TP N°1: Contador BCD de 4 dígitos con salida a display 7 segmentos

Vázquez, Matías - 91523
mfvazquez@gmail.com

10 de octubre de 2015

En el presente Trabajo Práctico se implementará en FPGA un sistema digital para un contador BCD de 4 dígitos con salida a un display de 7 segmentos

1. Especificaciones

Se implementó en lenguaje descriptor de hardware VHDL un contador BCD de 4 dígitos y un controlador para un display de 7 segmentos de 4 cifras. El contador se incrementá aproximadamente cada 1 segundo.

Se utilizó el kit de desarrollo "Spartan-3 Starter Board" de la empresa digilent. Utilizando una frecuencia de clock de 50 MHz.

1.1. Display de 7 segmentos de 4 cifras

Cada dígito comparte ocho señales de control para encender cada LED individual que corresponde a un segmento del carácter. Cada carácter tiene un ánodo asociado. Poniendo un '0' en el terminal de la FPGA que se conecta a ese ánodo, se selecciona el carácter a encenderse.

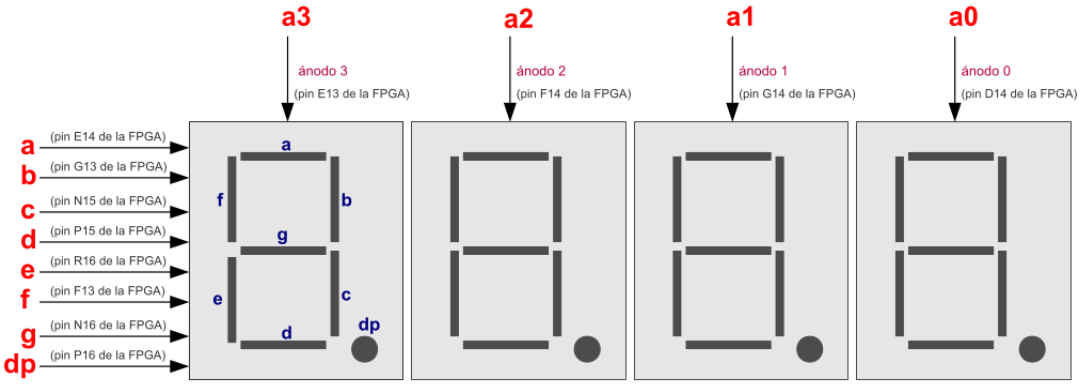

Figura 1: Conexión entre los I/Os de la FPGA y el display

2. Diseño

La implementación de cada bloque mostrado en el diagrama en bloques de la figura 2 se realizó lo más genérico posible de forma de poder reutilizar el mismo código a futuro.

2.1. Diagrama en bloques

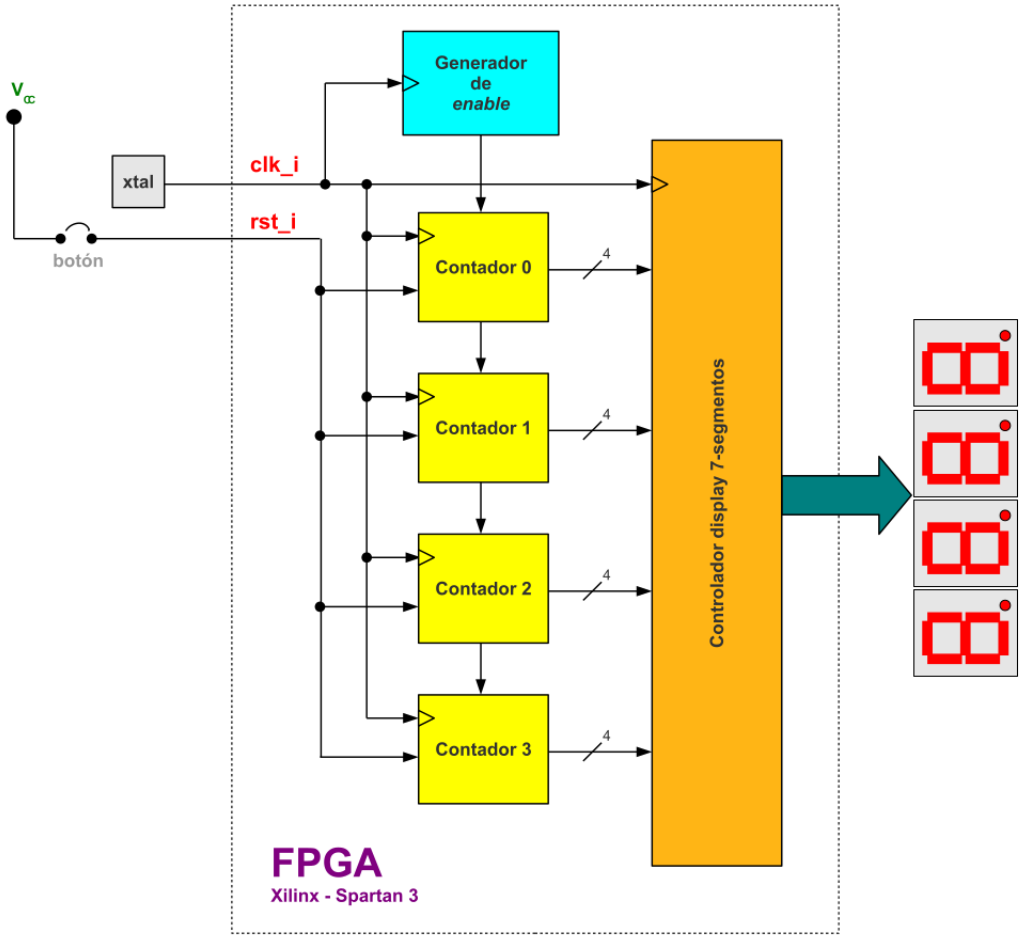

Figura 2: Diagrama en bloques de la arquitectura

2.2. Generador de enable

Este bloque divide la frecuencia de reloj de la placa a una frecuencia que tenga un período de aproximadamente 1 seg.

Para implementarlo se programó un contador de N bits, que devuelve un 1 en su salida durante un período del clock cuando los M bits más significativos estén en 1. Pasado este valor se resetea el contador.

Para que cuente aproximadamente 1 seg se fijaron los valores $N = 25$ y $M = 2$. Para así contar hasta que el bit 25 y 24 del contador estén en 1. El perído de la salida se puede calcular mediante:

$$T = \frac{2^{25} + 2^{24}}{50\,\text{MHz}} \approx 1\,\text{seg}$$

2.3. Contadores

Cuentan de 0 a 9, tienen una entrada de reset y una entrada de enable. Su salida es un contador de 4 bits y 1 bit para el flag de fin de cuenta, que solo devolverá 1 si el contador contó hasta 9 y la entrada de enable esta habilitada.

2.4. Controlador de display de 7 segmentos

En la figura 3 se muestra el diagrama en bloques de la arquitectura del controlador del display de 7 segmentos.

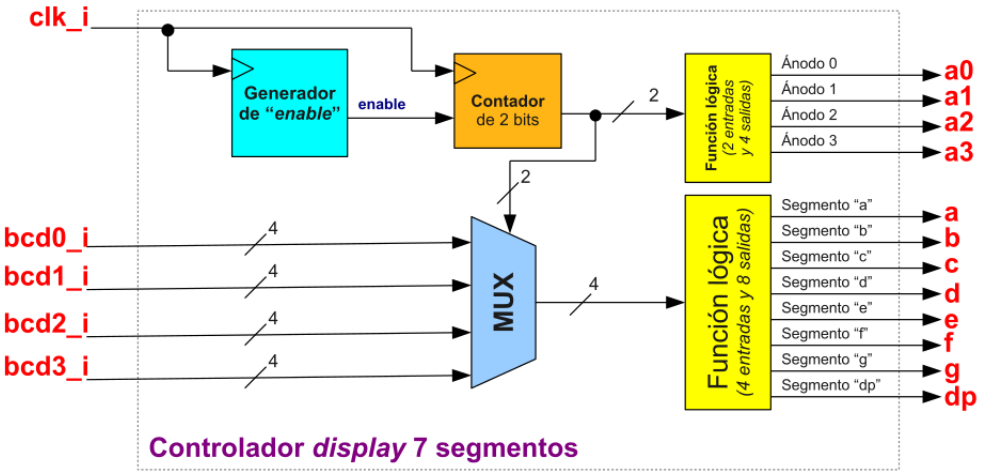

Figura 3: Arqutectura para el controlador del display de 7 segmentos

Cuenta con los siguientes elementos:

- **Multiplexor:** selecciona cuál de las entradas en formato BCD se imprimirá según lo que indica el contador.
- **Generador de enable:** Se utilizó el generador de enable genérico implementado, con $N = 17$ y $M = 21$ de forma que cada cifra del display se encienda aproximadamente a una frecuencia de 100 Hz. Como son 4 cifras, deberá devolver una salida con una frecuencia de 400 Hz aproximadamente. Por lo que la frecuencia de salida se la puede calcular mediante:

$$f = \frac{50\,\text{MHz}}{2^{17}} \approx 381,5\,\text{Hz}$$

- **Función lógica de 4 entradas y 8 salidas:** Es la encargada de mapear los números en formato BCD a los LEDs que deben encenderse para representar dicho número.
- **Función lógica de 2 entradas y 4 salidas:** Es la encargada de seleccionar el ánodo según el valor del contador
- **Contador de 2 bits:** Es el encargado de conmutar el ánodo para mantener las 4 cifras del display encendidas. Para hacerlo lo más genérico posible se implementó un contador de N bits, que cuenta hasta que todos sus bits sean 1.

3. Simulaciones

A continuación se muestran algunas simulaciones realizadas mediante pruebas.

3.1. Contador

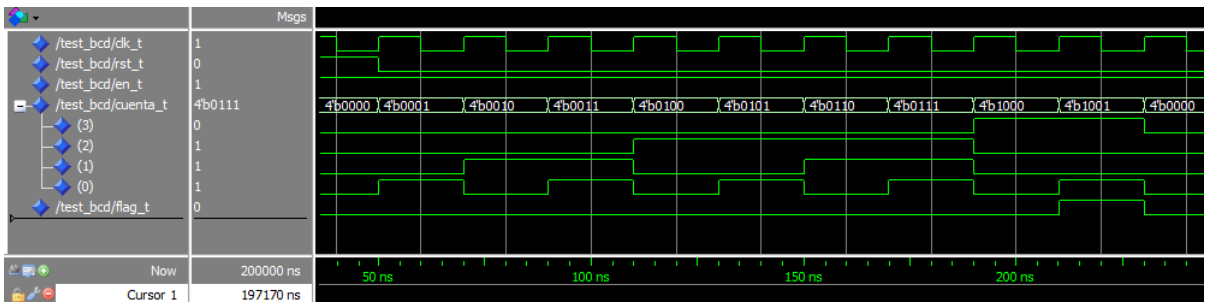

Figura 4: Simulación de BCD_test.vhd

3.2. Contador de N bits

Para esta prueba se utilizó $N = 4$, que es la cantidad de bits del contador.

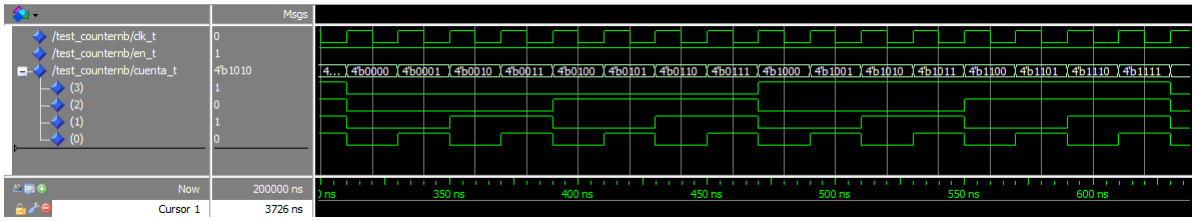

Figura 5: Simulación de CounterNb_test.vhd

3.3. Multiplexor

Para esta prueba las entradas del multiplexor eran: 1000, 0100, 0010 y 0001.

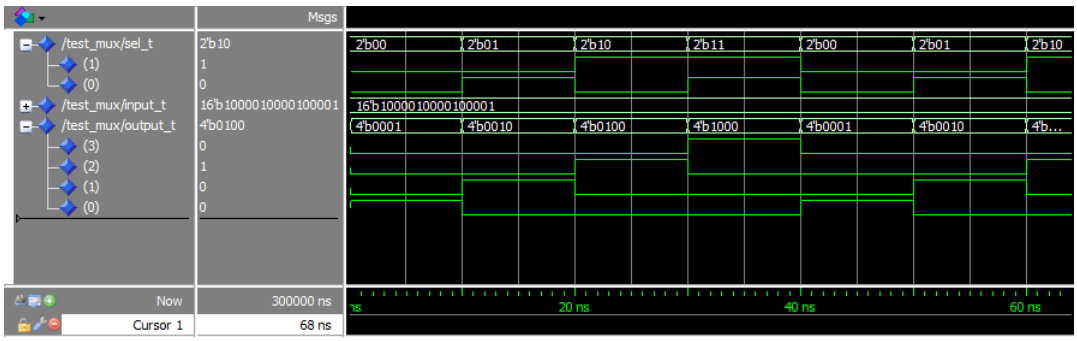

Figura 6: Simulación de Mux_test.vhd

3.4. Selector de display

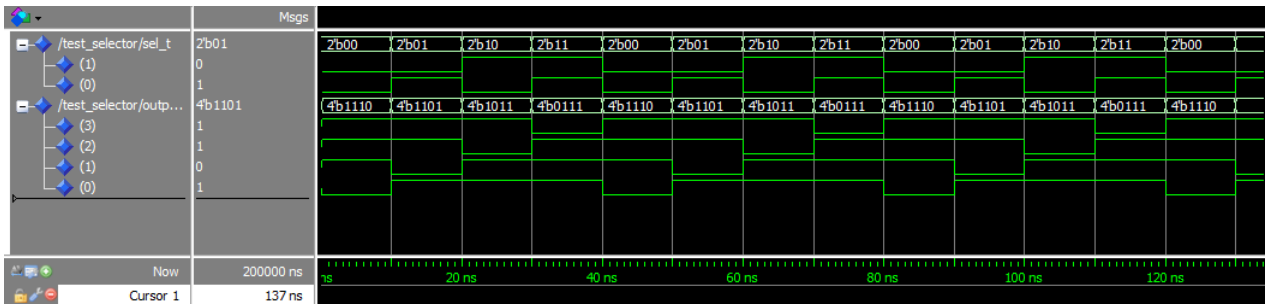

Figura 7: Simulación de Selector_test.vhd

3.5. Contador BCD de 4 digitos con salida a display de 7 segmentos

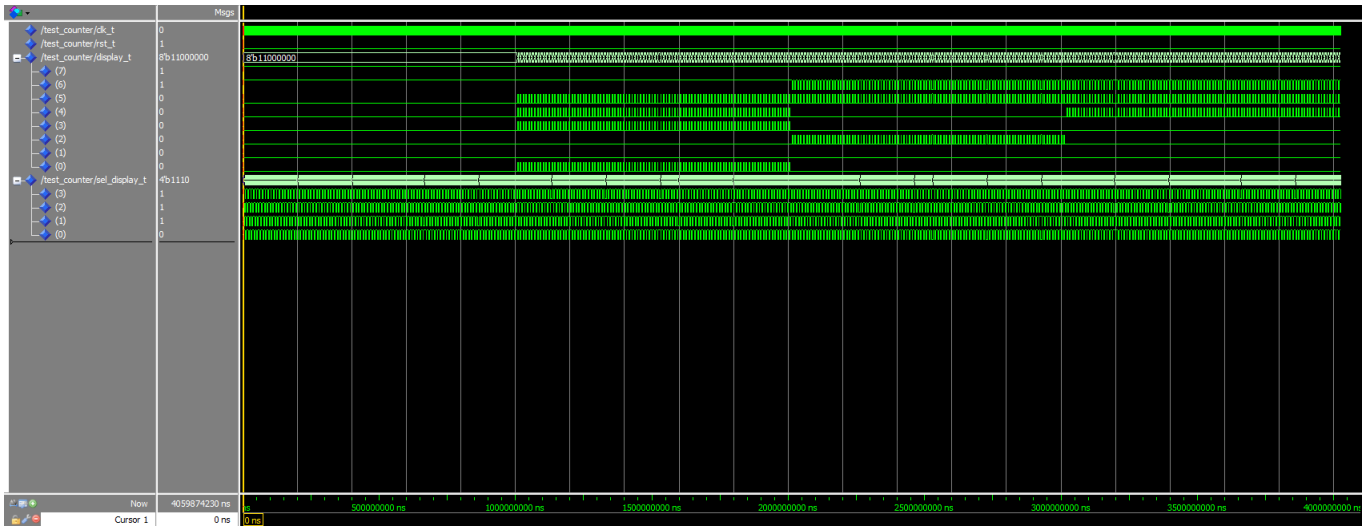

Figura 8: Simulación de Counter_test.vhd

4. Resumen de síntesis

Utilización Lógica	Usados	Utilización
Slices	69	3 %
Flip-Flops	68	1 %
LUTs	138	3 %
GCLK	1	12 %
frecuencia máxima	119,147 MHz	

5. Código fuente VHDL

5.1. GenEna.vhd

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

entity GenEna is
	generic(
		N: natural := 25;
		M: natural := 2
			);
	port(
		clk: in std_logic;
		en: out std_logic
	);
end;

architecture Beh of GenEna is
begin
	process(clk)
		variable cuenta: unsigned(N downto 0);
		variable comparador: unsigned(M-1 downto 0) := (others => '1');
		variable rst: std_logic := '1';
	begin
		if rising_edge(clk) then
			cuenta := cuenta + 1;
			if rst = '1' then
				cuenta := (others => '0');
				en <= '0';
				rst := '0';
			elsif comparador = cuenta(N downto N+1-M) then
				en <= '1';
				rst := '1';
			end if;
		end if;
	end process;
end;
```

5.2. GenEna_test.vhd

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


entity test_GenEna is
end;

architecture Beh of test_GenEna is
        signal clk_t: std_logic := '0';
        signal en_t: std_logic;

        component GenEna is
                generic (
                        N: natural := 25;
                        M: natural := 2
                );
                port (
                clk: in std_logic;
                en: out std_logic
                );
        end component;

begin
        inst_GenEna: GenEna
                generic map(
                        N => 4,
                        M => 1
                )
                port map(
                        clk => clk_t,
                        en => en_t
                );
        clk_t <= not clk_t after 10 ns;
end;
```

5.3. BCD.vhd

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

entity BCD is
	port(
		clk: in std_logic;
		rst: in std_logic;
		en: in std_logic;
		cuenta: out std_logic_vector(3 downto 0);
		flag: out std_logic
	);
end;

architecture BCD_arq of BCD is
	signal flag_aux: std_logic;
begin
	process(clk, rst, en)
		variable aux: unsigned(3 downto 0);
	begin
		if rst = '1' then
			aux := "0000";
			flag_aux <= '0';
		elsif rising_edge(clk) then
			if en = '1' then
				aux := aux + 1;
				if aux = "1001" then
					flag_aux <= '1';
				elsif aux = "1010" then
					aux := "0000";
					flag_aux <= '0';
				end if;
			end if;
		end if;
		cuenta <= std_logic_vector(aux);
	end process;
	flag <= flag_aux and en;
end;
```

5.4. BCD_test.vhd

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


entity test_BCD is
end;

architecture test_BCD1 of test_BCD is
        signal clk_t: std_logic := '0';
        signal rst_t: std_logic := '1';
        signal en_t: std_logic := '1';
        signal cuenta_t: std_logic_vector(3 downto 0);
        signal flag_t: std_logic;

        component BCD is
                port (
                clk: in std_logic;
                rst: in std_logic;
                en: in std_logic;
                cuenta: out std_logic_vector(3 downto 0);
                flag: out std_logic
                );
        end component;

begin
        inst_BCD: BCD
                port map(
                        clk => clk_t,
                        rst => rst_t,
                        en => en_t,
                        cuenta => cuenta_t,
                        flag => flag_t
                );
        clk_t <= not clk_t after 10 ns;
        rst_t <= '0' after 50 ns;
end;
```

5.5. BCD4_test.vhd

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


entity test_BCD4 is
end;

architecture test_BCD4 of test_BCD4 is
        signal clk_t: std_logic := '0';
        signal rst_t: std_logic := '1';
        signal en_t: std_logic_vector(4 downto 0);
        signal bcd_t: std_logic_vector(15 downto 0);


        component BCD is
                port(
                        clk: in std_logic;
                        rst: in std_logic;
                        en: in std_logic;
                        cuenta: out std_logic_vector(3 downto 0);
                        flag: out std_logic
                );
        end component;

begin

        en_t(0) <= '1';

        bcd_i :
        for i in 0 to 3 generate

        inst_BCD: BCD
                port map(
                        clk => clk_t,
                        rst => rst_t,
                        en => en_t(i),
                        cuenta => bcd_t(i*4+3 downto i*4),
                        flag => en_t(i+1)
                );

    end generate;

        clk_t <= not clk_t after 10 ns;
        rst_t <= '0' after 50 ns;
end;
```

5.6. Mux.vhd

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

entity Mux is
        generic(
                inputs: natural := 16;
                data_bus: natural := 4;
                sel_length: natural := 2
        );
        port(
                input: in std_logic_vector(inputs-1 downto 0);
                sel: in std_logic_vector(sel_length-1 downto 0);
                output: out std_logic_vector(data_bus-1 downto 0)
        );

end;

architecture Beh of Mux is
begin
        process(sel, input)
        variable sel_aux: unsigned(sel_length-1 downto 0);
        variable fin, inicio: natural;
        begin
                sel_aux := unsigned(sel);
                inicio := to_integer(sel_aux);
                inicio := inicio * data_bus;
                fin := inicio + data_bus - 1;

                output <= input(fin downto inicio);
        end process;
end;
```

5.7. Mux_test.vhd

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


entity test_Mux is
end;

architecture Beh of test_Mux is
        signal  sel_t: std_logic_vector(1 downto 0) := "00";
        signal  input_t: std_logic_vector(15 downto 0) := B"1000_0100_0010_0001";
        signal  output_t: std_logic_vector(3 downto 0);

        component Mux is
                generic(
                        inputs: natural := 16;
                        data_bus: natural := 4;
                        sel_length: natural := 2
                );
                port(
                        input: in std_logic_vector(inputs-1 downto 0);
                        sel: in std_logic_vector(sel_length-1 downto 0);
                        output: out std_logic_vector(data_bus-1 downto 0)
                );
        end component;

begin

        inst_Mux: Mux
                generic map(
                        inputs => 16,
                        data_bus => 4,
                        sel_length => 2
                )
                port map(
                        input => input_t,
                        sel => sel_t,
                        output => output_t
                );
        sel_t(0) <= not sel_t(0) after 10 ns;
        sel_t(1) <= not sel_t(1) after 20 ns;
end;
```

5.8. Serializer.vhd

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

entity Serializer is
	port(
		input: in std_logic_vector(3 downto 0);
		display: out std_logic_vector(7 downto 0)
	);

end;

architecture Serializer_arq of Serializer is
begin
	process(input)
	begin
		case input is
			when "0000" =>
				display <= "11000000";
			when "0001" =>
				display <= "11111001";
			when "0010" =>
				display <= "10100100";
			when "0011" =>
				display <= "10110000";
			when "0100" =>
				display <= "10011001";
			when "0101" =>
				display <= "10010010";
			when "0110" =>
				display <= "10000010";
			when "0111" =>
				display <= "11111000";
			when "1000" =>
				display <= "10000000";
			when "1001" =>
				display <= "10010000";
			when others =>
				display <= "11111111";
		end case;
	end process;
end;
```

5.9. Serializer_test.vhd

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


entity test_Serializer is
end;

architecture test_Serializer of test_Serializer is
        signal  input_t: std_logic_vector(3 downto 0) := "0000";
        signal  display_t: std_logic_vector(7 downto 0);


        component Serializer is
                port (
                        input: in std_logic_vector(3 downto 0);
                        display: out std_logic_vector(7 downto 0)
                );
        end component;

begin
        inst_Serializer: Serializer
                port map(
                        input => input_t,
                        display => display_t
                );
        input_t(0) <= not input_t(0) after 10 ns;
        input_t(1) <= not input_t(1) after 20 ns;
        input_t(2) <= not input_t(2) after 40 ns;
        input_t(3) <= not input_t(3) after 80 ns;
end;
```

5.10. Selector.vhd

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

entity Selector is
        generic(
                N: natural := 2;
                M: natural := 4
        );
        port(
                sel: in std_logic_vector(N-1 downto 0);
                output: out std_logic_vector(M-1 downto 0)
        );
end;

architecture Beh of Selector is
begin
        process(sel)
                variable sel_aux: natural;
        begin
                sel_aux := to_integer(unsigned(sel));
                output <= (others => '1');
                output(sel_aux) <= '0';
        end process;
end;
```

5.11. Selector_test.vhd

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


entity test_Selector is
end;

architecture test_SelDisplay of test_Selector is
        signal  sel_t: std_logic_vector(1 downto 0) := "00";
        signal  output_t: std_logic_vector(3 downto 0);


        component Selector is
                generic(
                        N: natural := 2;
                        M: natural := 4
                );
                port(
                        sel: in std_logic_vector(N-1 downto 0);
                        output: out std_logic_vector(M-1 downto 0)
                );
        end component;

begin
        inst_Selector: Selector
                port map(
                        sel => sel_t,
                        output => output_t
                );
        sel_t(0) <= not sel_t(0) after 10 ns;
        sel_t(1) <= not sel_t(1) after 20 ns;
end;
```

5.12. CounterNb.vhd

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

entity CounterNb is
        generic(
                N: natural := 2
                );
        port(
                clk: in std_logic;
                en: in std_logic;
                cuenta: out std_logic_vector(N-1 downto 0)
        );
end;

architecture Beh of CounterNb is
begin
        process(clk)
                variable cuenta_aux: unsigned(N-1 downto 0) := (others => '0');
                variable comparador: unsigned(N-1 downto 0) := (others => '1');
                variable rst: std_logic := '0';
        begin
                if en = '1' and rising_edge(clk) then
                        if rst = '1' then
                                cuenta_aux := (others => '0');
                                rst := '0';
                        else
                                cuenta_aux := cuenta_aux + 1;
                                if cuenta_aux = comparador then
                                        rst := '1';
                                end if;
                        end if;
                end if;
                cuenta <= std_logic_vector(cuenta_aux);
        end process;
end;
```

5.13. CounterNb_test.vhd

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


entity test_CounterNb is
end;

architecture beh of test_CounterNb is
        signal clk_t: std_logic := '0';
        signal en_t: std_logic := '1';
        signal cuenta_t: std_logic_vector(3 downto 0);

        component CounterNb is
                generic(
                        N: natural := 2
                );
                port (
                        clk: in std_logic;
                        en: in std_logic;
                        cuenta: out std_logic_vector(N-1 downto 0)
                );
        end component;

begin
        inst_CounterNb: CounterNb
                generic map(
                        N => 4
                )
                port map(
                        clk => clk_t,
                        en => en_t,
                        cuenta => cuenta_t
                );
        clk_t <= not clk_t after 10 ns;
end;
```

5.14. DisplayController.vhd

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

entity DisplayController is
	port(
		clk: in std_logic;
		bcd: in std_logic_vector(15 downto 0);
		sel_display: out std_logic_vector(3 downto 0);
		display: out std_logic_vector(7 downto 0)
	);
end;

architecture Beh of DisplayController is
	signal clk_aux: std_logic;
	signal en_aux: std_logic;
	signal sel_aux: std_logic_vector(1 downto 0);
	signal bcd_out_aux: std_logic_vector(3 downto 0);
	signal sel_display_aux: std_logic_vector(3 downto 0);
	signal display_aux: std_logic_vector(7 downto 0);
	signal input_mux_aux: std_logic_vector(15 downto 0);

	component GenEna is
		generic(
			N: natural := 17;
			M: natural := 1
				);
		port(
			clk: in std_logic;
			en: out std_logic
		);
	end component;

	component CounterNb is
		generic(
			N: natural := 2
			);
		port(
			clk: in std_logic;
			en: in std_logic;
			cuenta: out std_logic_vector(N-1 downto 0)
		);
	end component;

	component Mux is
		generic(
			inputs: natural := 16;
			data_bus: natural := 4;
			sel_length: natural := 2
		);
		port(
			input: in std_logic_vector(inputs-1 downto 0);
			sel: in std_logic_vector(sel_length-1 downto 0);
			output: out std_logic_vector(data_bus-1 downto 0)
		);
	end component;

	component Selector is
```

```
                generic(
                        N: natural := 2;
                        M: natural := 4
                );
                port(
                        sel: in std_logic_vector(N-1 downto 0);
                        output: out std_logic_vector(M-1 downto 0)
                );
        end component;

        component Serializer is
                port (
                        input: in std_logic_vector(3 downto 0);
                        display: out std_logic_vector(7 downto 0)
                );
        end component;

begin
        clk_aux <= clk;
        input_mux_aux <= bcd;
        display <= display_aux;
        sel_display     <= sel_display_aux;

        inst_GenEna: GenEna
                generic map(
                        N => 17,
                        M => 1
                )
                port map(
                        clk => clk_aux,
                        en => en_aux
                );

        inst_CounterNb: CounterNb
                port map(
                        clk => clk_aux,
                        en => en_aux,
                        cuenta => sel_aux
                );

        inst_Mux: Mux
                port map(
                        input => input_mux_aux,
                        sel => sel_aux,
                        output => bcd_out_aux
                );

        inst_Selector: Selector
                port map(
                        sel => sel_aux,
                        output => sel_display_aux
                );


        inst_Serializer: Serializer
                port map(
                        input => bcd_out_aux,
                        display => display_aux
                );
end;
```

5.15. DisplayController_test.vhd

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


entity test_DisplayController is
end;


architecture test_DisplayController of test_DisplayController is
        signal  clk_t: std_logic := '0';
        signal  bcd_t: std_logic_vector(15 downto 0);
        signal display_t: std_logic_vector(7 downto 0);
        signal sel_display_t: std_logic_vector(3 downto 0);
        signal cuenta_aux: unsigned(15 downto 0) := B"0000_0000_0000_0000";

        component DisplayController is
                port (
                        clk: in std_logic;
                        bcd: in std_logic_vector(15 downto 0);
                        sel_display: out std_logic_vector(3 downto 0);
                        display: out std_logic_vector(7 downto 0)
                );
        end component;

begin
        inst_DisplayController: DisplayController
                port map(
                        clk => clk_t,
                        bcd => bcd_t,
                        sel_display => sel_display_t,
                        display => display_t
                );
        clk_t <= not clk_t      after 10 ns;
        bcd_t <= std_logic_vector(cuenta_aux);
        cuenta_aux <= (cuenta_aux + 1) after 10 ms;
end;
```

5.16. Counter.vhd

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

entity Counter is
	port(
		clk: in std_logic;
		rst: in std_logic;
		sel_display: out std_logic_vector(3 downto 0);
		display: out std_logic_vector(7 downto 0)
	);

	attribute loc: string;

	attribute loc of clk:   signal is "B8";
	attribute loc of rst:   signal is "B18";
	attribute loc of sel_display: signal is "F15 C18 H17 F17";
	attribute loc of display:       signal is "C17 H14 J17 G14 D16 D17 F18 L18";

end;

architecture Beh of Counter is
	signal  clk_aux: std_logic;
	signal  rst_aux: std_logic;
	signal  en_aux: std_logic_vector(4 downto 0);
	signal  bcd_aux: std_logic_vector(15 downto 0);
	signal  sel_display_aux: std_logic_vector(3 downto 0);
	signal  display_aux: std_logic_vector(7 downto 0);

	component BCD is
		port(
			clk: in std_logic;
			rst: in std_logic;
			en: in std_logic;
			cuenta: out std_logic_vector(3 downto 0);
			flag: out std_logic
		);
	end component;

	component DisplayController is
		port(
			clk: in std_logic;
			bcd: in std_logic_vector(15 downto 0);
			sel_display: out std_logic_vector(3 downto 0);
			display: out std_logic_vector(7 downto 0)
		);
	end component;

	component GenEna is
	generic(
		N: natural := 25;
		M: natural := 2
	);
	port(
		clk: in std_logic;
		en: out std_logic
	);
	end component;
```

```
begin
        clk_aux <= clk;
        rst_aux <= rst;
        display <= display_aux;
        sel_display <= sel_display_aux;

        inst_GenEna: GenEna
                port map(
                        clk => clk_aux,
                        en => en_aux(0)
                );

        bcd_i :
        for i in 0 to 3 generate

        inst_BCD: BCD
                port map(
                        clk => clk_aux,
                        rst => rst_aux,
                        en => en_aux(i),
                        cuenta => bcd_aux(i*4+3 downto i*4),
                        flag => en_aux(i+1)
                );

    end generate;

        inst_DisplayController: DisplayController
                port map(
                        clk => clk_aux,
                        bcd => bcd_aux,
                        sel_display => sel_display_aux,
                        display => display_aux
                );

end;
```

5.17. Counter_test.vhd

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


entity test_Counter is
end;


architecture test_Counter of test_Counter is
        signal  clk_t: std_logic := '0';
        signal  rst_t: std_logic := '1';
        signal display_t: std_logic_vector(7 downto 0);
        signal sel_display_t: std_logic_vector(3 downto 0);

        component Counter is
                port(
                        clk: in std_logic;
                        rst: in std_logic;
                        sel_display: out std_logic_vector(3 downto 0);
                        display: out std_logic_vector(7 downto 0)
                );
        end component;

begin
        inst_Counter: Counter
                port map(
                        clk => clk_t,
                        rst => rst_t,
                        sel_display => sel_display_t,
                        display => display_t
                );
        rst_t <= '0' after 20 ns;
        clk_t <= not clk_t      after 10 ns;
end;
```