

INGENIERÍA DE COMPUTADORES 3

Solución al Trabajo Práctico - Convocatoria extraordinaria 2026

Ejercicio 1

Se desea diseñar un circuito digital que implemente las funciones X e Y mostradas a continuación, que dependen de las cuatro variables A, B, C y D:

$$X = A'B'C' + A'B'C + A'BC' + A'BC + AB'D + ABD$$

$$Y = A'B'D + A'BD + AB'D + ABD + ABCD'$$

- 1.a) (0.5 puntos) Escriba en VHDL la **entity** del circuito que implemente las dos funciones lógicas. Es decir, que tenga cuatro entradas A, B, C y D, y dos salidas X e Y.
- 1.b) (1 punto) Escriba en VHDL la **architecture** que describa el *comportamiento* del circuito utilizando las expresiones lógicas sin simplificar.
- 1.c) (0.5 puntos) Simplifique las funciones lógicas x e y. Dibuje el diagrama de un circuito que implemente las funciones simplificadas al nivel de puertas lógicas. No es necesario obtener una implementación mínima en número de puertas, pero sí debe corresponderse con las expresiones lógicas obtenidas en el apartado anterior.

A continuación, escriba en VHDL la **entity** y la **architecture** de cada una de las puertas lógicas que componen el circuito que acaba de dibujar.

1.d) (1 punto) Escriba en VHDL una **architecture** que describa la *estructura* del circuito que ha dibujado, instanciando y conectando las puertas lógicas que ha diseñado anteriormente.

1.e) (1 punto) Escriba en VHDL un banco de pruebas que permita visualizar, para todos los posibles valores de las entradas, las salidas de los circuitos diseñados en los Apartados 1.b y 1.d.

Compruebe mediante inspección visual que ambos diseños producen los mismos resultados para todas las combinaciones posibles de entrada. Incluya en la memoria la captura de pantalla de los cronogramas obtenidos al realizar con el simulador de VHDL la simulación del banco de pruebas con los circuitos diseñados en los Apartados 1.b y 1.d.

Solución al Ejercicio 1

El Código VHDL 1.1 y 1.2 es una posible solución a los Apartados 1.a y 1.b.

El diagrama al nivel de puertas lógicas pedido en el Apartado 1.c está representado en la Figura 1.1.

El Código VHDL 1.3 – 1.5 es una posible descripción de las puertas lógicas AND de tres entradas, NOT y OR de tres entradas. El Código VHDL 1.6 es una posible descripción estructural del circuito, que corresponde con el diagrama mostrado en la Figura 1.1.

```

1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3 entity Circuito is
4     Port (
5         X : out STD_LOGIC;
6         Y : out STD_LOGIC;
7         A : in  STD_LOGIC;
8         B : in  STD_LOGIC;
9         C : in  STD_LOGIC;
10        D : in  STD_LOGIC);
11 end Circuito;
```

Código VHDL 1.1: Apartado 1.a: **entity** del circuito.

```

1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3 architecture Behavioral of Circuito is
4 begin
5     X <= ((not A) and (not B) and (not C))
6         or ((not A) and (not B) and C)
7         or ((not A) and B and (not C))
8         or ((not A) and B and C)
9         or (A and (not B) and D)
10        or (A and B and D);
11
12    Y <= ((not A) and (not B) and D)
13        or ((not A) and B and D)
14        or (A and (not B) and D)
15        or (A and B and D)
16        or (A and B and C and (not D));
17 end Behavioral;

```

Código VHDL 1.2: Apartado 1.b: **architecture** que describe el comportamiento.

```

1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3 entity AND3_GATE is
4     Port (
5         Y : out STD_LOGIC;
6         A : in  STD_LOGIC;
7         B : in  STD_LOGIC;
8         C : in  STD_LOGIC);
9 end AND3_GATE;
10 architecture Behavioral of AND3_GATE is
11 begin
12     Y <= A and B and C;
13 end Behavioral;

```

Código VHDL 1.3: Apartado 1.c: puerta AND de tres entradas.

```

1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4 entity NOT_GATE is
5     Port (
6         Y : out STD_LOGIC;
7         A : in  STD_LOGIC
8     );
9 end NOT_GATE;
10
11 architecture Behavioral of NOT_GATE is
12 begin
13     Y <= not A;
14 end Behavioral;

```

Código VHDL 1.4: Apartado 1.c: puerta NOT.

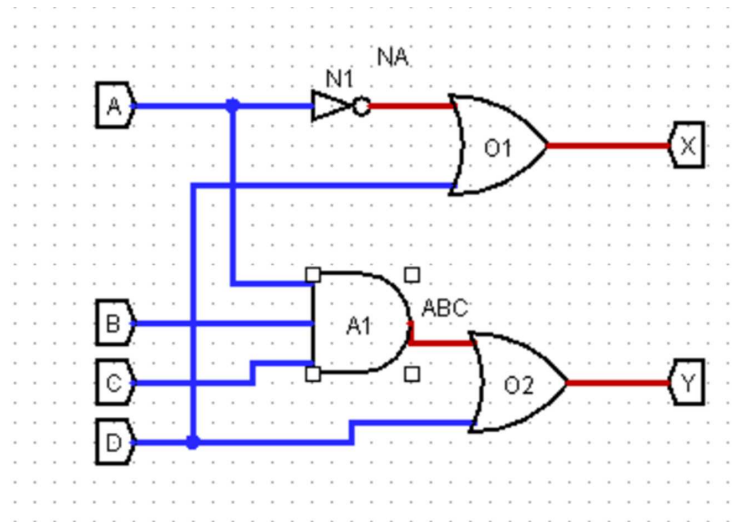


Figura 1.1: Apartado 1.c: diagrama al nivel de puertas lógicas.

```

1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4 entity OR2_GATE is
5     Port (
6         Y : out STD_LOGIC;
7         A : in  STD_LOGIC;
8         B : in  STD_LOGIC
9     );
10 end OR2_GATE;
11
12 architecture Behavioral of OR2_GATE is
13 begin
14     Y <= A or B;
15 end Behavioral;

```

Código VHDL 1.5: Apartado 1.c: puerta OR de dos entradas.

```

1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3 architecture Structural of Circuito is
4 --Funciones simplificada X=A'orD
5 --Y= D+(ABC)
6     signal NA : STD_LOGIC;
7     signal ABC : STD_LOGIC;
8     component NOT_GATE
9     Port (
10         Y : out STD_LOGIC;
11         A : in  STD_LOGIC);
12 end component;
13 component AND3_GATE
14 Port (
15     Y : out STD_LOGIC;

```

```

16         A : in   STD_LOGIC;
17         B : in   STD_LOGIC;
18         C : in   STD_LOGIC);
19     end component;
20     component OR2_GATE
21     Port (
22         Y : out  STD_LOGIC;
23         A : in   STD_LOGIC;
24         B : in   STD_LOGIC);
25     end component;
26 begin
27     N1 : NOT_GATE port map(Y => NA, A => A);
28     O1 : OR2_GATE port map(Y => X, A => NA, B => D);
29     A1 : AND3_GATE port map(Y => ABC, A => A, B => B, C => C);
30     O2 : OR2_GATE port map(Y => Y, A => D, B => ABC);
31 end Structural;

```

Código VHDL 1.6: Apartado 1.d: descripción estructural al nivel de puertas lógicas.

Finalmente, el Código VHDL 1.7 es un banco de pruebas para los diseños del circuito que implementa las dos funciones lógicas. En las Figura 1.2 se muestran las formas de onda obtenidas al simular el banco de pruebas con los dos diseños anteriormente realizados. La comprobación de que el diseño funciona correctamente debe hacerse en este caso mediante inspección visual.

```

1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.NUMERIC_STD.ALL;
4
5  entity TB_Circuito is
6  end TB_Circuito;
7
8  architecture Test of TB_Circuito is
9      signal A, B, C, D : STD_LOGIC;
10     signal X_beh, Y_beh : STD_LOGIC;
11     signal X_str, Y_str : STD_LOGIC;
12 begin
13     DUT1 : entity work.Circuito(Behavioral)
14         port map(
15             X => X_beh,
16             Y => Y_beh,
17             A => A,
18             B => B,
19             C => C,
20             D => D);
21     DUT2 : entity work.Circuito(Structural)
22         port map(
23             X => X_str,
24             Y => Y_str,
25             A => A,
26             B => B,

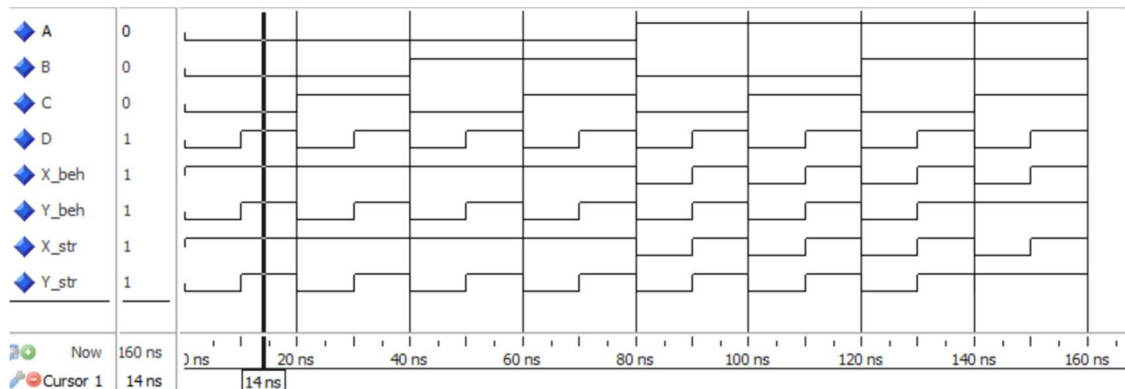
```

```

27         C => C,
28         D => D);
29     process
30         variable entradas : std_logic_vector(3 downto 0);
31     begin
32         for i in 0 to 15 loop
33             entradas := std_logic_vector(to_unsigned(i,4));
34             A <= entradas(3);
35             B <= entradas(2);
36             C <= entradas(1);
37             D <= entradas(0);
38             wait for 10 ns;
39             assert (X_beh = X_str)
40                 report "Error en la salida X";
41             assert (Y_beh = Y_str)
42                 report "Error en la salida Y";
43         end loop;
44         report "Fin de la simulacion";
45         wait;
46     end process;
47 end Test;

```

Código VHDL 1.7: Apartado 1.e: banco de pruebas.

**Figura 1.2:** Apartado 1.e: simulación del banco de pruebas del diseño describiendo el comportamiento del circuito.

Ejercicio 2

Se desea diseñar en VHDL una máquina expendedora capaz de vender tres productos distintos, devolver el cambio sobrante cuando exista sobrepago y permitir la cancelación de una compra en curso.

Los productos disponibles son:

Producto	Precio (euro)
A	1
B	1,50
C	2

La máquina acepta monedas de 50 céntimos y de 1 euro.

La entity del circuito es la siguiente:

```
entity VendingMachine is
  Port (
    DISP_A    : out STD_LOGIC;
    DISP_B    : out STD_LOGIC;
    DISP_C    : out STD_LOGIC;
    CHANGE50  : out STD_LOGIC;
    ERROR     : out STD_LOGIC;
    CLK       : in  STD_LOGIC;
    RST       : in  STD_LOGIC;
    M50       : in  STD_LOGIC;
    M100      : in  STD_LOGIC;
    SEL_A     : in  STD_LOGIC;
    SEL_B     : in  STD_LOGIC;
    SEL_C     : in  STD_LOGIC;
    CANCEL    : in  STD_LOGIC);
end VendingMachine;
```

Las señales de entrada y salida tienen la siguiente función:

- DISP_A: indica la dispensación del producto A.
- DISP_B: indica la dispensación del producto B.
- DISP_C: indica la dispensación del producto C.

- CHANGE50: salida utilizada para devolver dinero al usuario. Cada pulso equivale a la devolución de una moneda de 50 céntimos.
- ERROR: indica una condición de error.
- CLK: señal de reloj.
- RST: señal de reinicio síncrono.
- M50: inserción de una moneda de 50 céntimos.
- M100: inserción de una moneda de 1 euro.
- SEL_A: selección del producto A.
- SEL_B: selección del producto B.
- SEL_C: selección del producto C.
- CANCEL: cancelación de la compra y solicitud de devolución del dinero introducido.

El funcionamiento de la máquina se describe a continuación:

1. El usuario seleccionará un único producto mediante una de las señales SEL_A, SEL_B o SEL_C.
2. Si se activan simultáneamente dos o más señales de selección, la máquina deberá activar la señal ERROR.
3. Una vez seleccionado un producto, la máquina entrará en un estado de espera de monedas.
4. El saldo se almacenará internamente en múltiplos de 50 céntimos.
5. Los precios de los productos serán:
 - Producto A: 1 euro.
 - Producto B: 1,50 euros.
 - Producto C: 2 euros.
6. La máquina aceptará monedas de 50 céntimos y de 1 euro.
7. La activación simultánea de las entradas M50 y M100 se considerará una condición de error y provocará la transición al estado de error.

8. Cuando el saldo acumulado alcance o supere el precio del producto seleccionado, la máquina dispensará automáticamente el producto correspondiente.
9. Si el saldo acumulado supera el precio del producto, la diferencia deberá devolverse automáticamente al usuario utilizando la salida CHANGE50.
10. Si durante la compra se activa la entrada CANCEL, la máquina cancelará la operación y devolverá todo el saldo acumulado.
11. La devolución de dinero se realizará mediante pulsos sobre la salida CHANGE50. Cada pulso representará la devolución de una moneda de 50 céntimos.
12. Tras completar una dispensación, una devolución de cambio, una cancelación o una condición de error, la máquina volverá al estado inicial.

La implementación deberá utilizar obligatoriamente los siguientes estados:

- **IDLE**: estado inicial. La máquina espera una selección válida.
- **SELECTED**: espera de pago tras la selección de un producto.
- **DISPENSE**: activación de la salida de dispensación correspondiente.
- **DISPENSE_WAIT**: estado auxiliar para mantener visible la dispensación durante un ciclo adicional.
- **RETURN_MONEY**: estado encargado de devolver dinero mediante pulsos sobre CHANGE50.
- **RETURN_WAIT**: estado auxiliar para mantener visible cada devolución durante un ciclo adicional.
- **ERROR_STATE**: estado que activa la señal ERROR.
- **ERROR_WAIT**: estado auxiliar que mantiene visible la señal de error durante un ciclo adicional.

La codificación binaria de los estados queda a elección del alumno.

La máquina deberá seguir la siguiente lógica de funcionamiento:

1. El estado inicial es IDLE.
2. Desde IDLE, una selección válida provoca la transición a SELECTED.

3. Desde IDLE, una selección múltiple provoca la transición a ERROR_STATE.
4. En SELECTED se acumula el saldo introducido por el usuario.
5. Si se detecta la inserción simultánea de monedas mediante M50 y M100, la máquina pasa a ERROR_STATE.
6. Si durante SELECTED se activa CANCEL, la máquina pasa a RETURN_MONEY.
7. Cuando el saldo acumulado alcanza o supera el precio del producto seleccionado, la máquina pasa a DISPENSE.
8. Tras la dispensación, la máquina pasa a DISPENSE_WAIT.
9. Si existe cambio pendiente de devolución, la máquina pasa a RETURN_MONEY. En caso contrario regresa a IDLE.
10. La devolución de dinero se realizará alternando los estados RETURN_MONEY y RETURN_WAIT hasta devolver completamente el saldo pendiente.
11. Cuando no quede dinero por devolver, la máquina regresará al estado IDLE.
12. Desde ERROR_STATE se transitará a ERROR_WAIT.
13. Finalizada la gestión del error, la máquina volverá a IDLE.

Tenga en cuenta los siguientes requisitos de diseño:

1. La implementación deberá realizarse mediante una FSM síncrona gobernada por la señal CLK activa en el flanco de subida.
2. Todas las señales internas de almacenamiento deberán inicializarse exclusivamente mediante la señal RST.
3. Las salidas únicamente podrán activarse en los estados correspondientes.
4. Los estados DISPENSE y DISPENSE_WAIT deberán mantener visible la dispensación durante al menos un ciclo completo de reloj.
5. Los estados RETURN_MONEY y RETURN_WAIT deberán mantener visible la devolución de cada moneda durante al menos un ciclo completo de reloj.
6. Los estados ERROR_STATE y ERROR_WAIT deberán mantener visible la señal de error durante al menos un ciclo completo de reloj.

7. El cambio correspondiente a un sobrepago deberá devolverse automáticamente antes de regresar al estado IDLE.

2.a) (3 puntos)

Diseñe en VHDL la `architecture` que describa el comportamiento de la máquina expendedora. El alumno podrá realizar las consideraciones de diseño que estime oportunas para resolver situaciones no especificadas explícitamente en el enunciado, siempre que dichas decisiones no entren en contradicción con los requisitos funcionales establecidos ni alteren el comportamiento exigido de la máquina.

2.b) (3 puntos)

Programe en VHDL el banco de pruebas del circuito diseñado en el apartado anterior.

El banco de pruebas deberá comprobar, como mínimo, los siguientes escenarios:

- Compra exacta del producto A.
- Compra exacta del producto B.
- Compra exacta del producto C.
- Compra del producto B con sobrepago y devolución automática del cambio.
- Cancelación de una compra tras introducir una moneda de 50 céntimos.
- Selección simultánea de varios productos.
- Inserción simultánea de monedas de 50 céntimos y 1 euro.

Las comprobaciones deberán realizarse de forma automática mediante sentencias `assert` y deberán generar mensajes descriptivos cuando el comportamiento observado no coincida con el esperado.

Al finalizar la simulación, el banco de pruebas deberá mostrar un mensaje indicando que todas las pruebas han sido ejecutadas correctamente.

Incluya en la memoria el cronograma obtenido al realizar la simulación.

Solución al Ejercicio 2

El Código VHDL 1.8 muestra el diseño de la máquina de vending.

```

1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3 use IEEE.NUMERIC_STD.ALL;
4 architecture Behavioral of VendingMachine is
5     type state_t is (
6         IDLE, SELECTED,
7         DISPENSE, DISPENSE_WAIT,
8         RETURN_MONEY, RETURN_WAIT,
9         ERROR_STATE, ERROR_WAIT);
10    constant PRICE_A : integer := 2;
11    constant PRICE_B : integer := 3;
12    constant PRICE_C : integer := 4;
13    signal state : state_t := IDLE;
14    signal product : std_logic_vector(1 downto 0) := "00";
15    signal credit : integer range 0 to 10 := 0;
16    signal pending_change : integer range 0 to 10 := 0;
17 begin
18     process(CLK)
19         variable next_credit : integer range 0 to 10;
20         variable price_v : integer range 0 to 4;
21     begin
22         if rising_edge(CLK) then
23             DISP_A <= '0';
24             DISP_B <= '0';
25             DISP_C <= '0';
26             CHANGE50 <= '0';
27             ERROR <= '0';
28             if RST='1' then
29                 state <= IDLE;
30                 product <= "00";
31                 credit <= 0;
32                 pending_change <= 0;
33             else
34                 case product is
35                     when "00" => price_v := PRICE_A;
36                     when "01" => price_v := PRICE_B;
37                     when "10" => price_v := PRICE_C;
38                     when others => price_v := 0;
39                 end case;
40                 case state is
41                     when IDLE =>
42                         credit <= 0;
43                         pending_change <= 0;
44                     if SEL_A='1' and SEL_B='0' and SEL_C='0' then
45                         product <= "00";
46                         state <= SELECTED;
47                     elsif SEL_A='0' and SEL_B='1' and SEL_C='0' then
48                         product <= "01";
49                         state <= SELECTED;
50                     elsif SEL_A='0' and SEL_B='0' and SEL_C='1' then
51                         product <= "10";
52                         state <= SELECTED;
53                     elsif (SEL_A='1' and SEL_B='1') or
54                         (SEL_A='1' and SEL_C='1') or
55                         (SEL_B='1' and SEL_C='1') then

```

```

56     state <= ERROR_STATE;
57 end if;
58 when SELECTED =>
59     if CANCEL='1' then
60         pending_change <= credit;
61         state <= RETURN_MONEY;
62     elsif M50='1' and M100='1' then
63         state <= ERROR_STATE;
64     else
65         next_credit := credit;
66         if M50='1' then
67             next_credit := credit + 1;
68         elsif M100='1' then
69             next_credit := credit + 2;
70         end if;
71         credit <= next_credit;
72         if next_credit >= price_v then
73             pending_change <= next_credit - price_v;
74             credit <= 0;
75             state <= DISPENSE;
76         end if;
77     end if;
78 when DISPENSE =>
79     case product is
80         when "00" => DISP_A <= '1';
81         when "01" => DISP_B <= '1';
82         when "10" => DISP_C <= '1';
83         when others => null;
84     end case;
85     state <= DISPENSE_WAIT;
86 when DISPENSE_WAIT =>
87     case product is
88         when "00" => DISP_A <= '1';
89         when "01" => DISP_B <= '1';
90         when "10" => DISP_C <= '1';
91         when others => null;
92     end case;
93     if pending_change > 0 then
94         state <= RETURN_MONEY;
95     else
96         product <= "00";
97         state <= IDLE;
98     end if;
99 when RETURN_MONEY =>
100    if pending_change > 0 then
101        CHANGE50 <= '1';
102        pending_change <= pending_change - 1;
103        state <= RETURN_WAIT;
104    else
105        product <= "00";
106        credit <= 0;
107        state <= IDLE;
108    end if;
109 when RETURN_WAIT =>
110    CHANGE50 <= '1';

```

```

111     if pending_change > 0 then
112         state <= RETURN_MONEY;
113     else
114         product <= "00";
115         credit <= 0;
116         state <= IDLE;
117     end if;
118     when ERROR_STATE =>
119         ERROR <= '1';
120         state <= ERROR_WAIT;
121     when ERROR_WAIT =>
122         ERROR <= '1';
123         product <= "00";
124         credit <= 0;
125         pending_change <= 0;
126         state <= IDLE;
127     end case;
128 end if;
129 end if;
130 end process;
131 end Behavioral;

```

Código VHDL 1.8: Apartado 2.a: diseño de la máquina de vending.

El banco de pruebas del circuito se muestra en el Código VHDL 1.9. El cronograma obtenido al simular el banco de pruebas, usando como circuito a testear el diseño del Apartado 2.a, se muestra en la Figura 1.3.

```

1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  entity TB_VendingMachine is
4  end TB_VendingMachine;
5  architecture Test of TB_VendingMachine is
6      signal DISP_A    : STD_LOGIC;
7      signal DISP_B    : STD_LOGIC;
8      signal DISP_C    : STD_LOGIC;
9      signal CHANGE50  : STD_LOGIC;
10     signal ERROR     : STD_LOGIC;
11     signal CLK       : STD_LOGIC := '0';
12     signal RST       : STD_LOGIC := '0';
13     signal M50       : STD_LOGIC := '0';
14     signal M100      : STD_LOGIC := '0';
15     signal SEL_A     : STD_LOGIC := '0';
16     signal SEL_B     : STD_LOGIC := '0';
17     signal SEL_C     : STD_LOGIC := '0';
18     signal CANCEL    : STD_LOGIC := '0';
19 begin
20     DUT : entity work.VendingMachine
21     port map (
22         DISP_A    => DISP_A,
23         DISP_B    => DISP_B,
24         DISP_C    => DISP_C,

```

```

25     CHANGE50 => CHANGE50,
26     ERROR    => ERROR,
27     CLK      => CLK,
28     RST      => RST,
29     M50      => M50,
30     M100     => M100,
31     SEL_A    => SEL_A,
32     SEL_B    => SEL_B,
33     SEL_C    => SEL_C,
34     CANCEL   => CANCEL );
35 CLK <= not CLK after 10 ns;
36 process
37 begin
38     -----
39     -- RESET
40     -----
41     RST <= '1';
42     wait for 25 ns;
43     RST <= '0';
44     wait for 40 ns;
45     -----
46     -- CASO 1
47     -- Compra exacta producto A
48     -----
49     SEL_A <= '1';
50     wait until rising_edge(CLK);
51     SEL_A <= '0';
52     M100 <= '1';
53     wait until rising_edge(CLK);
54     M100 <= '0';
55     wait until rising_edge(CLK);
56     wait until rising_edge(CLK);
57     wait for 10 ns;
58     assert DISP_A = '1'
59     report "ERROR: No se ha dispensado el producto A";
60     -----
61     -- CASO 2
62     -- Compra exacta producto B
63     -----
64     wait until rising_edge(CLK);
65     SEL_B <= '1';
66     wait until rising_edge(CLK);
67     SEL_B <= '0';
68     M100 <= '1';
69     wait until rising_edge(CLK);
70     M100 <= '0';
71     M50 <= '1';
72     wait until rising_edge(CLK);
73     M50 <= '0';
74     wait until rising_edge(CLK);
75     wait until rising_edge(CLK);
76     wait for 10 ns;
77     assert DISP_B = '1'
78     report "ERROR: No se ha dispensado el producto B";
79     -----
80     -- CASO 3

```

```

81      -- Compra exacta producto C
82      -----
83      wait until rising_edge(CLK);
84      SEL_C <= '1';
85      wait until rising_edge(CLK);
86      SEL_C <= '0';
87      M100 <= '1';
88      wait until rising_edge(CLK);
89      M100 <= '0';
90      M100 <= '1';
91      wait until rising_edge(CLK);
92      M100 <= '0';
93      wait until rising_edge(CLK);
94      wait until rising_edge(CLK);
95      wait for 10 ns;
96      assert DISP_C = '1'
97      report "ERROR: No se ha dispensado el producto C";
98      -----
99      -- CASO 4
100     -- Cancelación tras introducir 50 céntimos
101     -----
102     wait until rising_edge(CLK);
103     SEL_B <= '1';
104     wait until rising_edge(CLK);
105     SEL_B <= '0';
106     M50 <= '1';
107     wait until rising_edge(CLK);
108     M50 <= '0';
109     CANCEL <= '1';
110     wait until rising_edge(CLK);
111     CANCEL <= '0';
112     wait until CHANGE50 = '1' for 500 ns;
113     assert CHANGE50 = '1'
114     report "ERROR: No se ha devuelto el dinero tras CANCEL";
115     -----
116     -- CASO 5
117     -- Selección múltiple
118     -----
119     SEL_A <= '0';
120     SEL_B <= '0';
121     SEL_C <= '0';
122     M50 <= '0';
123     M100 <= '0';
124     CANCEL <= '0';
125     wait until rising_edge(CLK);
126     wait until rising_edge(CLK);
127     SEL_A <= '1';
128     SEL_B <= '1';
129     wait until ERROR = '1' for 100 ns;
130     assert ERROR = '1'
131     report "ERROR: No se ha detectado selección múltiple";
132     SEL_A <= '0';
133     SEL_B <= '0';
134     -----
135     -- Fin
136     -----

```

```

137     report "TODAS LAS PRUEBAS FINALIZADAS";
138     wait;
139     end process;
140 end Test;

```

Código VHDL 1.9: Apartado 2.b: banco de pruebas del circuito.

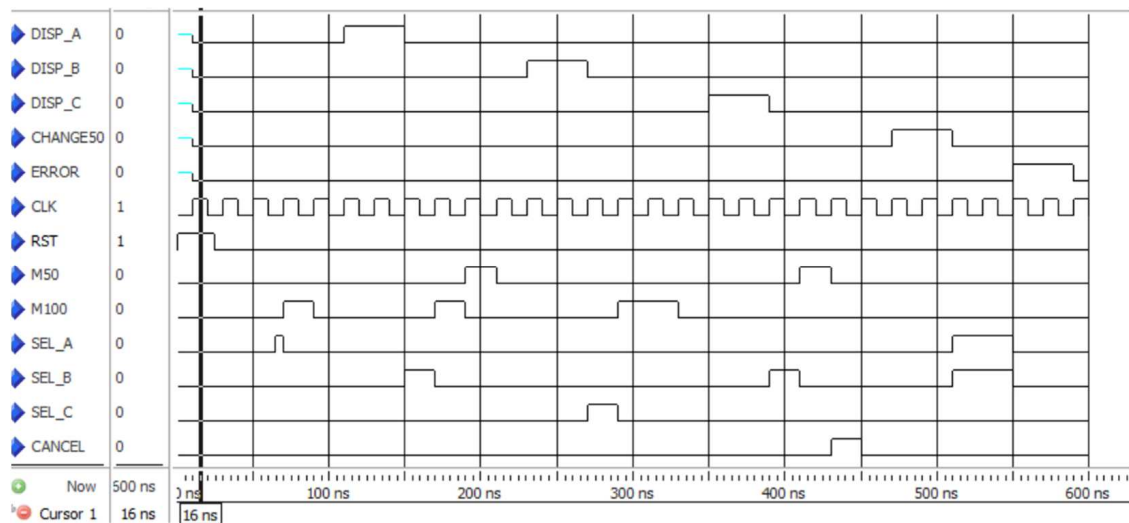


Figura 1.3: Cronograma del Apartado 2.b comprobando el comportamiento del circuito conversor.