Universidad de Córdoba

Máster en Inteligencia Computacional e Internet de las Cosas

Memoria de Práctica Final

Programación de un despertador en BASYS 3 - Artix-7 XC7A35T

Autor: Alberto Fernández Merchán
Asignatura: Computación de Altas Prestaciones

Profesor: Joaquin Olivares Bueno

Índice

1. Introducción **2**

2. Proyecto **2**
2.1. Declaración de los dígitos en el display 2
2.2. Reloj Digital 4
2.3. Puesta en hora 5
2.4. Cronómetro 6
2.5. Alarma 7

3. Problemas encontrados **9**

4. Conclusiones **9**

1. Introducción

El objetivo de este proyecto es diseñar e implementar un reloj digital con alarma y cronómetro utilizando la FPGA Basys3 y su display de 7 segmentos. El sistema debe permitir visualizar la hora actual, la puesta en hora manual, un cronómetro, y un sistema de alarma con visualización en LEDs. Además, la alarma debe tener un comportamiento progresivo, aumentando su intensidad conforme pasa el tiempo desde su activación.

Para la implementación se utilizará el lenguaje VHDL, desarrollando los distintos módulos que permitan controlar la multiplexación de los displays, gestionar entradas del usuario mediante interruptores y pulsadores, y generar señales para la visualización y control de alarmas.

2. Proyecto

2.1. Declaración de los dígitos en el display

En primer lugar, se desarrolló el módulo encargado de traducir los valores numéricos binarios codificados en BCD (Binary-Coded Decimal) a sus correspondientes patrones de activación para el display de 7 segmentos. La FPGA Basys3 dispone de cuatro displays de este tipo, los cuales permiten mostrar números y ciertos caracteres mediante la activación selectiva de los segmentos LED que forman cada dígito.

Cada segmento del display se representa mediante un bit en un vector de 7 posiciones. Por convención, un segmento encendido se representa con un `0`, mientras que un segmento apagado se representa con un `1`, ya que el display es de cátodo común. Por tanto, mostrar un número implica asignar la combinación adecuada de bits a la señal de salida que controla los segmentos (`SSEG_CA`).

La conversión de dígitos del 0 al 9 se realiza mediante una sentencia `case` dentro de un proceso sensible a la señal `digits`, que contiene el valor BCD a representar:

```
-- BCD a 7 segmentos
process(digits)
begin
    case digits is
        when "0000" => letra <= "1000000"; -- 0
        when "0001" => letra <= "1111001"; -- 1
        when "0010" => letra <= "0100100"; -- 2
        when "0011" => letra <= "0110000"; -- 3
        when "0100" => letra <= "0011001"; -- 4
        when "0101" => letra <= "0010010"; -- 5
        when "0110" => letra <= "0000010"; -- 6
        when "0111" => letra <= "1111000"; -- 7
        when "1000" => letra <= "0000000"; -- 8
        when "1001" => letra <= "0010000"; -- 9
        when others => letra <= "1111111"; -- apagado
    end case;
end process;

SSEG_AN <= "0111" when selection = 0 else
           "1011" when selection = 1 else
           "1101" when selection = 2 else
           "1110";

SSEG_CA <= letra;
LED <= led_out;
```

Para la visualización simultánea de varios dígitos, se implementó un sistema de multiplexación, activando cíclicamente uno de los cuatro displays disponibles. Esto se realiza a través de la señal `SSEG_AN`, que controla qué dígito del display está activo en cada instante. Mediante la variable `selection`, que se incrementa a

alta frecuencia mediante un contador, se consigue mostrar los cuatro dígitos de forma alterna. Debido a la persistencia de la visión humana, el resultado percibido es una visualización estable y continua de todos los dígitos a la vez.

A continuación, se muestra el código que implementa tanto el multiplexado como la lógica para determinar qué valor debe mostrarse según el modo activo (reloj, cronómetro o edición de alarma), considerando además el parpadeo del dígito seleccionado en modo edición:

```
-- Multiplexado display
process(CLK)
begin
    if rising_edge(CLK) then
        if clk_count = CLK_LIMIT then
            clk_count <= 0;
            selection <= (selection + 1) mod 4;
        else
            clk_count <= clk_count + 1;
        end if;
    end if;
end process;

-- Lógica visualización
process(selection, minutes, hours, chrono_mode, chrono_minutes, chrono_seconds,
        edit_mode, alarm_edit_mode, edit_position, blink)
    variable temp_digit : integer := 0;
    variable visible : boolean := true;
begin
    visible := not ((edit_mode = '1' or alarm_edit_mode = '1') and selection = edit_position and blink = '0');
    if chrono_mode = '1' then
        case selection is
            when 0 => temp_digit := chrono_minutes / 10;
            when 1 => temp_digit := chrono_minutes mod 10;
            when 2 => temp_digit := chrono_seconds / 10;
            when 3 => temp_digit := chrono_seconds mod 10;
            when others => temp_digit := 0;
        end case;
    elsif alarm_edit_mode = '1' then
        case selection is
            when 0 => temp_digit := alarm_hours / 10;
            when 1 => temp_digit := alarm_hours mod 10;
            when 2 => temp_digit := alarm_minutes / 10;
            when 3 => temp_digit := alarm_minutes mod 10;
            when others => temp_digit := 0;
        end case;
    else
        case selection is
            when 0 => temp_digit := hours / 10;
            when 1 => temp_digit := hours mod 10;
            when 2 => temp_digit := minutes / 10;
            when 3 => temp_digit := minutes mod 10;
            when others => temp_digit := 0;
        end case;
    end if;
    if visible then
        digits <= std_logic_vector(to_unsigned(temp_digit, 4));
    else
        digits <= "1111";
    end if;
end process;
```

2.2. Reloj Digital

El modo por defecto del sistema es el **modo reloj**, el cual muestra la hora actual utilizando el display de 7 segmentos. En este modo:

- El reloj se actualiza cada segundo.
- Se visualiza la hora y los minutos en formato `HHMM`.
- El sistema utiliza una señal de reloj de 100 MHz para contar los segundos.

El conteo de segundos se realiza con un contador que compara contra la constante `SEC_LIMIT`, que corresponde a 100 millones de ciclos:

```
constant SEC_LIMIT : integer := 100_000_000;
signal sec_count : integer := 0;
```

Cada vez que `sec_count` alcanza ese límite, se incrementan los segundos. Si los segundos llegan a 60, se reinician a 0 e incrementan los minutos, y así sucesivamente con las horas:

```
if sec_count = SEC_LIMIT - 1 then
    sec_count <= 0;
    seconds <= seconds + 1;

    if seconds = 59 then
        seconds <= 0;
        minutes <= minutes + 1;

        if minutes = 59 then
            minutes <= 0;
            hours <= hours + 1;

            if hours = 23 then
                hours <= 0;
            end if;
        end if;
    end if;
else
    sec_count <= sec_count + 1;
end if;
```

2.3. Puesta en hora

La puesta en hora del sistema se realiza mediante una combinación de pulsadores y switches de la placa Basys3. Para entrar en el modo de edición de la hora, el usuario debe pulsar el botón central `BTN(4)` cuando los switches `SW(0)` y `SW(1)` están desactivados. Esto activa el modo de edición general, indicado visualmente mediante el encendido del LED `LED(1)`.

El código para entrar en modo edición de hora es el siguiente:

```
1       -- BTN(4) acciones
2             if btn_edge(4) = '1' then
3
4                 if SW(0) = '1' and SW(1) = '0' then
5                     chrono_mode <= not chrono_mode;
6                     if chrono_mode = '1' then
7                         chrono_minutes <= 0;
8                         chrono_seconds <= 0;
9                         chrono_sec_count <= 0;
10                        saved_minutes <= minutes;
11                        saved_seconds <= seconds;
12                    end if;
13
14                elsif SW(0) = '1' and SW(1) = '1' then
15                    alarm_active <= '0';
16                    alarm_activated <= '0';
17                    alarm_led_counter <= 0;
18
19                else -- si ninguno de los SW está activo entonce entra en modo edición
20                    edit_mode <= not edit_mode;
21
22                end if;
23            end if;
```

Una vez dentro del modo de edición, el usuario puede seleccionar uno de los cuatro dígitos visibles en el display de 7 segmentos utilizando los botones izquierdo y derecho (`BTN(1)` y `BTN(2)`), que desplazan circularmente un selector sobre las posiciones de los dígitos (decenas de hora, unidades de hora, decenas de minuto, unidades de minuto).

El valor del dígito seleccionado puede incrementarse o decrementarse usando los botones superior e inferior (`BTN(0)` y `BTN(3)`). El incremento o decremento se realiza en pasos de 10 o 1, dependiendo del dígito seleccionado. Por ejemplo, si se selecciona la posición de las decenas de hora, el valor aumenta o disminuye en bloques de 10 dentro del rango permitido (0–23 horas, 0–59 minutos).

El código que realiza esta funcionalidad es el siguiente:

```
                -- Edición
            if edit_mode = '1' or alarm_edit_mode = '1' then
                led_out(1) <= '1';

                if btn_edge(1) = '1' then
                    edit_position <= (edit_position - 1 + 4) mod 4;
                elsif btn_edge(2) = '1' then
                    edit_position <= (edit_position + 1) mod 4;
                end if;

                if btn_edge(0) = '1' then
                    case edit_position is
                        when 0 => if alarm_edit_mode = '1' then alarm_hours <= (alarm_hours + 10)
                        ↪  mod 24; else hours <= (hours + 10) mod 24; end if;
                        when 1 => if alarm_edit_mode = '1' then alarm_hours <= (alarm_hours + 1)
                        ↪  mod 24; else hours <= (hours + 1) mod 24; end if;
                        when 2 => if alarm_edit_mode = '1' then alarm_minutes <= (alarm_minutes +
                        ↪  10) mod 60; else minutes <= (minutes + 10) mod 60; end if;
                        when 3 => if alarm_edit_mode = '1' then alarm_minutes <= (alarm_minutes +
                        ↪  1) mod 60; else minutes <= (minutes + 1) mod 60; end if;
                        when others => null;
                    end case;
                elsif btn_edge(3) = '1' then
                    case edit_position is
                        when 0 => if alarm_edit_mode = '1' then alarm_hours <= (alarm_hours - 10 +
                        ↪  24) mod 24; else hours <= (hours - 10 + 24) mod 24; end if;
                        when 1 => if alarm_edit_mode = '1' then alarm_hours <= (alarm_hours - 1 +
                        ↪  24) mod 24; else hours <= (hours - 1 + 24) mod 24; end if;
                        when 2 => if alarm_edit_mode = '1' then alarm_minutes <= (alarm_minutes -
                        ↪  10 + 60) mod 60; else minutes <= (minutes - 10 + 60) mod 60; end if;
                        when 3 => if alarm_edit_mode = '1' then alarm_minutes <= (alarm_minutes - 1
                        ↪  + 60) mod 60; else minutes <= (minutes - 1 + 60) mod 60; end if;
                        when others => null;
                    end case;
                end if;
```

El dígito actualmente seleccionado parpadea mediante un sistema de temporización que alterna su visibilidad cada 0,5 segundos, ayudando al usuario a identificar qué posición está modificando (Este código se puede ver en la sección 2.1).

El modo de edición se desactiva pulsando nuevamente el botón central `BTN(4)`, lo que devuelve el sistema al modo reloj. Durante este modo, el sistema mantiene el tiempo actualizado y continúa visualizándose en el display.

2.4. Cronómetro

El cronómetro se implementa mediante un contador de segundos independiente, que puede iniciarse, pausarse y reiniciarse utilizando los pulsadores de la placa. La visualización se realiza en el display de 7 segmentos, mostrando minutos y segundos en formato MM:SS.

Dado que el cronómetro comparte recursos con el reloj principal (como el display y el reloj del sistema), se implementa una máquina de estados para gestionar la conmutación entre modos de funcionamiento: reloj, edición y cronómetro. Esta máquina garantiza que cada modo tenga control exclusivo sobre los recursos mientras esté activo.

El siguiente fragmento de código muestra cómo se realiza el conteo de tiempo en modo cronómetro. Cada segundo, si el cronómetro está activo, se incrementa el valor de los segundos. Si se alcanza 59, se reinician

los segundos y se incrementan los minutos, asegurando un conteo correcto hasta 59:59.

```
-- Contador de segundos para el cronómetro
if chrono_mode = '1' then
    if chrono_seconds = 59 then
        chrono_seconds <= 0;
        chrono_minutes <= (chrono_minutes + 1) mod 60;
    else
        chrono_seconds <= chrono_seconds + 1;
    end if;
end if;
```

Por otro lado, el siguiente bloque de código ilustra cómo se activa o reinicia el cronómetro mediante un pulsador. Al presionar BTN(4) mientras el interruptor SW(0) está activado (y SW(1) desactivado), se alterna el modo cronómetro. Al activarse, se inicializan los contadores de tiempo y se guarda el valor actual del reloj principal para evitar interferencias en la visualización.

```
-- Activación del cronómetro (ejecutado al pulsar BTN(4) con SW(0) activado)
if btn_edge(4) = '1' and SW(0) = '1' and SW(1) = '0' then
    chrono_mode <= not chrono_mode;
    if chrono_mode = '1' then
        chrono_minutes <= 0;
        chrono_seconds <= 0;
        chrono_sec_count <= 0;
        saved_minutes <= minutes;
        saved_seconds <= seconds;
    end if;
end if;
```

2.5. Alarma

El sistema de alarma permite ajustar una hora concreta mediante un mecanismo similar al de la puesta en hora. Esta edición se habilita activando simultáneamente los interruptores `SW(0)` y `SW(1)`. En ese modo, el usuario puede modificar los valores de los minutos y horas de la alarma utilizando los botones de la placa, con un selector que parpadea sobre el dígito en edición.

Una vez configurada y desactivados los switches, la alarma queda programada. Cuando la hora actual del reloj coincide con la hora programada, se activa una señal visual a través de los LEDs de la placa.

Además, se ha implementado una **alarma progresiva**: a partir del momento en que se activa la alarma, los LEDs se encienden progresivamente cada cinco segundos. Esto simula un aumento de intensidad similar al volumen creciente en una alarma convencional.

El siguiente fragmento muestra cómo se detecta la coincidencia entre la hora actual y la programada, lo que activa la señal de alarma:

```
-- Comparación entre hora actual y hora de la alarma
if (hours = alarm_hours and minutes = alarm_minutes and seconds = 0) then
    alarm_active <= '1';
    alarm_timer  <= 0;
else
    if alarm_active = '1' then
        alarm_timer <= alarm_timer + 1;
    end if;
end if;
```

Una vez activada, el siguiente bloque controla la activación progresiva de los LEDs desde el `LED(15)` hacia abajo, encendiendo uno nuevo cada 5 segundos desde que se activó la alarma:

```
-- Alarma progresiva: encendido de LEDs de 15 a 10 cada 5 segundos
led_output <= (others => '0');

if alarm_active = '1' then
    if alarm_timer >=  0*5 and alarm_timer <  1*5 then
        led_output(15)              <= '1';
    elsif alarm_timer >= 1*5 and alarm_timer < 2*5 then
        led_output(15 downto 14)    <= "11";
    elsif alarm_timer >= 2*5 and alarm_timer < 3*5 then
        led_output(15 downto 13)    <= "111";
    elsif alarm_timer >= 3*5 and alarm_timer < 4*5 then
        led_output(15 downto 12)    <= "1111";
    elsif alarm_timer >= 4*5 and alarm_timer < 5*5 then
        led_output(15 downto 11)    <= "11111";
    elsif alarm_timer >= 5*5 then
        led_output(15 downto 10)    <= "111111";
    end if;
end if;
```

Este comportamiento se detiene cuando el usuario desactiva manualmente la alarma. Para ello, debe activar el modo edición de alarma, es decir, posicionar los interruptores `SW(0)` y `SW(1)` en alto, y pulsar el botón `BTN(4)`. Esta acción fuerza el apagado de la alarma, reinicia el contador de LEDs y evita que la alarma vuelva a activarse automáticamente hasta la próxima coincidencia horaria.

El siguiente fragmento de código muestra cómo se gestionan las distintas acciones asociadas al botón `BTN(4)`, según el estado de los interruptores:

```
-- BTN(4) acciones
if btn_edge(4) = '1' then
    if SW(0) = '1' and SW(1) = '0' then
        -- Activar/Desactivar cronómetro
        chrono_mode <= not chrono_mode;
        if chrono_mode = '1' then
            chrono_minutes     <= 0;
            chrono_seconds     <= 0;
            chrono_sec_count   <= 0;
            saved_minutes      <= minutes;
            saved_seconds      <= seconds;
        end if;

    elsif SW(0) = '1' and SW(1) = '1' then
        -- Desactivar alarma manualmente
        alarm_active       <= '0';
        alarm_activated    <= '0';
        alarm_led_counter  <= 0;

    else
        -- Entrar/salir de modo edición de hora
        edit_mode <= not edit_mode;
    end if;
end if;
```

3. Problemas encontrados

Uno de los principales problemas fue la gestión de múltiples modos de funcionamiento utilizando los botones e interruptores disponibles en la placa. Debido al número limitado de entradas, fue necesario diseñar una lógica que combinara estados y entradas de forma eficiente para permitir el cambio entre modos sin ambigüedad. Esto implicó varios ajustes en la máquina de estados y una planificación cuidadosa del comportamiento de cada botón.

Otro problema recurrente fue el rebote mecánico de los botones, un fenómeno por el cual una única pulsación puede generar múltiples transiciones de señal en cortos intervalos de tiempo. Este comportamiento causaba la ejecución no deseada de acciones múltiples con un solo pulso. Para mitigar este efecto, se implementó una lógica de detección de flanco acompañado de un sistema de anti-rebote basado en un contador, de forma que sólo se reconociera un nuevo pulso cuando el botón se mantuviera estable durante varios ciclos de reloj consecutivos.

4. Conclusiones

Este proyecto ha permitido aplicar de forma práctica los conocimientos adquiridos en diseño digital y programación en VHDL mediante la implementación de un despertador funcional sobre la placa Basys3. Se han integrado distintos módulos que gestionan el reloj, la edición de hora, el cronómetro y una alarma con activación progresiva, todo ello aprovechando los recursos disponibles en la FPGA como displays de 7 segmentos, LEDs, botones y switches.

El desarrollo del sistema ha requerido coordinar los distintos módulos para gestionar los modos de funcionamiento. Pese a las limitaciones, se ha logrado una interfaz intuitiva y un comportamiento fiable.