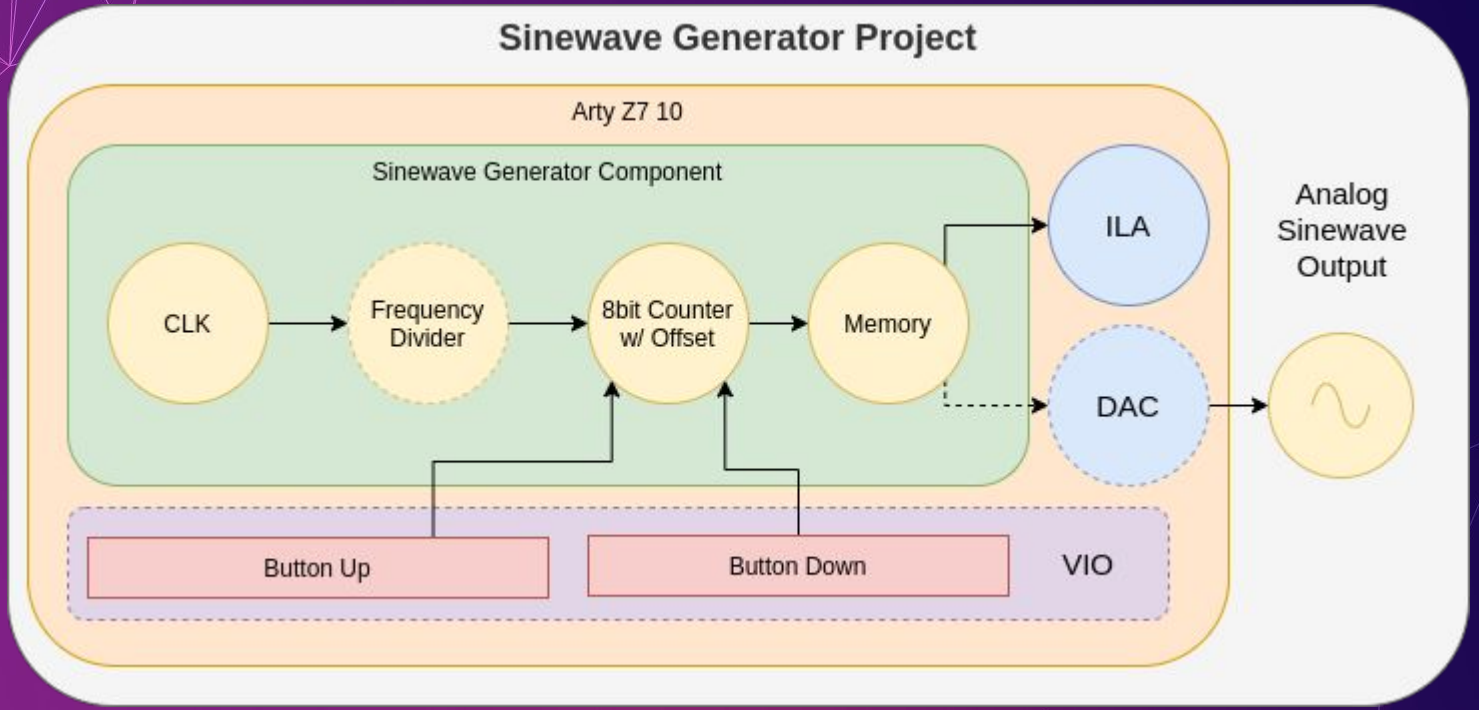
CIRCUITOS LÓGICOS PROGRAMABLES

GENERADOR DE SEÑALES SENOIDALES

Alumno: Lucas Orsi
Cohorte: 14va
Universidad de Buenos Aires

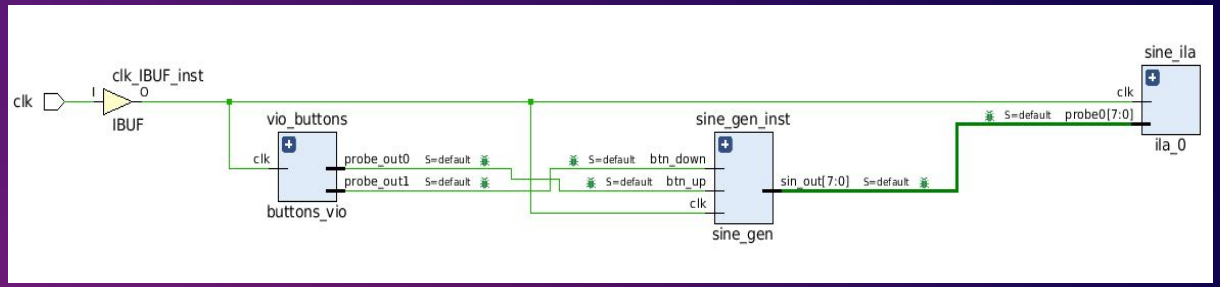
Introducción

Diagrama General del Sistema

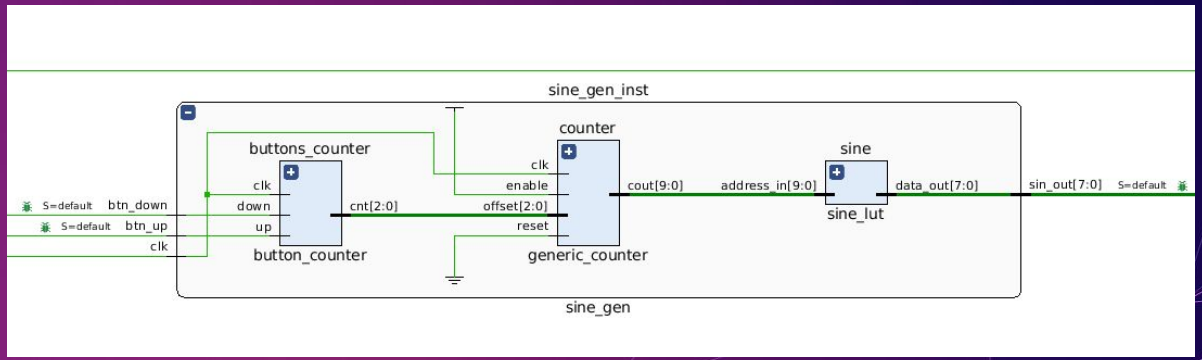


Esquemático

Esquemático General

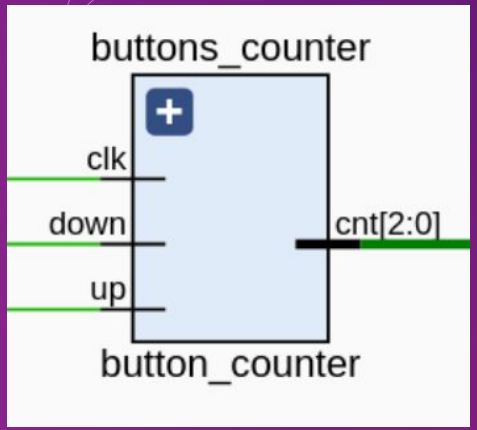


Generador de Senoidales



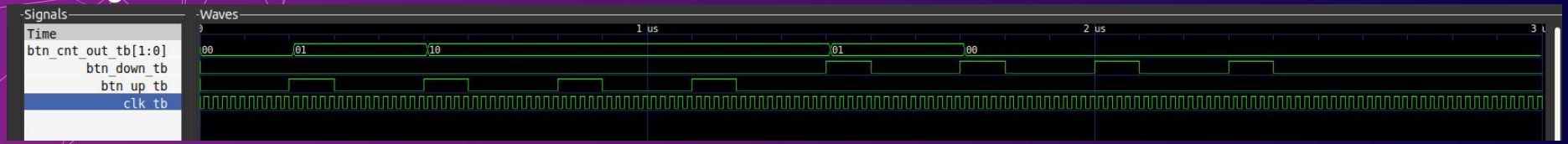
Bloques Funcionales

Manejador de Botones



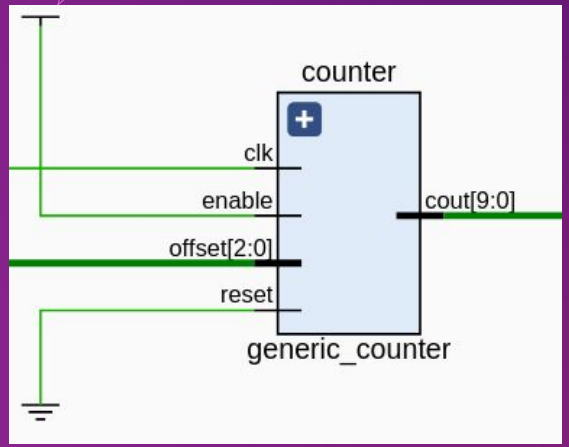

```
process(clk)
    begin
    if rising_edge(clk) then
        if up = '1' and last_up = '0'
            and internal_counter < N then
             internal_counter <= internal_counter + 1;
        end if;
        if down = '1' and last_down = '0'
            and internal_counter > 0 then
              internal_counter <= internal_counter - 1;
        end if;
        last_up <= up;
        last_down <= down;
    end if;
 end process;
 cnt <= std_logic_vector(internal_counter);
```

Manejador de Botones



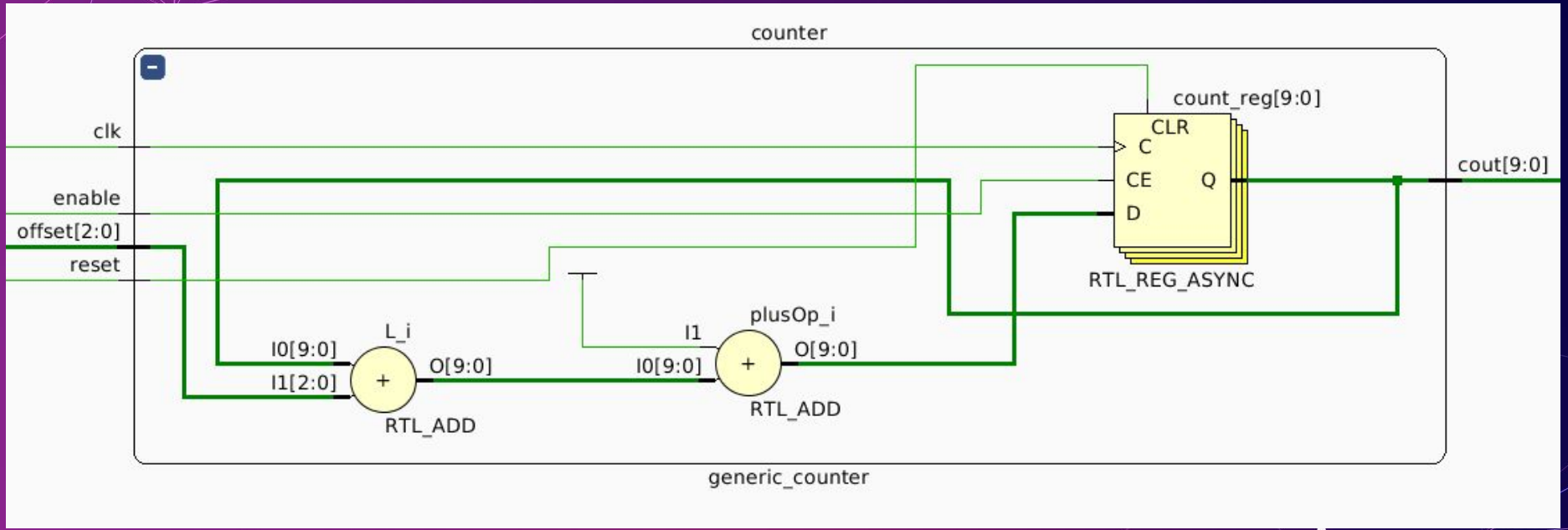

```
architecture button_counter_tb_arch of button_counter_tb is
    component button_counter is
        generic(N: natural := 2; N_BITS: natural := 2);
        port(
            clk:    in std_logic;
            up:     in std_logic;
            down:   in std_logic;
            cnt:    out std_logic_VECTOR (N_BITS - 1 downto 0)
        );
    end component;
```

Contador Principal

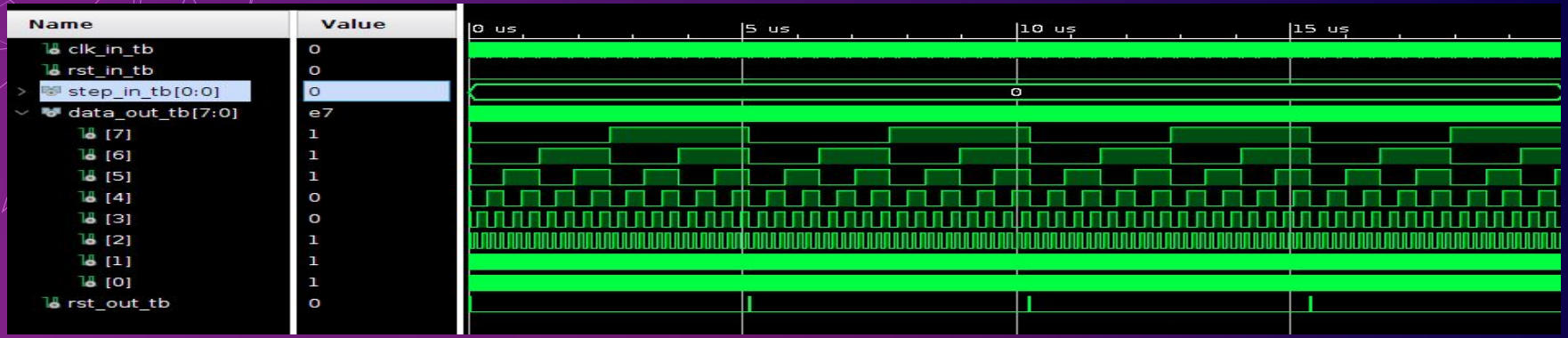



```
process (clk, reset) begin
     if (reset = '1') then
          count <= (others=>'0');
    elsif (rising_edge(clk)) then
         if (enable = '1') then
              count <= count + unsigned(offset) + 1;
          end if;
     end if;
 end process;
 cout <= std_logic_vector(count);
```

Contador Principal

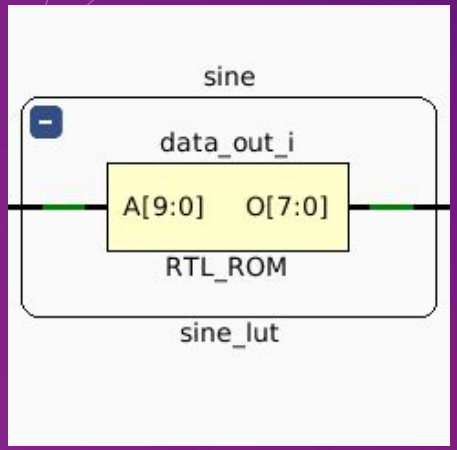


Contador Principal




```
component generic_counter is
     generic(OUTPUT_WIDTH: natural := 8; COUNT_OFFSET_WIDTH:  natural := 3);
     port (
         cout   :out std_logic_vector (OUTPUT_WIDTH - 1 downto 0);
         enable :in  std_logic;
         clk    :in  std_logic;
         offset :in  std_logic_vector(COUNT_OFFSET_WIDTH - 1 downto 0);
         reset  :in  std_logic
     );
 end component generic_counter;
```

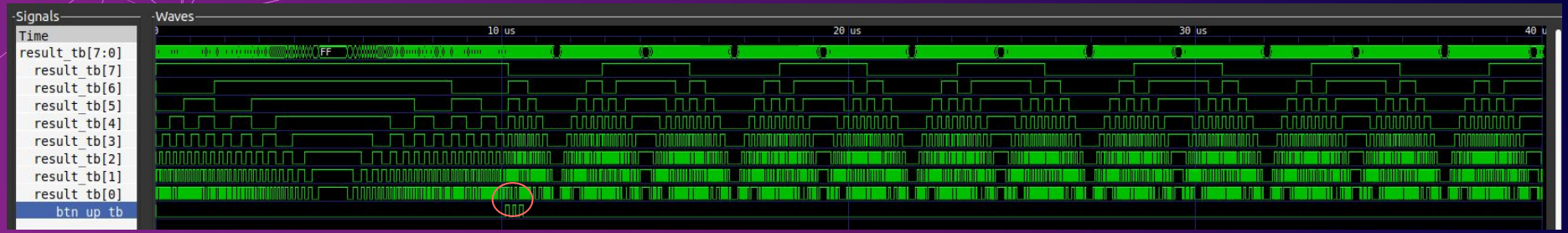
Tabla de Señal



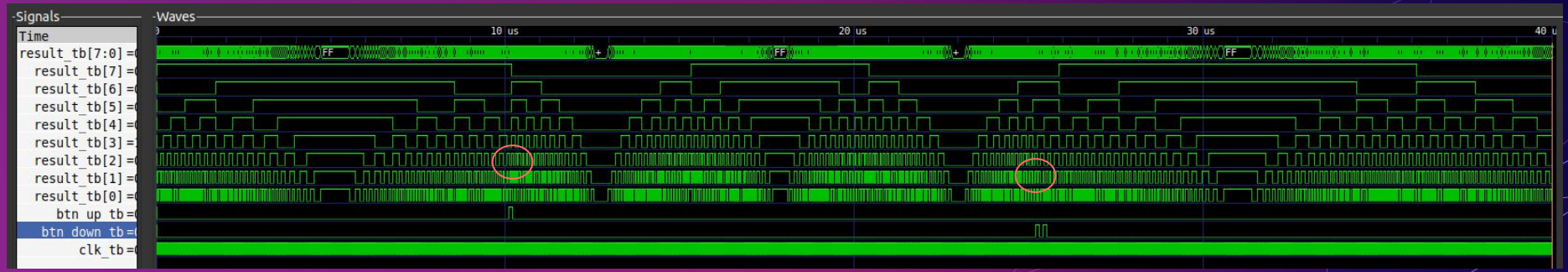

```
process(address_in) is
begin
    case address_in is
        when "0000000000" => data_out <= "01111111";
        when "0000000001" => data_out <= "10000000";
        ...
        when "1111111110" => data_out <= "01111111";
        when "1111111111" => data_out <= "01111111";
        when others => data_out <= (others => '0');
    end case;
end process;
```

Simulación del Sistema

Botón “Up”



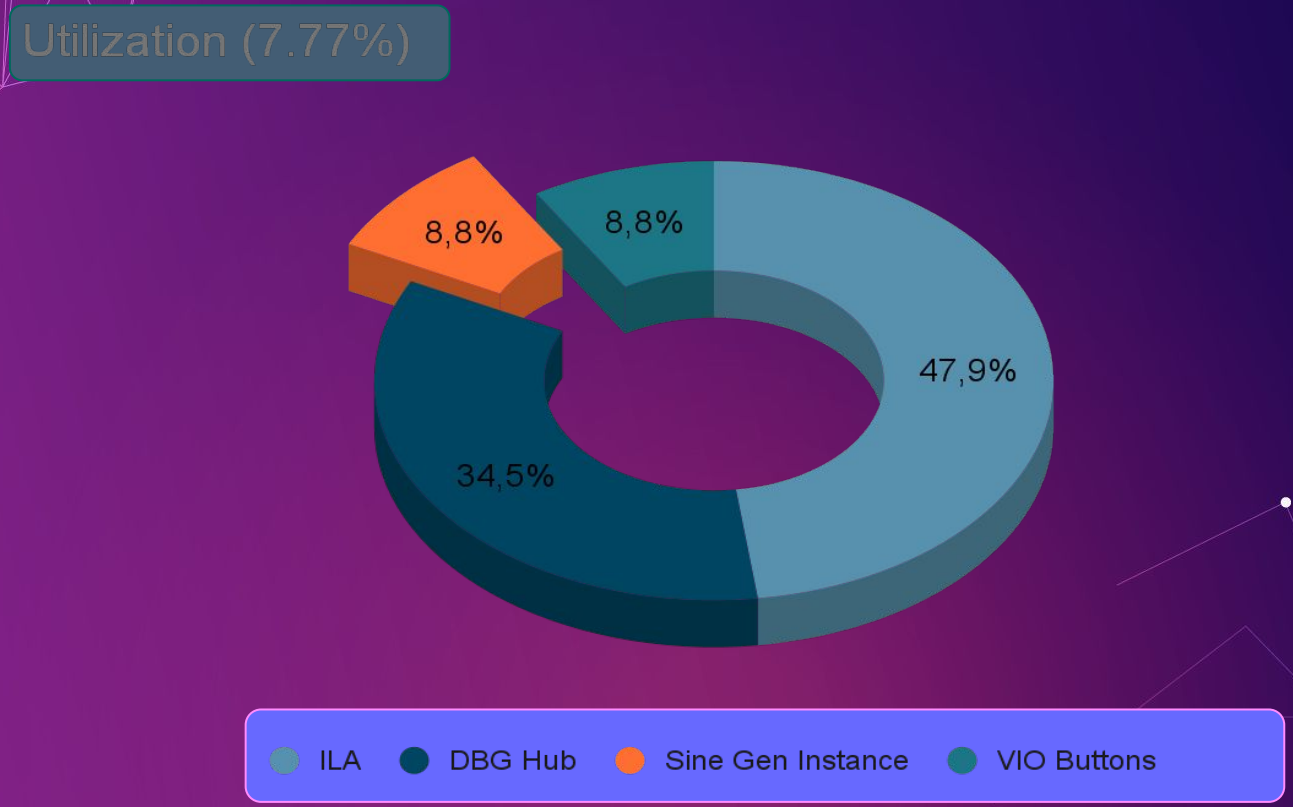
Botón “Up” y luego “Down”



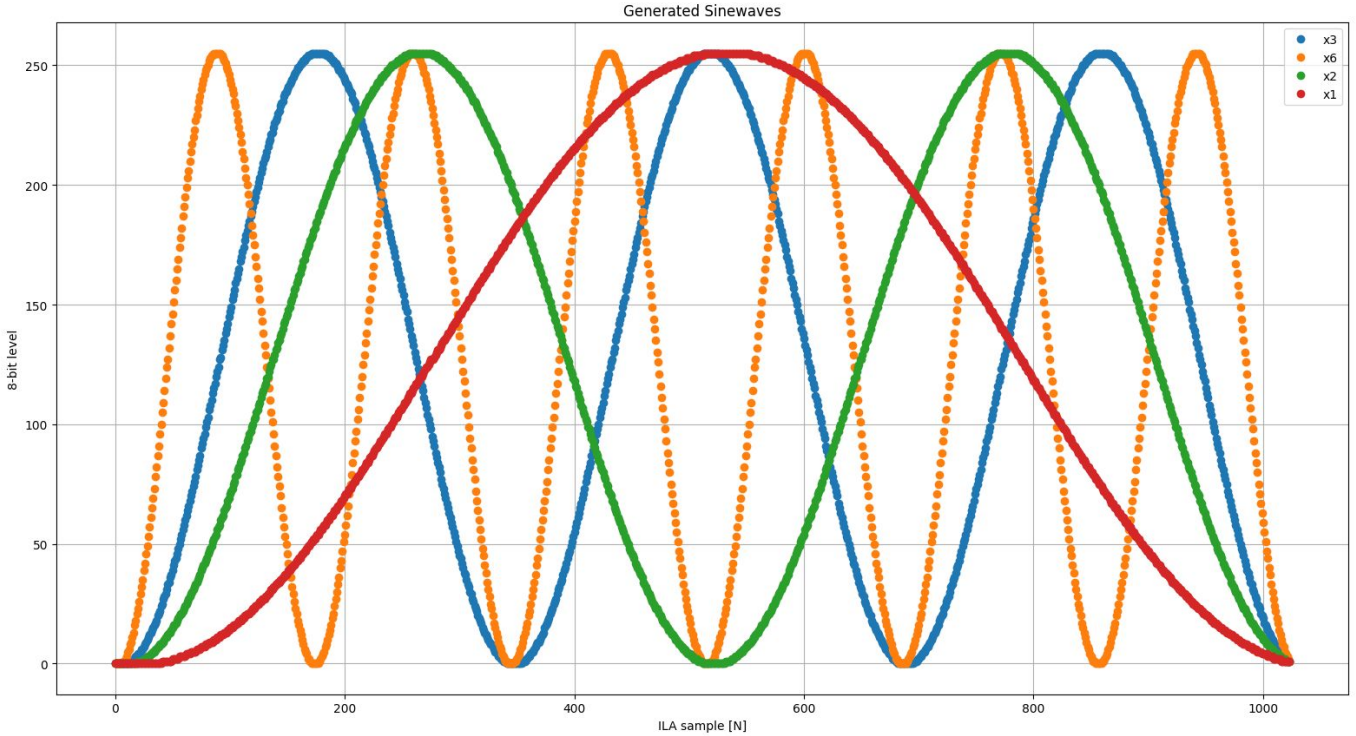
Prueba en Hardware

Análisis de Resultados

Reporte de Utilización



Senoidales Obtenidas



Herramientas Personalizadas

```
$ python3 tools/vhdlutils.py --help

usage: vhdlutils.py [-h] [--time TIME] [--compile-only] component

Compile and simulate VHDL sources.

positional arguments:
  component       Component whose testbench will be simulated

optional arguments:
  -h, --help      show this help message and exit
  --time TIME     dimulation stoptime in nanoseconds
  --compile-only  does not run any simulation
```

Compilación y Simulación de Fuentes

Generación de Tablas de Senoidales en VHDL

```
$  python3 tools/sine_table_gen.py

begin
    case address is
        when "0000000000" => result <= "01111111"; -- 0x7f
        when "0000000001" => result <= "10000000"; -- 0x80
        when "0000000010" => result <= "10000001"; -- 0x81
        when "0000000011" => result <= "10000010"; -- 0x82
        when "0000000100" => result <= "10000011"; -- 0x83
        when "0000000101" => result <= "10000011"; -- 0x83
        when "0000000110" => result <= "10000100"; -- 0x84
        ...
```

Gracias!

Preguntas?