Sistemas Digitales - 86.41

Trabajo Práctico 1

Contador BCD de 4 dígitos con salida a display 7 segmentos

Diego Almirón, 94051, diego90dionisio@gmail.com

27 de abril de 2017

Resumen

En el presente Trabajo Práctico se busco, a partir de una especificación y un diseño previo, describir una arquitectura, simular, sintetizar e implementar en FPGA un sistema digital para un contador BCD de 4 dígitos con salida a un display de 7 segmentos.

Desarrollo

Para llegar al objetivo que planteo el presente trabajo práctico, se tubo que describir, en lenguaje *VHDL*, varios elementos de hardware.

1. Clock:
   La función del clock es de generar una onda cuadrada cuyo periodo esta definido por un parámetro tau.

2. Generador de enable:
   Se usó principalmente Para habilitar el funcionamiento de otros componente. Ya que el *CLOCK* del Kit Nexis2 es de 50**MHz**, se necesito del generador de enable para evitar que todos los componentes como contadores y demás funcionen a una frecuencia menor.

3. Contadores:
   Se uso dos tipos de contadores, un contador BCD y un contador binario de 2 bits, el contador binario de 2 bits se uso para decidir que display 7 segmentos se prendía en qué tiempo.

4. Multiplexor:
   Este componente fue útil a la hora de decidir cual de las salidas de los contadores BCD era vista en el display.

5. Controladores:
   El controlador, en el sentido de que display se prendía en que tiempo, el controlador es básicamente un generador de enable más un contador de 2 bits más un decodificador. El controlador decide que contador muestra sus valores en que display 7 segmentos.

La figura **??** es el esquemático del contador BCD de 4 dígitos que generó el **ISE**, en él se puede ver que se usaron dos generadores de enable, uno conectado a los contadores BCD y otro conectado al contador de 2 bits que controla los display 7 segmentos. Además se usaron cuatro contadores BCD uno para cada dígito. Los contadores están conectados a un multiplexor que tiene como variable de control al contador de 2 bits. Por último se tienen dos decodificadores, el primero es el decodificador BCD a 7 segmentos y el segundo es es decodificador que controla los displays 7 segmentos alimentando o no sus anodos.

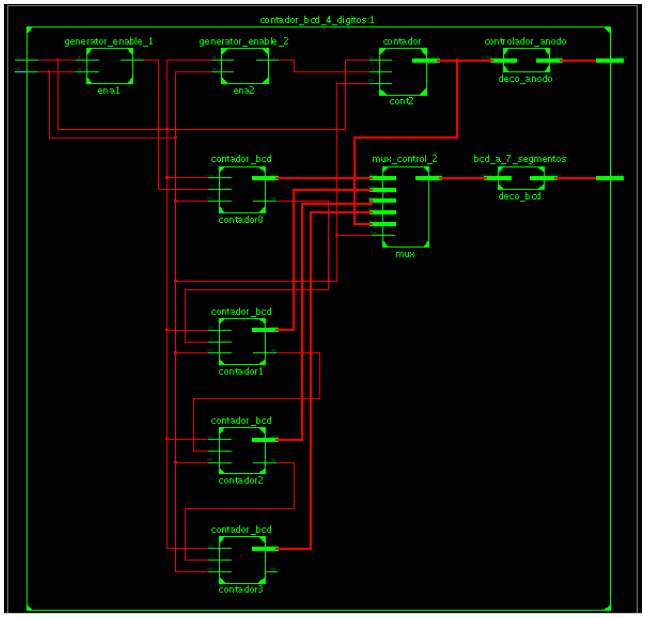

Figura 1: Diagrama en bloque de Contador BCD de 4 dígitos

Gráficos

A continuación se muestran los gráficos de varias simulaciones.

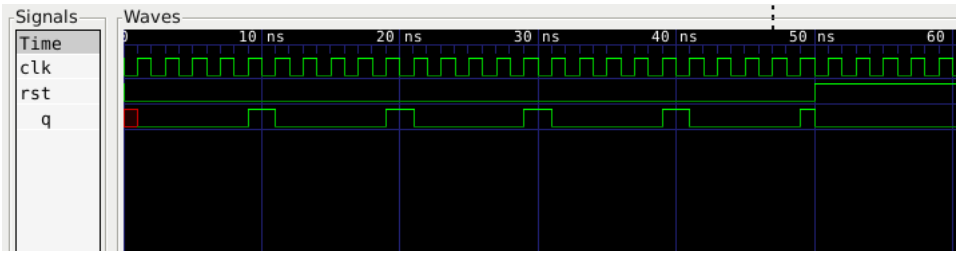

Figura 2: Simulación de generador de enable. Genera un pulso en **q** cada 5 pulsos de clock.

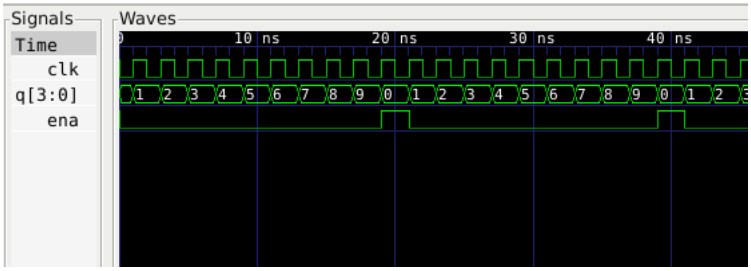

Figura 3: Simulación de contador BCD. Se genera un pulso de **ena** cada vez que la cuenta **q** pasa de 9 a 0.

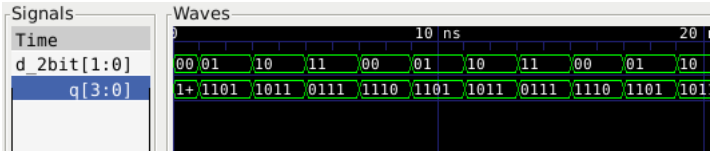

Figura 4: Simulación de controlador de ánodos

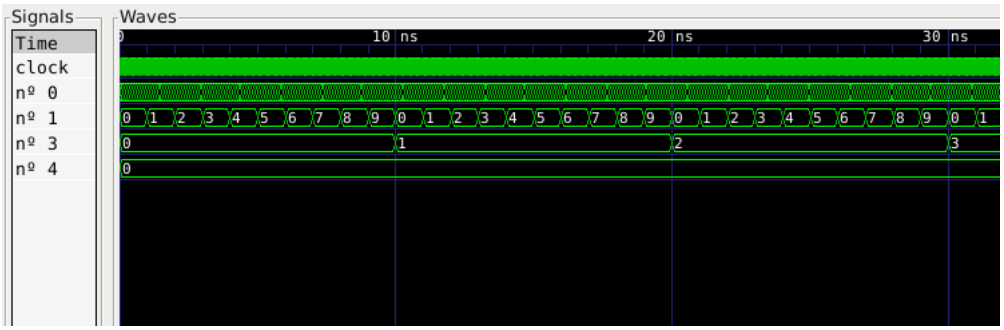

Figura 5: Simulación de contador BCD de 4 dígitos, incluye el clock más los cuatro números

Tabla de resumen de sintesis

Componente	Cantidad utilizada	Porcentaje de utilización
Slices	134	2 %
Flip-Flops	87	0 %
4 input LUTs	252	2 %
GCLKs	1	4 %
Frecuencia máxima de clock	109.727MHz	—

Codigo Fuente

Generador de enable

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

entity generator_enable is
    generic ( N : integer := 1000);
    port (
        clk : in std_logic;
        q : out std_logic;
        rst : in std_logic
    );
end entity generator_enable;


architecture enable_arq of generator_enable is
        signal count : integer := 0;
begin
    process(clk,rst)
    begin
        if rst = '1' then
            q <= '0';
            count <= 0;

        elsif rising_edge(clk) then
            count <= count +1;

            if count = N then
                q <= '1';
                count <= 0;
            else
                q <= '0';
            end if;
        end if;
    end process;
end architecture;
```

Multiplexor

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

entity mux_control_2 is
    generic( N : integer := 4);
    port(
        C0  : in std_logic_vector(N-1 downto 0);
        C1  : in std_logic_vector(N-1 downto 0);
        C2  : in std_logic_vector(N-1 downto 0);
        C3  : in std_logic_vector(N-1 downto 0);
        S   : in std_logic_vector(1 downto 0);
        Q   : out std_logic_vector(N-1 downto 0);
        RST : in std_logic
        );
end entity mux_control_2;

architecture mux_arq of mux_control_2 is
begin


    Q <= (OTHERS => '0') when rst = '1' else
         C0 when S = "00" else
         C1 when S = "01" else
         C2 when S = "10" else
         C3 when S = "11";


end architecture mux_arq;
```

Contador 2 bits

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

entity contador is
    generic( N : integer := 1);
    port(
        D   : in std_logic;
        CLK : in std_logic;
        Q   : out std_logic_vector(N-1 downto 0);
        RST : in std_logic
        );
end entity contador;

architecture cont_arq of contador is
    signal count : integer range 0 to 3 := 0;
begin
    process(clk,rst)
    begin
        if rst = '1' then
            count <= 0;
        elsif rising_edge(clk) then
            if D = '1' then
                if count = 3 then
                    count <= 0;
                else
                    count <= count + 1;
                end if;
            end if;
        end if;
    end process;
    Q   <= std_logic_vector( to_signed(count,N));
end architecture;
```

Controlador anodo

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

entity controlador_anodo is
    port(
        D_2BIT  : in std_logic_vector(1 downto 0);
        Q       : out std_logic_vector(3 downto 0)
        );
end entity controlador_anodo;


architecture controlador_arq of controlador_anodo is

begin
    with D_2BIT select
        Q       <= "1110" when "00",
                   "1101" when "01",
                   "1011" when "10",
                   "0111" when "11",
                   "1110" when others;
end architecture controlador_arq;
```

Contador bcd

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

entity contador_bcd is
    port(
        D   : in std_logic;                     --entrada de enable
        CLK : in std_logic;                     --entrada de clock
        Q   : out std_logic_vector(3 downto 0); --salida bcd
        RST : in std_logic;                     --entrada de rst
        ENA : out std_logic                     --salida de enable de 9 -> 0
    );
end entity contador_bcd;


architecture cont_arq of contador_bcd is

    signal count : integer range 0 to 9;
begin
    process(clk,rst)
    begin
        if rst = '1' then
            count <= 0;
            ena <= '0';

        elsif rising_edge(clk) then
            ena <= '0';
            if D = '1' then
                if count = 9 then
                    count <= 0;
                    ena <= '1';
                else
                    count <= count + 1 ;
                    ena <= '0';
                end if;
            end if;
        end if;
    end process;

    Q <= std_logic_vector( to_signed(count,4) );
end architecture;
```

Decodificador BCD a 7 segmentos

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


entity bcd_a_7_segmentos is
    port(
            BIN : in std_logic_vector(3 downto 0);
            DIG : out std_logic_vector(7 downto 0)
        );
end bcd_a_7_segmentos;


architecture bcd_7_seg_arq of bcd_a_7_segmentos is

begin
    with BIN select
        DIG     <=  "00000011" when "0000",
                    "10011111" when "0001",
                    "00100101" when "0010",
                    "00001101" when "0011",
                    "10011001" when "0100",
                    "01001001" when "0101",
                    "11000001" when "0110",
                    "00011111" when "0111",
                    "00000001" when "1000",
                    "00011001" when "1001",
                    "00000011" when others;
end architecture bcd_7_seg_arq;
```

Contador BCD 4 digitos

```
library ieee;
use     ieee.std_logic_1164.all;
--library tp1;
--use     tp1.my_package.all;
use work.my_package.all;


entity contador_bcd_4_digitos is
    port(
            CLK : in std_logic;
            RST : in std_logic;
            Q   : out std_logic_vector(7 downto 0);
            QAN : out std_logic_vector(3 downto 0)
        );
end entity;

architecture arq of contador_bcd_4_digitos is

    --constantes
    constant cuenta_contador : integer := 50000000;  -- hasta donde cuenta en enable de lo
    constant cuenta_display  : integer := 50000;     -- hasta donde cuenta el enable de lo

    --Propias del contador de 4 digitos
    signal clk_i    : std_logic;                     -- señal de clk
    signal rst_i    : std_logic;                     -- señal de rst
    signal Qdeco    : std_logic_vector(7 downto 0);  -- salida del decodificador bcd a 7 s
    signal anodo    : std_logic_vector(3 downto 0);  -- salida del controlador de anodos p

    --Señales auxiliares
    signal Qena1    : std_logic;                     -- enable de los contadores
    signal Qena2    : std_logic;                     -- enable de los display's
    signal Qcont0   : std_logic_vector(3 downto 0);  -- salida del contador bcd nro 0
    signal Qcont1   : std_logic_vector(3 downto 0);  -- salida del contador bcd nro 1
    signal Qcont2   : std_logic_vector(3 downto 0);  -- salida del contador bcd nro 2
    signal Qcont3   : std_logic_vector(3 downto 0);  -- salida del contador bcd nro 3
    signal Cont_ena : std_logic_vector(2 downto 0);  -- salida de enable de los contadores
    signal Cont2bit : std_logic_vector(1 downto 0);  -- salida del contador de 2 bit
    signal Qmux     : std_logic_vector(3 downto 0);  -- salida del multiplexor

begin
    -- vinculando puertos con señales
    clk_i <= CLK;
    rst_i <= RST;
    Q     <= Qdeco;
    QAN   <= anodo;

    -- generador de enable
    ena1    :   generator_enable
```

```
generic map ( N => cuenta_contador )
port map (
         CLK => clk_i,
         Q  => Qena1,
         RST => rst_i
         );
ena2    :   generator_enable
generic map ( N => cuenta_display)
port map(
        CLK => clk_i,
        Q  => Qena2,
        RST => '0'
        );

--decodificador de 2 a 4 para anodo
deco_anodo : controlador_anodo
port map (
         D_2BIT => Cont2bit,
         Q  => anodo
         );

--decodificador bcd a 7 segmentos
deco_bcd : bcd_a_7_segmentos
port map (
         BIN => Qmux,
         DIG => Qdeco
         );

--multiplexor de dos variables de control
mux     :   mux_control_2
port map (
         C0 => Qcont0,
         C1 => Qcont1,
         C2 => Qcont2,
         C3 => Qcont3,
         S  => Cont2bit,
         Q  => Qmux,
         RST => rst_i
         );

--contador dos dígitos
cont2   :   contador
generic map ( N => 2)
port map (
         D  =>  Qena2,
         CLK => clk_i,
         Q  =>  Cont2bit,
         RST => '0'
         );
```

```
    --cuatro contadores bcd
    contador0 : contador_bcd
    port map (
             D      => Qena1,
             CLK    => clk_i,
             Q      => Qcont0,
             RST    => rst_i,
             ENA    => Cont_ena(0)
             );
    contador1 : contador_bcd
    port map (
             D      => Cont_ena(0),
             CLK    => clk_i,
             Q      => Qcont1,
             RST    => rst_i,
             ENA    => Cont_ena(1)
             );
    contador2 : contador_bcd
    port map (
             D      => Cont_ena(1),
             CLK    => clk_i,
             Q      => Qcont2,
             RST    => rst_i,
             ENA    => Cont_ena(2)
             );
    contador3 : contador_bcd
    port map (
             D      => Cont_ena(2),
             CLK    => clk_i,
             Q      => Qcont3,
             RST    => rst_i,
             ENA    => Open
             );
end architecture arq;
```