Universidad de Alcalá

Departamento de Electrónica

Modelado de Sistemas Computacionales

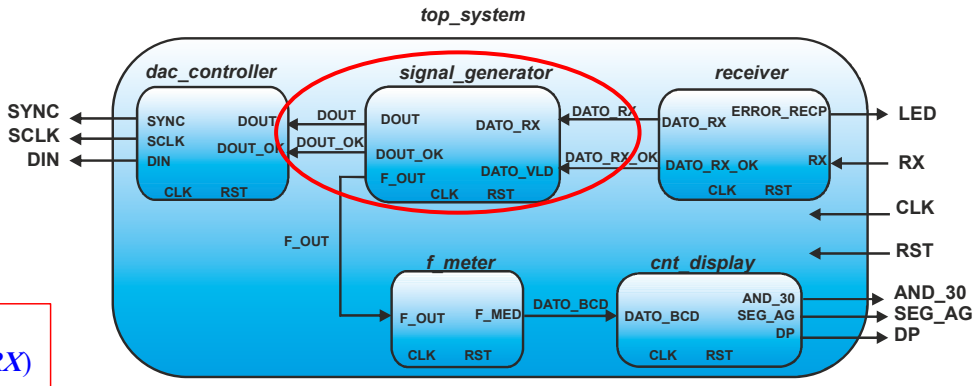
Grado en Ingeniería de Computadores

Práctica libre: Generador de señal controlado desde un puerto serie RS232

Apartado 3.

Entidad *signal_generator*.

- Entidad *signal_generator*.



Se encarga de decodificar el dato recibido por el puerto serie (***DATO_RX***) y generar los datos a enviar al DAC

```
entity signal_generator is
  port ( RST      : in  std_logic;
         CLK      : in  std_logic;
         DATO_RX  : in  std_logic_vector (7 downto 0);
         DOUT     : out std_logic_vector (11 downto 0);
         F_OUT    : out std_logic;
         DOUT_OK  : out std_logic);
end signal_generator;
```

Entidad *signal_generator*

Funcionalidad de los bits del dato recibido.

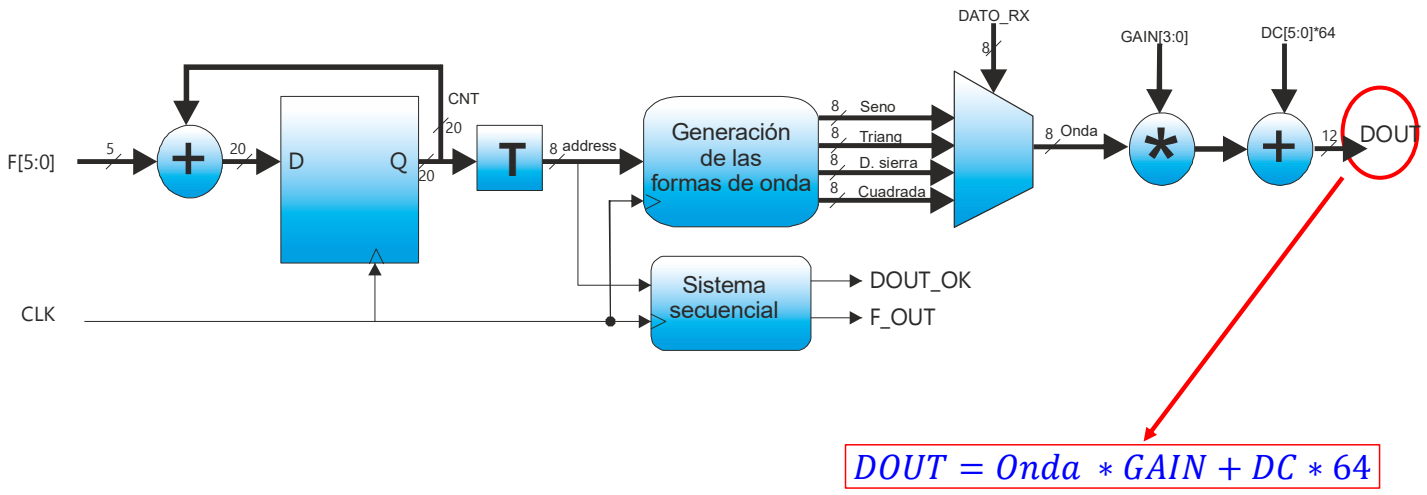
	Bit de *DATO_RX*								
	B_7	**B_6**	**B_5**	**B_4**	**B_3**	**B_2**	**B_1**	**B_0**	**Selección**
VOUT	0	0	X	X	X	1	1	0	Señal sinusoidal
	0	0	X	X	X	1	0	1	Señal triangular
	0	0	X	X	X	1	0	0	Diente de sierra
	0	0	X	X	X	0	1	1	Señal cuadrada D=50%
	0	0	X	X	X	0	1	0	Señal cuadrada D=25%
	0	0	X	X	X	0	0	1	Señal cuadrada D=75%
	0	1	F_5	F_4	F_3	F_2	F_1	F_0	Frecuencia de la señal
	1	0	X	X	G_3	G_2	G_1	G_0	Ganancia
	1	1	DC_5	DC_4	DC_3	DC_2	DC_1	DC_0	Nivel de continua (DC)

Valores por defecto de la señal generada

Tipo de señal	Sinusoidal
Frecuencia	Un valor distinto de cero
Ganancia	3
Nivel de continua (DC)	0

Práctica libre: Generador de señal controlado por el puerto serie RS232

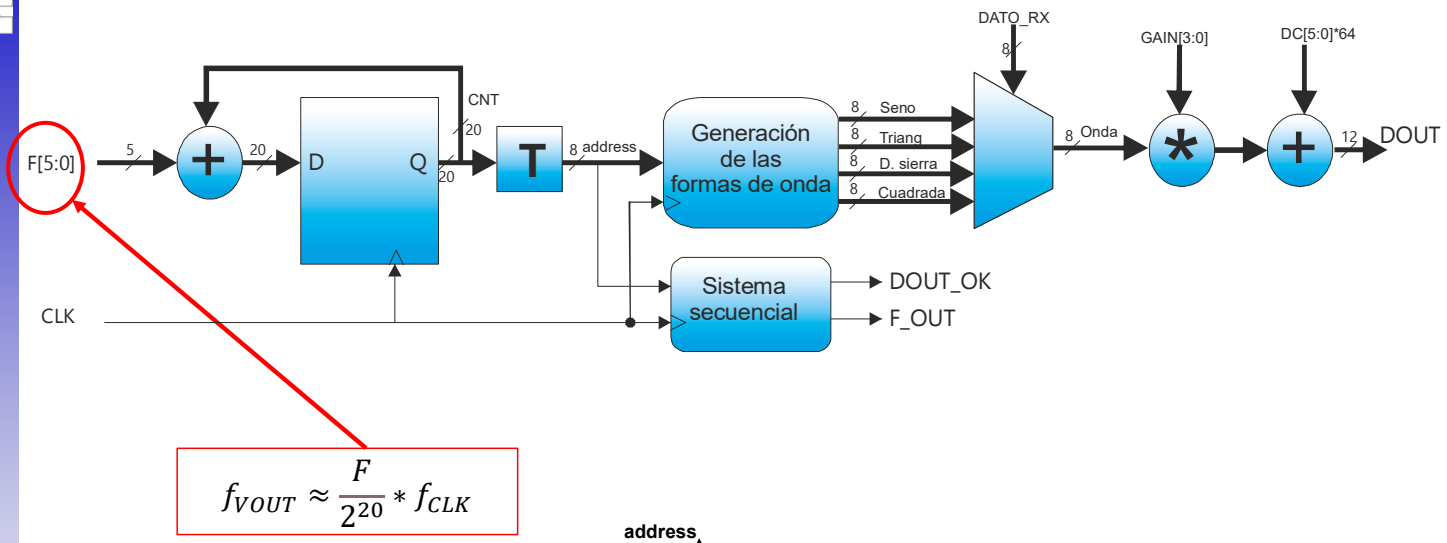
Entidad *signal_generator*



$$DOUT = Onda * GAIN + DC * 64$$

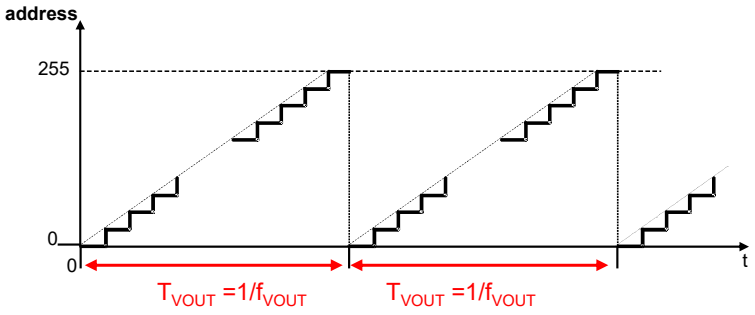
DOUT debe saturar a 2^{12}-1.

Entidad *signal_generator*



$$f_{VOUT} \approx \frac{F}{2^{20}} * f_{CLK}$$

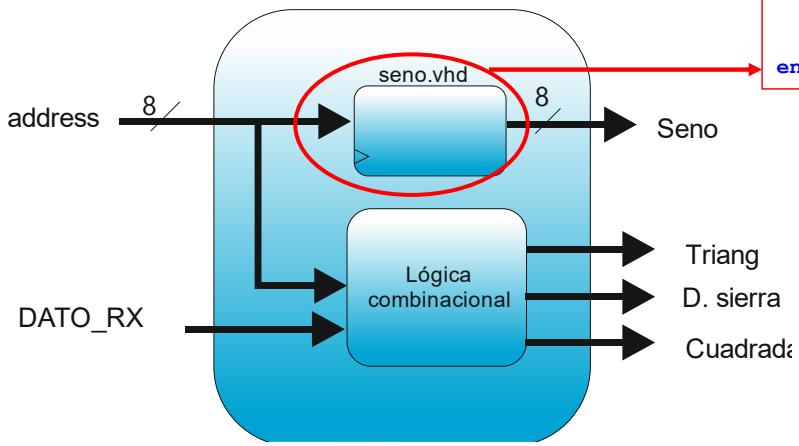
$$95\ Hz \leq f_{VOUT} \leq 6000\ Hz$$



Entidad *signal_generator*

Generación de las formas de onda.

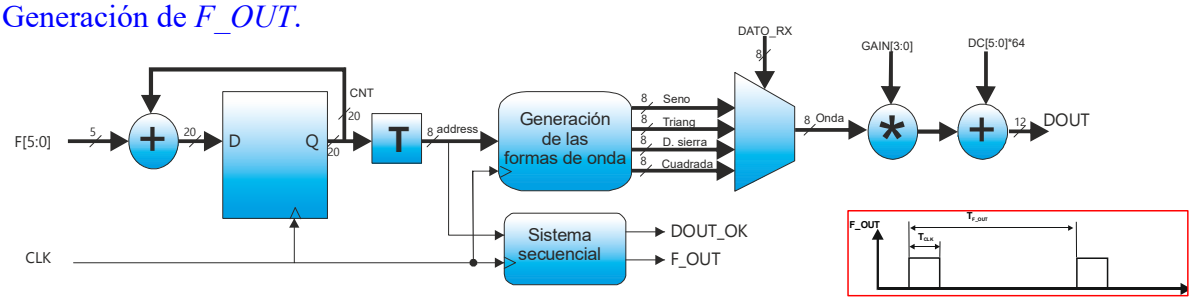
```
entity seno is
  port (
    ADDR : in  std_logic_vector(7 downto 0);
    CLK  : in  std_logic;
    DOUT : out std_logic_vector(7 downto 0));
end seno;
```

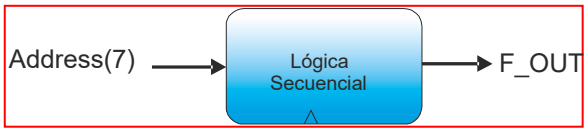


VOUT	Bit de *DATO_RX*								
	B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0	Selección
	0	0	X	X	X	1	1	0	Señal sinusoidal
	0	0	X	X	X	1	0	1	Señal triangular
	0	0	X	X	X	1	0	0	Diente de sierra
	0	0	X	X	X	0	1	1	Señal cuadrada D=50%
	0	0	X	X	X	0	1	0	Señal cuadrada D=25%
	0	0	X	X	X	0	0	1	Señal cuadrada D=75%

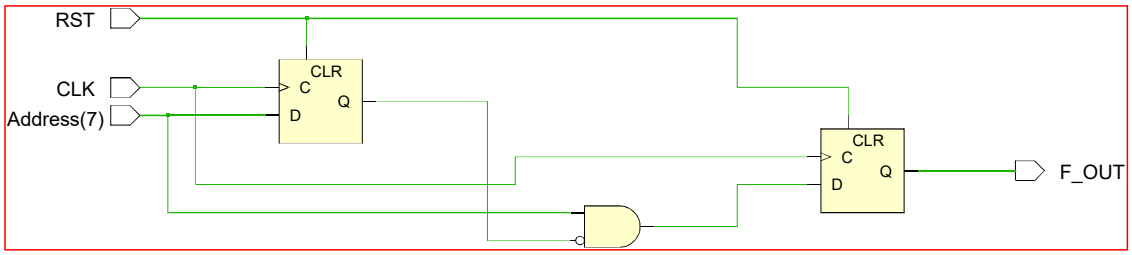
Entidad *signal_generator*

Generación de *F_OUT*.





¡Es un detector de flanco de subida o bajada!

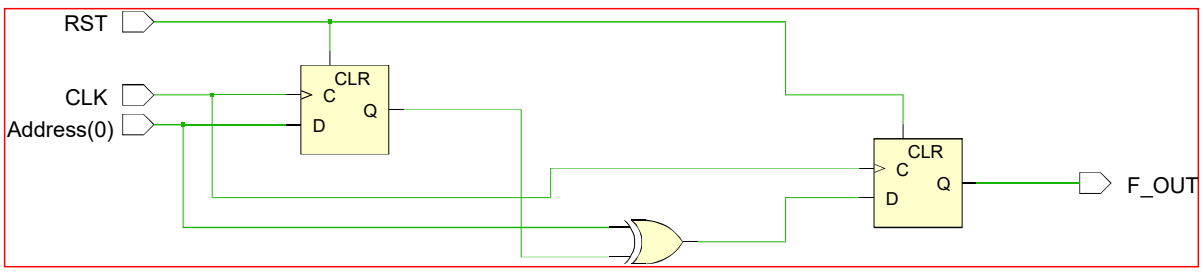


Práctica libre: Generador de señal controlado por el puerto serie RS232

Entidad *signal_generator*

Generación de *DOUT_OK*.

¡Es un detector de flanco de subida y bajada!



Entidad *signal_generator*

❑ Test-bench.

```
entity signal_generator_tb is

end signal_generator_tb;

architecture sim of signal_generator_tb is
. . . . . .
begin  -- sim

  DUT : entity work.signal_generator
    port map (
      RST         => RST_i,
      CLK         => CLK_i,
      DATO_RX     => DATO_RX_i,
      DATO_RX_OK  => DATO_RX_OK_i,
      DOUT        => DOUT_i,
      F_OUT       => F_OUT_i,
      DOUT_OK     => DOUT_OK_i);
. . . . . .

  process
    procedure act_dato_vld is

    begin
      wait until CLK_i = '0';
      DATO_RX_OK_i <= '1';
      wait for 1 us;
      DATO_RX_OK_i <= '0';
      wait for 1 us;
    end act_dato_vld;

  begin
    --ganancia inicial
    DATO_RX_i(7 downto 6)  <= "10";
    DATO_RX_i (5 downto 0) <= "100111";
    wait for 0.3 us;
    --forma de onda inicial
    DATO_RX_i(7 downto 6)  <= "00";
    DATO_RX_i (5 downto 0) <= "001110";
    act_dato_vld;
    wait for 3 ms;


    report "FIN CONTROLADO DE LA SIMULACION" severity failure;
  end process;

  . . . . . .
```

```
process (F_OUT_i)
    variable t_aux : time := 1 ns;
  begin  -- process
    if F_OUT_i'event and F_OUT_i = '1' then
      T <= now- t_aux;
      t_aux := now;
    end if;
  end process;

  F <= 1.0e-3*real((1 sec)/T) when T > 0 ns else 1.0;

end sim;
```


Práctica libre: Generador de señal controlado por el puerto serie RS232

Entidad *signal_generator*