Actividad 3.2

Se implementó el problema planteado logrando una sincronización con el reloj de referencia de entrada de 2.048Mbps. De esta manera el CLK de 17.92MHz se recupera en fase con este último sin necesariamente ser de 2.048Mbps exactamente. Se simularon sutiles diferencias en la frecuencia de referencia observando la perfecta adaptatibilidad del sistema. Para el PFD se utilizó la máquina de estados del M4044 (implementado en la Actividad 3.1.3) ya que no solo da referencia de fase sino también de frecuencia, y de esta manera se puede seleccionar entre división por 4 o por 5 para generar en fase los 17.92MHz desde los 72MHz internos.

Para analizar la estabilidad del reloj recuperado, en la simulación se implementó en paralelo un CLK de 17.92MHz, y se le asignó en el TestBench un contador de pulsos a cada uno (Counter_1 y Counter_2) para poder comparar a largo plazo si ambos contaban exactamente lo mismo o se perdían pulsos.

A continuación se presentarán los principales módulos. En este caso se adjuntará el proyecto en caso que se desee analizar en mayor detalle el sistema. Me pareció muy interesante en general, es por eso que me gustaría (dentro de lo posible) recibir un FeedBack sobre esta implementación.

En primer lugar se implementó el divisor de frecuencia racional en base a lo visto en la teoría para minimizar el ruido de fase basado en el Algoritmo de Bresenham.

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

entity En_Div_Bresenham is
    Port (  CLK_In    : in  STD_LOGIC;
            Reset     : in  STD_LOGIC;
            N         : in  STD_LOGIC_VECTOR(7 downto 0);
            Num       : in  STD_LOGIC_VECTOR(7 downto 0);
            Den       : in  STD_LOGIC_VECTOR(7 downto 0);
            Tick_Out : out STD_LOGIC);
end En_Div_Bresenham;

architecture Arq_Div_Bresenham of En_Div_Bresenham is
  signal DownCount      : unsigned(7 downto 0);
  signal Counter        : signed(8 downto 0);
  signal NextAdd        : signed(8 downto 0);
  signal NextCounter    : signed(8 downto 0);
begin

  -- Para un divisor donde DIV = N + Num/Den, sería que cada N o N+1 veces incre

  NextCounter <= Counter + NextAdd;

  process(CLK_In,Reset)
  begin
    if Reset = '1' then
      DownCount <= to_unsigned(0,DownCount'length);
      Counter   <= to_signed(0,Counter'length);
      NextAdd   <= signed('0' & Num);
    elsif rising_edge(CLK_In) then
      Tick_Out <= '0';
      if DownCount = to_unsigned(0,DownCount'length) then
        Tick_Out <= '1';
        Counter    <= NextCounter;
        if NextCounter > to_signed(0,NextCounter'length) then -- Tiene que ser M
          NextAdd   <= signed('0' & Num) - signed('0' & Den);
          DownCount <= unsigned(N);
        else
          NextAdd   <= signed('0' & Num);
          DownCount <= unsigned(N) - to_unsigned(1,DownCount'length);
        end if;
      else
        DownCount <= DownCount - to_unsigned(1,DownCount'length);
      end if;
    end if;
  end process;

end Arq_Div_Bresenham;
```

Donde Fout = Fin*Div, siendo Div = N + Num/Den.

La siguiente captura corresponde a la obtención de Rel2_240, R2048X y Rel17_92.

```
INS_Rel2_240: entity work.En_Div_Bresenham(Arq_DIV_Bresenham)
port map(
  CLK_IN   => sRel17_92,
  Reset    => Reset,
  N        => STD_LOGIC_VECTOR(to_unsigned(8,8)),
  Num      => STD_LOGIC_VECTOR(to_unsigned(0,8)),
  Den      => STD_LOGIC_VECTOR(to_unsigned(0,8)),
  Tick_Out => Rel2_240
  );

INS_Rel2_048: entity work.En_Div_Bresenham(Arq_DIV_Bresenham)
port map(
  CLK_IN   => sRel17_92,
  Reset    => Reset,
  N        => STD_LOGIC_VECTOR(to_unsigned(8,8)),
  Num      => STD_LOGIC_VECTOR(to_unsigned(3,8)),
  Den      => STD_LOGIC_VECTOR(to_unsigned(4,8)),
  Tick_Out => sR2048X
  );

Ins_Rel17_92: entity work.En_Div_Bresenham(Arq_DIV_Bresenham)
port map(
  CLK_IN   => Clk72MHz,
  Reset    => Reset,
  N        => DIV4_5,
  Num      => STD_LOGIC_VECTOR(to_unsigned(0,8)), -- Se desactiva la división racional ya que será manejado por el detector de fase
  Den      => STD_LOGIC_VECTOR(to_unsigned(25,8)),
  Tick_Out => sRel17_92
  );
```

Como puede verse, en la obtención de Rel17_92, N es el que definirá el detector de fase como 4 o 5, y Num = 0 para que actúe en divisor puro usando solamente N sin darle importancia a Den.

DIV4_5 es manejado de la siguiente manera (donde también se observa la obtensión de los flancos ascendentes Rel2048edge).

```
Rel2048edge <= (not sRel2_048) and Rel2_048;

INS_MC4044: entity work.En_MC4044(Arq_MC4044)
  Port map(
  R     => Rel2048edge,
  V     => sR2048X,
  U1    => open,
  D1    => sPFD_Slower,
  State => open
  );

DIV4_5 <=   STD_LOGIC_VECTOR(to_unsigned(5,8)) when sPFD_Slower = '0' else -- sPFD_Slower funciona con lógica negada en el M4044
            STD_LOGIC_VECTOR(to_unsigned(4,8));
```