Tabla de resumen de síntesis

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	—

Código Fuente

Clock

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

entity clock is
    generic( tau : time := 1 ns);
    port(
        RST : in std_logic;
        Q : out std_logic
    );
end entity clock;

architecture clock_arq of clock is
    signal Q_int : std_logic := '0';
    signal RST_int : std_logic;
begin
    -- Asignación de señales
    Q <= Q_int;
    RST_int <= RST;

    -- Lógica
    Q_int <= '0' when RST_int = '1' else
             not Q_int after tau;
end architecture clock_arq;
```

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
begin
    process(clk,rst)
        variable count : integer := 0;
    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,d)
    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;
```

0.1 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 10;
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;
```