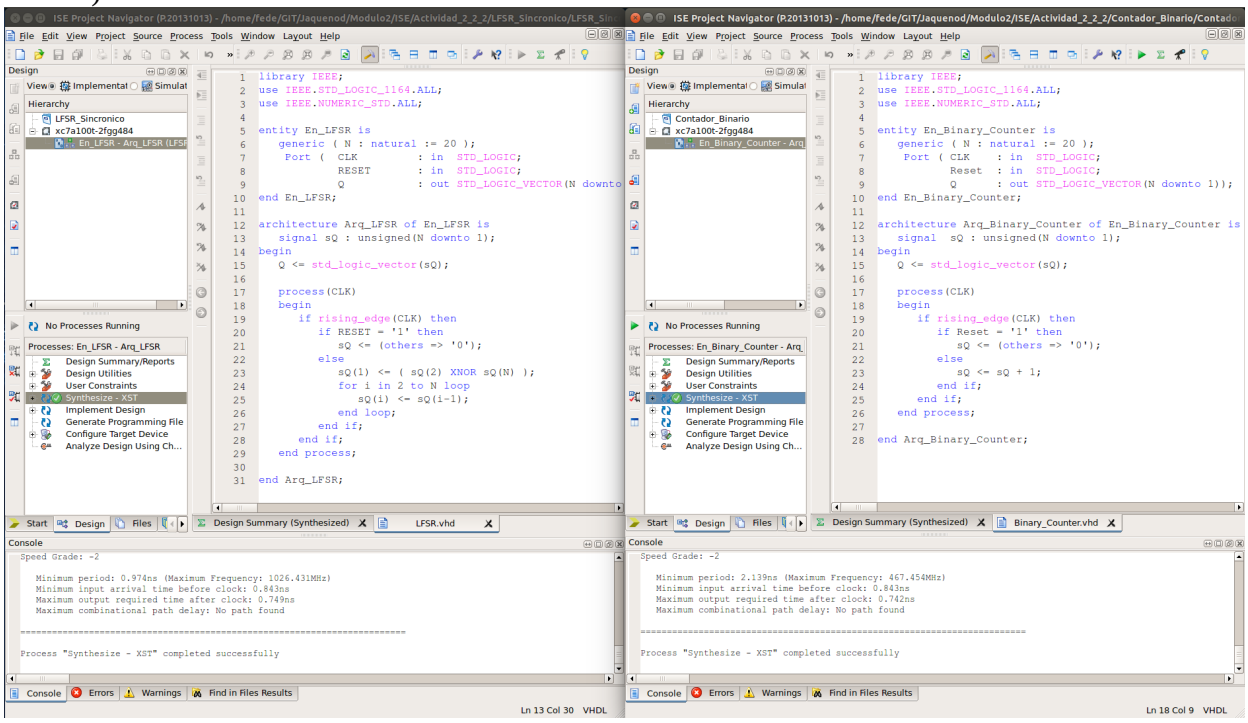
Actividad 2.1

1)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity En_LFSR is
  generic ( N : NATURAL := 7);
  port(
    cuentas : out STD_LOGIC_VECTOR(7 downto 1)
  );
end En_LFSR;

architecture Arq_LFSR of En_LFSR is
  subtype salida is STD_LOGIC_VECTOR(7 downto 1);

   function LFSR ( FB, n_stages, n_counts : integer; init_value : STD_LOGIC_VECTOR(N downto 1); type_of_FB : std_ulogic) return salida is -- type_of_FB = '0' es XOR, '1' es XNOR
     variable Q    : salida;
  begin
   Q     := init_value;

    loop1 : for i in 1 to n_counts loop
        Q := Q(n_stages-1 downto 1) & (type_of_FB XOR Q(FB) XOR Q(n_stages));
      end loop loop1;
    return Q;
  end LFSR;

begin

  cuentas <= LFSR(2,4,3,to_unsigned(0,N),'1');

end Arq_LFSR;
```

Se implementó una función que devuelve el valor del contador en **n_counts** desde el valor inicial **init_value**. El contador se implementa con doble realimentación (salidas de las etapas **FB** y **n_stages** –última etapa-). Además con **type_of_FB** se define si se desea realimentación XOR (**type_of_FB** = '1') o realimentación XNOR (**type_of_FB** = '0').

2)



Se implementaron dos contadores de longitud variable (en este caso N = 20), y se verificó la notable mayor velocidad máxima de operación en el LFSR en relación al contador binario natural en cada una de las síntesis variando las etapas desde 10 a 20.

NOTA: la implementación se hizo sobre una Artix-7 de Xilinx XC7A100T-2FGG484.

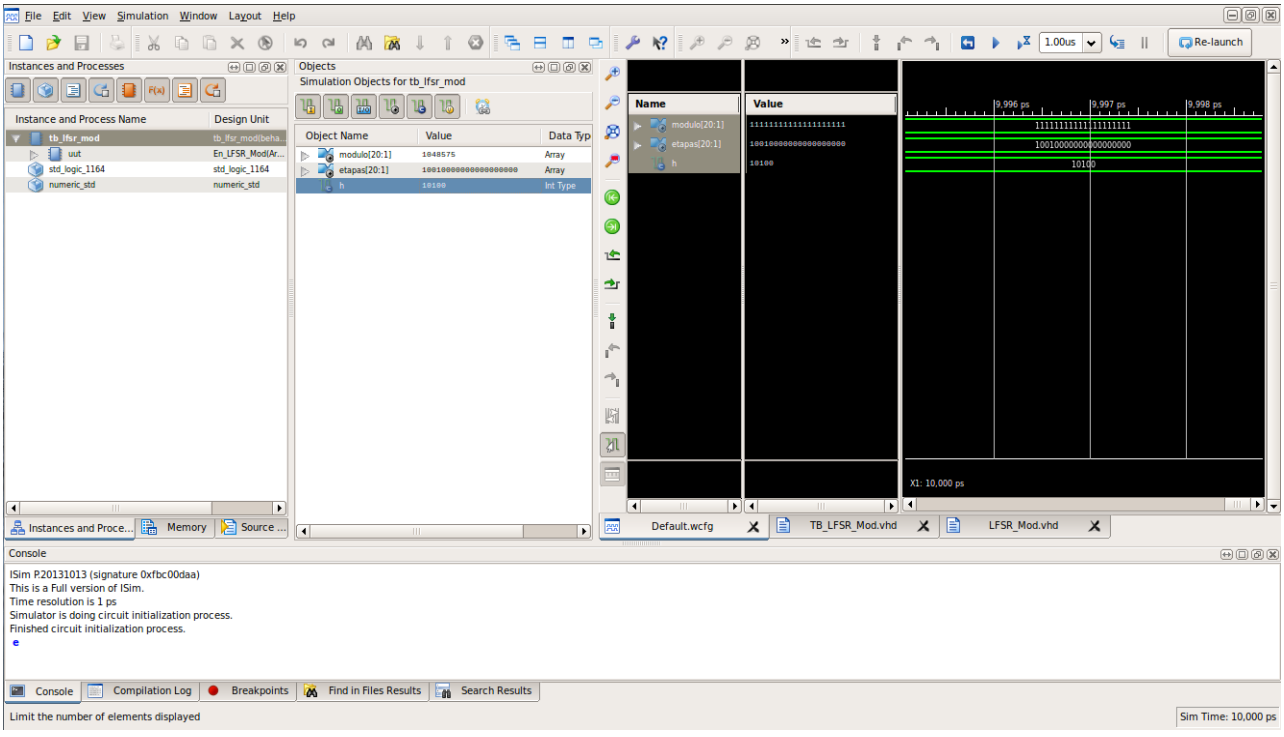
3) En primera instancia se realiza la implementación de una contador de módulo 1.000.000. Para optimizar la resolución se implementa un LFSR. Dado que, con la realimentación adecuada, dicho contador puede llegar a un módulo máximo de $2^N - 1$, siendo N el número de etapas, se toma como premisa que un LFSR de 20 etapas será suficiente (módulo máximo = 1.048.575).

Para lograr esto, se decidió buscar cuál es la mejor realimentación para lograr el módulo máximo del contador y así poder resetarlo cuando el mismo realice 1.000.000 cuentas.

Con la siguiente función, se logró bajo simulación encontrar la óptima combinación de doble realimentación para alcanzar el máximo módulo con un LFSR de 20 etapas.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

use IEEE.NUMERIC_STD.ALL;

entity En_LFSR_Mod is
    Generic ( N : NATURAL := 20 );
    Port (    modulo : out  STD_LOGIC_VECTOR(N downto 1);
              etapas : out  STD_LOGIC_VECTOR(N downto 1));
end En_LFSR_Mod;

architecture Arq_LFSR_Mod of En_LFSR_Mod is

  type salida is array (1 to 2) of STD_LOGIC_VECTOR(N downto 1);

  function LFSR_Mod_2(number_in : integer; type_of_FB : std_logic) return salida is
    variable Q          : STD_LOGIC_VECTOR(N downto 1) := STD_LOGIC_VECTOR(to_unsigned(number_in,N));
    variable modulo     : unsigned(N downto 1)    := to_unsigned(0,N);
    variable modulo_ant : unsigned(N downto 1)    := to_unsigned(1,N);
    variable best_FB    : unsigned(N-1 downto 1)  := to_unsigned(0,N-1);
    variable vSalida    : salida;
  begin

  loop1: for FB in 1 to N-1 loop
    loop2: while Q /= STD_LOGIC_VECTOR(to_unsigned(number_in,N)) or modulo = to_unsigned(0,N) loop
        Q := Q(N - 1 downto 1) & (type_of_FB XOR Q(FB) XOR Q(N));
        modulo := modulo + to_unsigned(1,N);
      end loop loop2;
      if modulo >= modulo_ant then
        modulo_ant := modulo;
        best_FB := to_unsigned(2**(FB-1),N-1);
      end if;
      Q    := STD_LOGIC_VECTOR(to_unsigned(number_in,N));
      modulo := to_unsigned(0,N);
    end loop loop1;

    vSalida(1) := '1' & STD_LOGIC_VECTOR(best_FB);
    vSalida(2) := STD_LOGIC_VECTOR(modulo_ant);
    return vSalida;
  end LFSR_Mod_2;

  signal LFSR_Return : salida;

  begin

  LFSR_Return <= LFSR_Mod_2(48,'1');

  etapas <= LFSR_Return(1);
  modulo <= LFSR_Return(2);

end Arq_LFSR_Mod;
```

Obteniendo la realimentación óptima con la simulación en N = 20.



De esta manera se ve que realimentando las etapas 20 y 17 con una XNOR se logra un módulo máximo de 1.048.575.

Con esto, se prepara el siguiente VHDL, teniendo en consideración las peticiones del inciso

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use work.Common_Types_Pck.all;

entity En_LFSR is
  generic ( N : NATURAL := 20);
  port(
    CLK       : in  STD_LOGIC;
    Reset     : in  STD_LOGIC;
    ocasiones : out STD_LOGIC_VECTOR(10 downto 1)
  );
end En_LFSR;

architecture Arq_LFSR of En_LFSR is
  subtype salida is STD_LOGIC_VECTOR(N downto 1);

  function LFSR ( FB, n_stages, n_counts : integer; init_value : salida; type_of_FB : std_ulogic) return salida is -- type_of_FB = '0' es XOR, '1' es XNOR
      variable Q  : salida := init_value;
  begin
    loop1 : for i in 1 to n_counts loop
        Q := Q(n_stages-1 downto 1) & (type_of_FB XOR Q(FB) XOR Q(n_stages));
      end loop loop1;
    return Q;
  end LFSR;

  signal number_to_find : buscar;
  signal Q : STD_LOGIC_VECTOR(N downto 1);
begin
  number_to_find(0)  <= LFSR(N-3,N,1000000-1,std_logic_vector(to_unsigned(0,N)),'1');
  number_to_find(1)  <= LFSR(N-3,N,47      ,std_logic_vector(to_unsigned(0,N)),'1');
  number_to_find(2)  <= LFSR(N-3,N,2555    ,std_logic_vector(to_unsigned(0,N)),'1');
  number_to_find(3)  <= LFSR(N-3,N,7981    ,std_logic_vector(to_unsigned(0,N)),'1');
  number_to_find(4)  <= LFSR(N-3,N,91466   ,std_logic_vector(to_unsigned(0,N)),'1');
  number_to_find(5)  <= LFSR(N-3,N,18965   ,std_logic_vector(to_unsigned(0,N)),'1');
  number_to_find(6)  <= LFSR(N-3,N,215543  ,std_logic_vector(to_unsigned(0,N)),'1');
  number_to_find(7)  <= LFSR(N-3,N,327333  ,std_logic_vector(to_unsigned(0,N)),'1');
  number_to_find(8)  <= LFSR(N-3,N,510321  ,std_logic_vector(to_unsigned(0,N)),'1');
  number_to_find(9)  <= LFSR(N-3,N,777666  ,std_logic_vector(to_unsigned(0,N)),'1');
  number_to_find(10) <= LFSR(N-3,N,892345  ,std_logic_vector(to_unsigned(0,N)),'1');

  process(CLK)
  begin
    if rising_edge(CLK) then
      if Reset = '1' or Q = number_to_find(0) then
        Q <= (others => '0');
      else
        Q <= Q(N-1 downto 1) & (Q(N-3) XNOR Q(N));
      end if;
    end if;
  end process;

  Ins_Salida: for i in 1 to 10 generate
    ocasiones(i) <= '1' when Q = number_to_find(i) else '0';
  end generate;

end Arq_LFSR;
```

Y simulando el mismo...