Documentación Técnica: Implementación y Simulación de Funciones Lógicas en VHDL para la práctica número 4 del laboratorio de electrónica

Kevin Esguerra Cardona
Código: `1004 699 587`

4 de abril de 2025

Resumen

Este documento detalla el planteamiento y desarrollo de un ejercicio práctico en VHDL, orientado a la implementación y verificación de funciones lógicas en dos formas canónicas: Suma de Productos (SOP) y Producto de Sumas (POS). Se presenta el paso a paso del proceso, incluyendo la generación de la tabla de verdad, el desarrollo de las ecuaciones lógicas, la elaboración de los circuitos correspondientes y la verificación mediante simulación en EDA Playground utilizando el simulador GHDL.

1. Planteamiento del ejercicio

El presente ejercicio consiste en el diseño y simulación de una función lógica en VHDL, a partir de un criterio personalizado basado en el número de identificación del estudiante (en adelante, ."el código"). El objetivo es construir una tabla de verdad y derivar las expresiones canónicas SOP (Suma de Productos) y POS (Producto de Sumas), implementando posteriormente dichas expresiones en código VHDL y verificando su funcionamiento mediante simulación en la plataforma EDA Playground, utilizando el simulador GHDL.

El procedimiento para definir los valores en los que la salida S será igual a 1 (y por tanto, codificados en la tabla de verdad) se realiza como sigue:

1. El número de identidad se separa en pares de dígitos consecutivos. Por ejemplo, si el código es `10154832`, los pares obtenidos serán: `10`, `15`, `48`, `32`.

2. Cada par se suma: $10 \rightarrow 1+0 = 1$, $15 \rightarrow 1+5 = 6$, $48 \rightarrow 4+8 = 12$, $32 \rightarrow 3+2 = 5$.

3. Si el número de identidad tiene una cantidad impar de dígitos, el último dígito se toma individualmente y se codifica directamente. Por ejemplo, si el código fuera `1015483`, el 3 se codifica como $3 \rightarrow 0011$.

4. Si la suma de un par es mayor a 15 (el valor máximo representable con 4 bits), se repite el proceso sumando los dígitos del resultado. Ejemplo: para el par `88`, la suma es $8 + 8 = 16$. Como 16 no puede representarse con 4 bits, se suman los dígitos de 16: $1 + 6 = 7$, por lo tanto, se toma el valor 7.

5. Si durante el proceso se obtienen valores repetidos, solo se consideran una única vez, es decir, no se codifican duplicados.

6. Una vez obtenidos todos los valores únicos, se construye una tabla de verdad de 4 entradas (A_3, A_2, A_1, A_0), donde cada valor calculado activa la salida $S = 1$ en la fila correspondiente a su representación binaria.

Este planteamiento tiene como finalidad generar una lógica personalizada por estudiante, manteniendo la coherencia entre análisis teórico y su implementación digital.

2. Paso 1: Generación de la Tabla de Verdad

La tabla de verdad se construye a partir de las variables de entrada definidas por el código (número de documento). Cada combinación de valores binarios es evaluada para determinar la salida de la función lógica.

A_3	A_2	A_1	A_0	S
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

3. Paso 2: Derivación de las Ecuaciones SOP

Con la tabla de verdad establecida, se procede a la extracción de la función lógica en forma de Suma de Productos (SOP). Esta forma consiste en sumar (OR) los términos que representan las combinaciones donde la salida es 1.

$$S = \overline{A_3} \cdot \overline{A_2} \cdot \overline{A_1} \cdot A_0 \; + \; \overline{A_3} \cdot A_2 \cdot \overline{A_1} \cdot \overline{A_0} \; + \; A_3 \cdot A_2 \cdot A_1 \cdot \overline{A_0} \; + \; A_3 \cdot A_2 \cdot A_1 \cdot A_0$$

4. Paso 3: Derivación de las Ecuaciones POS

De forma análoga, se obtiene la representación de la función lógica en Producto de Sumas (POS). Esta forma se centra en identificar las combinaciones donde la salida es 0 y formar la multiplicación (AND) de los sumandos correspondientes.

$$
\begin{aligned}
S = &(A_3 + A_2 + A_1 + A_0) \cdot (A_3 + A_2 + \overline{A_1} + A_0) \cdot (A_3 + A_2 + \overline{A_1} + \overline{A_0}) \cdot \\
&(A_3 + \overline{A_2} + A_1 + \overline{A_0}) \cdot (A_3 + \overline{A_2} + \overline{A_1} + A_0) \cdot (A_3 + \overline{A_2} + \overline{A_1} + \overline{A_0}) \cdot \\
&(\overline{A_3} + A_2 + A_1 + A_0) \cdot (\overline{A_3} + A_2 + A_1 + \overline{A_0}) \cdot (\overline{A_3} + A_2 + \overline{A_1} + A_0) \cdot \\
&(\overline{A_3} + A_2 + \overline{A_1} + \overline{A_0}) \cdot (\overline{A_3} + \overline{A_2} + A_1 + A_0) \cdot (\overline{A_3} + \overline{A_2} + A_1 + \overline{A_0})
\end{aligned}
$$

5. Paso 4: Diseño de los Circuitos Lógicos

Con las ecuaciones derivadas, se procede al diseño de los circuitos lógicos correspondientes. Se realizan dos implementaciones:

- **Circuito basado en la forma SOP.**
- **Circuito basado en la forma POS.**

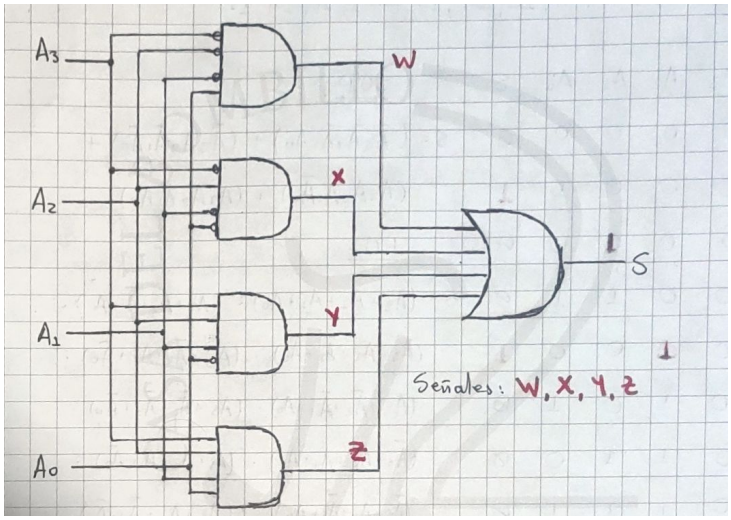


Figura 1: Diagrama del circuito implementado en forma SOP.

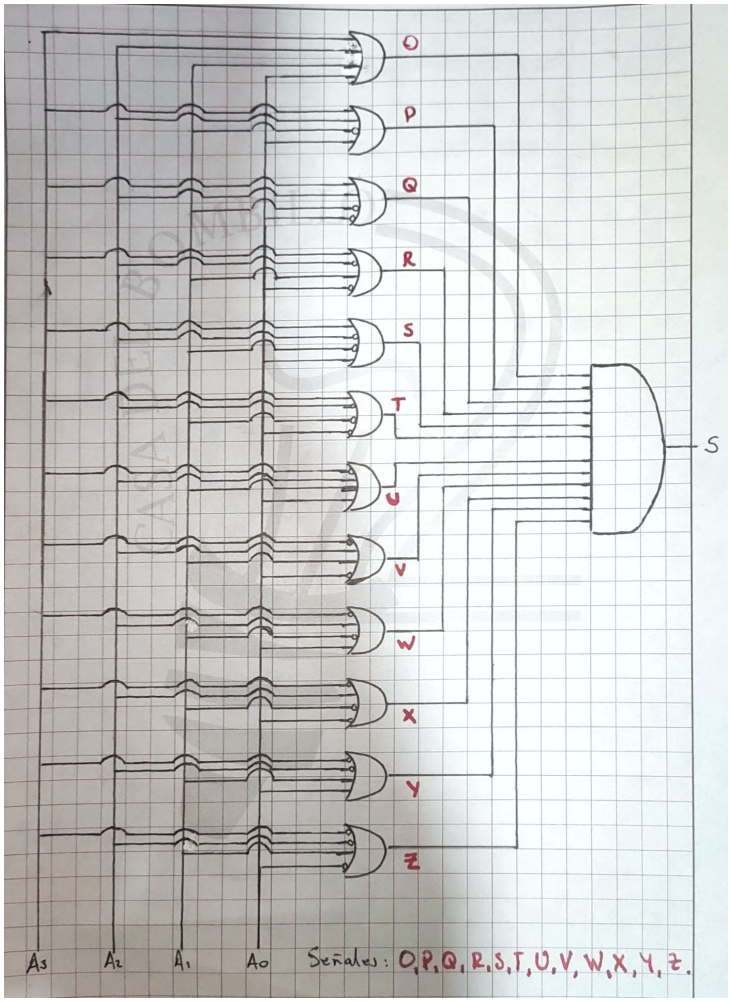


Figura 2: Diagrama del circuito implementado en forma POS.

6. Paso 5: Implementación en VHDL

La implementación se realizó en tres archivos VHDL:

- **pos.vhd:** Código que implementa la función en forma de POS.
- **sop.vhd:** Código que implementa la función en forma de SOP.
- **testBench.vhd:** Banco de pruebas utilizado para verificar el comportamiento de ambas implementaciones.

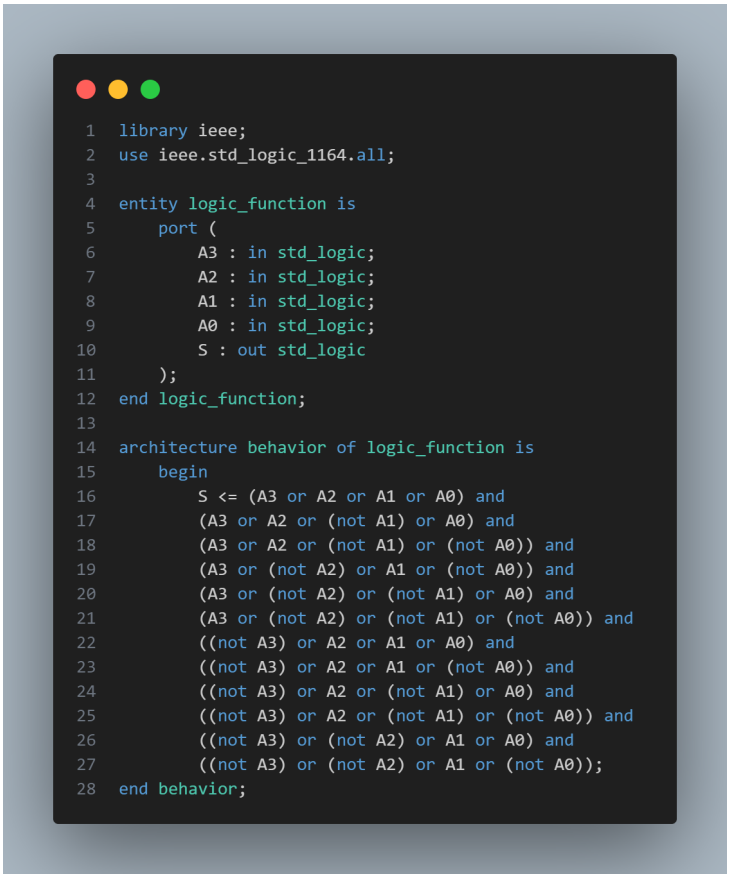
El código fue compilado y simulado en la plataforma EDA Playground utilizando el simulador GHDL, donde la entidad principal fue `logic_function_tb`.

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity logic_function is
    port (
        A3 : in std_logic;
        A2 : in std_logic;
        A1 : in std_logic;
        A0 : in std_logic;
        S  : out std_logic
    );
end logic_function;

architecture behavior of logic_function is
    begin
        S  <= ((not A3) and (not A2) and (not A1) and A0) or
        ((not A3) and A2 and (not A1) and (not A0)) or
        (A3 and A2 and (not A1) and (not A0)) or
        (A3 and A2 and A1 and A0);
end behavior;
```

Figura 3: Captura de pantalla del código SOP.



```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity logic_function is
    port (
        A3 : in std_logic;
        A2 : in std_logic;
        A1 : in std_logic;
        A0 : in std_logic;
        S : out std_logic
    );
end logic_function;

architecture behavior of logic_function is
    begin
        S <= (A3 or A2 or A1 or A0) and
        (A3 or A2 or (not A1) or A0) and
        (A3 or A2 or (not A1) or (not A0)) and
        (A3 or (not A2) or A1 or (not A0)) and
        (A3 or (not A2) or (not A1) or A0) and
        (A3 or (not A2) or (not A1) or (not A0)) and
        ((not A3) or A2 or A1 or A0) and
        ((not A3) or A2 or A1 or (not A0)) and
        ((not A3) or A2 or (not A1) or A0) and
        ((not A3) or A2 or (not A1) or (not A0)) and
        ((not A3) or (not A2) or A1 or A0) and
        ((not A3) or (not A2) or A1 or (not A0));
end behavior;
```

Figura 4: Captura de pantalla del código POS.

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity logic_function_tb is
end logic_function_tb;

architecture behavior of logic_function_tb is

    component logic_function is
        port (
            A3 : in std_logic;
            A2 : in std_logic;
            A1 : in std_logic;
            A0 : in std_logic;
            S  : out std_logic
        );
    end component logic_function;

    signal w_A3, w_A2, w_A1, w_A0, w_S : std_logic;

    begin
        and_gate_instance : logic_function
        port map (
            A3 => w_A3,
            A2 => w_A2,
            A1 => w_A1,
            A0 => w_A0,
            S => w_S
        );

        process begin
            -- Estimulación de las entradas
            w_A3 <= '0'; w_A2 <= '0'; w_A1 <= '0'; w_A0 <= '0';
            wait for 10 ns;

            w_A3 <= '0'; w_A2 <= '0'; w_A1 <= '0'; w_A0 <= '1';
            wait for 10 ns;

            w_A3 <= '0'; w_A2 <= '0'; w_A1 <= '1'; w_A0 <= '0';
            wait for 10 ns;

            w_A3 <= '0'; w_A2 <= '0'; w_A1 <= '1'; w_A0 <= '1';
            wait for 10 ns;

            w_A3 <= '0'; w_A2 <= '1'; w_A1 <= '0'; w_A0 <= '0';
            wait for 10 ns;

            w_A3 <= '0'; w_A2 <= '1'; w_A1 <= '0'; w_A0 <= '1';
            wait for 10 ns;

            w_A3 <= '0'; w_A2 <= '1'; w_A1 <= '1'; w_A0 <= '0';
            wait for 10 ns;

            w_A3 <= '0'; w_A2 <= '1'; w_A1 <= '1'; w_A0 <= '1';
            wait for 10 ns;

            w_A3 <= '1'; w_A2 <= '0'; w_A1 <= '0'; w_A0 <= '0';
            wait for 10 ns;

            w_A3 <= '1'; w_A2 <= '0'; w_A1 <= '0'; w_A0 <= '1';
            wait for 10 ns;

            w_A3 <= '1'; w_A2 <= '0'; w_A1 <= '1'; w_A0 <= '0';
            wait for 10 ns;

            w_A3 <= '1'; w_A2 <= '0'; w_A1 <= '1'; w_A0 <= '1';
            wait for 10 ns;

            w_A3 <= '1'; w_A2 <= '1'; w_A1 <= '0'; w_A0 <= '0';
            wait for 10 ns;

            w_A3 <= '1'; w_A2 <= '1'; w_A1 <= '0'; w_A0 <= '1';
            wait for 10 ns;

            w_A3 <= '1'; w_A2 <= '1'; w_A1 <= '1'; w_A0 <= '0';
            wait for 10 ns;

            w_A3 <= '1'; w_A2 <= '1'; w_A1 <= '1'; w_A0 <= '1';
            wait for 10 ns;

            -- Fin de la simulación
            wait;
        end process;
end behavior;

```


Figura 5: Captura de pantalla del código de simulación.

7. Paso 6: Análisis de Resultados y Verificación

La simulación permitió verificar que ambos circuitos (SOP y POS) responden correctamente a las diferentes combinaciones de entrada, de acuerdo con la tabla de verdad inicial. Se realizó un análisis comparativo de las salidas obtenidas y se constató la correcta implementación de la función lógica.

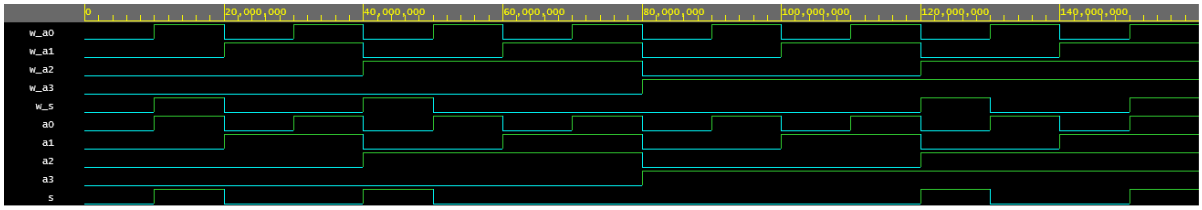


Figura 6: Resultados obtenidos de la simulación con SOP

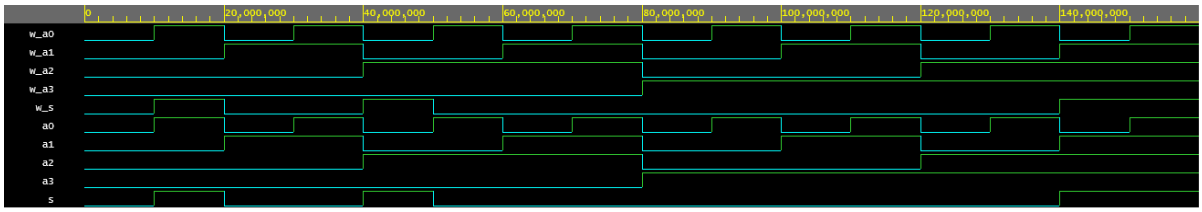


Figura 7: Resultados obtenidos de la simulación con POS

8. Conclusiones

El ejercicio permitió al estudiante:

- Comprender la relación entre la tabla de verdad, las ecuaciones lógicas y la implementación de circuitos digitales.
- Diferenciar y aplicar las metodologías SOP y POS en el diseño de circuitos.
- Desarrollar habilidades prácticas en el uso de VHDL para la descripción y simulación de sistemas digitales.
- Familiarizarse con herramientas de simulación como GHDL en EDA Playground.

Este proceso contribuye significativamente al entendimiento de los conceptos básicos del diseño digital y la lógica booleana, pilares fundamentales en la ingeniería electrónica y de sistemas.