje de programación y además la interpretación de ciertos datos a la hora de mostrarlos en la FPGA; por otro lado, tuvimos problemas con el tiempo y organización entre los miembros del grupo para reunirnos a programar, así como también dificultades a la hora de verificar el funcionamiento correcto del programa en la FPGA. Además, tuvimos problemas con una explicación que dio el monitor, la cual retrasó parte de la programación y traducción de los datos. El funcionamiento de añadir las horas y minutos respectivos a cada alarma fue desarrollada con éxito, pudimos, después de la elaboración del proyecto, no solo entender casi completamente el lenguaje de programación VHDL sino también aprender sobre la programación orientada a objetos y visualizar los resultados de manera física, comprendiendo que todo lo programado por nosotros tiene un funcionamiento físico explicito en los dispositivos. Agradecemos las explicaciones realizadas por parte del profesor que nos sirvieron de orientación y guía para la elaboración del proyecto, así como también la intervención del monitor quien resolvió algunas de nuestras dudas y nos oriento en el entendimiento mas profundo del lenguaje de programación VHDL y el funcionamiento y asignación de la FPGA.

ESTRUCTURA:


STANDBY
Alph = "STBY"
ON = 0
PROGRAMMING
Alph = " "
ON = 0
SLOT
Alph = "SLOT"
ON = 0
EXECUTE
NUM = TIME
Alph = "HHMM"
ON = 0
BORRAR
NUM = SLOT
Alph = "SLOT"
ON = 0
NAME
Alph = "NAME"
REPT
Alph = "REPT"
ON = 0
TIME
Alph = CNREPT
ON = 0
Button1 = 1
Button1= 0
Button2 = 1
Button2 = 0
Button3 = 1
Button4 = 1
Button4 = 0
Execute = 1
Execute = 0
Erase_Button = 0
Erase_Button = 1
Button3 = 1 && Counter == REPT
then Counter = 0
Button4 = 1
then min+1
Button3 = 1
then hora+1
Button2 = 1
then SLOT+1
Button4 = 1
then SLOT+1
Button3 = 1
then Erase SLOT
Button8 = 1 &&
Button2 = 1
then E+1
Button7 = 1 &&
Button2 = 1
then M+1
Button5 = 1 && Button2 = 1
then N+1
Button6 = 1 && Button2 = 1
then A+1
Button3 = 1
then REPT+1
Button2 =1
then Hour2 +1
Button1 = 1
then Min2 +1
Button3 = 1 && Counter < REPT
then Counter+1
Save
min == 60
then min = 0
hour + 1
hour == 25
then hour = 1
time == alarm
then ON = 1
ON = 1
then TURN_LIGHT_ON
ON = 0
seg == 60
then min + 1
seg + 1
seg < 60
then seg + 1


MENU
+getMode()
+setMode(int)
+getDayTime()
+setDayTime(int)
+getHour()
+increaseTime()
+setPrintTime()
+setPrintMensaje
SET_ALARM
+setMensaje(string)
+setSlot(int)
+setNumVeces(int)
+setHora(string)
1
DEL_ALARM
+getSelectAlarm()
+setSelectAlarm(int)
1
RUN_DAY
+setMensaje(string)
+setSlot(int)
+setNumVeces(int)
+setHora(string)
CHECK_ALARM
+setMensaje(string)
+setSlot(int)
+setNumVeces(int)
+setHora(string)
1

```c
int time = 0;
char ** alarms;
int ** hours;

int * alarmActivated() {
    int * on = (int *) malloc(sizeof(int) * 2);
    on[0] = -1;
    on[1] = -1;
    for(int i = 0; i < 5; i++) {
        for(int j = 0; j < 10; j++) {
            if(hours[i][j] == time) {
                on[0] = i;
                on[1] = j;
            }
        }
    }
    return on;
}

void setAlarm() {
    char * name = (char *) malloc(sizeof(char) * 15);
    printf("Mensaje que desea programar: ");
    scanf("%s", name);
    int repetition, position;
    for(int i = 0; i < 5; i++) {
        printf("%d. %s.\n", i + 1, alarms[i]);
    }
    printf("Posicion en donde desea programar la alarma. Ingrese un numero de 1 a 5: ");
    scanf("%d", &position);
    position = position - 1;
    alarms[position] = name;
    printf("Numero de veces que quiere programar la alarma. Maximo 10 veces: ");
    scanf("%d", &repetition);
    for(int i = 0; i < 10; i++) {
        if(i < repetition) {
            printf("Digite la hora que quiere programar: ");
            scanf("%d", &hours[position][i]);
        }
        else {
            hours[position][i] = -1;
        }
    }
```

```c
void deleteAlarm() {
    int position;
    for(int i = 0; i < 5; i++) {
        printf("%d. %s.\n", i + 1, alarms[i]);
    }
    printf("Posicion en donde desea eliminar la alarma. Ingrese un numero de 1 a 5: ");
    scanf("%d", &position);
    position = position - 1;
    alarms[position] = "Vacio";
    for(int i = 0; i < 10; i++) {
        hours[position][i] = -1;
    }
}

void printAlarms() {
    for(int i = 0; i < 5; i++) {
        printf("%d. %s.\n", i + 1, alarms[i]);
    }
}

void printHours() {
    for(int i = 0; i < 5; i++) {
        for(int j = 0; j < 10; j++) {
            printf("%d ", hours[i][j]);
        }
        printf("\n");
    }
}

int startTime() {
    int * on = alarmActivated();
    if(on[0] >= 0 && on[1] >= 1) {
        printf("%s", alarms[on[0]]);
        return 1;
    }
    printf("%d\n", time);
    time = time + 10;
    return startTime();
}
```

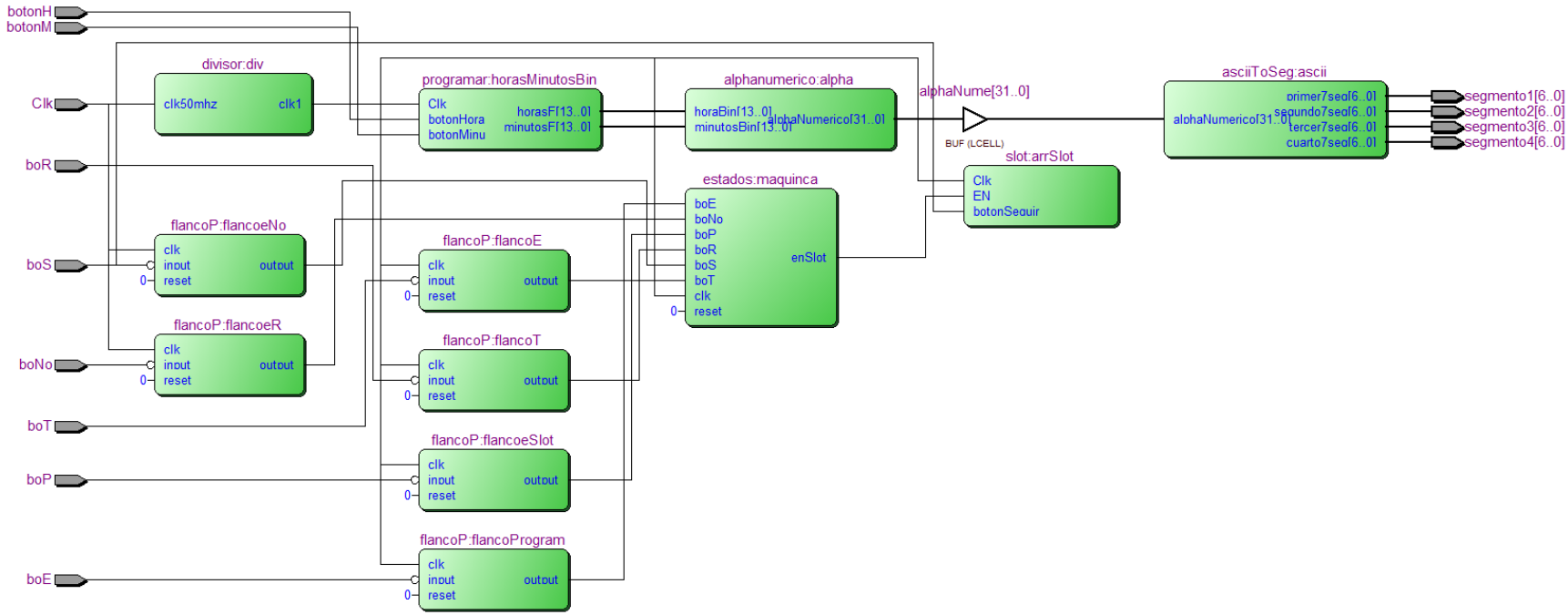
```
begin
   if reset = '0' then
      state <= espera;
   elsif (rising_edge(clk)) then
      case state is
         when espera=>
            if boE = '1' then
               state <= programar;
            else
               state <= espera;
            end if;
         when programar=>
            if boP = '1' then
               state <= slot;
            else
               state <= programar;
            end if;
         when slot=>
            if boS = '1' then
               state <= nombre;
            else
               state <= slot;
            end if;
         when nombre =>
            if boNo = '1' then
               state <= rept;
            else
               state <= nombre;
            end if;
         when rept =>
            if boR = '1' then
               state <= tiempo;
            else
               state <= rept;
```

```
architecture structural of programar is

   signal hora : natural;
   signal minutos : natural;


begin

   process (Clk) is
   begin
      if(rising_edge(Clk)) then
         if(botonHora = '0') then
            if(hora = 24) then
               hora <= 1;
            else
               hora <= hora + 1;
            end if;
         end if;

         if (botonMinu = '0') then
            if(minutos = 59) then
               minutos <= 0;
            else
               minutos <= minutos +1;
            end if;
         end if;
      end if;
   end process;

   horasF <= std_logic_vector(to_unsigned (hora,14));
   minutosF <= std_logic_vector(to_unsigned (minutos,14));

end structural;
```

```
architecture Behavioral of binToBCD is
begin
    proceso_bcd: process(num_bin)
        variable z: STD_LOGIC_VECTOR(29 downto 0);
    begin
        -- Inicialización de datos en cero.
        z := (others => '0');
        -- Se realizan los primeros tres corrimientos.
        z(16 downto 3) := num_bin;
        for i in 0 to 10 loop
            -- Unidades (4 bits).
            if z(17 downto 14) > 4 then
                z(17 downto 14) := z(17 downto 14) + 3;
            end if;
            -- Decenas (4 bits).
            if z(21 downto 18) > 4 then
                z(21 downto 18) := z(21 downto 18) + 3;
            end if;
            -- Centenas (4 bits).
            if z(25 downto 22) > 4 then
                z(25 downto 22) := z(25 downto 22) + 3;
            end if;
            -- Miles (4 bits).
            if z(29 downto 26) > 4 then
                z(29 downto 26) := z(29 downto 26) + 3;
            end if;
            -- Corrimiento a la izquierda.
            z(29 downto 1) := z(28 downto 0);
        end loop;
        -- Pasando datos de variable Z, correspondiente a
        num_bcd <= z(29 downto 14);
    end process;
end Behavioral;
```

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
entity divisor is
port (
   clk50mhz:   in STD_LOGIC;
   clk1:    out STD_LOGIC
);
end divisor;

architecture rtl of divisor is
   constant max_count: INTEGER := 5800000;
   signal count: INTEGER range 0 to max_count;
   signal clk_state: STD_LOGIC := '0';

begin
   gen_clock: process(clk50mhz, clk_state, count)
   begin
      if clk50mhz'event and clk50mhz='1' then
         if count < max_count then
            count <= count+1;
         else
            clk_state <= not clk_state;
            count <= 0;
         end if;
      end if;
   end process;

   persecond: process (clk_state)
   begin
      clk1 <= clk_state;
   end process;

end rtl;
```

```vhdl
--FLANCO
signal botonEspera: std_logic;
signal botonProg: std_logic;
signal botonS: std_logic;
signal botonNo: std_logic;
signal botonR: std_logic;
signal botonT: std_logic;
--ALPHANUMERIO
signal Clkd:std_logic;
signal horaBin : std_logic_vector(13 downto 0);
signal minutosBin : std_logic_vector(13 downto 0);
signal alphaNume : std_logic_vector(31 downto 0);
signal seg1 : std_logic_vector(6 downto 0);  -- primer siete segmentos derecha izquierda
signal seg2 : std_logic_vector(6 downto 0);
signal seg3 : std_logic_vector(6 downto 0);
signal seg4 : std_logic_vector(6 downto 0);
signal enSlot : std_logic;
signal arregloSlot : std_logic_vector(2 downto 0);
signal enEscritura : std_logic;
attribute keep : boolean;
attribute keep of alphaNume: signal is true;
-- REGISTROS SLOTS
   --PRIMER SLOT
signal hora1 : std_logic_vector(13 downto 0);
signal minuto1 : std_logic_vector(13 downto 0);
   --SEGUNDO SLOTS
signal hora2 : std_logic_vector(13 downto 0);
signal minuto2 : std_logic_vector(13 downto 0);
   --TERCER SLOTS
signal hora3 : std_logic_vector(13 downto 0);
signal minuto3 : std_logic_vector(13 downto 0);
   --CUARTO SLOT
signal hora4 : std_logic_vector(13 downto 0);
signal minuto4 : std_logic_vector(13 downto 0);
   --CINCO SLOT
```


botonH
botonM
Clk
boR
boS
boNo
boT
boP
boE
divisor:div
clk50mhz
clk1
programar:horasMinutosBin
Clk
botonHora
botonMinu
horasFf13..0]
minutosFf13..0]
alphanumerico:alpha
horaBinf13..0]
minutosBinf13..0]
alphaNumericof31..0]
alphaNume[31..0]
BUF (LCELL)
asciiToSeg:ascii
alphaNumericof31..0]
primer7seaf6..0]
segundo7seaf6..0]
tercer7seaf6..0]
cuarto7seaf6..0]
segmento1[6..0]
segmento2[6..0]
segmento3[6..0]
segmento4[6..0]
slot:arrSlot
Clk
EN
botonSeguir
estados:maquinca
boE
boNo
boP
boR
boS
boT
clk
reset
enSlot
flancoP:flancoeNo
flancoP:flancoE
flancoP:flancoeR
flancoP:flancoT
flancoP:flancoeSlot
flancoP:flancoProgram
clk
inout
reset
output
0