
# Trabajo práctico final

# Circuitos Logicos Programables

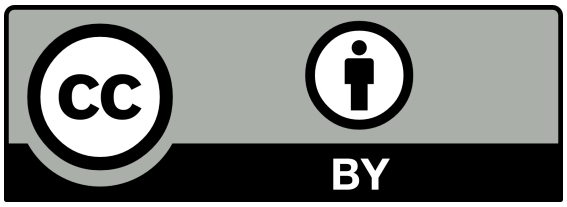
# Generador de Onda

Flavio Miravete

(flavio.miravete@gmail.com)

16/04/2024





# Índice de contenido

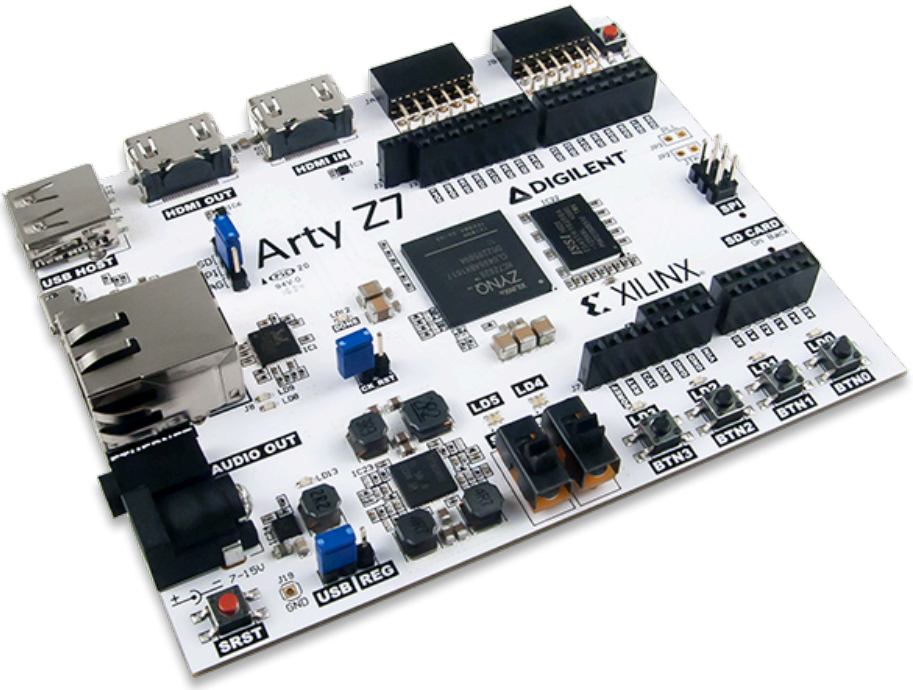
# 1. Introducción

## 1.1. Propósito

Este documento describe el trabajo final desarrollado para la materia Circuitos Logicos Programable del curso de especialización en sistemas embebidos.

## 1.2. Ámbito del Sistema

El trabajo desarrollado se realizó sobre una FPGA de la firma Xilinx. La placa de desarrollo utilizada es la Arty Z7-10 de la empresa Digilent.



# 2. Descripción del trabajo

## 2.1. Oscilador controlado numéricamente (NCO)

Un NCO es un sistema digital que se utiliza para sintetizar señales de frecuencia variable a partir de una base de tiempo fija.

Se compone de un acumulador que recibe como entrada el salto de fase que se acumulará con cada señal de reloj. El acumulador se almacena en un registro de N bits de ancho. El registro al llegar a su valor máximo reinicia en cero.

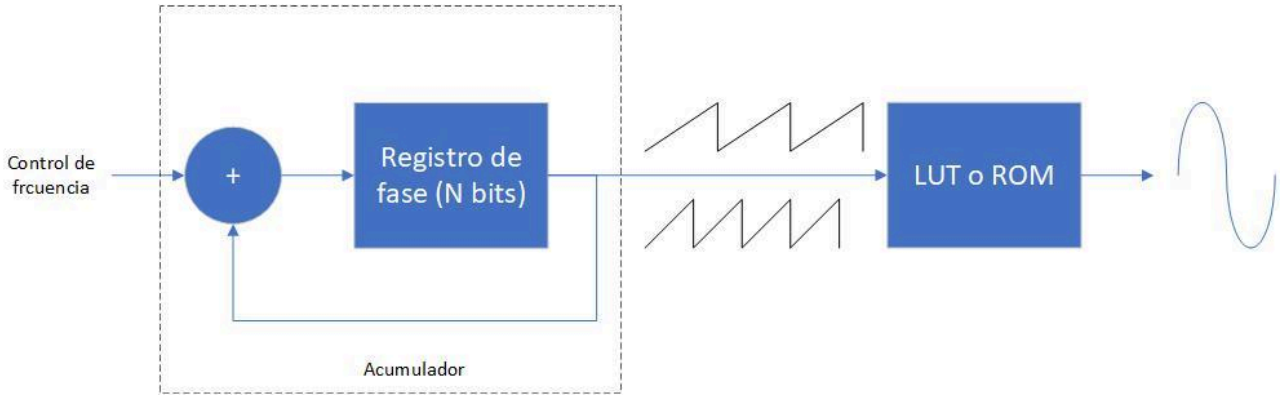



**Figura 1 - Diagrama en bloques del NCO**

La salida del registro de fase se utiliza para acceder a una memoria de 2**N direcciones donde se encuentran almacenados los valores de la forma de onda que se quiere generar.

El período de la señal de salida es igual al tiempo que le lleva al contador alcanzar su cuenta máxima. En este tiempo se tienen que obtener todas las muestras de la señal guardadas en la memoria.

$$Fo = \frac{Fclock \ . \ I}{2^N} [Hz]$$

El valor a la entrada del Registro de Control de Frecuencia, que se suma al valor del

Registro de Fase en cada ciclo de reloj, representa un incremento de fase de:

$$\Delta\Phi = \frac{360°.I}{2^N}$$

$$\Delta f = \frac{1}{Tclock.2^N} = \frac{Fclock}{2^N}[Hz]$$

## 2.2. Implementación

Para la implementación del sistema se definió un bloque llamado waveGenerator, descrito en VHDL. Este bloque contiene el acumulador del NCO e incorpora otro bloque también descrito en VHDL para la memoria ROM que contiene las formas de onda definidas.

El bloque waveGenerator posee la siguiente estructura:

```
22  entity waveGenerator is
23
24  port (
25    i_clk       : in  std_logic;
26    i_rst       : in  std_logic;
27    i_en        : in  std_logic;
28    i_phase     : in  std_logic_vector(11 downto 0); -- NCO incremento de fase
29
30    sin_out     : out std_logic_vector(11 downto 0);
31    cos_out     : out std_logic_vector(11 downto 0);
32    squ_out     : out std_logic_vector(11 downto 0);
33    saw_out     : out std_logic_vector(11 downto 0) );
34
35  end entity;
```

Este bloque genera 4 salidas digitales de 12 bits de ancho:

- Seno
- Coseno
- Cuadrada
- Diente de sierra

Como entradas posee: reloj, habilitación, reset y el incremento de fase.

Su comportamiento es el siguiente:

```
37  architecture waveGenerator_arch of waveGenerator is
38
39      component sincos_rom
40          port (
41              i_clk      : in  std_logic;
42              i_en       : in  std_logic;
43              i_addr     : in  std_logic_vector(11 downto 0);
44              o_sin      : out std_logic_vector(11 downto 0);
45              o_cos      : out std_logic_vector(11 downto 0) );
46      end component;
47
48      signal  nco_out       : std_logic_vector(11 downto 0);
49      signal  rom_addr      : std_logic_vector(11 downto 0);
50
51      begin
52
53      ------------------------------------------------------------------------...
59
60      proc_nco12: process(i_clk, i_rst)
61      begin
62          if i_rst = '0' then
63              nco_out <= (others => '0');
64          elsif i_clk'event and i_clk = '1' then
65              if i_en = '1' then
66                  nco_out <= unsigned(nco_out) + unsigned(i_phase);
67              end if;
68          end if;
69      end process proc_nco12;
70
71      rom_addr <= nco_out;
72
73      ------------------------------------------------------------------------...
77
78      rom: sincos_rom
79
80        port map (
81          i_clk   => i_clk,
82          i_en    => i_en,
83          i_addr  => rom_addr,
84          o_sin   => sin_out,
85          o_cos   => cos_out );
86      ------------------------------------------------------------------------...
89
90      squ_out <= "011111111111" when rom_addr(11) = '1' else "100000000000";
91
92      ------------------------------------------------------------------------...
95
96      saw_out <= rom_addr;
97
98  end waveGenerator_arch;
```

El bloque sincos_rom posee la siguiente estructura:

```
21  entity sincos_rom is
22
23  port (
24    i_clk      : in  std_logic;
25    i_en       : in  std_logic;
26    i_addr     : in  std_logic_vector(11 downto 0);
27    o_sin      : out std_logic_vector(11 downto 0);
28    o_cos      : out std_logic_vector(11 downto 0)
29    );
30
31  end entity;
```

Y su comportamiento es el siguiente:

```
33  architecture sincos_rom_arch of sincos_rom is
34
35  type rom_type is array (0 to 4095) of std_logic_vector (11 downto 0);
36
37  constant SIN_ROM : rom_type :=
38  (
39  X"000", X"003", X"006", X"009", X"00d", X"010", X"013", X"016",
40  X"019", X"01c", X"01f", X"023", X"026", X"029", X"02c", X"02f",
```

...

```
553   constant COS_ROM : rom_type :=
554   (
555   X"7ff", X"7ff", X"7ff", X"7ff", X"7ff", X"7ff", X"7ff", X"7ff",
556   X"7ff", X"7ff", X"7ff", X"7ff", X"7ff", X"7ff", X"7ff", X"7fe",
```

...

```
1069  begin
1070
1071  rom_select: process (i_clk)
1072  begin
1073    if i_clk'event and i_clk = '1' then
1074      if i_en = '1' then
1075        o_sin <= SIN_ROM(conv_integer(i_addr));
1076        o_cos <= COS_ROM(conv_integer(i_addr));
1077      end if;
1078    end if;
1079  end process rom_select;
1080
1081  end sincos_rom_arch;
```

Los diagramas esquemáticos obtenidos a través de la herramienta VIVADO son:

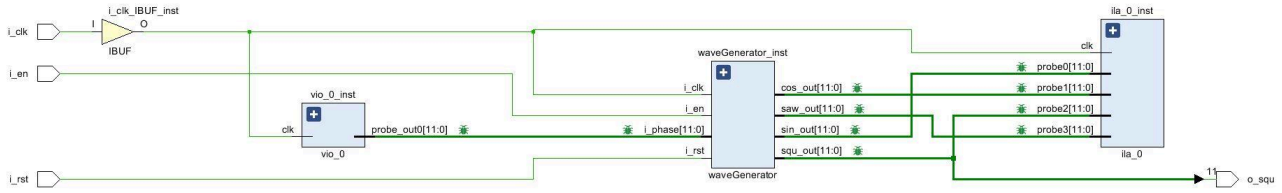



**Figura 2 - Diagrama en bloque general (incluye bloques ILA y VIO)**

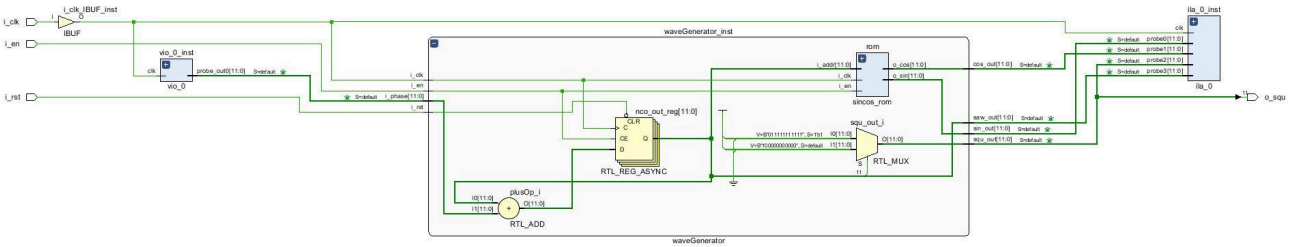



**Figura 3 - Detalle Bloque waveGenerator**

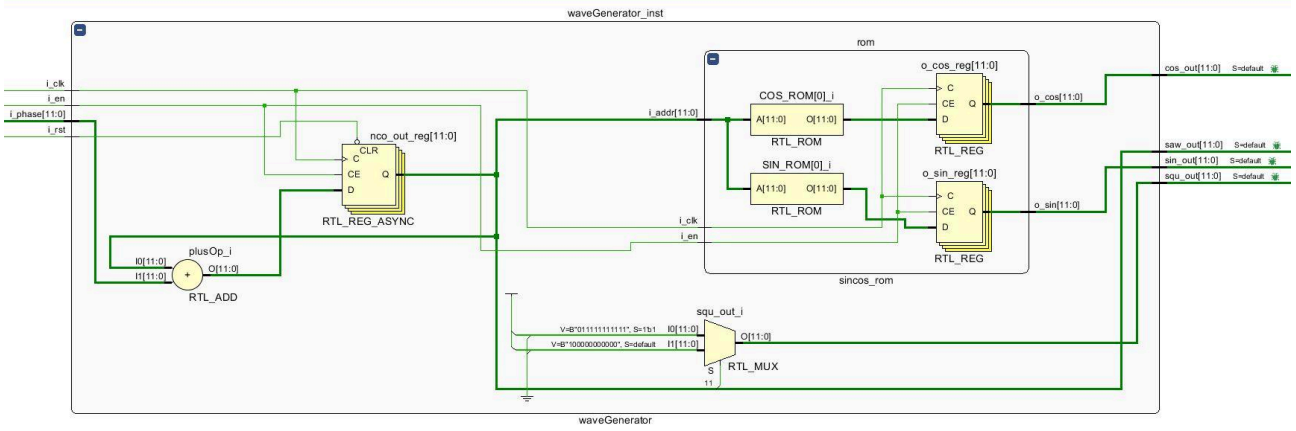



**Figura 4 - Detalle Bloque sincos_rom**

## 2.3. Pruebas y simulaciones

Para poder probar la implementación en la placa FPGA se aprovechó el uso de 2 bloques IP propios de la herramienta VIVADO (ver figura 2).

El bloque VIO que se utiliza para virtualizar entradas y salidas fue configurado para poder ingresar el valor de incremento de fase al NCO.

El bloque ILA que se utiliza para visualizar en tiempo real valores de salidas a traves del disparo de capturas de muestras. En este caso se conectaron las 4 salidas del bloque waveGenerator para poder visualizar los valores de cada una de las formas de onda.

Adicionalmente, en el archivo de restricciones se configuraron 3 entradas y una salida según el siguiente detalle:

- señal de clock: 125 Mhz
- Señal de reset al switch 0 (SW0)
- Señal de habilitación al switch 1 (SW1)
- Señal Cuadrada al pin 1 del conector Ja (pmod 2x6)

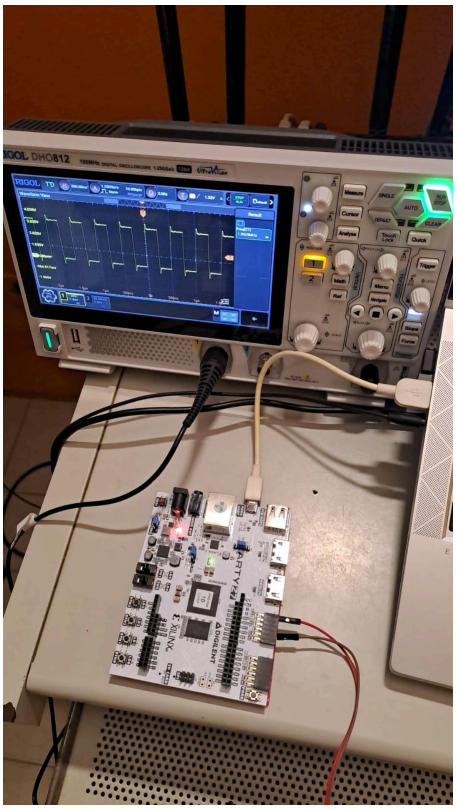


**Figura 5 - Foto del sistema**

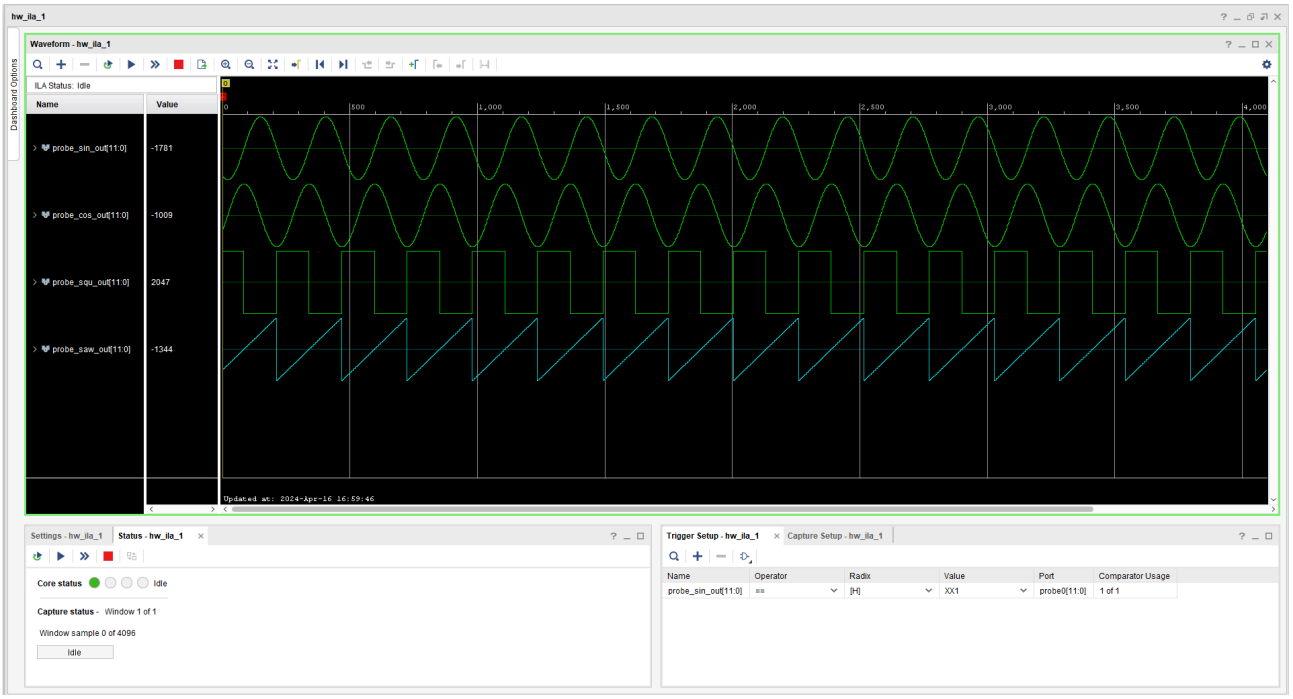



**Figura 6 - Captura ILA de las formas de onda**

## 2.4. Recursos usados en la FPGA

La herramienta VIVADO dispone de información referida a la utilización de los distintos recursos que contiene el chip FPGA.

**Utilization** Post-Synthesis | **Post-Implementation**

Graph | **Table**

| Resource | Utilization | Available | Utilization % |
|---|---|---|---|
| LUT | 1356 | 17600 | 7.70 |
| LUTRAM | 139 | 6000 | 2.32 |
| FF | 2407 | 35200 | 6.84 |
| BRAM | 8.50 | 60 | 14.17 |
| IO | 4 | 100 | 4.00 |
| BUFG | 2 | 32 | 6.25 |