UNIVERSITAT ROVIRA I VIRGILI

ESCOLA TÈCNICA SUPERIOR ENGINYERIA

GRAU D'ENGINYERIA ELECTRÒNICA INDUSTRIAL i AUTOMÀTICA

LABORATORIO DE ELECTRONICA DIGITAL

Curs 2021-2022

Alumnos: Chiara Coppola y Luisa Vicente Oliveira

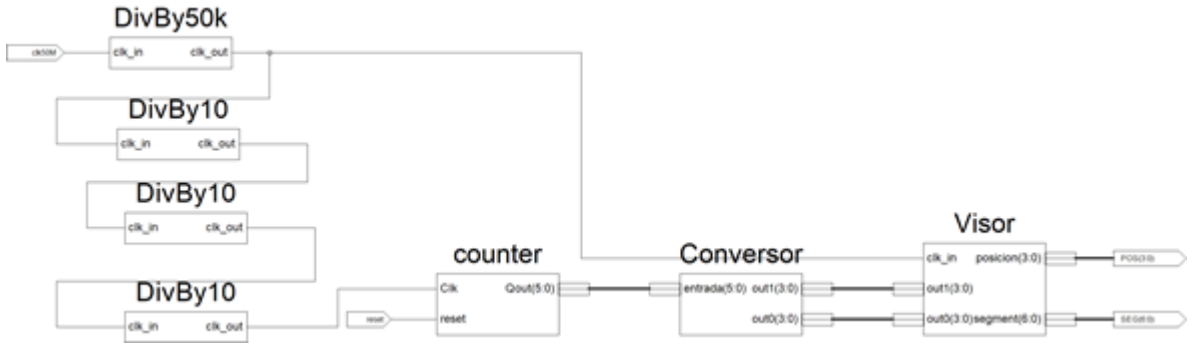
Grup: L5

Profesor: Enrique Fernando Cantó Navarro

Esteban del Castillo Pérez

MEMORIA DEL BLOQUE 1 DE LA PRÁCTICA

En esta primera parte de la práctica hemos creado un cronómetro que cuenta de 0 a 59 segundos.



Uno de los componentes necesarios para hacer el cronómetro es el divisor de frecuencia. Lo necesitamos porque el reloj que dispone la placa es de 50MHz y necesitamos dividir esa frecuencia para obtener diferentes *cloks* para otras partes del futuro reloj.
Por ejemplo, dividimos esa frecuencia inicial por 50K para tener una de 1KHz que la utilizamos como entrada en el visor para visualizar los dígitos.

```
begin
   process (clk_in)
   begin
      if (clk_in'event and clk_in = '1') then
         tmp <= nxt_tmp;
         cnt <= nxt_cnt;
      end if;
   end process;
   cmp <= '1' when cnt >= 24999 else '0';
   nxt_cnt <= 0 when cmp='1' else cnt+1;
   nxt_tmp <= not(tmp) when cmp='1' else tmp;
   clk_out <= tmp;
end Behavioral;
```

Entonces, lo que se hace para disminuir esa frecuencia es dividir ese periodo de modo que el flanco de reloj cambia a cada 25000 ciclos. Para eso utilizamos una señal ‘cnt’ que nos indica cuando ocurre la transición de nivel bajo a alto.

Aprovechando la frecuencia 1Khz, hacemos que esta sea la entrada de nuestro próximo divisor para que al final consigamos una frecuencia de 1Hz que sirve como reloj de entrada del contador ya que necesitamos contar a cada 1 segundo los números. En VHDL, se utiliza la misma lógica que la dicha anteriormente, pero en vez de contar de 0 a 24999, contamos de 0 a 4.

Para hacer el contador hemos creado un código que cuenta de 0 a 59 y hemos añadido un multiplexor para que compare las señales de modo que cuando llegue a 59 el valor sea 0 y pueda volver a contar otra vez.

```
signal actual, seguent:   unsigned (5 downto 0);
begin
   Qout <= std_logic_vector (actual);
   seguent <= "000000" when actual= "111011" else actual+1;
    process(Clk, reset)
    begin
       if (reset='1') then
            actual <= "000000";
       elsif (Clk'event and Clk='1') then
         actual <= seguent;
       end if;
    end process;

end Behavioral;
```

Como recibimos del contador una salida de 6 bits, necesitamos transformarlo mediante un conversor para tener los números en binario de 8 bits. Para eso, hemos creado dos vectores que separan la salida en 4 bits cada - las unidades y las decenas, que serán entradas en el visor.

```
architecture Behavioral of Conversor is
signal output:  STD_LOGIC_VECTOR (7 downto 0);
begin

out0<=output(3 downto 0);
out1<=output(7 downto 4);

output<= B"0000_0000" when entrada= "000000" else
         B"0000_0001" when entrada= "000001" else
         B"0000_0010" when entrada= "000010" else
         B"0000_0011" when entrada= "000011" else
```

En la parte del visor hemos tenido que crear un FSM porque necesitamos trabajar separadamente con las unidades y las decenas.

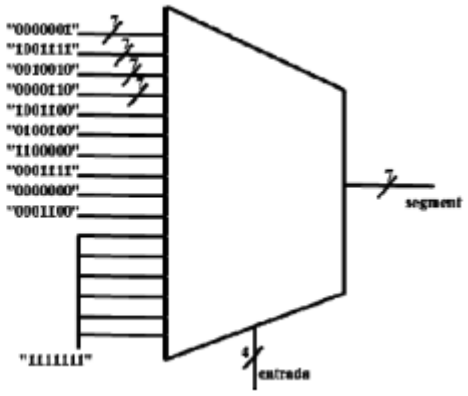
```
process(clk_in) is
begin
    if (clk_in'event and clk_in = '1') then
       if cnt = '0' then
            cnt <= '1';
        else
            cnt <= '0';
        end if;
    end if;
end process;
```

Con un multiplexor, entonces, decidimos cuándo trabaja con las unidades o con las decenas. Como se puede observar, la salida de este multiplexor es la entrada del decodificador.

```
entrada <= out0 when cnt = '0' else out1;
```

Para el decodificador de 7 segmentos visualizamos los dígitos decimales de 0 a 9 en el display de la placa. Para eso escribimos un código en VHDL como el de abajo.

```
segment <= "0000001" when entrada = "0000"
      else "1001111" when entrada = "0001"
      else "0010010" when entrada = "0010"
      else "0000110" when entrada = "0011"
      else "1001100" when entrada = "0100"
      else "0100100" when entrada = "0101"
      else "1100000" when entrada = "0110"
      else "0001111" when entrada = "0111"
      else "0000000" when entrada = "1000"
      else "0001100" when entrada = "1001"
      else "1111111";
```



Finalmente, basado en la construcción del FSM, creamos otro multiplexor que hace que se encienda el display de más a la derecha cuando tratamos las unidades y el anterior a él para las decenas. Como la placa trabaja con lógica negativa, cuando tenemos un ’0’ el display se enciende y con ‘1’ se apaga.

```
posicion <= "1110" when cnt = '0' else "1101";
```

PLANILLA UCF

Paralelamente al código VHDL, tuvimos que hacer una planilla UCF para hacer la conexión entre el software y el hardware. Observando el dibujo esquemático se ve que tenemos dos entradas (reloj de la placa y el *reset*) y dos salidas (la posición del display y sus segmentos).

```
NET "reset" LOC = A7;
NET "clk50M" LOC = B8;

NET "POS[3]" LOC = K14;
NET "POS[2]" LOC = M13;
NET "POS[1]" LOC = J12;
NET "POS[0]" LOC = F12;

NET "SEG[0]" LOC = M12;
NET "SEG[1]" LOC = L13;
NET "SEG[2]" LOC = P12;
NET "SEG[3]" LOC = N11;
NET "SEG[4]" LOC = N14;
NET "SEG[5]" LOC = H12;
NET "SEG[6]" LOC = L14;
```

MODIFICACIÓN DE LA GUÍA DE PRÁCTICAS

En la parte que creemos haber tenido más dificultad es la relacionada a la parte del conversor-visor porque no sé entiende muy bien la razón por la que tienes que cambiar los números de 6 bits a 8 bits. Falta un poco de información de cómo sales del contador a crear las partes que componen el visor.