Reloj digital basándonos en el lenguaje VHDL aplicado a la placa MachXO2 con salida a 6 displays de 7 segmentos de tipo ánodo.

Felipe-Prieto-de-la-Cruz, F, Servin-Hernando-Daniel, D, Rodríguez-Aguilar-Kathia, K

Diseño de sistemas digitales, Escuela Superior de cómputo, 07738, Ciudad de México, CDMX, Servin Daniel (androidx09@gmail.com), de la Cruz Felipe (felipe00@live.com.mx)

Innovación tecnológica: Reloj configurable usando el lenguaje de programación VHDL implementado para una placa MachXO2 mostrando resultado en 6 displays de 7 segmentos.

Área de aplicación industrial: Puede ser implementado para una área de producción la cual lleve el conteo de tiempo necesario para poder finalizar ciertos trabajos o contar el tiempo de proceso requerido en alguna máquina la cual realice trabajos basados en tiempo.

Para checar los tiempos de los turnos de los empleados que están en una empresa, otra de las aplicaciones que se pueden tomar en cuenta, es el área de control de calidad dado que algunas pruebas requieren tiempo, esta implementación de reloj puede lograr con el objetivo de tomar el tiempo y hacer las pruebas requeridos por el personal y tener resultados exitosos.

Puede ser implementado para el área telefónica la cual, en algunos casos se requiere hacer cierta cantidad de llamadas durante cierto periodo de tiempo para lograr ciertos objetivos.

Puede ser aplicado en el área de comida para ver el tiempo que se tiene para dar la entrega a los respectivos clientes.

Abstract

All electronic clocks use a pulse train of very precise frequency, generated by a quartz crystal or a timer, and successively divide this frequency to achieve very accurate pulses of a second or fraction. The pulses of seconds are counted in progressive order and when the count reaches sixty a minute pulse is produced. These pulses are then counted and after sixty seconds a pulse corresponding to one hour is delivered. Each time the pulses of seconds, minutes and hours are presented, they are shown in the corresponding 2 displays

A digital clock represents the time on a digital display in decimal numeration, presenting the time of day in the form: HH: MM, or HH: MM: SS, depending on whether the watch has a second hand or not.

For each of these cases there are two formats: 24 or 12 hours. In the 24-hour format, the day is divided into 24 hours starting at zero hours, and ending at twenty-three hours. In the twelve-hour format, the day is divided into morning (AM) and afternoon (PM), which in turn are divided into 12 hours from 1 to 12 each.

For this project we will make a digital clock in VHDL using a MachXO2 with output to 6 displays of 7 segments of anode type.

This project has a certain level of complexity since you have to program a file for each procedure that we are going to use, first you have to program the clock counter, which is a very important part because here you will designate how you are going to function.

Then we have to program the part that is known as the "clk" that for this case will be 1 HZ.
Followed by this it is necessary to program the part of the seven segments that will serve us. Because this unit contains what is necessary to display digits in the four displays.
Clock counter: This component will count from 0 to 23 for the hours and from 0 to 59 for the minutes.

The outputs correspond to the four digits: two to show the time and two to show the minutes.
1Hz clock: It will be used as an input signal in the counter (we will count every second).
There are several ways to make the clock counter, all with advantages and disadvantages. In this entry it was decided to use four independent counters (one per digit) to avoid further conversions of binary numbers to BCD.

The display is a component, which is used to represent numbers and some letters, this is composed of seven or eight leds assembled separately, combining elements to create symbols. The first seven segments are responsible for forming the symbol and with the eighth we can turn on and off the decimal point. There are two types of display one that is the common anode and the other that is the common cathode in this case to show the minutes and the We are going to use a common cathode display.
Keywords: Binary, BCD, Clk, Counter, Complexity, HZ, Hours, MachXO2, Clock, VHDL, display, elements, anode, counter.

Resumen

Todos los relojes electrónicos utilizan un tren de pulsos de frecuencia muy precisa,generado por un cristal de cuarzo o un timer, y dividen sucesivamente esta frecuencia hasta lograr pulsos muy exactos de un segundo o fracción. Los pulsos de segundos se cuentan en orden progresivo y cuando la cuenta llega a sesenta se produce un pulso de minuto. A continuación estos pulsos se cuentan y cuando han transcurrido sesenta se entrega un pulso correspondiente a una hora. Cada vez que se presentan los pulsos de segundos, minutos y horas, se muestran en los 2 displays correspondientes

Un reloj digital representa la hora en un display digital en numeración decimal, presentando la hora del día en la forma: HH:MM, o HH:MM:SS, según el reloj tenga o no segundero.

Para cada uno de estos casos hay dos formatos: 24 o 12 horas. En el formato de 24 horas, el día se divide en 24 horas comenzando a las cero horas, y finalizando a las veintitrés horas. En el formato de doce horas el día se divide en mañana (AM) y tarde (PM), que a su vez se dividen en 12 horas de 1 a 12 cada una.

Para este proyecto vamos a hacer un reloj digital en VHDL utilizando una MachXO2 con salida a 6 displays de 7 segmentos de tipo ánodo.

Este proyecto cuenta con cierto nivel de complejidad ya que se tiene que programar un archivo por cada procedimiento que vamos a utilizar, primero se tiene que programar el contador de reloj, que es una parte muy importante ya que aquí se va a designar como va a funcionar.

Después tenemos que programar la parte que se le conoce como el "clk" que para este caso va a ser de 1 HZ .
Seguido de ello es necesario que se programe la parte de los siete segmentos que nos servirá.

Porque esta unidad contiene lo necesario para mostrar dígitos en los cuatro visualizadores.
Contador de reloj: Este componente se encargará de contar de 0 a 23 para las horas y de 0 a 59 para los minutos.

Las salidas corresponden a los cuatro dígitos: dos para mostrar la hora y dos para mostrar los minutos.

Reloj de 1Hz: Se utilizará como señal de entrada en el contador (contaremos cada segundo).
Existen diversas formas para realizar el contador del reloj, todas con ventajas y desventajas. En esta entrada se optó por usar cuatro contadores independientes (uno por dígito) para evitar posteriores conversiones de números binarios a BCD.

El display es un componente, que es utilizado para representar números y algunas letras,este se encuentra compuesto por siete u ocho leds ensamblados por separado, logrando combinarlos elementos creando así símbolos. Los primeros siete segmentos se encargan de formar el símbolo y con el octavo podemos encender y apagar el punto decimal.Existen dos tipos de display uno que es el ánodo común y el otro que es el cátodo común en este caso para mostrar los minutos y las horas vamos ha usar un display cátodo común.

Palabra clave: Binarios, BCD, Clk,Contador,Complejidad, HZ, Horas, MachXO2,Reloj, VHDL, display, elementos, ánodo, contador.

Introducción

Desde hace muchos siglos el hombre establece su relación con el tiempo, basado sobretodo en los fenómenos naturales constantes como el día y la noche, el movimiento del sol, de los planetas y de las estrellas. De esta manera se realizaron construcciones, calendarios, y otros elementos útiles para medir el tiempo; fue el nacimiento del reloj. Los primeros relojes se construyeron utilizando la sombra del sol y su variación de acuerdo con la posición. Luego aparecieron los relojes mecánicos que han acompañado al hombre durante muchos años y de los cuales se han realizado verdaderas obras de arte.En el nacimiento y desarrollo de la tecnología electrónica, no podía faltar su aporte a la medición del tiempo. Esta ciencia ha facilitado la elaboración de relojes de todo tipo, desde modelos personales de muy bajo costo hasta sistemas altamente sofisticados cuya operación está controlada por un microprocesador miniatura con un tamaño de unos pocos milímetros

Damos por comienzo la parte del desarrollo de nuestro proyecto, anteriormente no teníamos idea de como presentar nuestro proyecto, pero basándonos en investigación pudimos lograr sacar esta maravillosa idea, de principal el problema a resolver fue plasmar la idea del equipo, de momento queríamos hacer una marquesina la cual esta nos mostraba los nombres de los integrantes de nuestro equipo en 6 displays, por los tiempos y demás decidimos crear este pequeño pero funcional proyecto, lo cual es un reloj que se muestra en 6 displays de tipo ánodo, de cierto momento en la primera etapa del desarrollo tuvimos problemas al armado de nuestro circuito, dado que este no quería agarrar como debería, después de muchos momentos de pruebas y fallos, logramos hacer que nuestro reloj funcionara, uno de los aspectos importantes a mencionar es que podemos ajustar la hora la cual nos permite tener varias horas o ajustarlas de acuerdo a nuestra necesidad, es importante mencionar que este tipo de prácticas se puede implementar con la placa MatchOSX2 o bien en una gal22v10 nosotros por comodidad y rapidez, además como de más puertos de entradas y salidas preferimos usar esta.

Como parte del proceso a continuación se muestran los procesos a seguir para poder hacer el reloj digital usando el software lattice Diamond para usar la MachXO2, así que iniciaremos el proceso de instalación de Lattice Diamond, los paso a seguir son los siguientes

Instalación de Lattice Diamond

Principalmente hay que registrarnos en la página principal la cual el enlace es el siguiente: http://www.latticesemi.com/en/Accounts/AccountRegister es importante mencionar que hay que llenar un formaluario bastante largo para que nos puedan proporicionar nuestra licencia para que el programa funcione, es importante mencionar que todo el contenido en la página viene en el idioma Ingles. Una vez hecho eso haro vamos a la parte de la descarga del software, de cierto modo a que se supone que el usuario está logueado, de tal manera que no no lo está lea con con atención el párrafo de la introducción

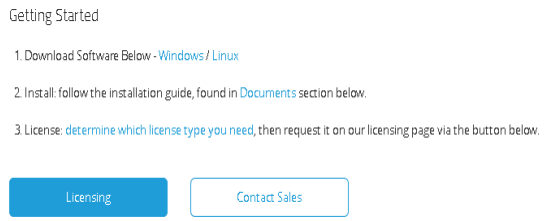


Figura 1: Botón de descarga para software lattice

Ahora daremos click en la parte que viene en azul y dice Windows seguido de eso, darle en la primera opción o verificar que tipo de sistema se maneja en su sistema operativo.

Figura 2: Descargar software lattice.

Después terminando tendremos que aceptar los términos y licencia otorgados por lattice, una vez aceptado descargamos y seguimos el proceso de instalación correspondiente al programa.

Al terminar nuestra correspondiente instalación procederemos a activar nuestro producto, en la parte donde se descarga el programa en la parte de abajo viene un boton azul que dice Licensing.

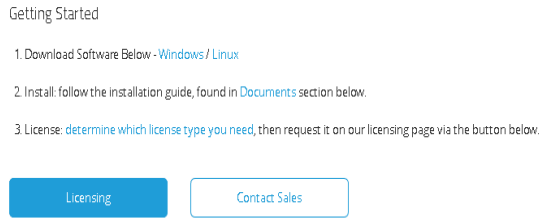


Figura 3: El botón para descargar la licencia.

Le damos click en la parte donde dice Licensing y nos debe de aparecer algo como en la siguiente imagen:

Figura 2: Descargar licencia para laticce

Después al darle click nos aparece una ventana como la siguiente, **NOTA es importante conocer nuestra dirección física (MAC ADDRESS) ya que de esto va a depender que se active nuestro programa, en este caso estamos usando windows lo cual usamos el comando ipconfig /all para saber o tener mas detalles sobre nuestra hardware de red.**

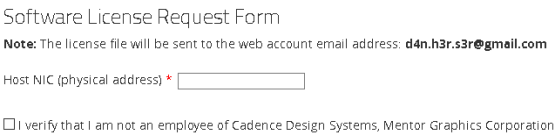


Figura 4: Campo para ingresar nuestra dirección física.

Una vez proporcionada la información e ingresado la licencia en el programa podemos realizar nuestra práctica.

Desarrollo de la práctica

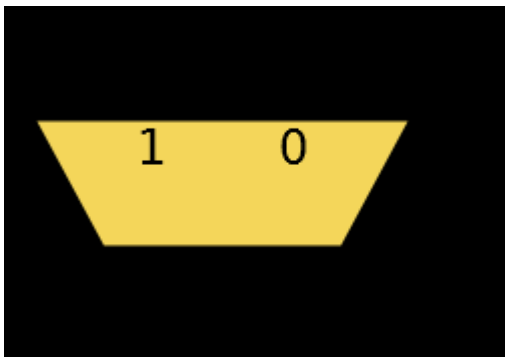
Primero iniciamos programando nuestro archivo principal que es el "contador_reloj.vhd" el cual tendrá la arquitectura de cómo va a funcionar y la asignación del display.

Después procedemos a crear el archivo "siete segmentos_completo.vhd" Será utilizado un multiplexor que nos permite que acepta varias entradas y solamente permite a una de ellas alcanzar la salida. La figura 1 muestra el diagrama de un multiplexor, donde se observa que la salida *Z* puede tomar el valor de *A* o *B*, pero no de ambos a la vez, en base al valor del parámetro de selección S_0. Un claro ejemplo de un multiplexor se encuentra en la televisión, donde solamente se muestra

en pantalla el canal de deportes o el canal

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

entity contador_reloj is
    PORT (
        clk  : IN  STD_LOGIC; --Reloj de 1Hz.
        reset: IN  STD_LOGIC; --Señal de reset.
        H1   : OUT STD_LOGIC_VECTOR(2 DOWNTO 0); --Segundo digito de la hora.
        H0   : OUT STD_LOGIC_VECTOR(3 DOWNTO 0); --Primer digito de la hora.
        M1   : OUT STD_LOGIC_VECTOR(2 DOWNTO 0); --Segundo digito de los minutos.
        M0   : OUT STD_LOGIC_VECTOR(3 DOWNTO 0)  --Primer digito de los minutos.
    );
end contador_reloj;
```

de música, pero no ambos a la vez (al menos hasta hace unos años, claro está).



Después creamos el clk para que podamos llevar un control en la manera van cambiando los dígitos.

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

entity clk1Hz is
    Port (
        entrada: in  STD_LOGIC;
        reset  : in  STD_LOGIC;
        salida : out STD_LOGIC
    );
end clk1Hz;

architecture Behavioral of clk1Hz is
    signal temporal: STD_LOGIC;
    signal contador: integer range 0 to
    24999999 := 0;
begin
    divisor_frecuencia: process (reset,
    entrada) begin
        if (reset = '1') then
            temporal <= '0';
            contador <= 0;
        elsif rising_edge(entrada) then
            if (contador = 24999999)
            then
                temporal <= NOT(
                temporal);
```

La entidad del contador se describe en la figura 2, donde se observa que solamente tiene dos entradas: el reloj de 1 Hz (o 1 segundo) y la señal de *reset*. Las salidas corresponden a los cuatro digitos: dos para mostrar la hora y dos para mostrar los minutos.

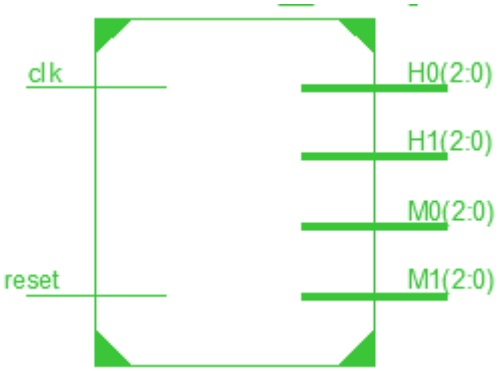


Figura 2: Entradas y salidas del contador.

Se anexan el código de los siete segmentos para el multiplexor.

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

entity siete_segmentos is
    PORT (
        entrada: IN
STD_LOGIC_VECTOR(5 downto 0);
        salida : OUT
STD_LOGIC_VECTOR(7 downto 0)
    );
end siete_segmentos;

architecture Behavioral of siete_segmentos
is
begin
    visualizador: process (entrada) begin
        case entrada is
            when "000000" =>  salida <=
x"C0"; -- 0
            when "000001" =>  salida <=
x"F9"; -- 1
```

```
        when "000010" =>  salida <= x"A4"; -- 2
        when "000011" =>  salida <= x"B0"; -- 3
        when "000100" =>  salida <= x"99"; -- 4
        when "000101" =>  salida <= x"92"; -- 5
        when "000110" =>  salida <= x"82"; -- 6
        when "000111" =>  salida <= x"F8"; -- 7
        when "001000" =>  salida <= x"80"; -- 8
        when "001001" =>  salida <= x"98"; -- 9
        when "001010" =>  salida <= x"88"; -- A
        when "001011" =>  salida <= x"83"; -- B
        when "001100" =>  salida <= x"C6"; -- C
        when "001101" =>  salida <= x"A1"; -- D
        when "001110" =>  salida <= x"86"; -- E
        when "001111" =>  salida <= x"8E"; -- F
        when "010000" =>  salida <= x"90"; -- G
        when "010001" =>  salida <= x"89"; -- H
        when "010010" =>  salida <= x"E6"; -- I
        when "010011" =>  salida <= x"E1"; -- J
        when "010100" =>  salida <= x"85"; -- K
        when "010101" =>  salida <= x"C7"; -- L
        when "010110" =>  salida <= x"C8"; -- M
        when "010111" =>  salida <= x"AB"; -- N
        when "011000" =>  salida <= x"C0"; -- O
        when "011001" =>  salida <= x"8C"; -- P
        when "011010" =>  salida <= x"98"; -- Q
        when "011011" =>  salida <= x"AF"; -- R
        when "011100" =>  salida <= x"92"; -- S
        when "011101" =>  salida <= x"87"; -- T
        when "011110" =>  salida <= x"E3"; -- U
        when "011111" =>  salida <= x"C1"; -- V
        when "100000" =>  salida <= x"E2"; -- W
        when "100001" =>  salida <= x"8F"; -- X
        when "100010" =>  salida <= x"91"; -- Y
        when "100011" =>  salida <= x"B6"; -- Z
        when "100100" =>  salida <= x"BF"; -- -
        when "100101" =>  salida <= x"F7"; -- _
        when "100110" =>  salida <= x"7F"; -- .
        when others   =>  salida <= x"FF"; -- Nada
    end case;
  end process;
end Behavioral;
```

Una vez que tenemos una primera parte que es el multiplexor, procedemos con el clk.

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

entity clk1Hz is
  Port (
```

```
        entrada: in  STD_LOGIC;
        reset  : in  STD_LOGIC;
        salida : out STD_LOGIC
    );
end clk1Hz;

architecture Behavioral of clk1Hz is
    signal temporal: STD_LOGIC;
    signal contador: integer range 0 to
24999999 := 0;
begin
    divisor_frecuencia: process (reset,
entrada) begin
        if (reset = '1') then
            temporal <= '0';
            contador <= 0;
        elsif rising_edge(entrada) then
            if (contador = 24999999) then
                temporal <= NOT(temporal);
                contador <= 0;
            else
                contador <= contador+1;
            end if;
        end if;
    end process;

    salida <= temporal;
end Behavioral;
```

Una vez que tenemos los dos partes que necesitamos para que nuestro código funcione.
Lo unimos todo en nuestra función principal que es el reloj.

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

LIBRARY lattice;       --para habilitar el
sínbolo de +
USE lattice.components.all;

entity reloj is
port (
                    display                 :
          out STD_LOGIC_VECTOR(7
downto 0);
                    mux
          :         out
STD_LOGIC_VECTOR(5 downto 0);
                    ajustemin               :
          in STD_LOGIC;
                    ajustehra               :
          in STD_LOGIC;
                    reset                   :
          in STD_LOGIC --AQUI
AGREEGUÉ EL PUNTO Y COMA QUE
GENERA UN ERROR
);
end reloj;

architecture arq of reloj is

          signal clk50mhz : std_logic;

          --Oscilador machxo2
          COMPONENT OSCH
      GENERIC(
          NOM_FREQ: string := "53.20");
      PORT(
          STDBY    : IN  STD_LOGIC;
          OSC      : OUT STD_LOGIC;
          SEDSTDBY : OUT STD_LOGIC);
          END COMPONENT;


   --bajar  max count para que el reloj vaya
más rápido
   --velz:
   --2000000 aún se percibe el cambio en
los segundos
   --1000000

  constant max_count: INTEGER :=
25000001 ; -- 50000000/2 -----menos 1 ->
24999999
```

```
  constant max_refresh_count: INTEGER
:= 100000; -- 50Mzh/100000=500Hz

  signal count: INTEGER range 0 to
max_count;
  signal refresh_count: INTEGER range 0
to max_refresh_count;
  signal refresh_state:
STD_LOGIC_VECTOR(2 downto 0) :=
(others => '0');
  signal clk_state: STD_LOGIC := '0';
  signal display_sel:
STD_LOGIC_VECTOR(5 downto 0) :=
(others => '0');
  shared variable hora1, hora2, min1,
min2,seg1,seg2: INTEGER range 0 to 10
:= 0;
  shared variable segundos: INTEGER
range 0 to 59 := 0;
  function digito(numero:INTEGER)
return STD_LOGIC_VECTOR is variable
salida: STD_LOGIC_VECTOR(7 downto
0);

  begin

                --Para la salida de los
números en el display
                case numero is
                        when 0 => salida :=
"11000000";
                        when 1 => salida :=
"11111001";
                        when 2 => salida :=
"10100100";
                        when 3 => salida :=
"10110000";
                        when 4 => salida :=
"10011001";
                        when 5 => salida :=
"10010010";
                        when 6 => salida :=
"00000010";
                        when 7 => salida :=
"11111000";
                        when 8 => salida :=
"10000000";
                        when 9 => salida :=
"10010000";
                        when others =>
salida := "11111111";

                end case;
                return(salida); --regresa la
asignación del número que es.

  end digito;

  begin
    OSCInst0: OSCH
      GENERIC MAP (NOM_FREQ  =>
"53.20")
      PORT MAP (STDBY => '0', OSC =>
clk50mhz, SEDSTDBY => OPEN);
      mux <= display_sel;

    gen_clock: process(clk50mhz, clk_state,
count)
    begin

        if clk50mhz'event and
clk50mhz='1' then
                -- contador de 1Hz
                if count < max_count then
                        count <= count + 1;
                else
                        clk_state <= not
clk_state;
                        count <= 0;
                end if;

                -- contador 500Hz (para el
rifresh del display)
                if refresh_count <
max_refresh_count then
                        refresh_count <=
refresh_count + 1;
```

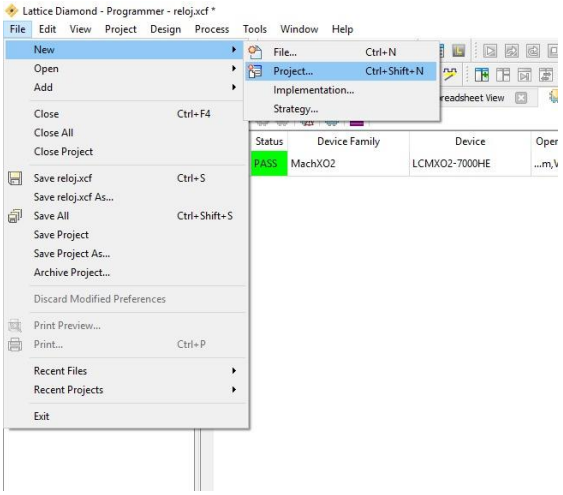
```
                else
                        refresh_state <=
refresh_state + 1;
                        if refresh_state
="111" then
                                refresh_state
<="000";
                        end if;
                        refresh_count <= 0;
            end if;
      end if;
   end process;

         --Para mostrar en el display
indicado

          show_display:
process(refresh_state)

                        begin
                        -- selección del
display
                         case refresh_state is
                                when
"000" => display_sel <= "111110";
                                when
"001" => display_sel <= "111101";
                                when
"010" => display_sel <= "111011";
                                when
"011" => display_sel <= "110111";
                                when
"101" => display_sel <= "101111";
                                when
"110" => display_sel <= "011111";
                                when
others => display_sel <= "000000";
                                end case;

                                -- mostrar
hora
                                case
display_sel is
                                        when
"111110" => display <= digito(hora2);
                                        when
"111101" => display <= digito(hora1);
                                        when
"111011" => display <= digito(min2);
                                        when
"110111" => display <= digito(min1);
                                        when
"101111" => display <= digito(seg2);
                                        when
"011111" => display <= digito(seg1);

                                        when
others => display <= "11111111";
                                end case;


          end process;

 persecond: process (clk_state)
          begin

                if clk_state'event and
clk_state='1' then

                        ----------
hora2hora1:min2min1


                        --Para volver a ceros
el reloj regresamos todos los contadores

                        if reset = '1' then
                                seg1:=0;
                                seg2:=0;
                                min2 := 0;
                                min1 :=0;
                                hora2 := 0;
                                hora1 := 0;
                        end if;

                        --para el ajuste de
los minutos
```

```
                    if ajustemin='1' then
                        min1 :=
min1+1;
                        if(min1  =
10) then

                            min2
:= min2 + 1;

        if(min2 = 6 and min1 = 10) then

        min2 := 0;
                            end
if;

                            min1
:= 0;
                        end if;


                    end if;


                    ----ajuste de la hora,
se deben tratar los casos por separado

                    if ajustehra='1' then
                        hora2 :=
hora2+1;

                        if (hora2 =
10 AND hora1 < 2 ) then

        hora1 := hora1 + 1;

        hora2 := 0;
                            end
if;

                        if (hora1 = 2
AND hora2 = 4) then    --
hora1hora2:min2min1
                            hora2
:= 0;
                            hora1
:=0;
                        end if;

                end if;


                -- contador de
segundos
                if segundos < 59
then
                    segundos :=
segundos + 1;

        seg1:=seg1+1;
                    if(seg1 = 10)
then
                        seg2
:= seg2 + 1;

        seg1:= 0;
                    end if;
                else

                    seg1:=0;
                    seg2:=0;
                    segundos :=
0;
                    min1 :=
min1 + 1; -- +1 minuto

                    if min1 = 10
then
                        min1
:= 0;
                        min2
:= min2 + 1;
```

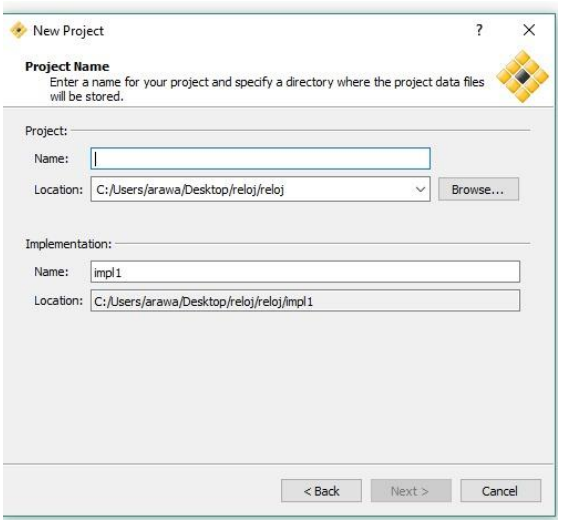
```
                                        --
primer dígito hora    hora2hora1:min2min1
                                            if
min2 = 6 then

        hora2 := hora2 + 1;

        min2 := 0;



        if (hora2 = 10 AND hora1 < 2 )
then

                hora1 := hora1 + 1;

                hora2 := 0;

        end if;


        if (hora1 = 2 AND hora2 = 4) then
--hora1hora2:min2min1

                hora2 := 0;

                hora1 :=0;

        end if;


                                            end
if;


                                end if;


                        end if;


                end if;
        end process;

end arq;
```

El proceso para poder compilar el programa es el siguiente:

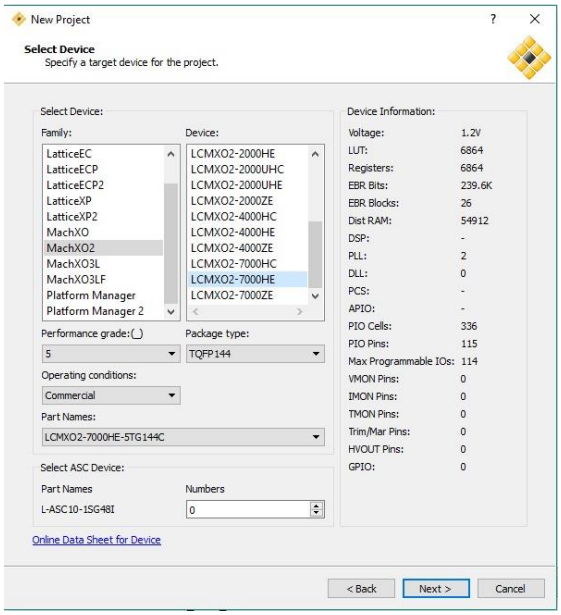
1.- Vamos a crear un nuevo proyecto



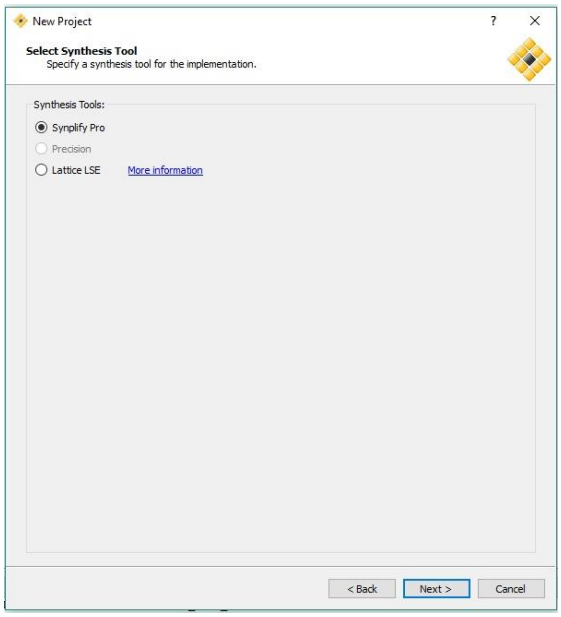
2.- Elegimos la ruta donde lo vamos a guardar nuestro archivo.

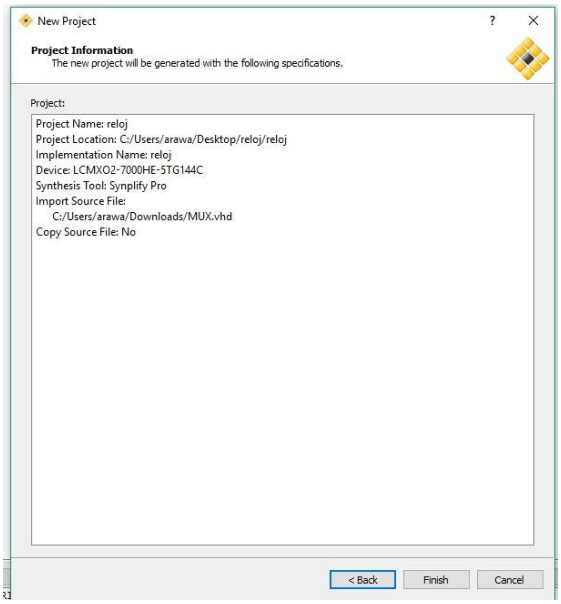


Seleccionamos nuestro dispositivo tarjeta

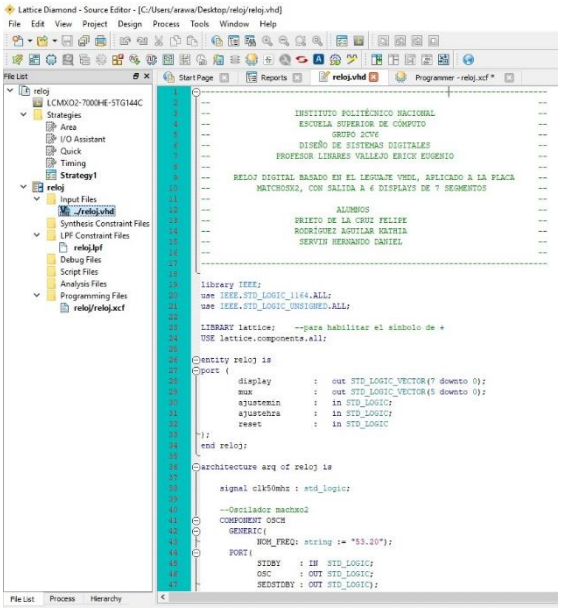


Seleccionamos symplify pro

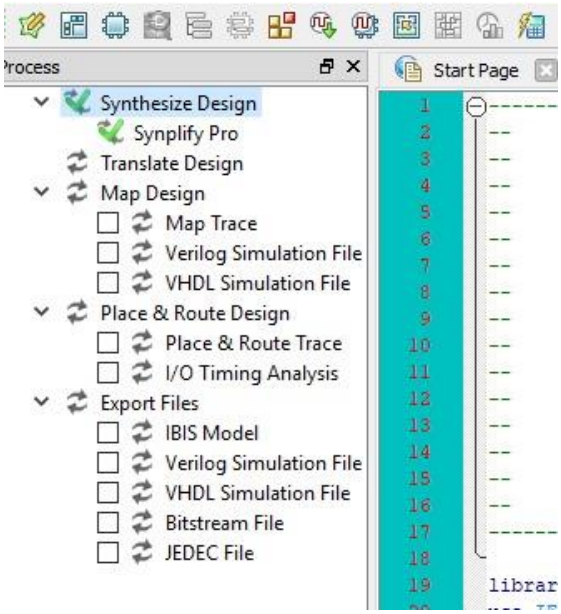




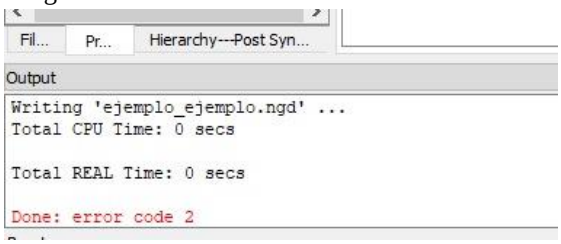
Una vez que tenemos nuestro archivo listo y creamos nuestro proyecto, podemos compilarlo.



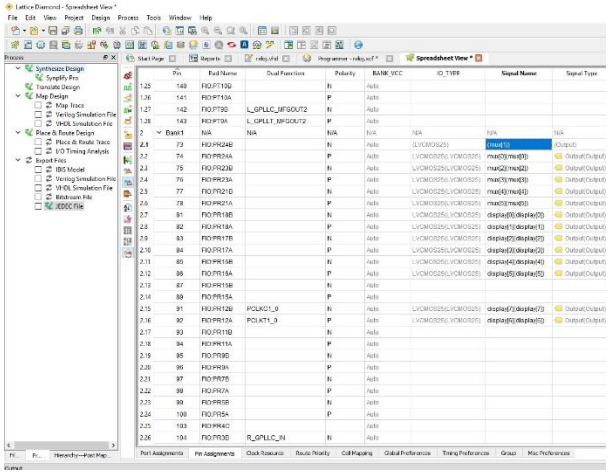
Primero tienes que compilar el JDEC file para que salga correctamente deben importarse todos los archivos.



Si no tenemos errores nos debe mostrar la siguiente imagen.

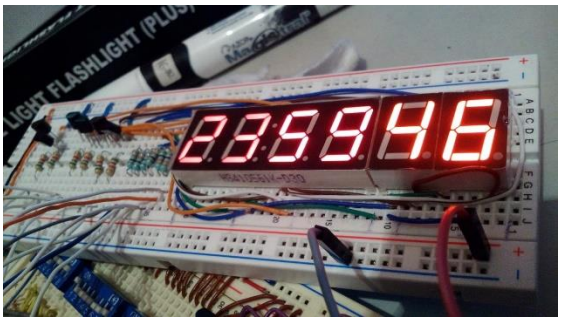


Compilamos y asignamos los pines para terminar.



.

Al final después de asignar los pines y compilar el código nuestro proyecto final quedaría.
Así:



Agradecimiento: Agradecemos la oportunidad que tenemos para poder tener mayor habilidades tanto en la creación de proyectos que involucren tecnología, como también la manera en que se nos está dando el formato para poder aumentar nuestras habilidades para crear proyectos de calidad, por otro lado, también agradecemos al profesor Erick Eugenio Linares Vallejo por toda la atención y supervisión que tuvo al guiarnos por este proyecto, a la compañera Kathia que siempre estuvo al pendiente de que todo quedara a la perfección, al compañero Felipe por tomar Iniciativa e interés en el proyecto, a Daniel Servin por contribuir al proyecto. Pero lo más importante, darle gracias a todas las personas que creyeron en nosotros durante esta etapa de la realización del proyecto. Fue muy difícil documentar pero siempre tratamos de dar lo mejor de nosotros.