Reloj digital con alarma FPGA (Noviembre de 2023)

Laura D. Hernández R, Omar David Garzón H., Daniel A. Morales G.
Facultad de Ingeniería
Departamento de Ingeniería Eléctrica y Electrónica
Universidad Nacional de Colombia
Bogotá DC-Colombia
lhernandezra@unal.edu.co,omgarzon@unal.edu.co,dmoralesgi@unal.edu.co

Resumen - Este informe detalla el diseño y la ejecución de un reloj digital con función de alarma en un dispositivo FPGA utilizando el lenguaje de descripción de hardware VHDL. El objetivo de este proyecto es desarrollar un sistema capaz de mostrar la hora actual y activar una alarma cuando se alcance un tiempo predefinido.

Índice de Términos – FPGA, VHDL

I. INTRODUCCIÓN

El objetivo de este proyecto es diseñar e implementar un reloj digital utilizando VHDL (VHSIC Hardware Description Language). VHDL es un lenguaje de descripción de hardware que permite modelar sistemas digitales a nivel de comportamiento y estructura. Este lenguaje es ampliamente utilizado en la industria para el diseño de circuitos integrados y sistemas embebidos.

En este proyecto, se utilizará VHDL para describir el comportamiento de un reloj digital. El reloj mostrará horas, minutos y segundos en formato de 24 horas. Además, contará con funciones para ajustar la hora y reiniciar el contador.

El diseño se realizará utilizando una metodología top-down, comenzando con una descripción de alto nivel del sistema y refinándola hasta llegar a los componentes individuales. Se utilizarán herramientas de simulación para verificar el correcto funcionamiento del diseño antes de proceder con la implementación física.

Este proyecto proporcionará una excelente oportunidad para aprender y aplicar conceptos fundamentales de electrónica digital y diseño de sistemas digitales con VHDL.

II. DESCRIPCIÓN

Se pretende desarrollar un circuito digital que simule el funcionamiento de un reloj digital avanzado mediante el uso de una FPGA. El objetivo principal de este diseño es permitir dos funcionalidades específicas: ajuste de hora y programación de alarma. Para el ajuste de hora, los usuarios podrán modificar la hora actual del reloj a través de algún método de entrada, como botones o interruptores conectados a la FPGA. Por otro lado, se incorporará la capacidad de programar horas específicas para la activación de una alarma, la cual se activará mediante una bocina como dispositivo de salida. Se espera que el diseño resultante sea preciso, configurable y eficiente en la visualización del tiempo y la activación de la alarma, utilizando la capacidad de la FPGA para procesar información digital y controlar dispositivos de salida. Este proyecto combina conceptos de electrónica digital, programación en VHDL y el uso de la FPGA como plataforma para la implementación del sistema.

III. PLANIFICACIÓN DEL PROYECTO

- ¿Cuál es el objetivo de este proyecto?
  El objetivo de este proyecto es diseñar e implementar un reloj digital utilizando VHDL.

- ¿Cuáles son los indicadores clave de rendimiento (KPI)?
  Los KPIs podrían incluir la precisión del reloj, el tiempo de respuesta al ajustar la hora, y la eficiencia del código VHDL.

- ¿Cuál es el alcance?
  El alcance del proyecto es el diseño, simulación, síntesis, implementación y prueba del reloj digital en VHDL.

- ¿Cuál es el presupuesto?
  El presupuesto dependerá de los recursos necesarios para la implementación física del diseño, como un FPGA o un circuito integrado personalizado.

- ¿Cuáles son los riesgos?
  Los riesgos pueden incluir retrasos en el cronograma, dificultades técnicas durante la implementación, y errores en el diseño.

- ¿Qué miembros del equipo participan?
  Los 3 integrantes somos estudiantes de ingeniería electrónica.

- ¿Qué tareas conlleva?
  Las tareas incluyen la escritura del código VHDL, la simulación del diseño, la síntesis del código en un

diseño de circuito, la implementación física del diseño y las pruebas finales.

- ¿Qué hitos hay que cumplir?
  Los hitos podrían incluir la finalización de la simulación, la finalización de la síntesis, la implementación física y las pruebas finales.

IV. Objetivos SMART:

- Completar el desarrollo del proyecto en el tiempo establecido gestionando eficientemente los recursos y las tareas para cumplir con los plazos, comparando constantemente los tiempos de entrega con los tiempos del cronograma.
- Aumentar la calidad del proyecto sincronizando el reloj de tal forma que su desface no supere 1s por cada 10min.
- Desarrollar un diseño modular del reloj digital que logre explicar el funcionamiento lógico del reloj.
- Completar la implementación del diseño lógico en código y medir la efectividad del código por medio de ensayos en versiones beta del proyecto.
- Integrar de manera exitosa los periféricos requeridos para el proyecto antes de las ultimas 2 semanas del proyecto.

V. Análisis PESTAL

1) Político:
   - La constante alerta de tener un paro de actividades académicas por alguna problemática social que cobre relevancia dentro de la Universidad Nacional.
   - La disponibilidad de incentivos gubernamentales para proyectos de tecnología electrónica que puede impactar sobre los recursos disponibles para el proyecto.

2) Económico:
   La variación de los precios de componentes electrónicos importados que afecten al proyecto, tales como los periféricos requeridos.

3) Social:
   Considerar las preferencias y tendencias de los consumidores en cuanto a relojes digitales, como características deseadas, diseño y marca.

4) Tecnológico:
   - La implementación eficiente del lenguaje de programación para el desarrollo del proyecto, entendiendo las ventajas y desventajas del lenguaje para optimizar el funcionamiento del reloj.
   - Comprender el ciclo de vida de la tecnología utilizada en el proyecto para la planificación a largo plazo.

5) Ambiental:
   Se deben considerar aspectos como el reciclaje de componentes y la gestión de residuos electrónicos cuando el producto haya cumplido con su vida útil.

6) Legal:
   - Asegurarse de que no se infrinjan derechos de propiedad intelectual al utilizar tecnología o patentes en el proyecto.
   - Cumplir con estándares de seguridad del proyecto para evitar problemas legales y proteger a los consumidores.

VI. Procedimiento

1) Simulación

Una vez que el diseño VHDL esté completo, el primer paso en la ejecución del proyecto será la simulación. Utilizaremos un simulador VHDL para probar nuestro diseño y verificar su funcionamiento. Durante esta fase, es importante verificar todas las funcionalidades del reloj, incluyendo la visualización de la hora y las funciones de ajuste.

2) Síntesis

Después de verificar que nuestro diseño funciona como se esperaba en la simulación, el siguiente paso es la síntesis. Durante la síntesis, nuestro código VHDL se traducirá a un diseño de circuito que puede ser implementado en un dispositivo físico con los periféricos.

3) Implementación

Una vez sintetizado el diseño, se procederá con la implementación física. Esto puede implicar la programación de un FPGA con nuestro diseño, o la fabricación de un circuito integrado personalizado con los periféricos correspondientes.

4) Pruebas

Finalmente, después de implementar el diseño, realizaremos pruebas en el hardware para asegurarnos de que funciona correctamente. Esto implicará verificar que el reloj muestra la hora correcta y que las funciones de ajuste funcionan como se esperaba.

VII. Diseño de Módulos

1) Módulo de reloj digital:

La funcionalidad principal del reloj digital depende en gran medida del módulo que se encarga de llevar un registro preciso del tiempo y gestionar la funcionalidad de la alarma.

La importancia de este módulo radica en su capacidad para implementar un contador que se incrementa de manera consistente cada segundo, sincronizado con la frecuencia del reloj del sistema.

El contador es la base del módulo, ya que permite mantener un seguimiento preciso del tiempo actual, lo que a su vez permite que el reloj cumpla su función principal de mostrar horas, minutos y segundos de manera exacta. Al ser impulsado por la frecuencia del reloj del sistema, el contador garantiza una medición temporal precisa y constante.

Además, la función de alarma añade una dimensión adicional a la utilidad del reloj. El módulo compara constantemente el tiempo actual registrado por el contador con la hora establecida para la alarma. Cuando estas dos variables coinciden, el módulo activa una señal de alarma, proporcionando una característica fundamental para aquellos usuarios que desean ser alertados en momentos específicos.

2) Módulo de ajuste de velocidad del reloj

Este módulo es fundamental para la gestión del tiempo en un reloj digital, ya que permite la modificación de la frecuencia del reloj del sistema, lo que a su vez posibilita la aceleración o ralentización del tiempo. Su implementación puede involucrar técnicas como divisores de frecuencia o el uso de un oscilador controlado por voltaje (VCO). Esta capacidad de variar la frecuencia resulta muy valiosa en diversas aplicaciones, tales como simulaciones detalladas, pruebas de rendimiento y ahorro de energía en sistemas embebidos. En resumen, el módulo de variación de frecuencia añade flexibilidad al reloj, adaptándose a diferentes necesidades y escenarios mediante el ajuste dinámico de la velocidad del tiempo.

3) Módulo de la implementación salida de audio

El módulo generador de señal de audio es esencial para convertir información digital en una forma analógica reproducible mediante altavoces o auriculares. Utiliza un convertidor digital-analógico (DAC), un amplificador y un filtro para lograr una reproducción de audio de alta calidad. Este componente es crucial en dispositivos multimedia y sistemas de entretenimiento, asegurando una experiencia auditiva precisa y nítida. En resumen, su función principal es traducir señales digitales en sonidos audibles mediante una combinación de componentes clave.

4) Módulo de LCD

El módulo LCD tiene la responsabilidad de controlar la interfaz de visualización en un Liquid Crystal Display. Entre sus funciones principales se encuentran la transferencia de datos desde el sistema a la pantalla LCD, la gestión de la activación/desactivación de píxeles para la representación visual y el control del cursor para una presentación precisa. Este módulo se comunica con el sistema a través de VHDL y se integra con otros componentes, como el contador de tiempo y la lógica de la alarma, para coordinar la presentación de información en la pantalla LCD. La implementación en VHDL abarca la lógica necesaria para la transferencia de datos y el control de la interfaz con el LCD.

VII. Código VHDL

```
library IEEE; -- Importa las bibliotecas estándar de VHDL para trabajar con lógica y aritmética
use IEEE.std_logic_1164.all; -- Proporciona el tipo de datos std_logic
use IEEE.numeric_std.all; -- Proporciona funciones aritméticas para std_logic_vector
use IEEE.std_logic_unsigned.all; -- Proporciona funciones aritméticas para std_logic_vector sin signo
use IEEE.std_logic_arith.all; -- Proporciona funciones aritméticas para std_logic_vector

USE WORK.COMANDOS_LCD_REVC.ALL; -- Define la entidad LIB_LCD_INTESC_REVC con varias entradas y salidas
entity LIB_LCD_INTESC_REVC is
PORT(CLK: IN STD_LOGIC; -- Reloj de entrada
   RS : OUT STD_LOGIC; -- Señal de registro de selección de salida
   RW : OUT STD_LOGIC; -- Señal de lectura/escritura de salida
   ENA : OUT STD_LOGIC; -- Señal de habilitación de salida
   LED_SIGNAL: OUT STD_LOGIC; -- Señal de salida para el LED
   parlante: OUT STD_LOGIC; -- Señal de salida para el parlante
          CORD : IN STD_LOGIC; -- Señal de entrada
   CORI : IN STD_LOGIC; -- Señal de entrada
   DATA_LCD: OUT STD_LOGIC_VECTOR(7 DOWNTO 0); -- Datos de salida para la pantalla LCD
   BLCD :  OUT STD_LOGIC_VECTOR(7 DOWNTO 0) -- Señal de salida para la retroiluminación de la pantalla LCD
);
end LIB_LCD_INTESC_REVC;

-- Define la arquitectura Behavioral de la entidad LIB_LCD_INTESC_REVC
architecture Behavioral of LIB_LCD_INTESC_REVC is
   -- Define un nuevo tipo de datos llamado RAM que es un array de std_logic_vector
   TYPE RAM IS ARRAY (0 TO  60) OF STD_LOGIC_VECTOR(8 DOWNTO 0);
   -- Define una señal llamada INST de tipo RAM
SIGNAL INST : RAM;

COMPONENT PROCESADOR_LCD_REVC is -- Define un componente llamado PROCESADOR_LCD_REVC

PORT( -- Define las entradas y salidas del componente
CLK : IN STD_LOGIC; -- Señal de reloj de entrada
   VECTOR_MEM : IN STD_LOGIC_VECTOR(8 DOWNTO 0); -- Vector de memoria de entrada
```

```
   INC_DIR : OUT INTEGER RANGE 0 TO 1024; --
Dirección incremental de salida
   CORD : IN STD_LOGIC; -- Señal de entrada
   CORI : IN STD_LOGIC; -- Señal de entrada
   RS : OUT STD_LOGIC; -- Señal de salida
   RW : OUT STD_LOGIC; -- Señal de salida
   DELAY_COR : IN INTEGER RANGE 0 TO 1000; --
Retardo de entrada
   BD_LCD : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
-- Salida de datos para la pantalla LCD
   ENA  : OUT STD_LOGIC; -- Señal de habilitación de
salida
C1A,C2A,C3A,C4A : IN STD_LOGIC_VECTOR(39
DOWNTO 0); -- Entradas de datos
   C5A,C6A,C7A,C8A : IN STD_LOGIC_VECTOR(39
DOWNTO 0); -- Entradas de datos
   DATA : OUT STD_LOGIC_VECTOR(7 DOWNTO 0) --
Salida de datos
);

end  COMPONENT PROCESADOR_LCD_REVC; -- Fin de
la definición del componente

-- Define un componente llamado
CARACTERES_ESPECIALES_REVC
COMPONENT CARACTERES_ESPECIALES_REVC is
-- Define las entradas y salidas del componente
PORT(
   C1,C2,C3,C4:OUT STD_LOGIC_VECTOR(39 DOWNTO
0); -- Salidas de datos
   C5,C6,C7,C8:OUT STD_LOGIC_VECTOR(39 DOWNTO
0); -- Salidas de datos
   CLK : IN STD_LOGIC -- Señal de reloj de entrada
);
end COMPONENT CARACTERES_ESPECIALES_REVC;
-- Fin de la definición del componente

-- Define una constante llamada CHAR1 y le asigna el valor 1
CONSTANT CHAR1 : INTEGER := 1;

-- Define una constante llamada CHAR2 y le asigna el valor 2
CONSTANT CHAR2 : INTEGER := 2;
CONSTANT CHAR3 : INTEGER := 3;


-- Define una constante llamada CHAR3 y le asigna el valor 3
CONSTANT CHAR3 : INTEGER := 3;

-- Define una constante llamada CHAR4 y le asigna el valor 4
CONSTANT CHAR4 : INTEGER := 4;

-- Define una constante llamada CHAR5 y le asigna el valor 5
CONSTANT CHAR5 : INTEGER := 5;

-- Define una constante llamada CHAR6 y le asigna el valor 6
CONSTANT CHAR6 : INTEGER := 6;

-- Define una constante llamada CHAR7 y le asigna el valor 7
CONSTANT CHAR7 : INTEGER := 7;

-- Define una constante llamada CHAR8 y le asigna el valor 8
CONSTANT CHAR8 : INTEGER := 8;

-- Define una señal llamada DIR con un rango de 0 a 1024 y
un valor inicial de 0
SIGNAL DIR : INTEGER RANGE 0 TO 1024 := 0;

-- Define una señal llamada VECTOR_MEM_S que es un
vector de 9 bits
SIGNAL VECTOR_MEM_S : STD_LOGIC_VECTOR(8
DOWNTO 0);

-- Define tres señales de un solo bit llamadas RS_S, RW_S y
E_S
SIGNAL RS_S, RW_S, E_S : STD_LOGIC;

-- Define una señal llamada DATA_S que es un vector de 8
bits
SIGNAL DATA_S : STD_LOGIC_VECTOR(7 DOWNTO 0);

-- Define una señal llamada DIR_S con un rango de 0 a 1024
SIGNAL DIR_S : INTEGER RANGE 0 TO 1024;

-- Define una señal llamada DELAY_COR con un rango de 0
a 1000
SIGNAL DELAY_COR : INTEGER RANGE 0 TO 1000;

-- Define cuatro señales de 40 bits llamadas C1S a C4S
SIGNAL C1S,C2S,C3S,C4S : STD_LOGIC_VECTOR(39
DOWNTO 0);

-- Define cuatro señales de 40 bits llamadas C5S a C8S
SIGNAL C5S,C6S,C7S,C8S : STD_LOGIC_VECTOR(39
DOWNTO 0);

-- Define una señal llamada conta_retardo con un rango de 0 a
escala_1s y un valor inicial de 0
SIGNAL conta_retardo : INTEGER RANGE 0 TO escala_1s
:= 0;

-- Define una señal de un solo bit llamada b
signal b : std_logic := '0';

-- Define dos señales llamadas unidades y decenas con un
rango de 0 a 9 y un valor inicial de 0
signal unidades, decenas : integer range 0 to 9 :=0;

-- Define cuatro señales llamadas unidades_minuto,
decenas_minuto, unidades_hora y decenas_hora con un rango
apropiado y un valor inicial de 0
signal unidades_minuto : integer range 0 to 9 := 0;
signal decenas_minuto : integer range 0 to 5 := 0;
signal unidades_hora : integer range 0 to 9 := 0;
signal decenas_hora : integer range 0 to 2 := 0;

-- Define una señal de un solo bit llamada alarma con un valor
inicial de '0'
signal alarma : std_logic := '0';
```

```
-- Define una señal llamada tiempo_alarma con un rango de 0 a 86399 y un valor inicial de 10
signal tiempo_alarma : integer range 0 to 86399 := 10;

-- Inicia la descripción de la arquitectura
BEGIN
-- Instancia el componente PROCESADOR_LCD_REVC y mapea sus puertos a las señales correspondientes
U1 : PROCESADOR_LCD_REVC PORT MAP(
   CLK  => CLK, -- Mapea la señal CLK al puerto CLK del componente
   VECTOR_MEM => VECTOR_MEM_S, -- Mapea la señal VECTOR_MEM_S al puerto VECTOR_MEM del componente
   RS  => RS_S, -- Mapea la señal RS_S al puerto RS del componente
   RW  => RW_S, -- Mapea la señal RW_S al puerto RW del componente
   ENA => E_S, -- Mapea la señal E_S al puerto ENA del componente
   INC_DIR => DIR_S, -- Mapea la señal DIR_S al puerto INC_DIR del componente
   DELAY_COR => DELAY_COR, -- Mapea la señal DELAY_COR al puerto DELAY_COR del componente
   BD_LCD => BLCD, -- Mapea la señal BLCD al puerto BD_LCD del componente
   CORD => CORD, -- Mapea la señal CORD al puerto CORD del componente
   CORI => CORI, -- Mapea la señal CORI al puerto CORI del componente
   C1A =>C1S, -- Mapea la señal C1S al puerto C1A del componente
   C2A =>C2S, -- Mapea la señal C2S al puerto C2A del componente
   C3A =>C3S, -- Mapea la señal C3S al puerto C3A del componente
   C4A =>C4S -- Mapea la señal C4S al puerto C4A del componente
);

            -- Continuación del mapeo de puertos para el componente PROCESADOR_LCD_REVC
C5A =>C5S, -- Mapea la señal C5S al puerto C5A del componente
C6A =>C6S, -- Mapea la señal C6S al puerto C6A del componente
C7A =>C7S, -- Mapea la señal C7S al puerto C7A del componente
C8A =>C8S, -- Mapea la señal C8S al puerto C8A del componente
DATA  => DATA_S -- Mapea la señal DATA_S al puerto DATA del componente
);

-- Instancia el componente CARACTERES_ESPECIALES_REVC y mapea sus puertos a las señales correspondientes
U2 : CARACTERES_ESPECIALES_REVC PORT MAP(
   C1 =>C1S, -- Mapea la señal C1S al puerto C1 del componente
   C2 =>C2S, -- Mapea la señal C2S al puerto C2 del componente
   C3 =>C3S, -- Mapea la señal C3S al puerto C3 del componente
   C4 =>C4S, -- Mapea la señal C4S al puerto C4 del componente
   C5 =>C5S, -- Mapea la señal C5S al puerto C5 del componente
   C6 =>C6S, -- Mapea la señal C6S al puerto C6 del componente
   C7 =>C7S, -- Mapea la señal C7S al puerto C7 del componente
   C8 =>C8S, -- Mapea la señal C8S al puerto C8 del componente
   CLK => CLK -- Mapea la señal CLK al puerto CLK del componente
);

-- Asigna el valor de la señal DIR_S a la señal DIR
DIR <= DIR_S;

-- Asigna el valor del elemento del array INST en la posición DIR a la señal VECTOR_MEM_S
VECTOR_MEM_S <= INST(DIR);

-- Asigna el valor de las señales RS_S, RW_S, E_S y DATA_S a las señales RS, RW, ENA y DATA_LCD respectivamente
RS <= RS_S;
RW <= RW_S;
ENA <= E_S;
DATA_LCD <= DATA_S;
DELAY_COR <= 600; --Modificar esta variable para la velocidad del corrimiento.

-- Inicia un proceso que se activa en el flanco ascendente de la señal de reloj
process(CLK)
begin
   if rising_edge(CLK) then
      -- Incrementa el contador de retardo en cada ciclo de reloj
      conta_retardo <= conta_retardo+1;
      -- Si el contador de retardo alcanza el valor de escala_1s, lo reinicia a 0 y establece la señal b en '1'
      if conta_retardo = escala_1s then
         conta_retardo <= 0;
         b <= '1';
      else
         b <= '0';
      end if;
   end if;
end process;

-- Inicia otro proceso que se activa en el flanco ascendente de la señal de reloj
process(CLK)
begin
   if rising_edge(CLK) then
```

```
      -- Si la señal b es '1', incrementa las unidades y realiza varias comprobaciones para incrementar las decenas, minutos y horas
      if b = '1' then
         unidades <= unidades+1;
         if unidades = 9 then
            unidades <= 0;
            decenas <= decenas +1;
            if decenas = 5 then
               decenas <= 0;
               unidades_minuto <= unidades_minuto + 1;
               if unidades_minuto = 9 then
                  unidades_minuto <= 0;
                  decenas_minuto <= decenas_minuto + 1;
                  if decenas_minuto = 5 then
                     decenas_minuto <= 0;
                     unidades_hora <= unidades_hora + 1;
                     if unidades_hora = 9 then
                        unidades_hora <= 0;
                        decenas_hora <= decenas_hora + 1;
                        if decenas_hora = 2 then
                           decenas_hora <= 0;
                        end if;
                     end if;
                  end if;
               end if;
            end if;
         end if;
      end if;
      -- Comprueba si es la hora de la alarma y establece la señal de alarma y las señales de salida correspondientes
      if decenas_hora*36000 + unidades_hora*3600 + decenas_minuto*600 + unidades_minuto*60 + decenas*10 + unidades = tiempo_alarma then
         alarma <= '1';
         parlante <= '0';
         LED_SIGNAL <= '1';  -- Enciende el LED
      else
         alarma <= '0';
         parlante <= '1';
         LED_SIGNAL <= '0';  -- Apaga el LED
      end if;
   end if;
end process;

-- Inicia otro proceso que se activa en el flanco ascendente de la señal de reloj
process(CLK)
begin
   if rising_edge(CLK) then
      -- Si la alarma está activada, establece las señales de salida correspondientes
      if alarma = '1' then
         INST(0)  <= LCD_INI("11");
         INST(1)  <= LCD_INI("00");
-- Si la alarma está activada, establece las señales de salida para mostrar "ALARMA" en la pantalla LCD
         INST(2)  <= CHAR(A);
         INST(3)  <= CHAR(L);
         INST(4)  <= CHAR(A);
         INST(5)  <= CHAR(R);
          INST(6)  <= CHAR(M);
         INST(7)  <= CHAR(A);
         INST(8)  <= CODIGO_FIN(1);
      else
-- Si la alarma no está activada, establece las señales de salida para mostrar la hora actual en la pantalla LCD
         INST(0)  <= LCD_INI("11");
         INST(1)  <= LCD_INI("00");
         INST(2)  <= CHAR(H);
         INST(3)  <= CHAR(O);
         INST(4)  <= CHAR(r);
         INST(5)  <= CHAR(a);
         INST(6)  <= CHAR_ASCII(X"3A"); -- Carácter ':'
         INST(7)  <= BUCLE_INI(1);
         INST(8)  <= POS(2,1);
         INST(9)  <= INT_NUM(decenas_hora);
         INST(10) <= INT_NUM(unidades_hora);
         INST(11) <= CHAR_ASCII(X"3A"); -- Carácter ':'
         INST(12) <= INT_NUM(decenas_minuto);
         INST(13) <= INT_NUM(unidades_minuto);
         INST(14) <= CHAR_ASCII(X"3A"); -- Carácter ':'
         INST(15) <= INT_NUM(decenas);
         INST(16) <= INT_NUM(unidades);
         INST(17) <= BUCLE_FIN(1);
         INST(18) <= CODIGO_FIN(1);
      end if;
   end if;
end process;

end Behavioral; -- Fin de la descripción de la arquitectura
```

VIII. Implementación Física

Figura: Montaje físico del reloj digital con alarma en la FPGA

IX. Análisis

1) Importancia Fundamental de Cada Módulo:

La importancia fundamental de cada módulo en el diseño del reloj digital y cómo su inclusión estratégica contribuye a la consecución de los objetivos del proyecto. La atención meticulosa a la interconexión y la interoperabilidad de los módulos demuestra un enfoque sistemático y metódico que asegura una funcionalidad sin fisuras y una experiencia del usuario enriquecedora. Cada módulo es esencial para el rompecabezas del diseño del reloj digital contribuye a la eficiencia operativa, la precisión y la funcionalidad intuitiva del sistema en su conjunto.

2) Contribución a la Eficiencia Operativa:

La implementación precisa y coordinada de los módulos que conforman el reloj digital es un factor determinante en su eficiencia. Este dispositivo, al ser el núcleo del sistema, establece el ritmo y la sincronización precisa gracias al ajuste de velocidad. Además, la salida de audio y el LCD no solo ofrecen una interfaz intuitiva, sino que también contribuyen a mejorar la experiencia del usuario. La selección y disposición cuidadosa de estos módulos demuestran una comprensión profunda de cómo cada uno de ellos contribuye a mejorar la eficiencia operativa del sistema en su conjunto.

3) Enfoque Sistemático y Metódico:

La implementación de estos módulos no es producto del azar, sino el resultado de un enfoque sistemático y metódico. Cada módulo se incorpora de manera deliberada para garantizar una funcionalidad sin interrupciones. Desde el reloj digital que establece la base temporal hasta los módulos periféricos que mejoran la experiencia del usuario, la atención meticulosa a la interconexión y la interoperabilidad revela un diseño que va más allá de la simple acumulación de características, demostrando un compromiso con la coherencia y la simplicidad funcional.

4) Alineación con Objetivos de Eficiencia, Precisión y Funcionalidad:

La ejecución de estos módulos no solo es de carácter técnico, sino también estratégico. Cada uno de ellos se encuentra alineado con los objetivos específicos del proyecto, que abarcan desde la optimización de las operaciones hasta la precisión en la medición del tiempo y la funcionalidad intuitiva. Esta alineación evidencia una clara comprensión de los requisitos del proyecto y cómo cada módulo contribuye de manera exitosa a la consecución de dichos objetivos. La implementación no se limita únicamente a la ejecución de código, sino que se trata de un medio intencional para lograr resultados específicos y deseados.

X. Conclusión

La relevancia estratégica de cada módulo se hace evidente en su implementación, la cual se lleva a cabo con un enfoque riguroso y sistemático. Este enfoque asegura no solo la funcionalidad técnica, sino también la alineación precisa con los objetivos del proyecto, lo que se traduce en un reloj digital eficiente y completamente funcional.

XI. References

[1] Academia del Caos, “Reloj con alarma en VHDL para la tarjeta FPGA DE2-115,” YouTube. Dec. 03, 2018. [Online]. Available: https://www.youtube.com/watch?v=1-NZ5vME4FIW.-K. Chen, *Linear Networks and Systems* (Book style). Belmont, CA: Wadsworth, 1993, pp. 123–135.

[2] “Reloj Digital en VHDL | Ejercicios de Circuitos Digitales | Docsity.” https://www.docsity.com/es/reloj-digital-en-vhdl/5021895/B. Smith, “An approach to graphs of linear forms (Unpublished work style),” unpublished.

[3] Freelancer, “Reloj, cronometro y alarma en VHDL para Basys2 | Freelancer,” Freelancer. https://www.freelancer.es/projects/engineering/reloj-cronometro-alarma-vhdl-para/J. Wang, “Fundamentals of erbium-doped fiber amplifiers arrays (Periodical style—Submitted for publication),” *IEEE J. Quantum Electron.*, submitted for publication.