UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DISEÑO DIGITAL MODERNO

PROFESORA

M.I. Norma Elva Chávez Rodríguez

Equipo No.4

NOMBRES DE LOS ALUMNOS

Gómez Luna Alejandro

Guzmán Mercado Sergio Francisco

Pardo Reyna Anelissa Allizon

GRUPO 1

PROYECTO FINAL. RELOJ DIGITAL

CALIFICACIÓN

_____

OBSERVACIONES:

Cd Universitaria a 14 de julio del 2021

Índice

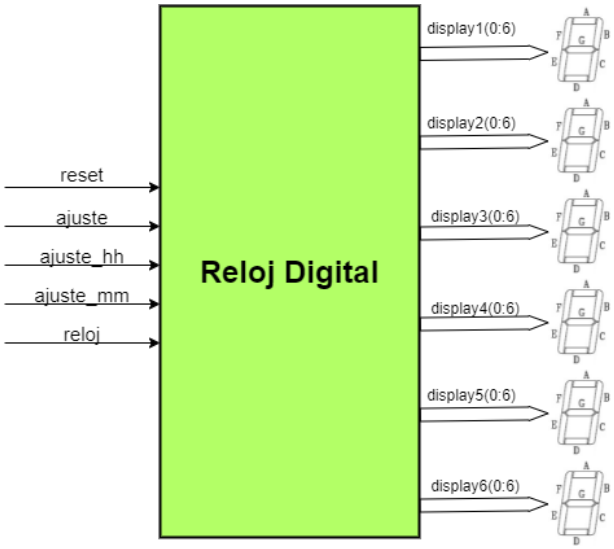
Especificaciones.

Diseñar un reloj digital, empleando 6 displays de 7 segmentos, de los cuales los displays menos significativos corresponden a los segundos, los displays centrales a los minutos y los más significativos a las horas.

- Los displays correspondientes a los segundos deben contar del 0 al 59.
- Los displays correspondientes a los minutos deben contar del 0 al 59.
- Los displays correspondientes a las horas deben contar del 0 al 23.

Además se deben implementar tanto un botón que incremente los minutos y otro que incremente las horas, así como una entrada que permita reiniciar el reloj digital.

Diagrama de bloques.



En el diagrama de bloques, se requieren como entradas un dip switch que reseteará todos los displays a 00:00:00, un segundo dip switch que coloque el reloj en modo de ajuste, este servirá para que el reloj pueda ser ajustado haciendo uso de dos botones (*ajuste_hh* y *ajuste_mm*) y la señal de reloj de la tarjeta FPGA DE10-lite de 50 MHz, el cual se debe ajustar con un divisor de frecuencia para que otorgue una señal a 1 Hz exacto, el cual es equivalente a 1 segundo. Las salidas corresponden a 6 displays de 7 segmentos, donde podrá ser visualizado el reloj de la siguiente manera:

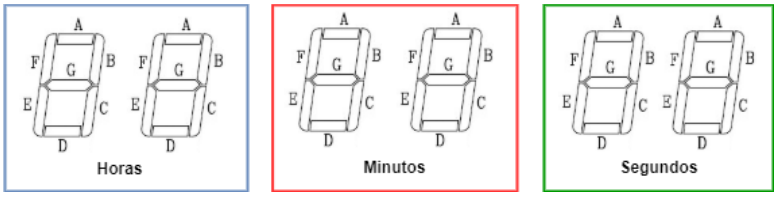
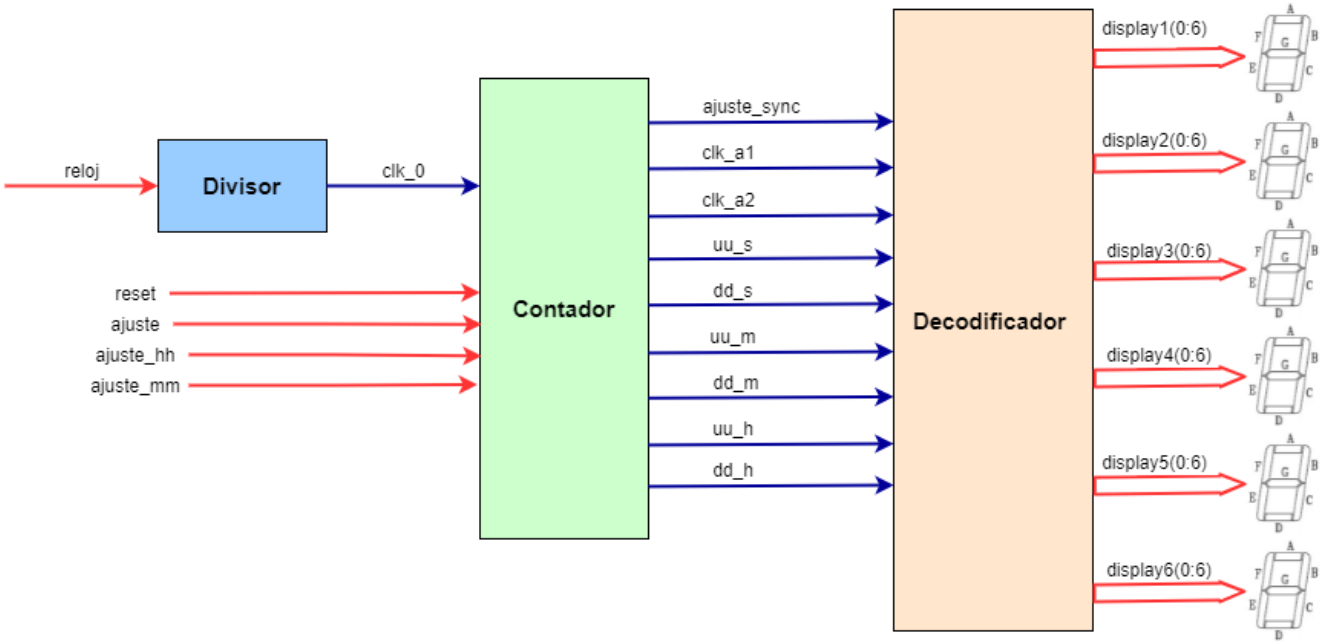


Diagrama de bloques funcionales.



Para el proyecto se requieren 5 entradas físicas, la entrada de reloj de 50MHz de la tarjeta FPGA DE10-lite, dos dip switch, el primero, al ser puesto en bajo resetea la salida del reloj a 00:00:00, mientras que el segundo, al ponerlo en alto el reloj entra en modo de ajuste de manera que los minutos y las horas puedan ser modificados haciendo uso de dos push buttons, ajuste_hh para las horas y ajuste_mm para los minutos.

La entrada de reloj de 50MHz ingresa al bloque funcional *Divisor*, dicho bloque se encarga de sacar una señal 1 Hz, equivalente a 1 segundo, esta señal será empleada para obtener 5 señales síncronas, con las cuales sera posible incrementar las unidades y decenas de los segundos, minutos y horas obteniendo así 6 variables a decodificar, además en este mismo bloque se obtienen como salida 3 señales, las cuales corresponden al modo de ajuste y a los respectivos push buttons, en otras palabras se define el suceso que ocurre si se oprime determinado botón o dip switch. Finalmente las señales y variables anteriores son decodificadas para su correcta visualización en los displays de 7 segmentos haciendo uso del bloque *Decodificador*.

Código documentado.

```
library IEEE;
Use IEEE.STD_LOGIC_1164.ALL;
Use IEEE.STD_LOGIC_ARITH.ALL;
Use IEEE.STD_LOGIC_UNSIGNED.ALL;

Entity reloj_digital is

    port(reloj:in std_logic;
         reset,ajuste:in std_logic;
         ajuste_hh,ajuste_mm:in std_logic;
         display1,display2,display3,display4,display5,display6: out std_logic_vector(0 to 6));

end reloj_digital;
```

Una vez declaradas las bibliotecas, definimos la entidad, la cual, siguiendo el diagrama de bloques funcionales se tienen como entradas *std_logic* el reloj de la tarjeta FPGA DE10-lite, los dip switch de reset y ajuste, y los push buttons que modificaran los minutos y las horas (ajuste_hh, ajuste_mm), teniendo así un total de 5 entradas.

Como salidas se tienen 6 displays de 7 segmentos, de esta forma cada display será un vector que va del 0 al 6.

```
architecture Behavioral of reloj_digital is

    --Señales de reloj para controlar cada bit referido al reloj
    signal clk0,clk1,clk2,clk3,clk4,clk5: std_logic := '0'; --señales del contador para cada display

    signal clk_a1,clk_a2:STD_LOGIC:='0'; --señales de ajuste minutos y horas de push button
    signal ajuste_sync:std_logic; -- señal de modo de ajuste en dipswitch

    constant max: integer := 50000000;
    constant med: integer := max/2;

    --Cuenta
    signal cuenta: integer range 0 to max; -- se asigna la señal de reloj

    -- Decodificador:

    function Deco(digito:integer) return std_logic_vector is

       variable display:std_logic_vector(0 to 6);

       begin
          case digito is
             when 0 => display := "0000001";--0
             when 1 => display := "1001111";--1
             when 2 => display := "0010010";--2
             when 3 => display := "0000110";--3
             when 4 => display := "1001100";--4
             when 5 => display := "0100100";--5
             when 6 => display := "1100000";--6
             when 7 => display := "0001111";--7
             when 8 => display := "0000000";--8
             when 9 => display := "0000100";--9
             when others => null;
          end case;
       return(display);
       end Deco;

Begin
```

A continuación declaramos las señales que se emplearan dentro de los bloques funcionales, se tienen 6 señales `clk0` a `clk5`, estas señales se comportaran de manera síncrona es decir serán las encargadas de llevar los flancos de reloj de modo que se puedan incrementar en determinado momentos unidades y decenas de segundos, minutos y horas.

Las señales `clk_a1` y `clk_a2` corresponden a los eventos asociados a los push buttons, de igual forma la señal `ajuste_sync`, corresponde al evento del dip switch para entrar en modo de ajuste.

```
signal clk_a1,clk_a2:STD_LOGIC:='0'; --señales de ajuste minutos y horas de push button
signal ajuste_sync:std_logic; -- señal de modo de ajuste en dipswitch
```

Las siguientes declaraciones serán útiles para realizar el divisor de frecuencia, es por ello que se requieren dos constantes, las cuales delimitaran la señal de 50 MHz y la mitad de esta, es decir 25 MHz, de igual forma una señal que se emplea únicamente internamente dentro del bloque funcional *Divisor* y no como salida para su manipulación por otros bloques funcionales.

```
constant max: integer := 50000000;
constant med: integer := max/2;

--Cuenta
signal cuenta: integer range 0 to max; -- se asigna la señal de reloj
```

Las siguientes sentencias corresponden a una función, la cual servirá como decodificador, ya que las operaciones que se realizan dentro de los bloques funcionales están operadas a números enteros, y la salida a una representación *std_logic_vector*. De esta manera, con la función es posible transformar números enteros y obtener una salida vectorial que correspondería a un display de 7 segmentos genérico con cualquier representación numérica del 0 al 9.

```
function Deco(digito:integer) return std_logic_vector is

   variable display:std_logic_vector(0 to 6);

   begin
      case digito is
         when 0 => display := "0000001";--0
         when 1 => display := "1001111";--1
         when 2 => display := "0010010";--2
         when 3 => display := "0000110";--3
         when 4 => display := "1001100";--4
         when 5 => display := "0100100";--5
         when 6 => display := "1100000";--6
         when 7 => display := "0001111";--7
         when 8 => display := "0000000";--8
         when 9 => display := "0000100";--9
         when others => null;
      end case;
   return(display);
   end Deco;
```

El siguiente bloque de código corresponde al bloque funcional *Divisor*

```
52  Begin
53
54  -- Se establece el reloj a 1Hz lo equivalente a 1 segundo exacto
55
56  Divisor: process (reloj)
57
58  begin
59     if(rising_edge(reloj) and reloj='1') then
60        if(cuenta < max) then
61           cuenta <= cuenta + 1; --incrementa la cuenta
62        else
63           cuenta <= 1; --reinicia la cuenta
64        end if;
65        if (cuenta <= med) then
66           clk0 <= '0';
67        else
68           clk0 <='1';
69        end if;
70     end if;
71  end process;
72
```

Como se puede observar, se toma como argumento la señal de reloj así como las dos constantes que delimitaran a la señal, su punto máximo y medio, la señal llamada *cuenta* únicamente servirá para establecer una señal de reloj a 1 Hz (evidentemente con flancos positivos y negativos) asignada a `clk0`, de esta manera se obtiene un divisor de frecuencia a 1 segundo exacto.

El bloque funcional de *Contador* se muestra a continuación:

```
73    -- Se definen los elementos del reloj
74
75    Contador: process (clk0,reset,ajuste,ajuste_hh,ajuste_mm)
76
77       variable uu_s: integer range 0 to 10;
78       variable dd_s: integer range 0 to 6;
79       variable uu_m: integer range 0 to 10;
80       variable dd_m: integer range 0 to 6;
81       variable uu_h: integer range 0 to 10;
82       variable dd_h: integer range 0 to 3;
83
84     begin
85
86     -- Se establece el reseteo del reloj
87
88    if (reset='0') then
89       uu_s:=0;
90       dd_s:=0;
91       uu_m:=0;
92       dd_m:=0;
93       uu_h:=0;
94       dd_h:=0;
95
96    else
97
98     -- Se establece el contador de segundos a 60 (unidades)
99
100      if (rising_edge(clk0)  and clk0='1') then
101
102         uu_s:= uu_s+1;
103         clk1 <= '0';
104         if (uu_s = 10) then
105            uu_s := 0;
106            clk1 <= '1';
107
108         end if;
109      end if;
110
111    -- (decenas)
112
113      if (rising_edge(clk1) and clk1='1') then
114
115         dd_s := dd_s + 1;
116         clk2 <= '0';
117         if (dd_s = 6) then
118            dd_s := 0;
119            clk2 <= '1';
120
121         end if;
122      end if;
```

Dentro del contador, primero se declaran las variables (ya que al ser síncronas, su valor permanecerá en espera hasta un evento y luego cambiara), como se puede ver cada variable se delimita a su definición, es decir las unidades de segundo solo pueden tomar valores del 0 al 9, mientras que las decenas del 0 al 5, pues conforme transcurran los flancos de reloj a 1 Hz lo máximo que pueden formar es un 59 y posteriormente regresar a 00, lo mismo ocurrirá con los minutos, sin embargo quedan acotados a flancos de reloj mas lentos y para las horas, las unidades de hora solo pueden tomar valores del 0 al 9, pero las decenas de hora solo pueden tomar valores del 0 al 2, al ser un formato de 24 horas, lo máximo que pueden formar

es un 23, y posteriormente cambiar a 00, evidentemente con flancos de reloj más lentos en comparación con los minutos.

```
variable uu_s: integer range 0 to 10;
variable dd_s: integer range 0 to 6;
variable uu_m: integer range 0 to 10;
variable dd_m: integer range 0 to 6;
variable uu_h: integer range 0 to 10;
variable dd_h: integer range 0 to 3;
```

La primer instrucción que realiza el bloque funcional es la de resetear el reloj a 00:00:00 en caso de que se oprima el botón asociado al *reset*.

```
-- Se establece el reseteo del reloj

if (reset='0') then
   uu_s:=0;
   dd_s:=0;
   uu_m:=0;
   dd_m:=0;
   uu_h:=0;
   dd_h:=0;
```

En caso de que no se tenga el evento de reseteo, se establece el contador para los segundos.

```
else

-- Se establece el contador de segundos a 60 (unidades)

   if (rising_edge(clk0)  and clk0='1') then

      uu_s:= uu_s+1;
      clk1 <= '0';
      if (uu_s = 10) then
         uu_s := 0;
         clk1 <= '1';

      end if;
   end if;

-- (decenas)

   if (rising_edge(clk1) and clk1='1') then

      dd_s := dd_s + 1;
      clk2 <= '0';
      if (dd_s = 6) then
         dd_s := 0;
         clk2 <= '1';

      end if;
   end if;
```

Para el contador de las unidades de segundos se emplea la señal dada por el divisor, es decir `clk0` la cual trabaja a 1 Hz. De este bloque de sentencias se destaca que se forma una nueva señal síncrona cuyos flancos positivos dependen de la variable *uu_s*, esta señal `clk1` servirá para el contador de la variable *dd_s* por su naturaleza de sincronía.

De la misma manera se forma otra señal síncrona, `clk2`, la cual depende de la variable *dd_s*.

```
124   -- Se establece la señal de ajuste de los segundos
125
126   -- Sincronizacion de segundos
127
128      if (falling_edge(clk0) and clk0='0') then
129         ajuste_sync <= ajuste;
130      end if;
131
132      clk_a1 <= (clk2 and not ajuste_sync) or (ajuste_mm and ajuste_sync);
133
```

Con esta sentencia se declara que si se tiene un flanco negativo de la señal `clk0`, entonces se operará en modo de ajuste, ahora bien, la sentencia de la línea 132 , al entrar en modo de ajuste, para los minutos se debe asociar su push botton, entonces existen dos casos, cuando `clk2` ya existe pero no se entro en modo ajuste entonces se hace una copia de la señal dependiente del contador de las decenas de segundo o en caso de que exista un evento de ajuste para los minutos presionando su botón asociado y además se esta operando en modo de ajuste, entonces la señal `clk_a1` operará como la responsable de ajustar directamente los minutos con cada evento del botón.

Ahora el contador de los minutos toma como entrada `clk_a1` independientemente de que esta señal opere en modo de ajuste o no lo haga (ya que ahora se tiene una señal síncrona dependiente del contador de las decenas de segundo, es decir `clk2`).

```
134   -- Se establece el contador de minutos a 60 (Unidades)
135
136      if (rising_edge(clk_a1) and clk_a1='1') then
137
138         uu_m:= uu_m+1;
139         clk3 <= '0';
140
141         if (uu_m=10) then
142            uu_m := 0;
143            clk3 <= '1';
144         end if;
145      end if;
146
147      --Decenas
148
149      if (rising_edge(clk3) and clk3='1') then
150
151         dd_m := dd_m + 1;
152         clk4 <= '0';
153
154         if (dd_m = 6) then
155            dd_m := 0;
156            clk4 <= '1';
157         end if;
158      end if;
```

De este bloque de sentencias se destaca que se forma una nueva señal síncrona cuyos flancos positivos dependen de la variable *uu_m*, esta señal `clk3` servirá para el contador de la variable *dd_s* por su naturaleza de sincronía.

De la misma manera se forma otra señal síncrona, `clk4`, la cual depende de la variable dd_m.

Es importante notar que si `clk_a1` entra en modo de ajuste, las siguientes líneas operaran de forma normal, ya que lo que se busca con el ajuste es que la cuenta se pueda manipular físicamente por un botón, de esta manera se incrementarían las unidades y decenas de minuto.

Mismo caso de ajuste para los minutos se tendrá para las horas con la siguiente sentencia:

```
160   -- Se establece la señal de ajuste a los minutos
161
162      clk_a2 <= (clk4 and not ajuste_sync) or (ajuste_hh and ajuste_sync);
163
```

Al entrar en modo de ajuste, para las horas se debe asociar su push botton, entonces existen –nuevamente- dos casos, cuando `clk4` ya existe pero no se entró en modo ajuste entonces se hace una copia de la señal dependiente del contador de las decenas de minuto o en caso de que exista un evento de ajuste para las horas presionando su botón asociado y además se está operando en modo de ajuste, entonces la señal `clk_a2` operará como la responsable de ajustar directamente las horas con cada evento del botón.

Ahora el contador de las horas toma como entrada `clk_a2` independientemente de que esta señal opere en modo de ajuste o no lo haga (ya que ahora se tiene una señal síncrona dependiente del contador de las decenas de minuto, es decir `clk4`).

```
164   -- Se establece la señal de ajuste a las horas (unidades)
165
166      if (rising_edge(clk_a2) and clk_a2='1') then
167         uu_h := uu_h + 1;
168         clk5 <= '0';
169
170         if (uu_h = 10) then
171            uu_h := 0;
172            clk5 <= '1';
173         end if;
174      end if;
175
176      --Decenas
177
178      if (rising_edge(clk5) and clk5 = '1') then
179         dd_h := dd_h+1;
180      end if;
181      if (dd_h = 2 and uu_h = 4) then
182         uu_h := 0;
183         dd_h := 0;
184      end if;
185   end if;
```

De este bloque de sentencias se destaca que se forma una nueva señal síncrona cuyos flancos positivos dependen de la variable *uu_h*, esta señal `clk5` servirá para el contador de la variable *dd_h* por su naturaleza de sincronía. En este punto ya no se crea una nueva señal síncrona dependiente, pues ya no existe otro contador, sin embargo se valida que si el contador de las decenas y las unidades toma los valores de 2 y 4 (formato de 24) entonces se coloque un 0 tanto para las unidades de hora como para las decenas de hora.

Es importante notar que si `clk_a2` entra en modo de ajuste, las siguientes líneas operaran de forma normal, ya que lo que se busca con el ajuste es que la cuenta se pueda manipular físicamente por un botón, de esta manera se incrementarían las unidades y decenas de hora.

Finalmente se hacen llamadas a la función decodificadora para que transforme los valores enteros (correspondientes a las unidades y decenas de segundo, minuto y hora) a vectores para que puedan ser visualizados en cada respectivo display de 7 segmentos.

```
189     display1 <= Deco(uu_s);
190     display2 <= Deco(dd_s);
191     display3 <= Deco(uu_m);
192     display4 <= Deco(dd_m);
193     display5 <= Deco(uu_h);
194     display6 <= Deco(dd_h);
195
196   end process;
197   end Behavioral;
198
```

Código fuente en VHDL.

```
library IEEE;
Use IEEE.STD_LOGIC_1164.ALL;
Use IEEE.STD_LOGIC_ARITH.ALL;
Use IEEE.STD_LOGIC_UNSIGNED.ALL;

Entity reloj_digital is

        port(reloj:in std_logic;
                  reset,ajuste:in std_logic;
                  ajuste_hh,ajuste_mm:in std_logic;
                  display1,display2,display3,display4,display5,display6: out std_logic_vector(0 to 6));

end reloj_digital;

architecture Behavioral of reloj_digital is

        --Señales de reloj para controlar cada variable referida al reloj
        signal clk0,clk1,clk2,clk3,clk4,clk5: std_logic := '0';

        signal clk_a1,clk_a2:STD_LOGIC:='0'; --señales de ajuste minutos y horas de push button
        signal ajuste_sync:std_logic; -- señal de modo de ajuste en dip switch

        constant max: integer := 50000000;
        constant med: integer := max/2;

        --Cuenta
        signal cuenta: integer range 0 to max; -- se asigna la señal de reloj

        -- Decodificador:

        function Deco(digito:integer) return std_logic_vector is

                variable display:std_logic_vector(0 to 6);

                begin
                        case digito is
                                when 0 => display := "0000001";--0
                                when 1 => display := "1001111";--1
                                when 2 => display := "0010010";--2
                                when 3 => display := "0000110";--3
                                when 4 => display := "1001100";--4
                                when 5 => display := "0100100";--5
                                when 6 => display := "1100000";--6
                                when 7 => display := "0001111";--7
                                when 8 => display := "0000000";--8
                                when 9 => display := "0000100";--9
                                when others => null;
                        end case;
                return(display);
                end Deco;

Begin
```

```
-- Se establece el reloj a 1Hz lo equivalente a 1 segundo exacto
Divisor: process (reloj)

begin
        if(rising_edge(reloj) and reloj='1') then
                if(cuenta < max) then
                        cuenta <= cuenta + 1; --incrementa la cuenta
                else
                        cuenta <= 1; --reinicia la cuenta
                end if;
                if (cuenta <= med) then
                        clk0 <= '0';
                else
                        clk0 <='1';
                end if;
        end if;
end process;

-- Se definen los elementos del reloj
Contador: process (clk0,reset,ajuste,ajuste_hh,ajuste_mm)

        variable uu_s: integer range 0 to 10;
        variable dd_s: integer range 0 to 6;
        variable uu_m: integer range 0 to 10;
        variable dd_m: integer range 0 to 6;
        variable uu_h: integer range 0 to 10;
        variable dd_h: integer range 0 to 3;

begin

-- Se establece el reseteo del reloj
if (reset='0') then
        uu_s:=0;
        dd_s:=0;
        uu_m:=0;
        dd_m:=0;
        uu_h:=0;
        dd_h:=0;
else

-- Se establece el contador de segundos a 60 (unidades)

        if (rising_edge(clk0)  and clk0='1') then

                uu_s:= uu_s+1;
                clk1 <= '0';
                if (uu_s = 10) then
                        uu_s := 0;
                        clk1 <= '1';

                end if;
        end if;

-- (decenas)

        if (rising_edge(clk1) and clk1='1') then

                dd_s := dd_s + 1;
                clk2 <= '0';
                if (dd_s = 6) then
                        dd_s := 0;
                        clk2 <= '1';

                end if;
        end if;

-- Se establece la señal de ajuste de los segundos

-- Sincronizacion de segundos
```

```
        if (falling_edge(clk0) and clk0='0') then
                ajuste_sync <= ajuste;
        end if;

        clk_a1 <= (clk2 and not ajuste_sync) or (ajuste_mm and ajuste_sync);

-- Se establece el contador de minutos a 60 (Unidades)

        if (rising_edge(clk_a1) and clk_a1='1') then

                uu_m:= uu_m+1;
                clk3 <= '0';

                if (uu_m=10) then
                        uu_m := 0;
                        clk3 <= '1';
                end if;
        end if;

        --Decenas

        if (rising_edge(clk3) and clk3='1') then

                dd_m := dd_m + 1;
                clk4 <= '0';

                if (dd_m = 6) then
                        dd_m := 0;
                        clk4 <= '1';
                end if;
        end if;

-- Se establece la señal de ajuste a los minutos

        clk_a2 <= (clk4 and not ajuste_sync) or (ajuste_hh and ajuste_sync);

-- Se establece la señal de ajuste a las horas (unidades)

        if (rising_edge(clk_a2) and clk_a2='1') then
                uu_h := uu_h + 1;
                clk5 <= '0';

                if (uu_h = 10) then
                        uu_h := 0;
                        clk5 <= '1';
                end if;
        end if;

        --Decenas

        if (rising_edge(clk5) and clk5 = '1') then
                dd_h := dd_h+1;
        end if;
        if (dd_h = 2 and uu_h = 4) then
                uu_h := 0;
                dd_h := 0;
        end if;
end if;

-- Salida en display, usando la funcion Deco

display1 <= Deco(uu_s);
display2 <= Deco(dd_s);
display3 <= Deco(uu_m);
display4 <= Deco(dd_m);
display5 <= Deco(uu_h);
display6 <= Deco(dd_h);

end process;
end Behavioral;
```

Conclusiones generales.

Con la elaboración del proyecto final logramos poner en práctica lo aprendido a lo largo del curso de diseño digital moderno, tanto de laboratorio como de teoría, desde nuestro punto de vista, el realizar el reloj digital fue de cierta forma un reto ya que logramos conjuntar en un solo bloque funcional lo que correspondería a 6 bloques funcionales (1 por cada unidad y decena de segundos, minutos y horas), ahorrando de esta forma procedimientos repetitivos. De igual forma logramos diseñar un algoritmo que permitiera al divisor de frecuencia obtener una salida a 1 segundo exacto de manera que la visualización de la hora en los displays fuera acorde a un estándar. Desde nuestro punto de vista gracias a este proyecto tenemos la noción de como es que trabajan y funcionan los relojes digitales comerciales, además de que no es tan complicado diseñarlos en cuanto a su respectivo funcionamiento.

Referencias.

Baker, G. (febrero de 2002). *The rising_edge function*. Obtenido de https://www2.cs.sfu.ca/~ggbaker/reference/std_logic/1164/rising_edge.html

Lenguaje descripción de hardware: VHDL. (s.f.). Obtenido de http://www1.frm.utn.edu.ar/tecnicad1/_private/Apuntes/VHDL.pdf

Múltiples Procesos en una Arquitectura. (s.f.). Obtenido de https://www.utm.mx/~merg/AC/vhdl/7_multiples_procesos.pdf

Sánchez Élez, M. (julio de 2014). *INTRODUCCIÓN A LA PROGRAMACIÓN EN VHDL*. Obtenido de https://eprints.ucm.es/id/eprint/26200/1/intro_VHDL.pdf