.UBAfiuba
FACULTAD DE INGENIERÍA

Presentación del trabajo final del curso de Circuitos Lógicos Programables

Oscilador senoidal controlado numéricamente (NCO senoidal)

Autor: Ing. Guillermo F. Caporaletti
Docente: Ing. Nicolás Álvarez

Junio de 2023.

Descripción general

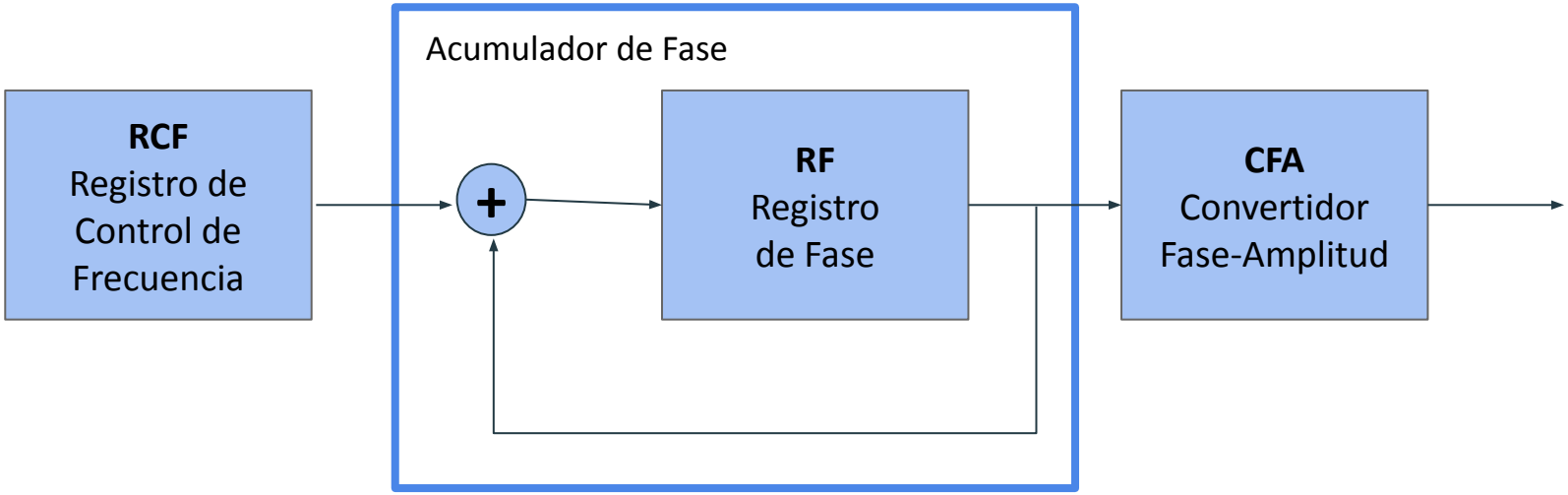
1. Un NCO senoidal implementa una señal senoidal a partir de una base de tiempo y componentes digitales.

2. Frente a los circuitos analógicos, uno de los objetivos de esta forma de implementación es evitar la necesidad de calibración debido a los cambios en los componentes producidos con el paso del tiempo.

3. Por otra parte, puede permitir un amplio rango de frecuencias sin sumar excesiva complejidad.

4. Además facilita el control remoto del sistema.

Diagrama básico



Contador Módulo N

Tabla de Búsqueda (TdB)
o *Lookup Table* (LUT)

Fuente: Presentación de Sistemas Digitales (66.17), FIUBA.

Ecuaciones básicas

N	Bits del contador
2^N	Cantidad de muestras por período
$T_{Reloj} = \frac{1}{F_{Reloj}}$	Período del reloj o base de tiempo del sistema [seg]
$T_0 = T_{Reloj} * 2^N$	Período de la señal de salida [seg]
$F_0 = \frac{F_{Reloj}}{2^N}$	Frecuencia de la señal de salida [Hz]

Variando la frecuencia

- Para variar la frecuencia introducimos un parámetro *I* para variar el incremento del contador.
- Las ecuaciones quedan:

$$T_0 = \frac{T_{Reloj} * 2^N}{I}$$

Período de la señal de salida [seg]

$$F_0 = \frac{F_{Reloj} * I}{2^N}$$

Frecuencia de la señal de salida [Hz]

- Nótese que la frecuencia es proporcional al incremento *I*.
- El incremento *I* puede ser una fracción.

Resolución y rango de frecuencia

- Tomaremos en nuestro caso $N = 10$ (que equivale a 1024 muestras) y $F_0 = 125$ MHz (frecuencia de reloj de nuestro FPGA).

- Según esto, para $l = 1$ nuestra frecuencia sería $F_0 = 122{,}1$ kHz. Ésta sería nuestra resolución en frecuencia y nuestra frecuencia mínima si tomamos l como entero.

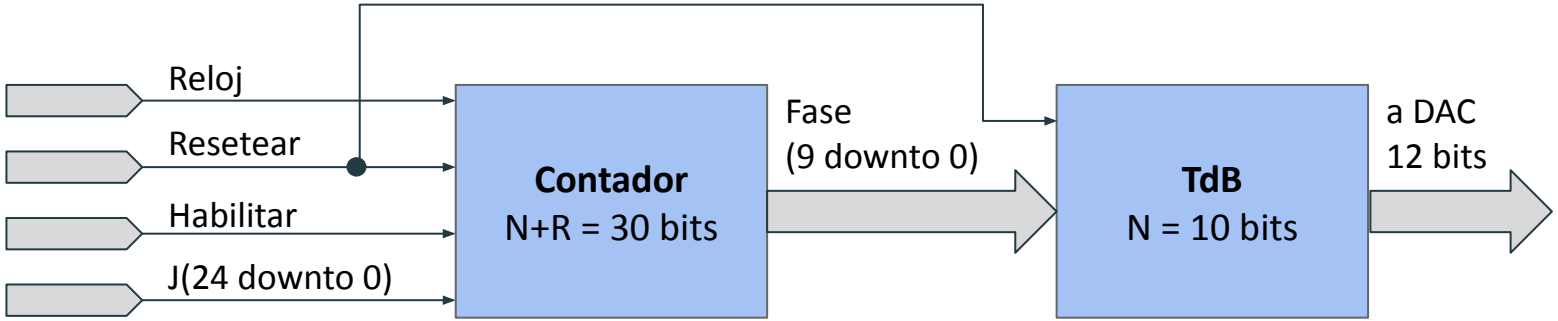
- La frecuencia máxima depende de cuánto se pueda degradar la señal achicando la cantidad de muestras. Tomaremos un valor máximo de $l_{MAX} = 17$. Esto equivale a una señal senoidal con 60 muestras en nuestro caso; y una frecuencia máxima de $F_0 = 2{,}075$ MHz.

Aumentando resolución

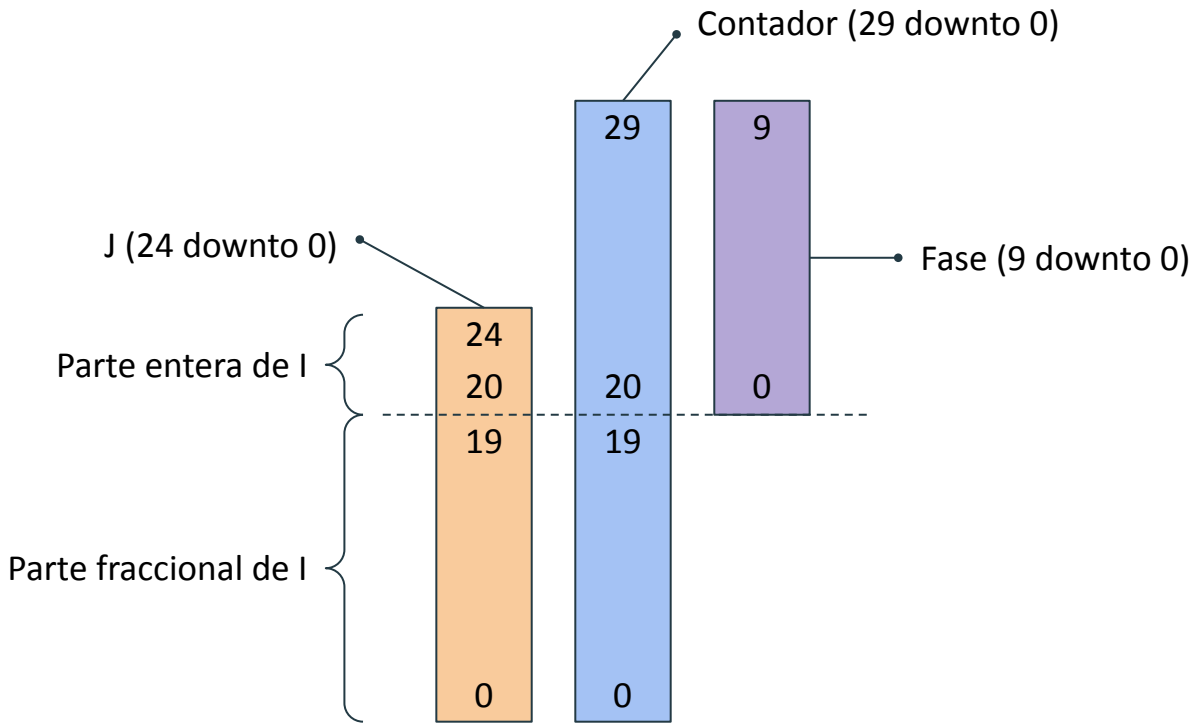
- Agregaremos *R* = 20 bits para la parte fraccional de *I*.
- Con esto la frecuencia mínima y resolución quedará establecida en:

$$F_{MIN} = \frac{F_{Reloj} * I_{MIN}}{2^N} = \frac{F_{Reloj}}{2^N * 2^R} = \frac{F_{Reloj}}{2^{N+R}} = 0,1164\ Hz$$

Implementación



Correspondencia de bits



Implementación del CFA

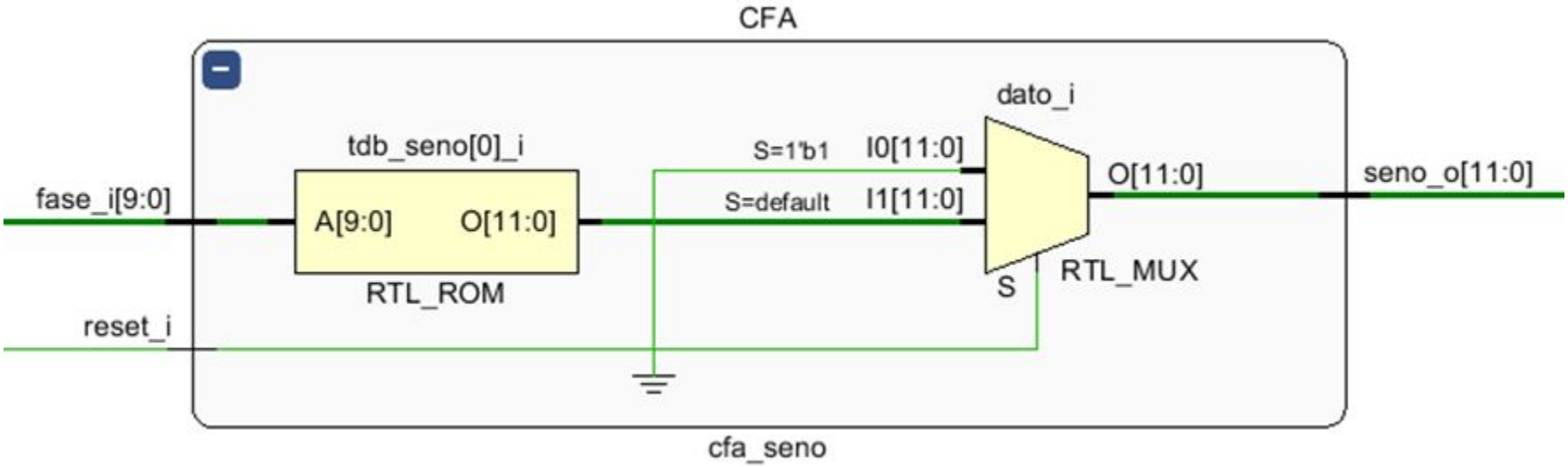
```
entity cfa_seno is
    generic (
        Q_BITS_DATO     :  positive := 12;       -- Número de bits de cada dato-valor de salida (de 1 a 12)
        N_BITS_MUESTRAS :  positive := 10        -- Bits necesario para las N muestras de señal por período
                                                 -- 2^N_BITS_MUESTRAS debe ser >= 1024
    );
    port (
        reset_i  : in  std_logic;
        fase_i   : in  std_logic_vector(N_BITS_MUESTRAS - 1 downto 0);
        seno_o   : out std_logic_vector(Q_BITS_DATO - 1 downto 0)
    );
end entity cfa_seno;
```

- Constante que define el mapeo de la señal senoidal en memoria:

```
type tdb type is array (0 to N MUESTRAS - 1) of std_logic_vector(Q_BITS_TDB - 1 downto 0);
constant tdb seno    : tdb type := (
      "011111111111", "100000001100", "100000011000", "100000100101", "100000110001", "100000111110", "100001001010", "100001010111",
      "100001100011", "100001110000", "100001111101", "100010001001", "100010010110", "100010100010", "100010101111", "100010111011",
      "100011001000", "100011010100", "100011100001", "100011101101", "100011111010", "100100000110", "100100010011", "100100011111",
      "100100101011", "100100111000", "100101000100", "100101010001", "100101011101", "100101101001", "100101110110", "100110000010",
      "100110001110", "100110011011", "100110100111", "100110110011", "100111000000", "100111001100", "100111011000", "100111100100",
      "100111110001", "100111111101", "101000001001", "101000010101", "101000100001", "101000101101", "101000111001", "101001000101",
      "101001010001", "101001011101", "101001101001", "101001110101", "101010000001", "101010001101", "101010011001", "101010100101",
```

- Constante que define el mapeo de la señal senoidal en memoria:

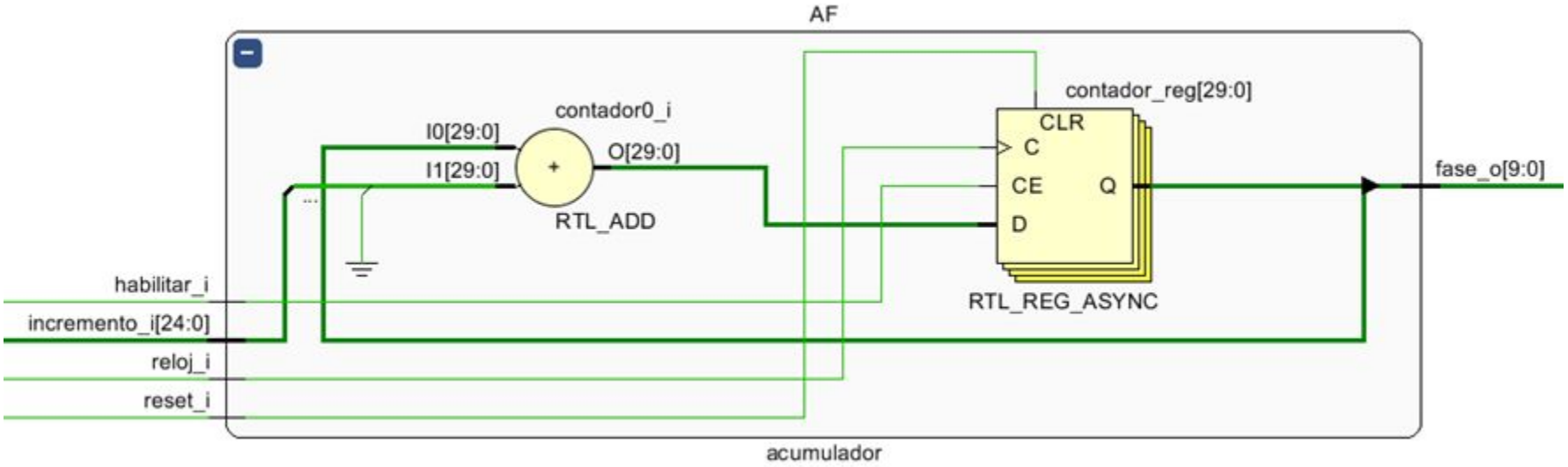
```
begin
    process (fase_i, reset_i)
    begin
        if reset i = '1' then
            dato <= (others => '0');
        else
            dato <= tdb_seno(to_integer(unsigned(fase_i)));
        end if;
    end process;
    seno_o <= dato(Q_BITS_TDB-1 downto Q_BITS_TDB-Q_BITS_DATO);
end architecture cfa_seno_arq;
```



Esquemático RTL del CFA

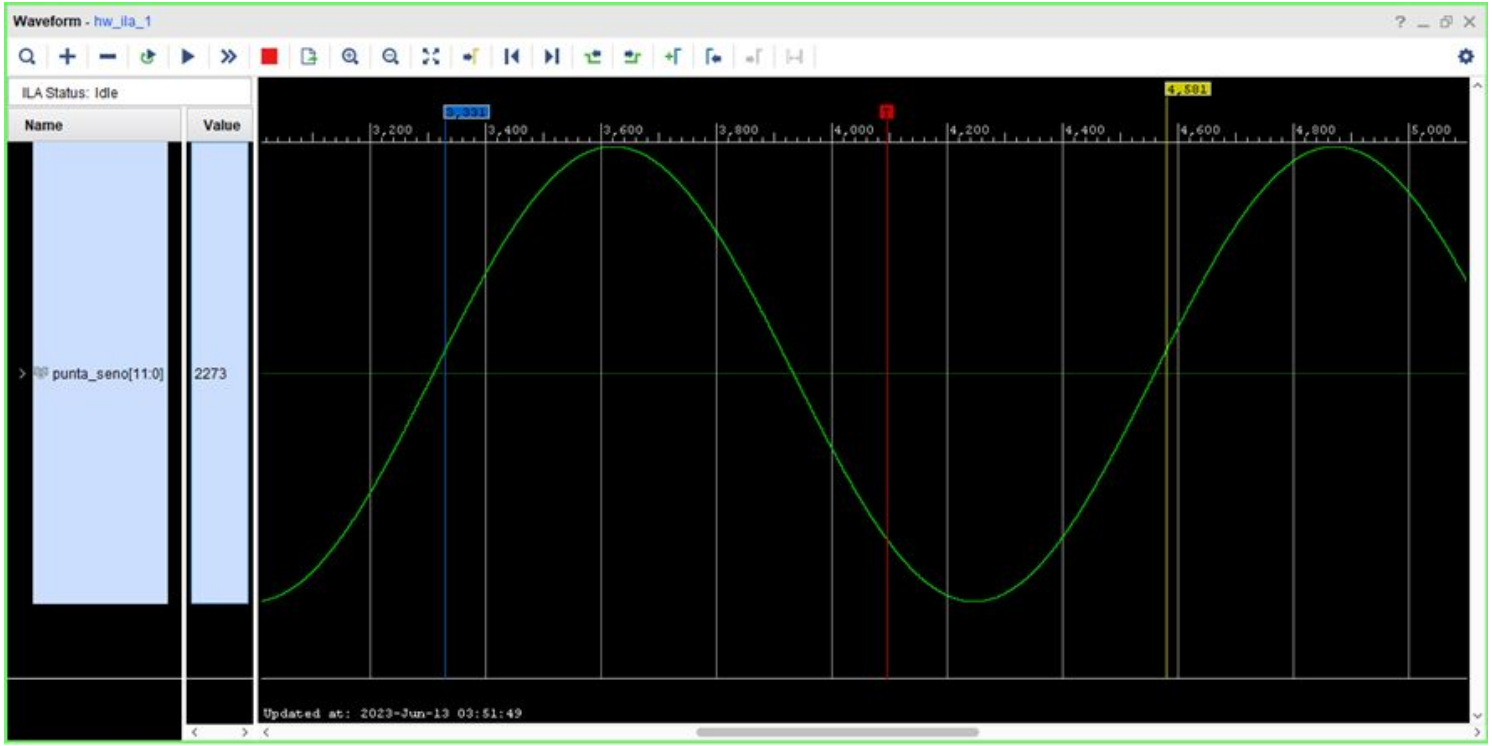
Implementación del AF

```
process (reloj_i, reset_i, incremento_i)
begin
    if reset_i = '1' then                -- El reset es asincrónico
        contador <= (others => '0');

    elsif rising_edge(reloj_i) then      -- El reloj es sincrónico
        if habilitar_i = '1' then        -- Habilitación para contar
            contador <= contador + (ceros &  unsigned(incremento_i));
            -- Si contador desborda, vuelve a contar desde el inicio.
        end if;

    end if;
end process;
```

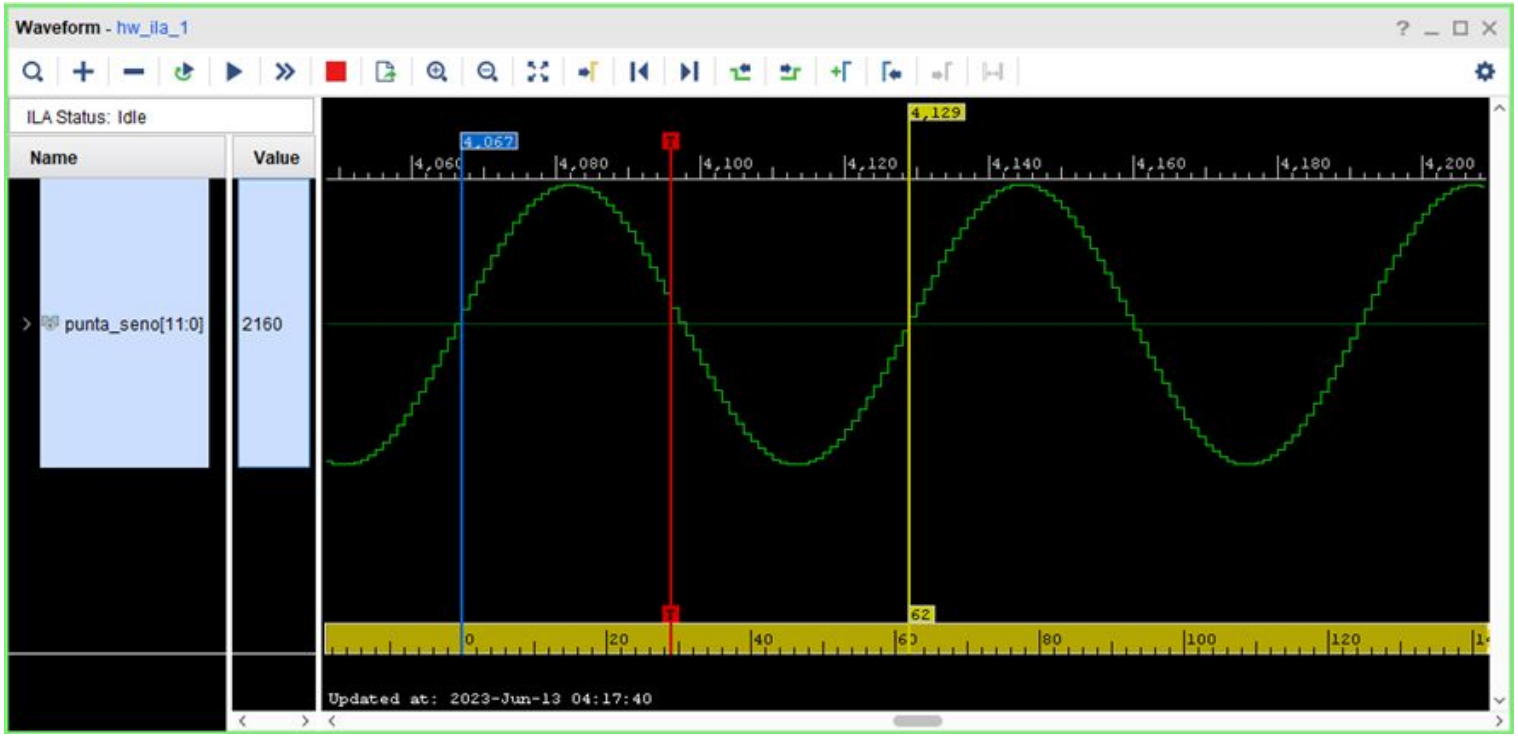


Esquemático RTL del AF

Prueba práctica



Prueba práctica: 2 MHz



¡Gracias!