Trabajo Práctico Final Circuitos logicos programables

Abraham Rodriguez

Octubre, 2024

.UBAfiuba
FACULTAD DE INGENIERÍA

Índice

Funcionalidad e Implementación

En esta sección se proporciona una breve explicación del NCO implementado, así como diagramas y bloques de código relevantes.

Repositorio

El trabajo realizado se encuentra en GitHub: `https://github.com/AbeRodz/Numeric-Controlled-Oscillator`

Numeric Controlled Oscillator (NCO)

Un NCO consiste de un acumulador de fase y convertidor de fase de amplitud. Diagrama interno:

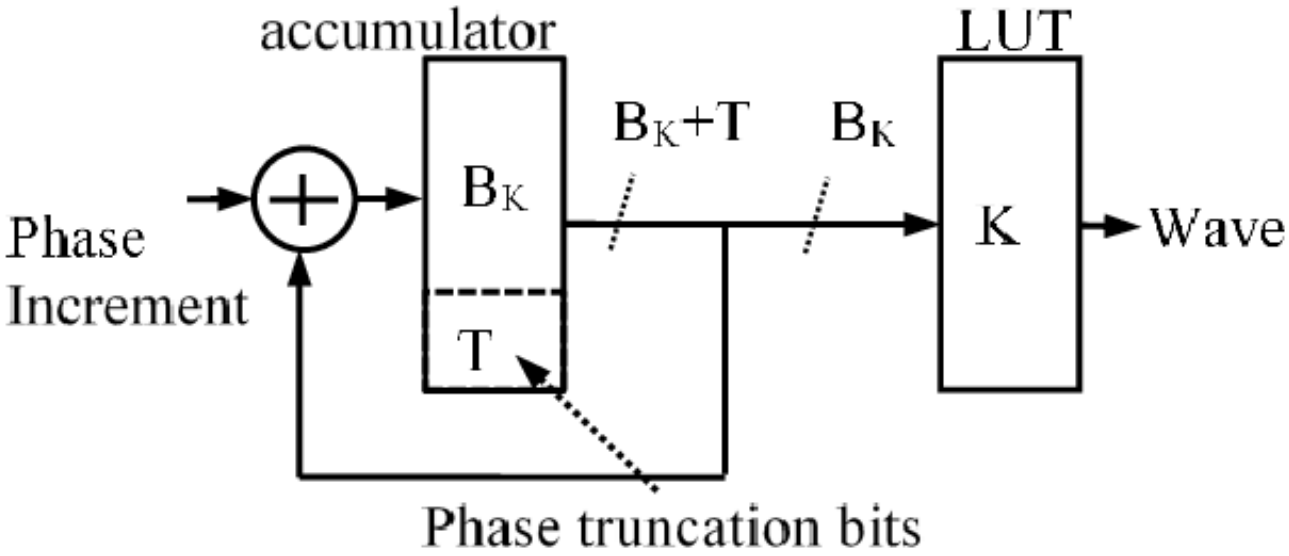


Figura: Diagrama interno NCO.

Métodos de Generación

- CORDIC
- Aproximación de Taylor
- Aproximación lineal
- Direct Digital Synthesis (DDS)

Tipos de Onda

Tipos de onda generados:

- Seno
- Cuadrada
- Triangular
- Sierra

Acumulador

El acumulador aplica la operación de suma de la fase o *Frequency Tuning/Control Word* actual con el estado anterior. Código VHDL simplificado:

```
process (c_i)
begin
    if rising_edge(c_i) then
        if en_i = '1' then
        phase_acc <= phase_acc + unsigned(freq_word_i);
        addr <= std_logic_vector(phase_acc(31 downto 22));
        end if;
    end if;
end process;
```

SineLUT

Para generar los valores de la función seno, se realizó un script de Python:

```
def generate_uint16_sine_table(
num_samples: int = 1024,
amplitude: int = 32767,
offset:int = 32767) -> tuple[np.ndarray]:
    num_values = num_samples
    x = np.linspace(0, 2 * np.pi, num_values, endpoint=False)
    y = np.round(amplitude * np.sin(x) + offset).astype(int)
    return x,y
```

Diagrama de Muestra de Seno

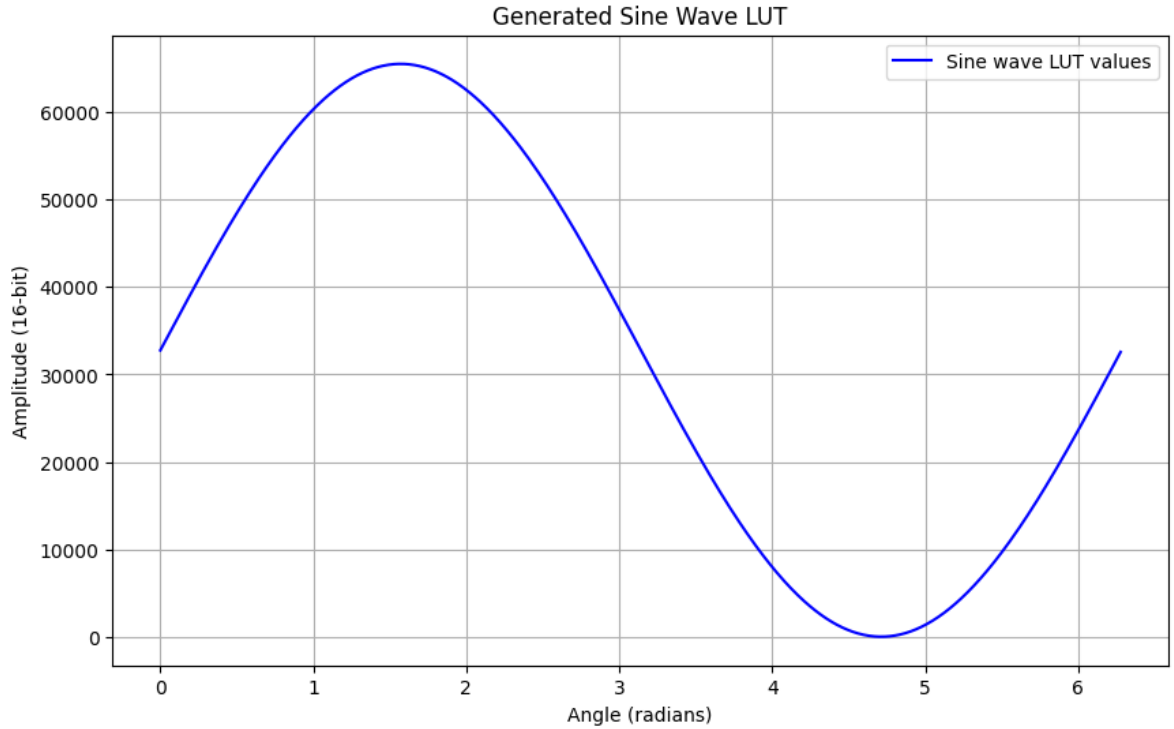


Figura: Muestra de la función seno.

NCO Implementada

La NCO consiste en una instancia de sineLUT y una señal como acumulador:

```
begin
    if rising_edge(c_i) then
        if en_i = '1' then
            phase_acc <= phase_acc + unsigned(freq_word_i);
            addr <= std_logic_vector(phase_acc(31 downto 22));
            ...
        end if;
    end if;
end process;
```

Diagrama de Bloques

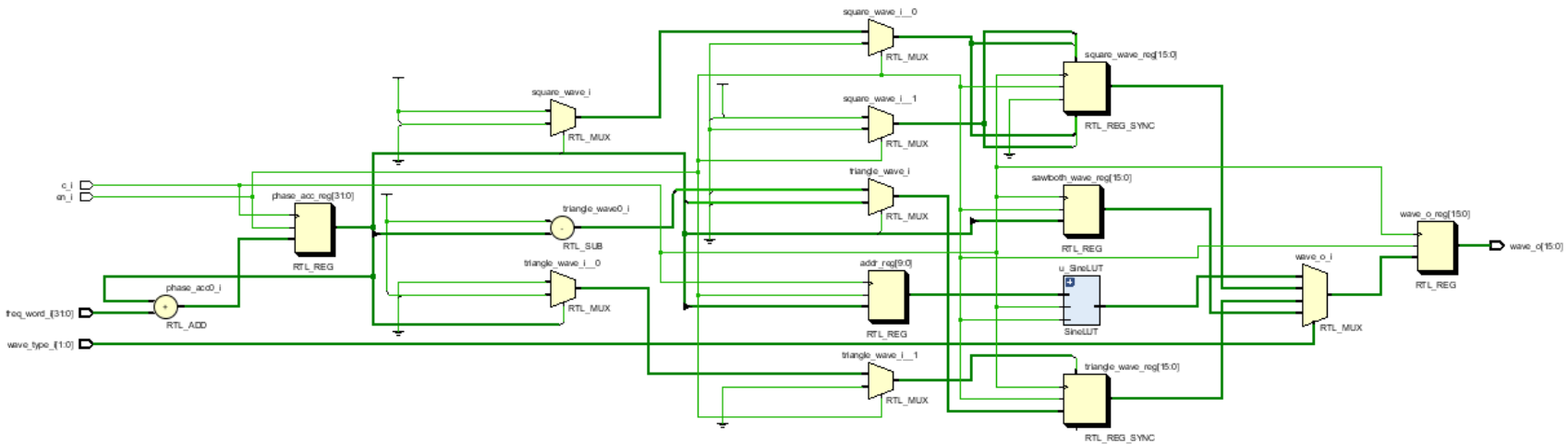
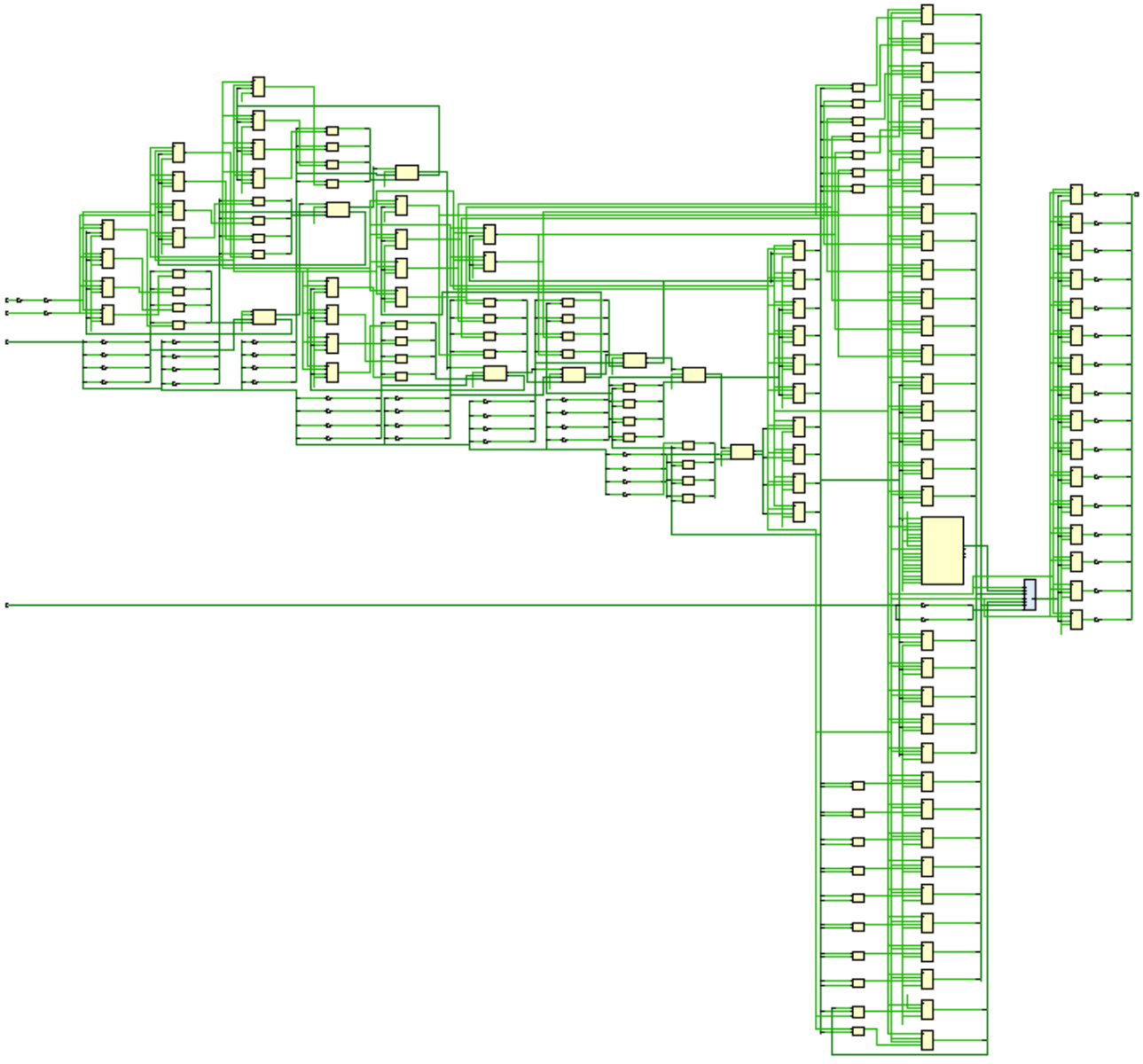


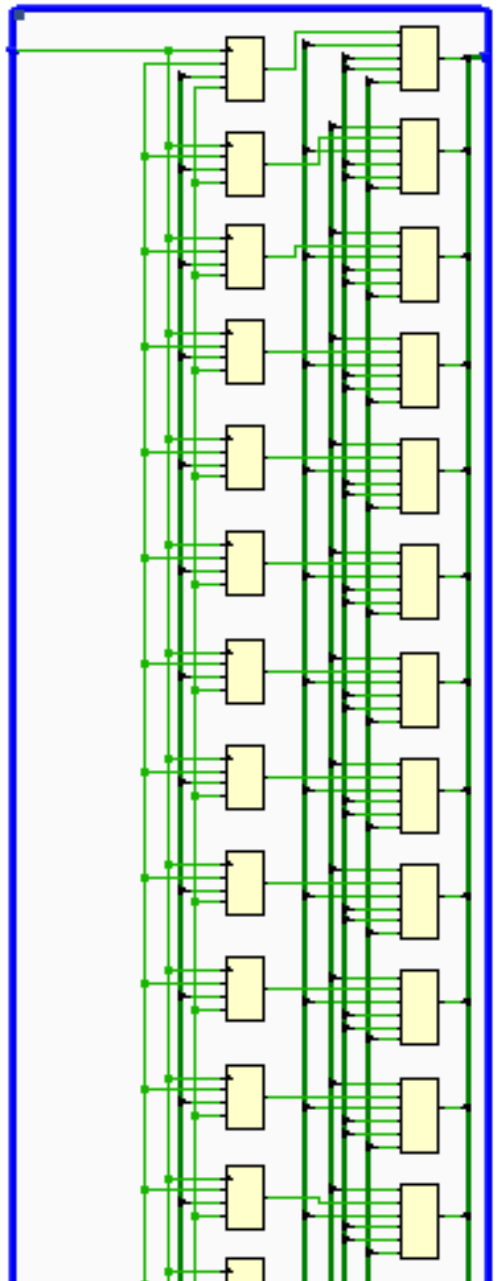
Figura: Diagrama de bloques NCO.

Síntesis

Se generaron los siguientes diagramas para la NCO y sineLUT:



Síntesis



Simulaciones

Se simularon distintas formas de onda:

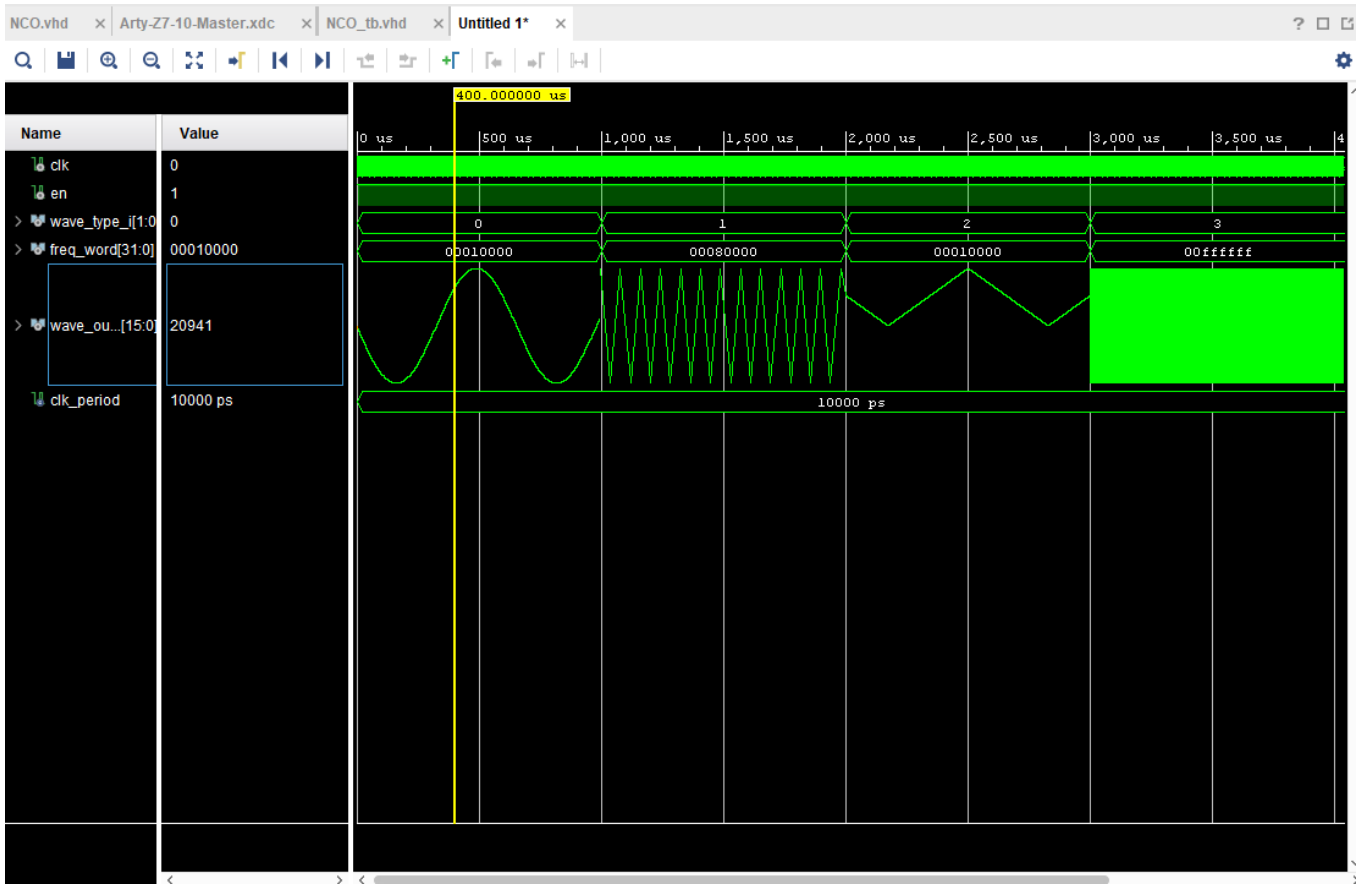


Figura: Simulaciones.

Testbench

El testbench cuenta con estímulos para generar ondas a distintas frecuencias:

```
-- Stimulus process to apply different test cases
stimulus : process
begin
    wave_type_i <= "00"; -- Sine wave
    freq_word   <= x"00010000"; -- Low frequency
    wait for 1000 us;
    ...
end process;
```

Tabla de Recursos

La tabla de recursos generada por Vivado denota una alta utilización de IO:

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

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	59	17600	0.34
FF	97	35200	0.28
BRAM	0.50	60	0.83
IO	52	100	52.00
BUFG	1	32	3.13

Figura: Tabla de recursos.

Pruebas con FPGA remota

Se configuro el VIO y la ILA y se creo el siguiente diagrama

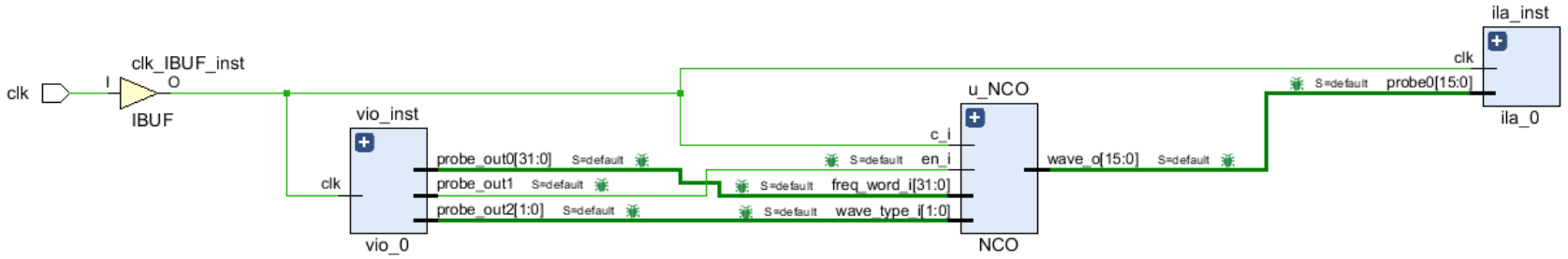


Figura: VIO e ILA.

Se genero el bitstream y se realizaron pruebas con la FPGA remota mediante el server lse, se probo generando distintas ondas a diferentes frecuencias:

Pruebas con FPGA remota

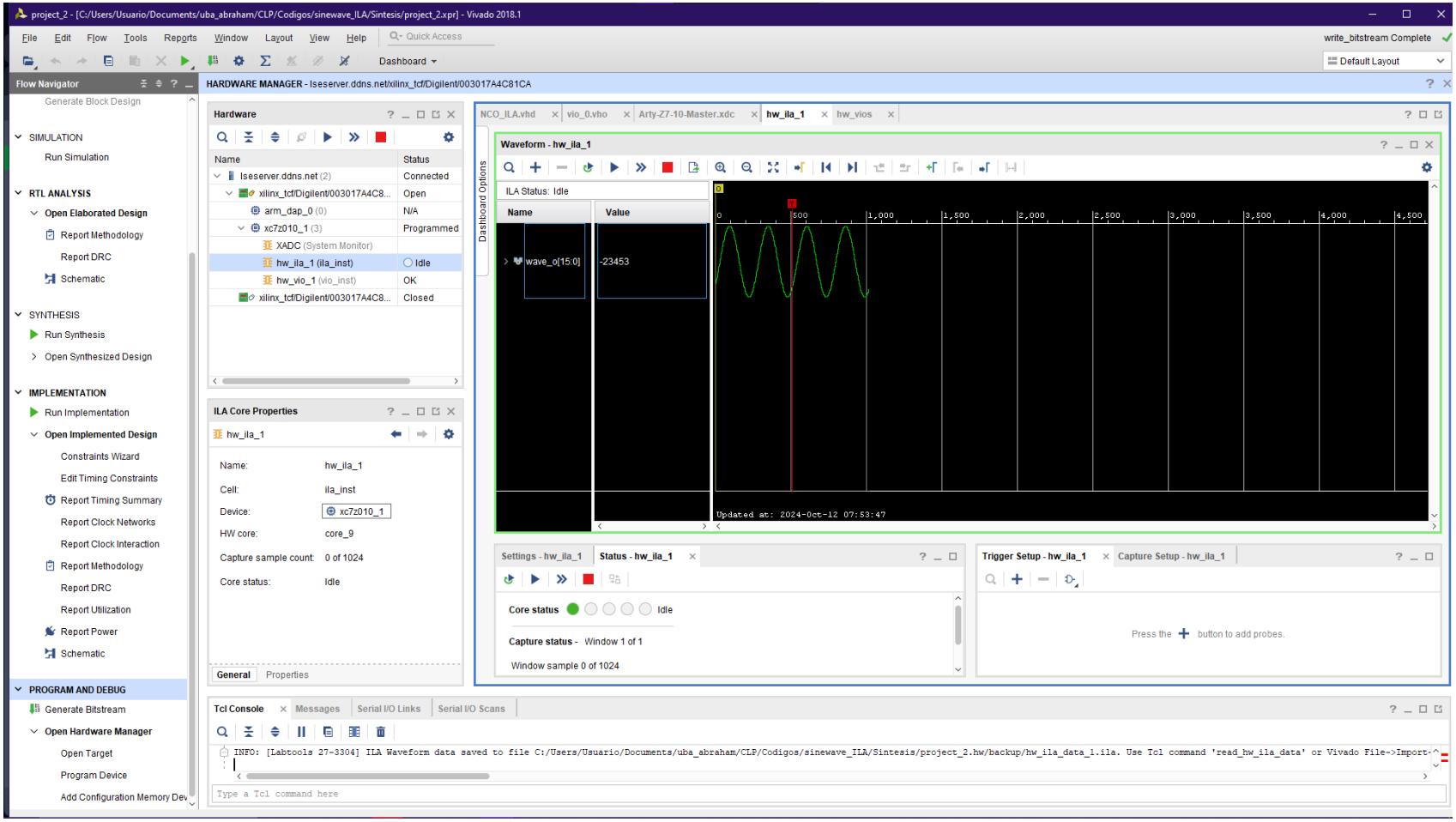


Figura: Seno en FPGA remota.

Pruebas con FPGA remota

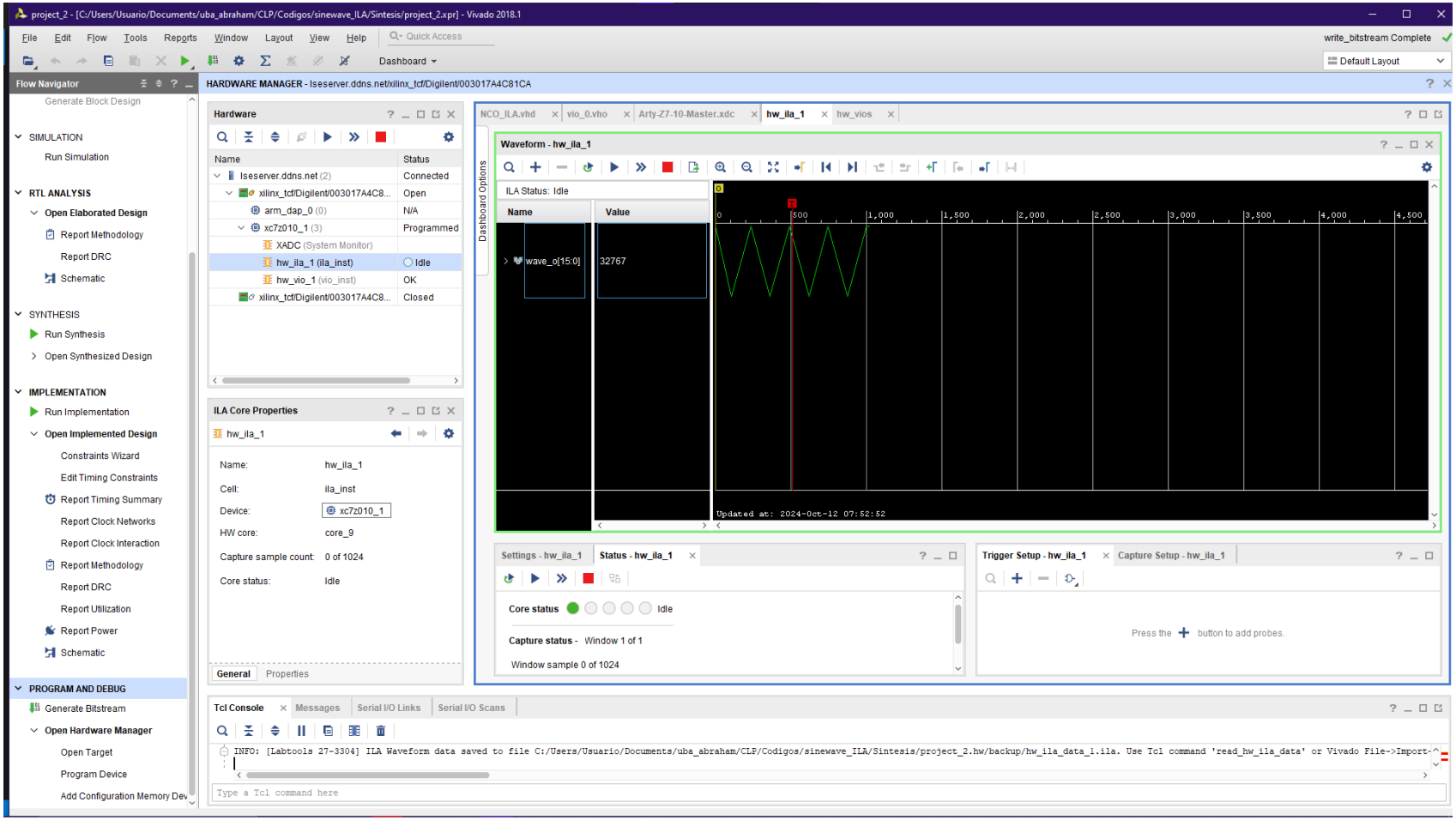
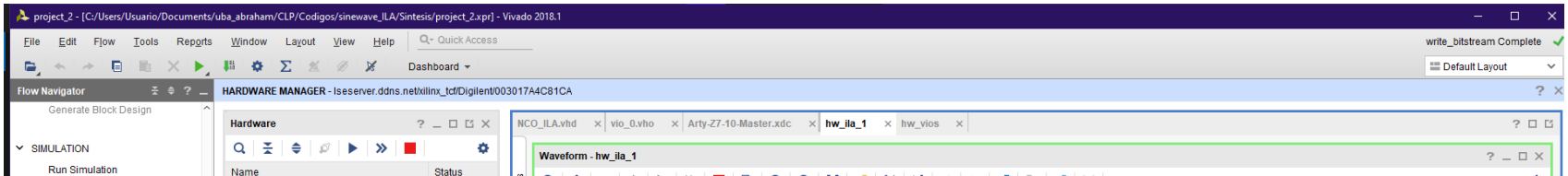


Figura: Triangular en FPGA remota.



Pruebas con FPGA remota

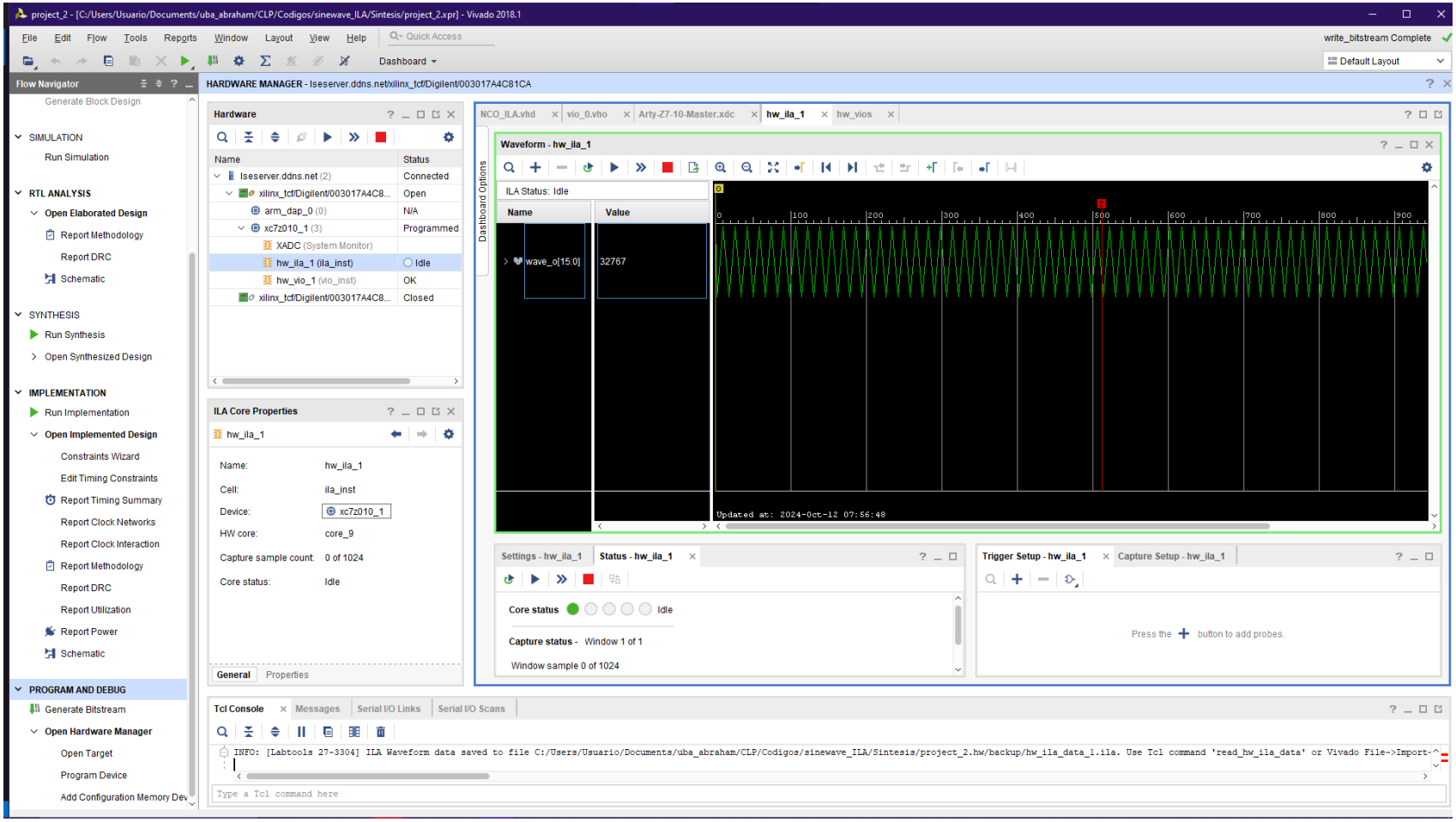


Figura: Triangular de mayor frecuencia en FPGA remota.

Gracias por su atencion