Módulo Generador de PWM

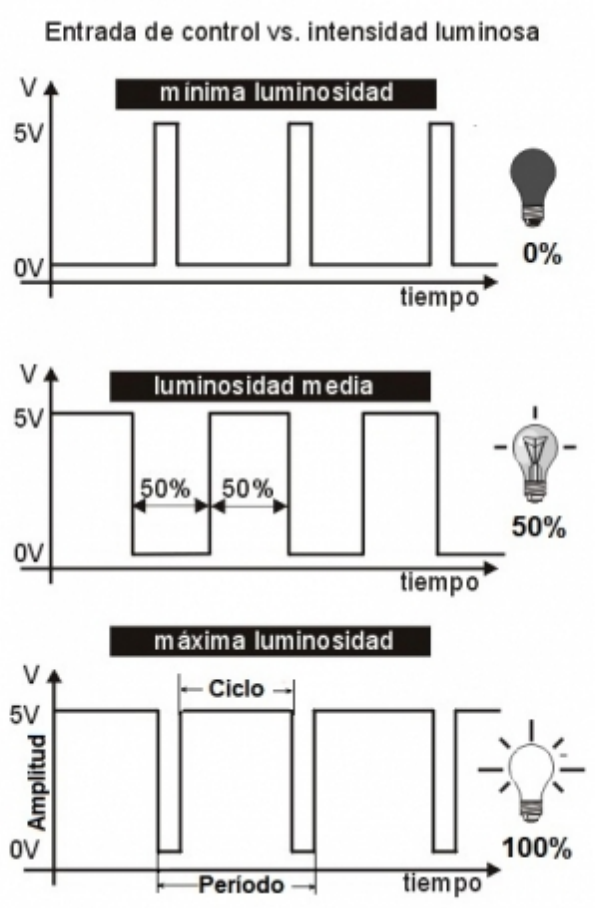
Circuitos Lógicos Programables

CESE

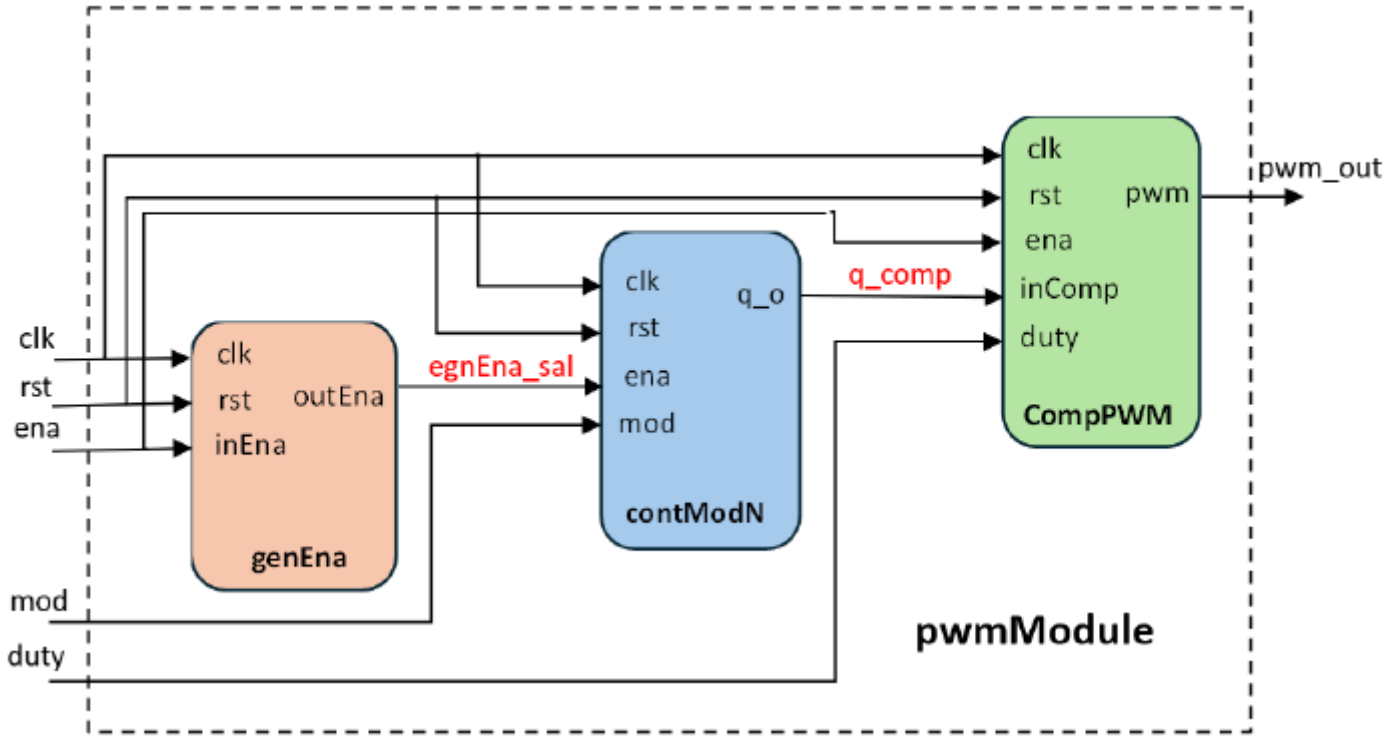
Prof. Nicolás Álvarez

Alumno: Rubén Mansilla

La señal PWM


Entrada de control vs. intensidad luminosa
mínima luminosidad
V
5V
0V
tiempo
0%
luminosidad media
50%
50%
50%
máxima luminosidad
Ciclo
Amplitud
Período
100%

Esquema del Módulo Generador de PWM



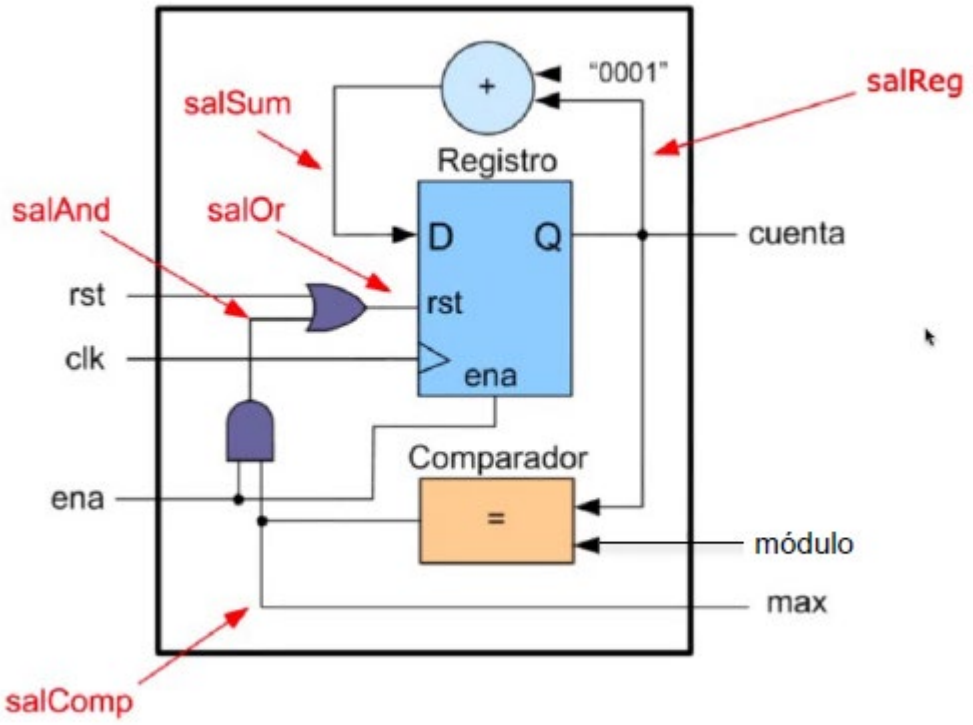
Resumen de las señales del módulo PWM:

- **Entradas**:
  - ***clk*** → Clock de la placa (125 MHz) [1 bit]
  - ***rst*** → Reset: Fuerza a “0” todas las salidas de los módulos del componente. [1 bit]
  - ***ena*** →Enable: Habilita el funcionamiento de todos los módulos del componente. [1 bit]
  - ***mod*** → Módulo: Valor que establece el período de la señal PWM. [N bits]
  - ***duty*** → Duty cicle: Valor que establece la cantidad de cuentas que la salida PWM permanecerá en “1”. [N bits]
- **Salidas:**

  pwm_out → Salida PWM [1 bit] serial.

El contador de Módulo N: contModN

Utiliza una variable interna **N** que determinará la cantidad de bits que se desee que tenga el contador



Implementación VHDL del Generador de habilitación genEna

Se utiliza una variable interna ***NC*** que determinará cada cuantos pulsos de clock la salida *q_o saca un "1"*

```
entity genEna is
    generic(
        NC: natural := 4
    );
    port(
        clk_i : in std_logic;
        rst_i : in std_logic;
        ena_i : in std_logic;
        q_o   : out std_logic
    );
end;
```

```
architecture genEna_arq of genEna is
    -- Parte declarativa
    signal cuenta : integer;

begin
    -- Parte descriptiva

    process(clk_i)
        variable aux: integer;
    begin
        if rising_edge(clk_i) then
            if rst_i = '1' then
                aux := 0;
                q_o <= '0';
            elsif ena_i = '1' then
                aux := aux + 1;
                if aux = NC then
                    q_o <= '1';
                    aux := 0;
                else
                    q_o <= '0';
                end if;
            end if;
        end if;
        cuenta <= aux;
    end process;

end;
```

Implementación VHDL de contModN

Utiliza una variable interna **N** que determinará la cantidad de bits que se desee que tenga el contador

```vhdl
entity contModN2 is
    generic(
        N : natural := 4
    );
    port(
        clk_i : in std_logic;
        rst_i : in std_logic;
        ena_i : in std_logic;
        mod_i : in std_logic_vector(N-1 downto 0); -- Mod
        q_o   : out std_logic_vector(N-1 downto 0)
        --max_o : out std_logic
    );
end;
```

```vhdl
architecture contModN2_arq of contModN2 is
    -- Parte declarativa

    component reg is
        generic(
            N: in natural := 4
        );
        port(
            clk_i : in std_logic;
            rst_i : in std_logic;
            ena_i : in std_logic;
            d_i   : in std_logic_vector(N-1 downto 0);
            q_o   : out std_logic_vector(N-1 downto 0)
        );
    end component;

    signal salReg : std_logic_vector(N-1 downto 0);
    signal salSum : std_logic_vector(N-1 downto 0);
    signal salOr  : std_logic;
    signal salAnd : std_logic;
    signal salComp:std_logic;
```

```vhdl
begin
    -- Parte descriptiva

    reg_inst: reg
        generic map(
            N => 4
        )
        port map(
            clk_i => clk_i,
            rst_i => salOr,
            ena_i => ena_i,
            d_i   => salSum,
            q_o   => salReg
        );

    -- mod_i <= "1111"

    salSum <= std_logic_vector(unsigned(salReg) + "0001");

    salComp <= '1' when salReg = mod_i else '0';

    salAnd <= ena_i and salComp;

    salOr <= rst_i or salAnd;

    --max_o <= salComp;

    q_o <= salReg;
end;
```

Implementación VHDL del Módulo pwmModule

La variable interna **N** determina la cantidad de bits de las entradas *mod_i*, *duty_i* y la salida *q_o* y debe ser consistente con el valor adoptado en el contador **contMosN**.

```
entity pwmModule is
    generic(
        N : natural := 4
    );
    port(
        clk_i   : in  std_logic;
        rst_i   : in  std_logic;
        mod_i   : in  std_logic_vector(N-1 downto 0);     -- Período del PWM
        ena_i   : in  std_logic;
        duty_i  : in  std_logic_vector(N-1 downto 0);     -- Ciclo de trabajo
        q_o     : out std_logic_vector(N-1 downto 0); -- Salida del contador contModN2
        pwm_out : out std_logic
    );
end;
architecture pwmModule_arq of pwmModule is
    -- Parte declarativa
    signal q_internal : std_logic_vector(N-1 downto 0);

    component contModN2 is
        generic(
            N : natural := 4
        );
        port(
            clk_i : in std_logic;
            rst_i : in std_logic;
            ena_i : in std_logic;
            mod_i : in std_logic_vector(N-1 downto 0); -- Modulo de cuenta --> Período
            q_o   : out std_logic_vector(N-1 downto 0)
        );
    end component;

    signal genEna_sal : std_logic;
```

Implementación VHDL del Módulo pwmModule

```vhdl
begin
    -- Parte descriptiva

    contModN2_inst: contModN2
        generic map(
            N => 4
        )
        port map(
            clk_i => clk_i,
            rst_i => rst_i,
            ena_i => genEna_sal,
            mod_i => mod_i,
            q_o   => q_internal
        );

    genEna_inst: entity work.genEna
        generic map(
            NC => 3 --N
        )
        port map(
            clk_i => clk_i,
            rst_i => rst_i,
            ena_i => ena_i,
            q_o   => genEna_sal
        );
```

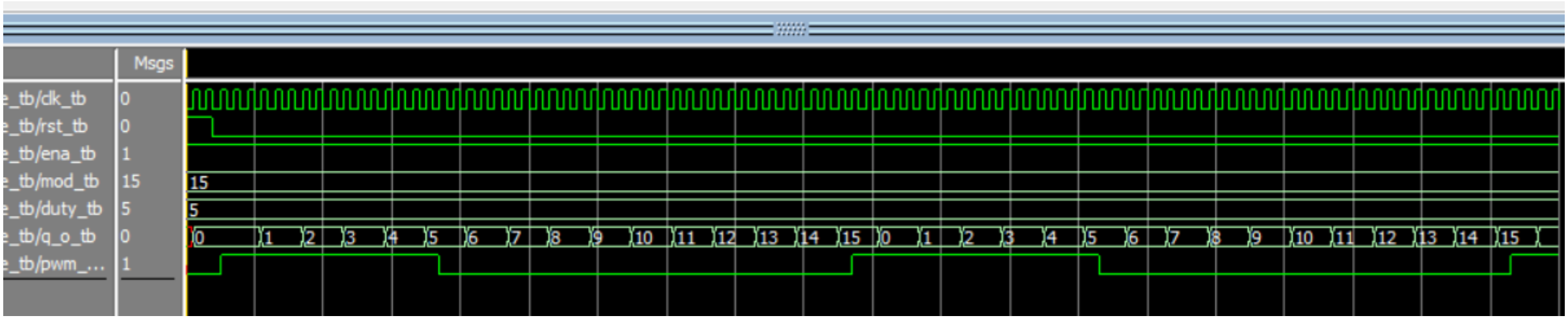
```vhdl
    -- Aqui ponemos las operaciones propias del Modulo pwm
    process (clk_i, rst_i)
    begin
        if rst_i = '1' then
            pwm_out <= '0';
        elsif rising_edge(clk_i) then
            if ena_i = '1' then
                if unsigned(q_internal) < unsigned(duty_i) then
                    pwm_out <= '1';
                else
                    pwm_out <= '0';
                end if;
            end if;
            if unsigned(q_internal) >= unsigned(mod_i) then
                pwm_out <= '1';
            end if;
        end if;
    end process;

    q_o <= q_internal;

end;
```

La señal *q_internal* se utiliza para poder visualizar en el simulador el valor de la cuenta de salida del contador **contModN**.

Pruebas y Simulaciones



modulo	15	16 cuentas
duty	5	6 cuentas
N	4	bits
NC	3	Ciclos
clock	20 ns	Periodo

Salida PWM	37,5% duty cicle
	6 cuentas en "1"
	10 cuentas en "0"

Cada 3 ciclos de clock el contador hace una cuenta.

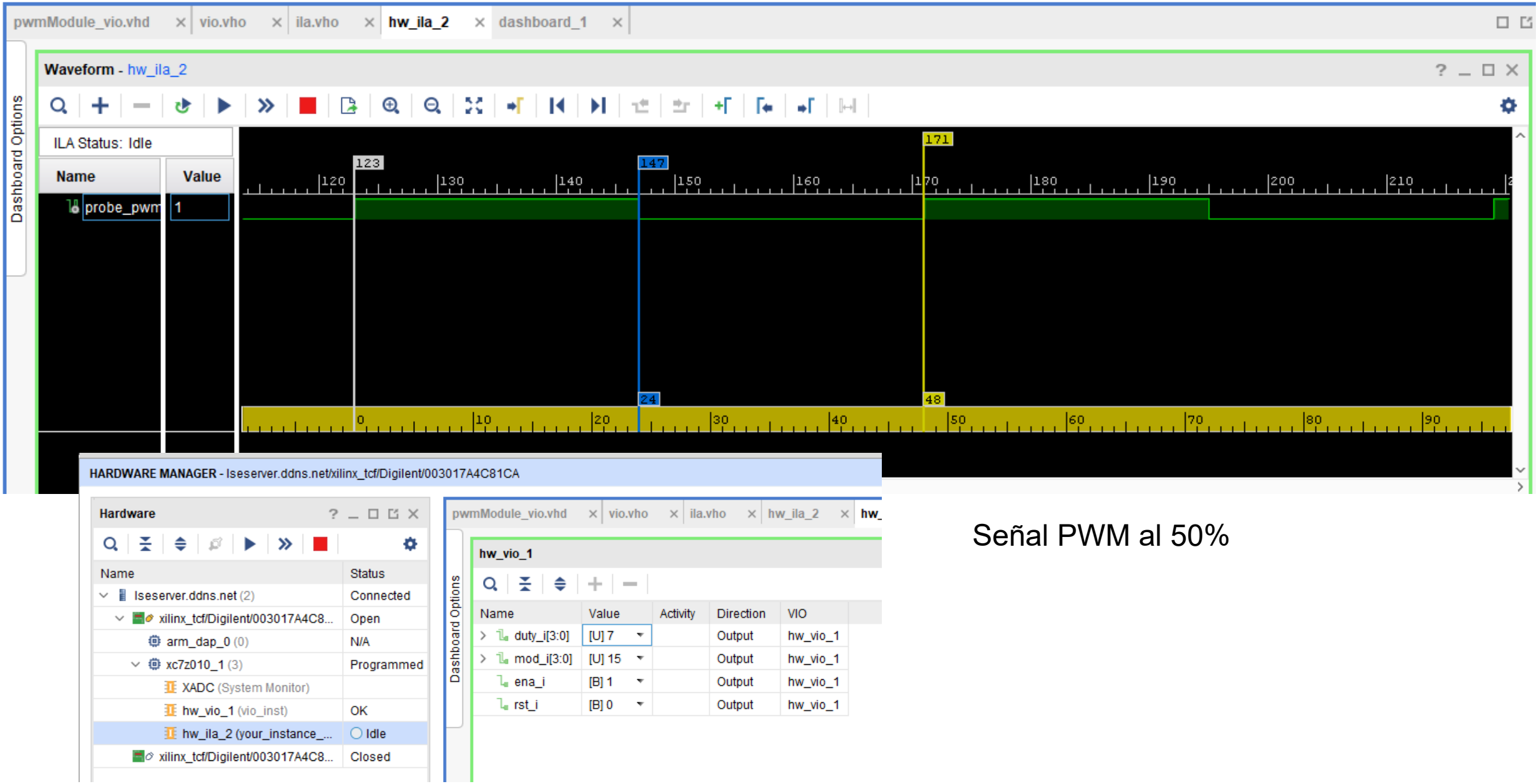
$$\text{Tiempo de cuenta} = 20\ ns\ \times 3 = 60\ ns$$

$$\text{Período de PWM} = 60\ ns\ \times 16 = 960\ ns$$

$$\text{Duty cicle} = 60\ ns\ \times 6 = 360\ ns$$

$$\text{Salida PWM} = \frac{Duty\ cicle}{Periodo\ PWM} = \frac{6}{16} \times 100 = 37{,}5\%$$

Prueba de funcionamiento en Vivado



Señal PWM al 50%

Prueba de funcionamiento en Vivado

Una cuenta de **contModN** son 3 marcas en el grafico – 16 cuentas conforman un periodo de 48 marcas.

Frecuencia clock = 125 MHz → Período clock $= \frac{1}{125}\ \mu s = 0{,}008\ \mu s = 8\ ns$

$$\text{Tiempo de cuenta} = 8\ ns\ \times 3 = 24\ ns$$

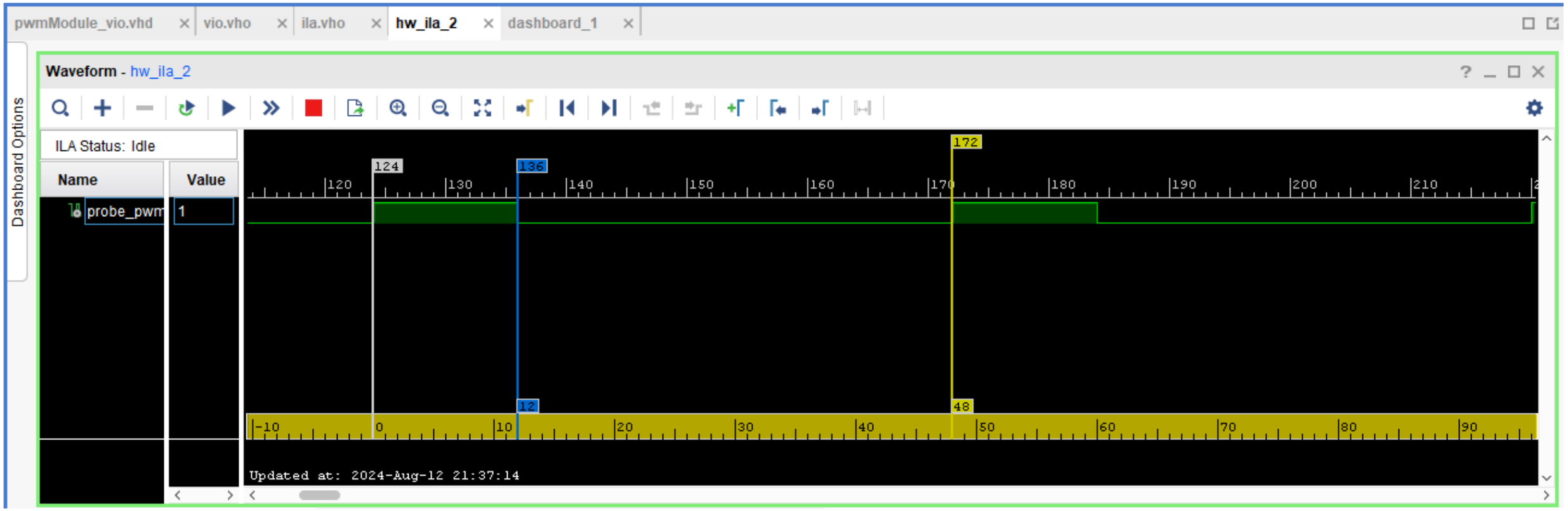
$$\text{Período de PWM} = 24\ ns\ \times 16 = 384\ ns$$

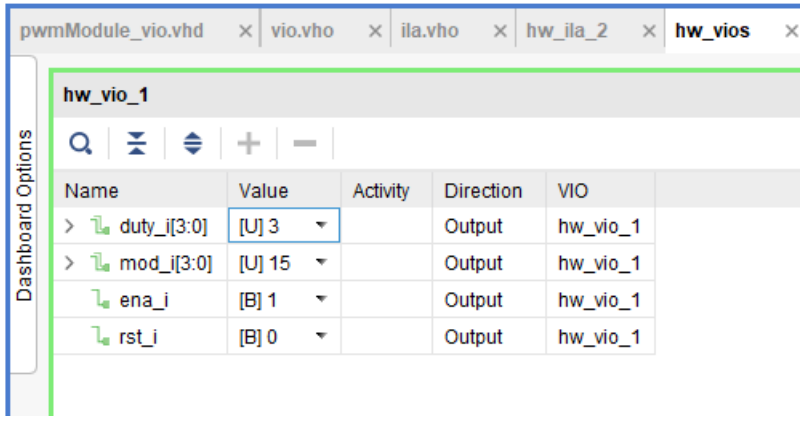
$$\text{Duty cicle} = 24\ ns\ \times 12 = 288\ ns$$

$$\text{Salida PWM} = \frac{Duty\ cicle}{Periodo\ PWM} = \frac{12}{16} \times 100 = 75\%$$

$$\text{1 marca en el grafico} = \frac{tiempo\ cuenta}{3} = \frac{24\ ns}{3} = 8\ ns$$

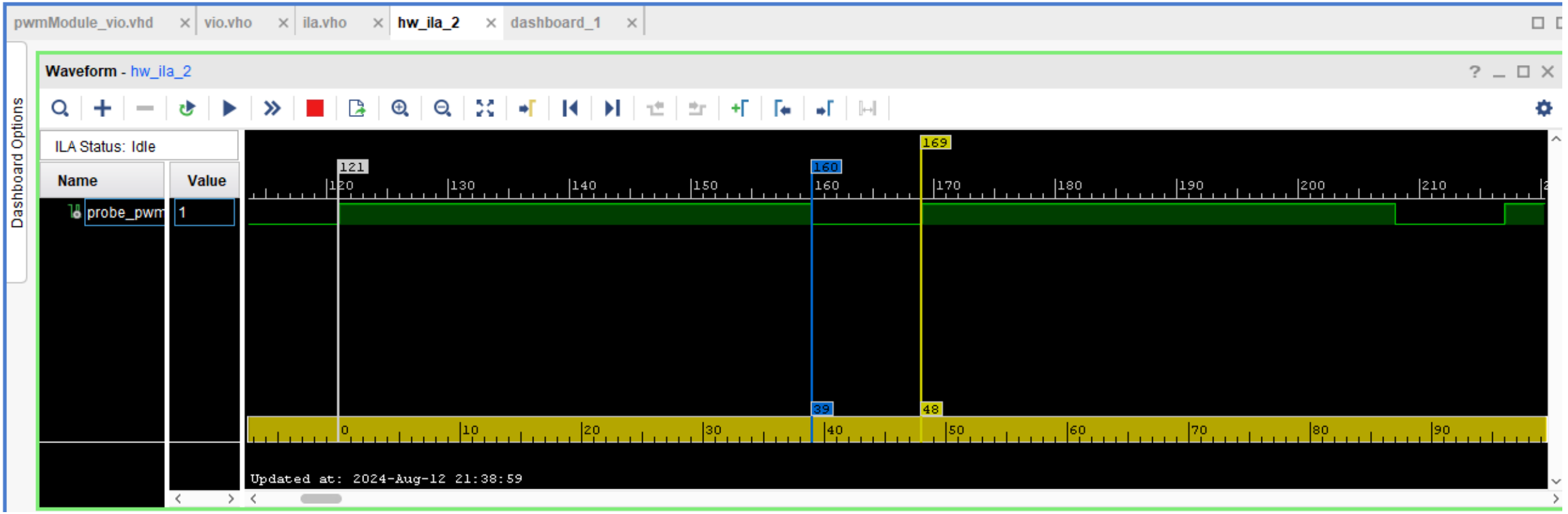
Prueba de funcionamiento en Vivado

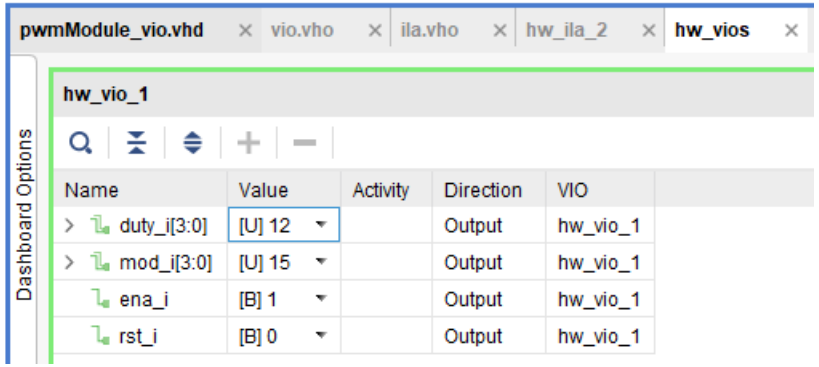




Señal pwm al 25%

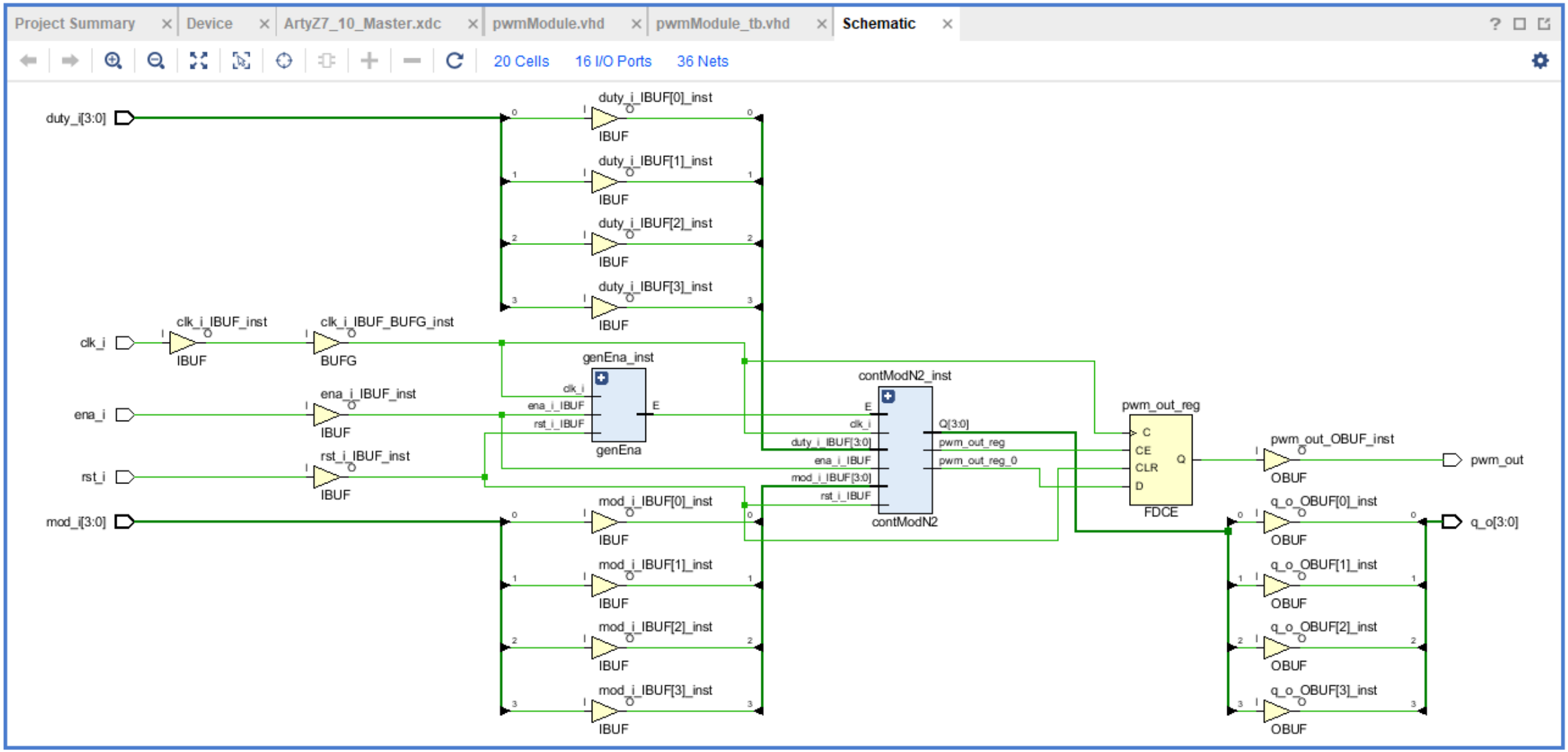
Prueba de funcionamiento en Vivado



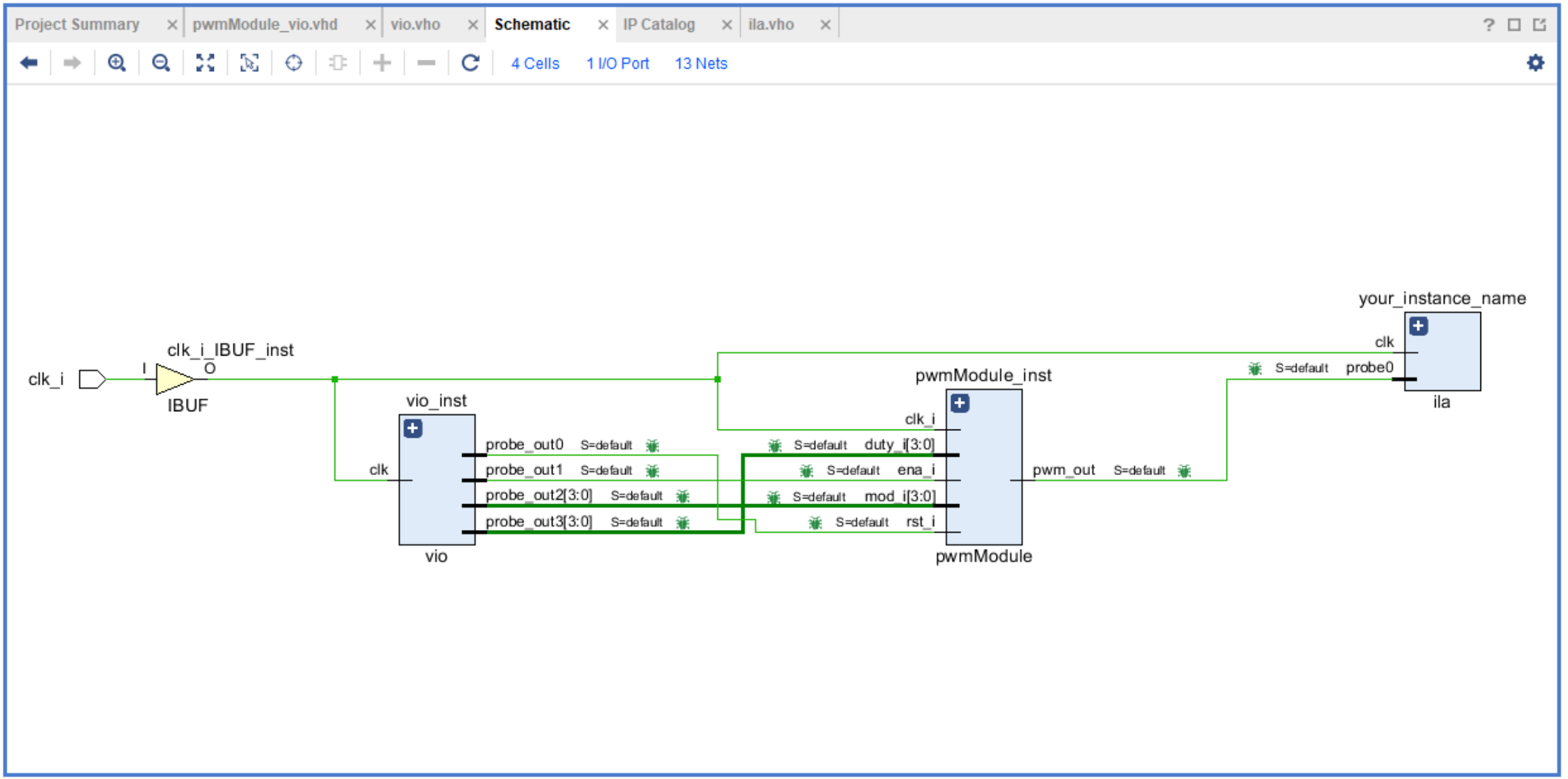


Señal PWM al 75%

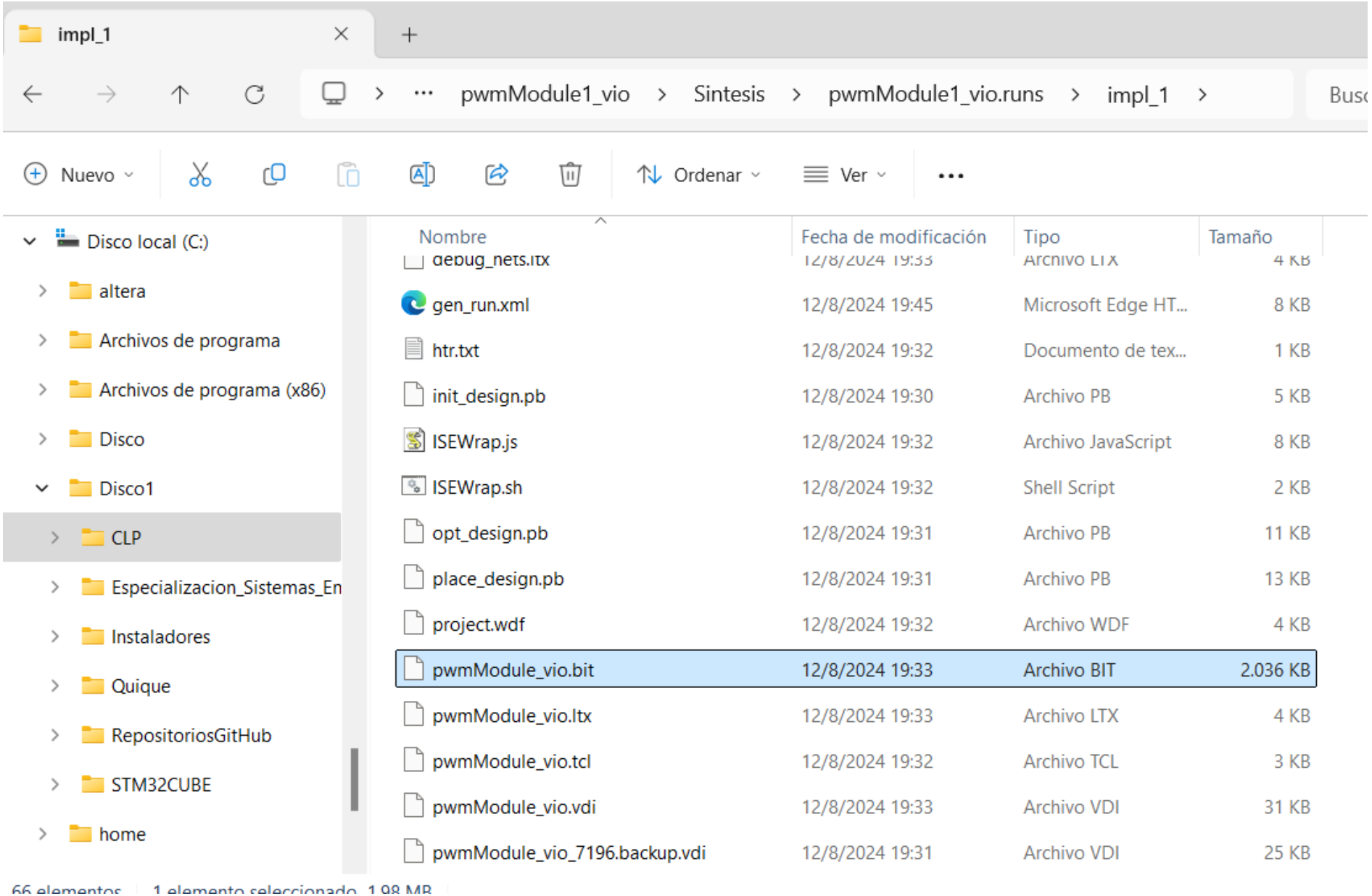
Esquemático de la implementación en Vivado del componente **pwmModule**



Esquemático de implementación en Vivado del conjunto **pwmModule – VIO - ILA**

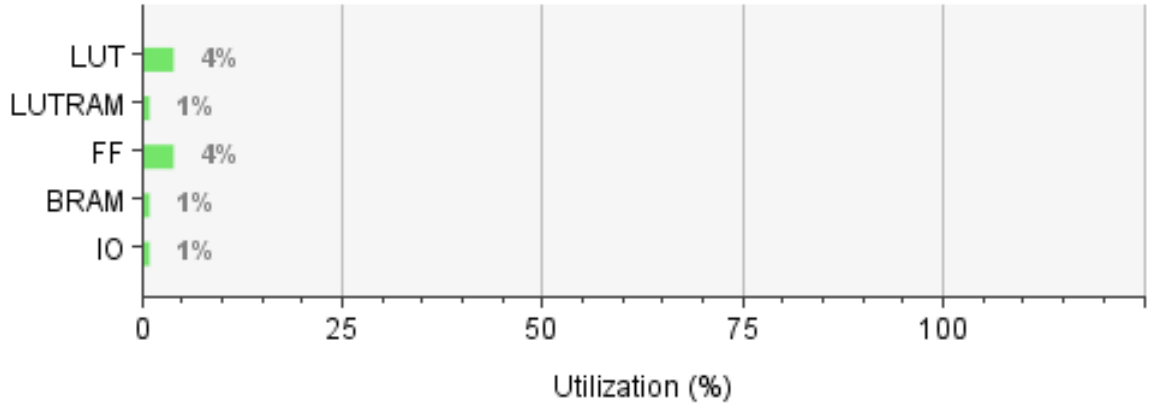


Generación del archivo *bitstream*



Utilización de recursos post sintesis

Resource	Utilization	Available	Utilization %
LUT	722	17600	4.10
LUTRAM	61	6000	1.02
FF	1365	35200	3.88
BRAM	0.50	60	0.83
IO	1	100	1.00



Utilización de recursos post implementación

Resource	Utilization	Available	Utilization %
LUT	1177	17600	6.69
LUTRAM	85	6000	1.42
FF	2028	35200	5.76
BRAM	0.50	60	0.83
IO	1	100	1.00