Facultad de Ingeniería
UNAM

DISEÑO DIGITAL VLSI

Grupo: 2

PRACTICA 1

DOCUMENTACION

Profesor teoría

M.C. GERARDO ARIEL CASTILLO GARCIA

EQUIPO: 3

EJERCICIO 1.6 RELOJ 23 HORAS 59 MINUTOS

Fecha: 20/Agosto/2024

EJERCICIO

Realizar un reloj 23 (hrs) :59 (min) utilizando procesos concurrentes.
Para abordar este ejercicio se comenzó definiendo nuestro primer código main, definiendo nuestras entradas y salidas.

- clk: Señal de reloj.
- ssg0, ssg1, ssg2, ssg3: Salidas para cada display de 7 segmentos.

Además de definir las señales internas dentro de nuestra arquitectura del main.

- clkl: Señal de reloj para el contador, dividida para reducir la frecuencia.
- count1, count2, count3, count4: Contadores para las unidades y decenas de minutos y horas.
- flag1, flag2, flag3, flag4: Banderas que indican cuándo un contador ha alcanzado su valor máximo.
- rst: Señal de reinicio.

```
library ieee;
use ieee.std_logic_1164.all;

entity main is
  port (
    -- Reloj de entrada
    clk   : in std_logic;
    -- Salidas a cada display de 7 segmentos (uno para cada dígito)
    ssg0  : out std_logic_vector(6 downto 0);
    ssg1  : out std_logic_vector(6 downto 0);
    ssg2  : out std_logic_vector(6 downto 0);
    ssg3  : out std_logic_vector(6 downto 0)
  );
end entity;

architecture arqmain of main is
    -- Señal interna para el reloj dividido (señal lenta para los contadores)
  signal clkl                          : std_logic := '0';

-- Señales internas para almacenar los valores de cada dígito del reloj (4 dígitos en total)
  signal count1, count2, count3, count4 : integer   := 0;

  -- Señales de bandera que indican cuándo cada contador ha alcanzado su valor máximo
  signal flag1, flag2, flag3, flag4     : std_logic := '0';

  -- Señal de reset para reiniciar los contadores cuando se alcanza la hora 23:59
  signal rst                           : std_logic := '0';
```

Para continuar con el código definimos dentro de nuestro bloque begin

- Se instancia el módulo divfreq, que se encarga de dividir la frecuencia del reloj original (clk) para generar una señal de reloj más lenta (clkl).
- La señal de entrada clk es la señal de reloj del sistema, y la señal clkl es la salida del divisor de frecuencia, que se usará para los contadores.
- Definimos los contadores que usaremos para representar los valores solicitados, cada uno con su limite representando los segundo, minutos y horas por unidades, las representamos como c1, c2, c3, c4.

Para concluir con este código definimos los módulos de salida con su asignación

- d1 se conecta al valor de las unidades de segundos (count1) y controla el display ssg3.
- d2 se conecta al valor de las decenas de segundos (count2) y controla el display ssg2.
- d3 se conecta al valor de las unidades de horas (count3) y controla el display ssg1.
- d4 se conecta al valor de las decenas de horas (count4) y controla el display ssg0.

```
begin
  -- Entidad del divisor de frecuencias
  u1 : entity work.divfreq(rtl) port map (clk, clkl);

  -- Etidades para cada uno de los contadores
  c1 : entity work.cont(arqcont) -- Unidades segundos
    generic map(max_count => 9)
    port map(clkl, count1, rst, flag1);

  c2 : entity work.cont(arqcont) -- Decenas segundos
    generic map(max_count => 5)
    port map(flag1, count2, rst, flag2);

  c3 : entity work.cont(arqcont) -- Unidades hora
    generic map(max_count => 9)
    port map(flag2, count3, rst, flag3);

  c4 : entity work.cont(arqcont) -- Decenas hora
    generic map(max_count => 2)
    port map(flag3, count4, rst, flag4);

  -- Salidas para cada display
  d1 : entity work.ss7(arqss7) port map(count1, ssg3);
  d2 : entity work.ss7(arqss7) port map(count2, ssg2);
  d3 : entity work.ss7(arqss7) port map(count3, ssg1);
  d4 : entity work.ss7(arqss7) port map(count4, ssg0);
```

Para concluir nuestro código main, definimos las condiciones para activar nuestro reset con un proceso que verifica si el contador de horas ha llegado a 23:59 (count4 = 2 y count3 = 4). Si es así, activa la señal de reinicio (rst).

```
  -- Proceso para verificar cuando llegue a 23:59, se considera el contador
 -- de las decenas llega a 2 y el de las unidades a 4, en ese momento se reinician todos los contadores
  process (clk, rst)
  begin
    if rising_edge(clk) then
      if count4 = 2 and count3 = 4 then
        rst <= '1';
      else
        rst <= '0';
      end if;
    end if;
  end process;

end architecture;
```

En el segundo código implementamos el divisor de frecuencia

Este módulo divide la frecuencia del reloj.

Entradas y salidas:

- clk: Señal de reloj de entrada.
- clkl: Señal de reloj dividida de salida.

Arquitectura

Contador es una señal interna del tipo integer, que se utiliza para contar los ciclos del reloj clk. El rango del contador es de 0 a 25,000,000, lo que indica que el módulo está diseñado para dividir la frecuencia del reloj de entrada de manera significativa.

Dentro del begin marcamos el proceso principal, este proceso es sensible a la señal de reloj clk. Cada vez que hay un flanco ascendente (rising_edge) en clk, se ejecuta el código dentro del proceso.

Utilizando la condicional de if Si el contador ha alcanzado el valor de 750,000, se reinicia (contador <= 0;), y se invierte (not) el estado de clk1. Esta inversión de clk1 genera una señal de reloj con la mitad de la frecuencia original.

```
library ieee;
use ieee.std_logic_1164.all;

entity divfreq is
  port
  (
    clk  : in std_logic;
    clk1 : buffer std_logic
  );
end entity;

architecture rtl of divfreq is
  signal contador : integer range 0 to 25000000;

begin
  process (clk)
  begin
    if (rising_edge(clk)) then
      -- if (contador = 25000000) then
        if (contador = 750000) then
        contador <= 0;
        clk1     <= not clk1;
      else
        contador <= contador + 1;
      end if;
    end if;
  end process;
end architecture;
```

Pasamos al siguiente código implementado llamado ss7 el cual es un convertidor para decodificar las señales en entendibles para el display de 7 segmentos

Entradas y salidas:

- value: Valor del contador (0-9).
- display_out: Salida que controla los segmentos del display.

Funcionamiento:

- Utiliza una sentencia with...select para asignar la configuración correcta de segmentos en función del valor de entrada.

```
library ieee;
use ieee.std_logic_1164.all;

entity ss7 is
  port
  (
    value : in integer;
    display_out : out std_logic_vector(6 downto 0)
  );
end entity;

architecture arqss7 of ss7 is

begin

  with value select
    display_out <= "1000000" when 0,
    "1111001" when 1,
    "0100100" when 2,
    "0110000" when 3,
    "0011001" when 4,
    "0010010" when 5,
    "0000010" when 6,
    "1111000" when 7,
    "0000000" when 8,
    "0011000" when 9,
    "1111111" when others;
end architecture;
```

Como ultimo se reutilizo parte del contador implementado en los ejercicios pasados, sin embargo, de igual manera se explicara.

- entity cont is: Define la entidad cont, que es un contador genérico.
- max_count : integer := 9;: Parámetro genérico que define el valor máximo del contador.
- clk : in std_logic;: Señal de entrada de reloj.
- counter : buffer integer;: Contador que se incrementa en cada ciclo de reloj.
- reset : in std_logic;: Señal de reinicio que resetea el contador.
- carry : out std_logic := '0';: Señal de salida que indica cuando el contador ha alcanzado su valor máximo.

Arquitectura

- architecture arqcont of cont is: Inicia la arquitectura arqcont de la entidad cont.
- signal flag : std_logic := '1';: Señal interna que se utiliza para indicar cuando el contador ha alcanzado su valor máximo.

Proceso de conteo.

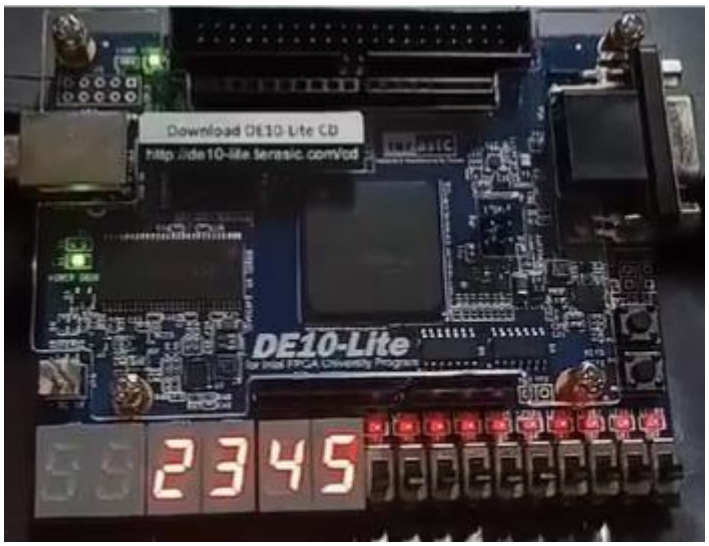
- process (clk, reset): Define un proceso sensible al flanco ascendente de clk y a la señal de reinicio reset.
- if reset = '1' then: Si la señal de reinicio está activa, el contador se reinicia a 0.
- elsif rising_edge(clk) then: Si no hay reinicio, se verifica el flanco ascendente de clk.
- if (counter = max_count) then: Si el contador ha alcanzado su valor máximo (max_count), se reinicia.
- counter <= 0; flag <= '1';: Reinicia el contador y activa la bandera flag.
- else counter <= counter + 1; flag <= '0';: Si no ha alcanzado el valor máximo, incrementa el contador y desactiva la bandera.
- carry <= flag;: La señal carry se activa cuando el contador alcanza su valor máximo (flag = '1').

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity cont is
  generic (
    max_count : integer := 9
  );
  port (
    clk     : in std_logic;
    counter : buffer integer;
    -- Se agrega una entrada para reiniciar el contador
    reset   : in std_logic;
    carry   : out std_logic := '0'
  );
end entity;

architecture arqcont of cont is
  signal flag : std_logic := '1';
begin
  -- Se toma en cuenta esta entrada en la lista se sensibilidad del process
  process (clk, reset)
  begin
    -- Se comprueba primero si el reset está activo, de ser el caso,
    -- el contador se reinicia
    if reset = '1' then
      counter <= 0;
    elsif rising_edge(clk) then
      if (counter = max_count) then
        counter <= 0;
        flag    <= '1';
      else
        counter <= counter + 1;
        flag    <= '0';
      end if;
    end if;
  end process;
  carry <= flag;
end architecture;
```

RESULTADO



REFERENCIAS

- *Ingeniería, D. (2023, 1 febrero). Contador Ascendente con VHDL. Domando Ingeniería. https://domandoingenieria.com/index.php/2022/03/01/contador-ascendiente-con-vhdl/*
- *Admin, C. (2020, 5 enero). Contador con Display de 7 Segmentos (0 – 99). https://www.electroallweb.com/index.php/2020/01/05/contador-con-display-de-7-segmentos-0-99/#google_vignette*
- *ManualsLib. (2018, 27 agosto). TERASIC DE10-LITE USER MANUAL Pdf download. https://www.manualslib.com/manual/1429115/Terasic-De10-Lite.html*