

Implementación de reloj digital

Tyrone Novillo

Facultad de ingeniería

Universidad de Cuenca

Cuenca, Ecuador

tyrone.novillo@ucuenca.edu.ec

Resumen—This project focuses on developing a digital clock using VHDL and the Basys 3 FPGA board. The clock displays minutes and seconds on 7-segment displays, addressing the technical challenge of showcasing different numbers on different displays while utilizing a shared data bus for all four displays. Individual control of each display via its common anode is employed. The 'reloj' entity is defined, specifying necessary input and output ports along with generic variables to manage frequency division counters. The clock's architecture includes generating 1-second and 1-millisecond clock signals, logic for minute and second counting, and a multiplexing technique to efficiently update display values.

Keywords— Basys 3 FPGA, VHDL, digital clock, multiplexing, 7-segment displays, frequency division.

I. INTRODUCCIÓN

VHDL (VHSIC Hardware Description Language), es un lenguaje de descripción de hardware ampliamente utilizado en el diseño y la simulación de circuitos digitales. VHDL permite describir la estructura, el comportamiento y la funcionalidad de sistemas digitales complejos, como controladores, procesadores, y periféricos, a nivel de abstracción alto. Con VHDL, es posible modelar circuitos digitales de manera precisa y eficiente, facilitando la verificación y el desarrollo de sistemas electrónicos.

El siguiente proyecto se centra en la implementación de un reloj digital utilizando una placa Basys 3 y lenguaje VHDL. Este reloj muestra los minutos y segundos en displays de 7 segmentos, enfrentándose al desafío técnico de compartir un único bus de datos para los cuatro displays. Esto se logra mediante el control individual de cada display a través de su ánodo común. Para ello, se define la entidad `reloj` que especifica los puertos de entrada y salida necesarios, junto con variables genéricas para controlar los contadores de división de frecuencia. La arquitectura del reloj se basa en la generación de señales de reloj de 1 segundo y 1 milisegundo, la lógica de conteo para minutos y segundos, y una técnica de multiplexación para actualizar los valores en los displays de manera eficiente.

II. DESARROLLO

Se requiere hacer un reloj que muestre en los displays 7 segmentos de la placa Basys 3 los minutos y segundos. El problema consiste en mostrar diferentes números a la vez en los diferentes displays de la placa debido a que el bus de datos en la entrada de los displays es el mismo para los cuatro displays. Solamente el encendido y apagado de cada uno de los displays es individual para cada uno controlado por ánodo común. La Figura 1 muestra lo que se pretende implementar en la placa.



Figura 1: Reloj digital

Previo a la implementación, es necesario definir los puertos de entrada y salida dentro de la entidad llamada `reloj`.

II-A. Declaración de la entidad

Para describir la funcionalidad del reloj se ha creado el archivo de diseño `reloj.vhd`. En este diseño, se define la entidad `reloj` que representa el componente y describe sus puertos de entrada y salida.

La entidad `reloj` tiene los siguientes puertos:

- `clk_in`: entrada de reloj de 100 MHz.
- `activ_display`: salida que define que display 7 segmentos está encendido.
- `Display_7_segmentos`: Array de salida que define el valor BCD para escribir sobre el display.

Adicionalmente, se definieron variables genéricas para controlar los contadores que hacen la división de frecuencia.

- `divisor_sec`: Entero que define cada cuanto se alterna la señal de 1 segundo.
- `divisor_ms`: Entero que define cada cuanto se alterna la señal de 1 milí segundo.

II-B. Definición de la arquitectura

Previo a empezar la arquitectura, se definen las siguientes señales intermedias:

- signal `clk_temp_sec`: Señal que contendrá la señal de reloj de 1 segundo
- signal `clk_temp_ms`: Señal que contendrá la señal de reloj de 1ms segundo para mostrar diferentes valores en los displays.
- `contador_sec`, `contador_ms` y `contador`: Contadores enteros para hacer la division de frecuencias.
- `decenas_sec`, `unidades_sec`, `decenas_min`, `unidades_min` : contadores para ponerlos en cada uno de los displays
- signal `Display_BCD` : entero para escribir un valor en los displays

II-B1. Generación de señales de reloj: Posteriormente, se crean dos procesos para generar señales de reloj de 1 segundo y 1 milisegundo a partir de la señal de reloj de entrada de 10MHz (`clk_in`). En el primer proceso, se utiliza un contador (`contador_sec`) para contar los ciclos de la señal de reloj de entrada hasta alcanzar un el divisor `divisor_sec` que representa un segundo. Cuando el flanco de subida del reloj de entrada ocurre, se incrementa el contador hasta que alcanza el divisor, momento en el cual se invierte una señal temporal `clk_temp_sec` y se reinicia el contador para empezar nuevamente.

En el segundo proceso, se realiza un procedimiento similar pero a una escala de milisegundos. El contador `contador_ms` cuenta los ciclos del reloj de entrada hasta llegar al divisor de milisegundos. Cuando se alcanza este divisor, se invierte una señal temporal `clk_temp_ms` y se reinicia el contador. Ambos procesos utilizan el flanco de subida del reloj de entrada para sincronizar sus operaciones y generar las señales de reloj deseadas en las frecuencias especificadas.

II-B2. Reloj digital: Para generar la lógica del reloj, se trabaja con un proceso que cuenta los segundos y minutos basado en una señal de reloj de 1 segundo generada previamente. Se utilizan contadores para las unidades y decenas de segundos y minutos, incrementándolos adecuadamente para reflejar el paso del tiempo.

Cuando ocurre un flanco de subida en la señal temporal de segundos, el proceso verifica el valor de las unidades de segundos; si es 9, se reinicia a 0 y se incrementa la unidad de decenas de segundos. Si la unidad de decenas de segundos alcanza 5, se reinicia a 0 y se incrementa la unidad de minutos. De manera similar, se manejan los minutos en unidades y decenas, llevando un seguimiento del tiempo transcurrido.

II-C. Escritura de valores de reloj en Displays 7 Segmentos

Para mostrar los valores de unidades y decenas de minutos y segundos, se ejecutarán dos procesos para implementar una técnica de multiplexación que permite mostrar unidades y decenas de segundos y minutos en un display de 7 segmentos. El primer proceso se activa con el flanco de subida de una señal de reloj temporal de milisegundos (clk_temp_ms). Este proceso incrementa un contador que define a qué display al cual se está asignando un valor. Se asegura que el contador se reinicie después de contar hasta 3. Esto crea una tasa de refresco de aproximadamente 1 ms, lo que es imperceptible para el ojo humano y evita parpadeos visibles en el reloj.

El segundo proceso se activa cuando el valor del contador es 0, 1, 2 o 3. Dependiendo del valor del contador, se selecciona qué número BCD mostrar en el display de 7 segmentos, correspondiendo a las unidades y decenas de segundos y minutos. Este proceso de multiplexación garantiza que cada segmento del display se actualice a una velocidad suficiente para que el usuario perciba una visualización estable y sin parpadeos notables, gracias a la tasa de refresco de 1 ms controlada por el primer proceso.

II-D. Constraints

El archivo constraints.xdc proporciona las instrucciones necesarias para establecer la conexión entre los pines físicos de la tarjeta Basys 3 y las entradas y salidas definidas en la entidad VHDL. Dentro del archivo, se realizan las siguientes configuraciones:

Se establece configuraciones para la señal de reloj (clk_in) que considera la señal proporcionada por la Basys 3 de 10MHz.

En cuanto al display de 7 segmentos, se asignan los pines correspondientes a cada segmento del display (Display_7segments) y a las señales de activación de los displays (activ_displays). Estos pines también se configuran como LVCMOS33.

III. CONCLUSIONES

En conclusión, el desarrollo de este proyecto ha demostrado la viabilidad de implementar un reloj digital en la placa Basys 3 utilizando VHDL. La técnica de multiplexación empleada, junto con la generación de una señal de 1 milisecondo como tasa de refresco, ha permitido mostrar diferentes valores en diferentes displays de manera eficiente. Esta estrategia de iterar a la tasa de refresco cada display ha garantizado una visualización estable y sin parpadeos notables, lo cual es crucial para la legibilidad y usabilidad del reloj digital.

Además, la división de frecuencia desempeña un papel fundamental en el funcionamiento del reloj, al permitir sincronizar las diferentes operaciones y garantizar que los valores se actualicen con la precisión necesaria. La definición de contadores y la gestión de las señales de reloj de 1 segundo y 1 milisecondo han sido clave para lograr un sistema coherente y funcional.

IV. ANEXOS

IV-A. Archivo contador.vhd

-- Carrera: Ingeniería en Telecomunicaciones
-- Instituto: Universidad de Cuenca

```
-- Autor: Tyrone Novillo
-- Nombre del diseño: reloj
-----
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity reloj is
generic (
divisor_sec : integer := 50_000_000; -- para CK 1[s]
divisor_ms  : integer := 50_000 --para CK 1[ms]
);
port (
clk_in : in std_logic;
activ_displays : out std_logic_vector(3 downto 0);
Display_7segments : out std_logic_vector(6 downto 0)
);
end reloj;

architecture Behavioral of reloj is
signal clk_temp_sec : std_logic;
signal clk_temp_ms : std_logic;
signal contador_sec : integer := 0;
signal contador_ms : integer := 0;
-- contador para activación de los 4
signal contador: integer := 0;
signal Display_BCD : integer range 0 to 9 := 0;
-- Rango de 0 a 5 para decenas de segundos
signal decenas_sec : integer range 0 to 5 := 0;
-- Rango de 0 a 9 para unidades de segundos
signal unidades_sec : integer range 0 to 9 := 0;
-- Rango de 0 a 5 para decenas de minutos
signal decenas_min : integer range 0 to 5 := 0;
-- Rango de 0 a 9 para unidades de minutos
signal unidades_min : integer range 0 to 9 := 0;
begin

-- Proceso para obtener clock de 1 segundo
process (clk_in)
begin
if rising_edge(clk_in) then
if contador_sec < divisor_sec then
contador_sec <= contador_sec + 1;
else
clk_temp_sec <= not clk_temp_sec;
contador_sec <= 0;
end if;
end if;
end process;

-- Proceso para obtener clock de 1m segundo
process (clk_in)
begin
if rising_edge(clk_in) then
if contador_ms < divisor_ms then
contador_ms <= contador_ms + 1;
else
clk_temp_ms <= not clk_temp_ms;
contador_ms <= 0;
end if;
end if;
end process;

-----
-- Conversión BCD para segundos
process (clk_temp_sec)
begin
```

```

if rising_edge(clk_temp_sec) then
if unidades_sec = 9 then
if decenas_sec = 5 then
decenas_sec <= 0;
if unidades_min = 9 then
if decenas_min = 5 then
decenas_min <= 0;
else
decenas_min <= decenas_min +1;
end if;
unidades_min <= 0;
else
unidades_min <= unidades_min + 1;
end if;
else
decenas_sec <= decenas_sec +1;
end if;
unidades_sec <= 0;
else
unidades_sec <= unidades_sec + 1;
end if;

end if;
end process;

-----
process (clk_temp_ms) is
begin
if (clk_temp_ms'event and clk_temp_ms = '1') then
if contador < 3 then
contador <= contador + 1;
else
contador <= 0;
end if;
end if;
end process;

process (contador)
begin
if contador = 0 then
Display_BCD <= unidades_sec;
activ_displays <= "1110";
elsif contador = 1 then
Display_BCD <= decenas_sec;
activ_displays <= "1101";
elsif contador = 2 then
Display_BCD <= unidades_min ;
activ_displays <= "1011";
else
Display_BCD <= decenas_min ;
activ_displays <= "0111";
end if;
end process;

-- Conversión BCD a 7 segmentos para segundos
with Display_BCD select
Display_7segments <=
"0000001" when 0,
"1001111" when 1,
"0010010" when 2,
"0000110" when 3,
"1001100" when 4,
"0100100" when 5,
"0100000" when 6,
"0001111" when 7,
"0000000" when 8,
"0000100" when 9,
"1111111" when others;
end Behavioral;

```

IV-B. Archivo de Constraints Basys-3-Master.xdc

```

# Clock signal
set_property -dict { PACKAGE_PIN W5
    ↪ IOSTANDARD LVCMOS33 } [get_ports clk_in]
create_clock -add -name sys_clk_pin -period
    ↪ 10.00 -waveform {0 5} [get_ports clk_in]

## Segment Display
set_property -dict { PACKAGE_PIN W7
    ↪ IOSTANDARD LVCMOS33 } [get_ports
    ↪ {Display_7segments[6]}]
set_property -dict { PACKAGE_PIN W6
    ↪ IOSTANDARD LVCMOS33 } [get_ports
    ↪ {Display_7segments[5]}]
set_property -dict { PACKAGE_PIN U8
    ↪ IOSTANDARD LVCMOS33 } [get_ports
    ↪ {Display_7segments[4]}]
set_property -dict { PACKAGE_PIN V8
    ↪ IOSTANDARD LVCMOS33 } [get_ports
    ↪ {Display_7segments[3]}]
set_property -dict { PACKAGE_PIN U5
    ↪ IOSTANDARD LVCMOS33 } [get_ports
    ↪ {Display_7segments[2]}]
set_property -dict { PACKAGE_PIN V5
    ↪ IOSTANDARD LVCMOS33 } [get_ports
    ↪ {Display_7segments[1]}]
set_property -dict { PACKAGE_PIN U7
    ↪ IOSTANDARD LVCMOS33 } [get_ports
    ↪ {Display_7segments[0]}]

set_property -dict { PACKAGE_PIN U2
    ↪ IOSTANDARD LVCMOS33 } [get_ports
    ↪ {activ_displays[0]}]
set_property -dict { PACKAGE_PIN U4
    ↪ IOSTANDARD LVCMOS33 } [get_ports
    ↪ {activ_displays[1]}]
set_property -dict { PACKAGE_PIN V4
    ↪ IOSTANDARD LVCMOS33 } [get_ports
    ↪ {activ_displays[2]}]
set_property -dict { PACKAGE_PIN W4
    ↪ IOSTANDARD LVCMOS33 } [get_ports
    ↪ {activ_displays[3]}]

## Configuration options, can be used for all
→ designs
set_property CONFIG_VOLTAGE 3.3
→ [current_design]
set_property CFGBVS VCCO [current_design]

## SPI configuration mode options for QSPI
→ boot, can be used for all designs
set_property BITSTREAM.GENERAL.COMPRESS TRUE
→ [current_design]

```

```
set_property BITSTREAM.CONFIG.CONFIGRATE 33  
    [current_design]  
set_property CONFIG_MODE SPIx4  
    [current_design]
```

IV-C. Funcionamiento del reloj en la Basys 3

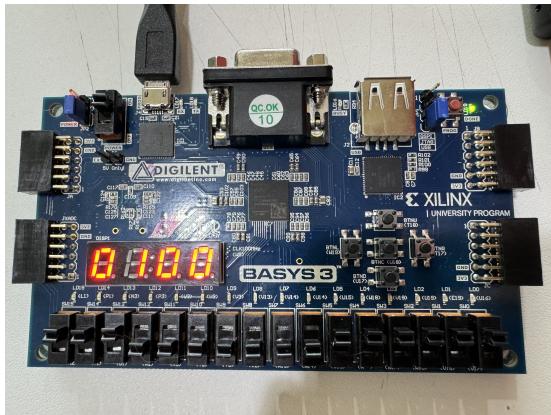


Figura 2: Reloj en 1 minuto

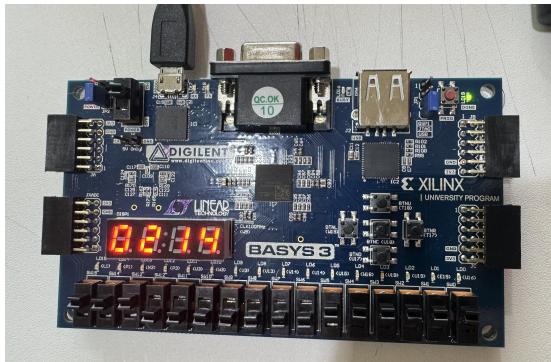


Figura 3: Reloj en 2 minutos, 14 segundos

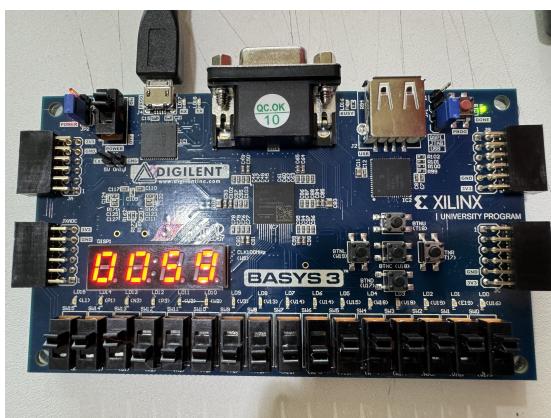


Figura 4: Reloj en 59 segundos