Tema 3: Diseño de Circuitos Combinacionales

Escuela Técnica Superior de Ingeniería

Departamento de
Ingeniería Electrónica, Sistemas Informáticos y Automática

Circuitos Combinacionales

Los *Circuitos y Sistemas Combinacionales* son aquellos en los que en cada instante, el estado lógico de su salida dependen única y exclusivamente de sus entradas.

Metodología clásica de diseño de Circuitos Combinacionales sencillos

El diseño se realiza generalmente a partir del planteamiento de un problema.

1. **SE OBTIENE LA TABLA DE VERDAD DE CADA UNA DE LAS SALIDAS Y, OPCIONALMENTE, LAS EXPRESIONES CANÓNICAS**
2. **SE PROCEDE A LA SIMPLIFICACIÓN PARA OBTENER UNA EXPRESIÓN BOOLEANA MÍNIMA PARA CADA FUNCIÓN**
3. **POR ÚLTIMO SE REALIZA EL DIAGRAMA LÓGICO DEL CIRCUITO DE MÍNIMO TAMAÑO OBTENIDO MEDIANTE LA SIMPLIFICACIÓN.**

Ejemplo:

Para abrir una caja fuerte se dispone de tres llaves, la caja se abre si:

- **Están giradas A y B independientemente de si lo está C.**
- **Cuando estando girada C, estén giradas A o B.**

Expresión canónica:

$F(C,B,A) = m_3 + m_5 + m_6 + m_7 = \Sigma_3(3,5,6,7)$

$$F(C,B,A) = \overline{C}BA + C\overline{B}A + CB\overline{A} + CBA$$

C	B	A	F(C,B,A)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Métodos de simplificación

La simplificación de funciones lógicas tiene por objeto obtener:

1. **Menor número de términos en la función** (que equivalen a menor número de PUERTAS LÓGICAS)
2. **Menor número de variables en cada término** (que equivalen a menor número de ENTRADAS DE LAS DIVERSAS PUERTAS).

Los métodos más usuales se describen brevemente a continuación.

SIMPLIFICACIÓN TABULAR

Este tipo de simplificación se realiza mediante tablas y mapas que representan la tabla de verdad. Útil para funciones con hasta 5 variables. El método más usual es el **MAPA DE KARNAUGH**. Este método nos garantiza también la máxima simplificación posible.

SIMPLIFICACIÓN NUMÉRICA DE QUINE-McCLUSKEY

Es un método numérico que permite escoger de todas las simplificaciones posibles de una función, la que pueda ser implementada con el menor número de elementos. Nos garantiza la máxima simplificación posible, y se usa para funciones con muchas variables y/o multisalidas. Puede realizarse por ordenador.

SIMPLIFICACIÓN ALGEBRAICA

Uno de los métodos de simplificación posibles es la simplificación algebraica. Aunque no es un método propiamente dicho, aplicando los postulados y teoremas del Álgebra de Boole podemos obtener una versión simplificada de la función, aunque sin garantías de que la simplificación obtenida sea la óptima.

Ejemplo: $F(C,B,A) = \overline{C}BA + C\overline{B}A + CB\overline{A} + CBA$

$F(C,B,A) = \overline{B}A + C\overline{A} + CB$

Método de Karnaugh (I)

Se realiza mediante un diagrama de cuadros o celdas dónde cada una de ellas representa una línea de la tabla de verdad de la función, o sea, un mintérmino o un maxtérmino.

La principal característica del mapa es que las celdas adyacentes físicamente, corresponden a términos adyacentes lógicamente, o sea, la diferencia entre una celda y las adyacentes es el cambio en una y solo una de las variables.

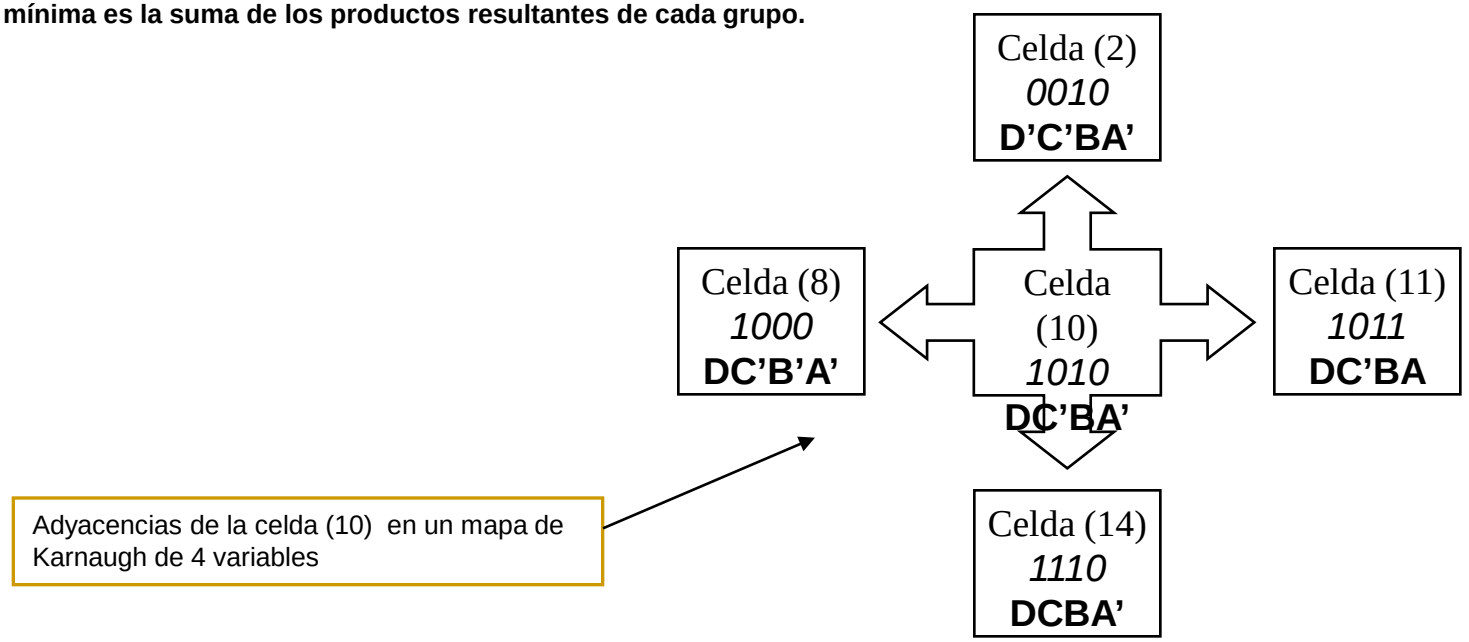
	D	C	B	A	F
(0)	0	0	0	0	
(1)	0	0	0	1	
(2)	0	0	1	0	
(3)	0	0	1	1	
(4)	0	1	0	0	
•	•	•	•	•	
(15)	1	1	1	1	

CD \ AB	00	01	11	10
00	(0)	(2)	(3)	(1)
01	(8)	(10)	(11)	(9)
11	(12)	(14)	(15)	(13)
10	(4)	(6)	(7)	(5)

Método de Karnaugh (II)

OBTENCIÓN DE LA EXPRESIÓN MÍNIMA EN FORMA DE SUMA DE PRODUCTOS

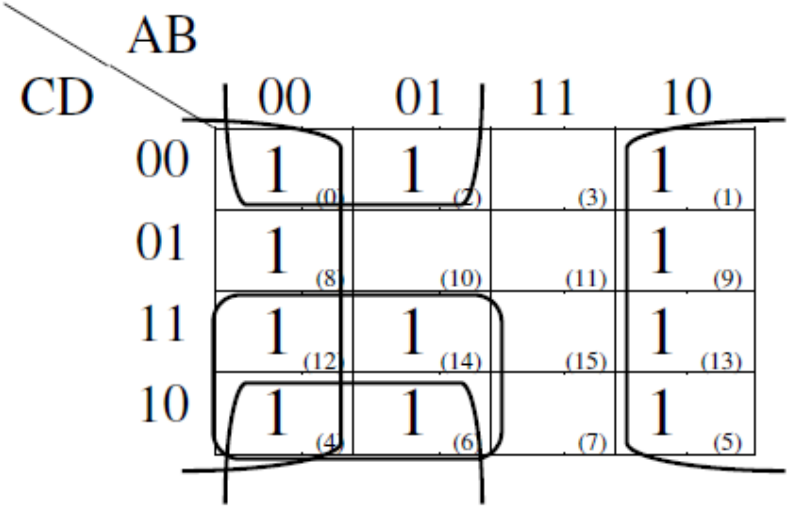
1. **Marcamos en el mapa un '1' en cada mintérmino que representa la función.**
2. **Mediante líneas cerradas hacemos agrupaciones de 'unos' adyacentes. Estos grupos pueden contener un número de 'unos' correspondiente a potencias de 2, o sea, 1, 2, 4, 8,...**
3. **Se deben escoger el menor número de grupos, pero que contengan el mayor número de 'unos', de manera que todos ellos queden cubiertos.**
4. **Para obtener la expresión, cada grupo representa un producto. Las variables que cambien de valor dentro del grupo quedan eliminadas. El producto se obtiene asignando la variable sin negar al '1' y la negada al '0'.**
5. **La expresión mínima es la suma de los productos resultantes de cada grupo.**

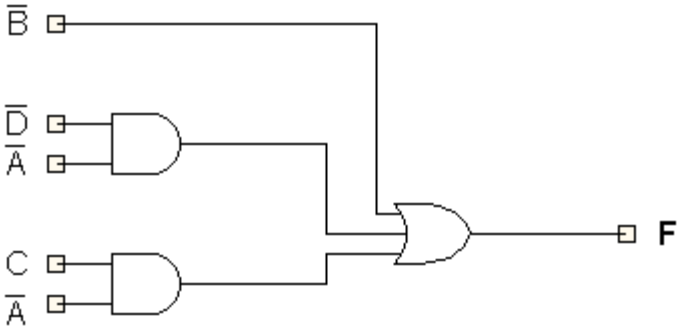


Método de Karnaugh (III)

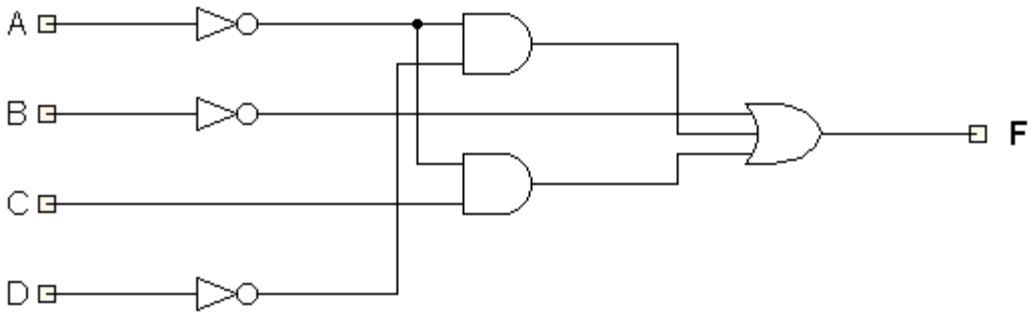
Ejemplo:

- Simplificar la función $F(D,C,B,A) = \Sigma_4(0,1,2,4,5,6,8,9,12,13,14)$.





- Hemos formado 3 grupos:
  - **{(0),(8),(12),(4),(1),(9),(13),(5)} ⇒ B'**
  - **{(0),(2),(4),(6)} ⇒ D'A'**
  - **{(12),(14),(4),(6)} ⇒ CA'**



La función simplificada queda así:

$$F = \overline{D}\,\overline{A} + C\overline{A} + \overline{B}$$

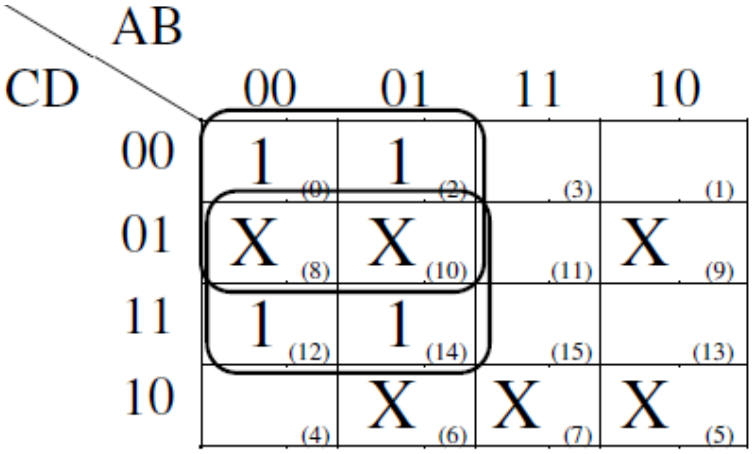
Método de Karnaugh (IV)

EXPRESIÓN MÍNIMA PARA FUNCIONES INCOMPLETAS

1. **Para funciones incompletas, aquellas que no están definidas para algunas combinaciones de las variables de entrada, marcamos en el mapa de Karnaugh una 'X' en los lugares correspondientes a esas combinaciones, también llamadas condiciones "no importa" (don't care).**
2. **Estas 'X' podemos considerarlas '0' ó '1' y proceder como en los casos anteriores.**
3. **Solo es necesario cubrir todos los unos (si estamos realizando la simplificación como suma de productos), o todos los ceros (si estamos obteniendo producto de sumas). Las condiciones de no importa sólo deben cogerse si son útiles para que el grupo contenga un número mayor de términos.**

Ejemplo:

Simplificar la función $F(D,C,B,A) = \Sigma_4(0,2,12,14) + \Sigma_\Phi(5,6,7,8,9,10)$



Hemos formado 2 grupos:

{(0),(2),(8),(10)} ⇒ C'A'
{(8),(10),(12),(14)} ⇒ DA'

Por tanto ⇒ **F = DA' + C'A'**

Como se ve, hemos tratado las condiciones no importa (8) y (10) como unos y el resto como ceros.

Método de Karnaugh (V)

OBTENCIÓN DE LA EXPRESIÓN MÍNIMA EN FORMA DE PRODUCTO DE SUMAS

1. **Marcamos en el mapa un '0' en cada maxtérmino que representa la función..**
2. **Mediante líneas cerradas hacemos agrupaciones de ceros adyacentes. Estos grupos pueden contener un número de 'ceros' correspondiente a potencias de 2, o sea, 1, 2, 4, 8,...**
3. **Se deben escoger el menor número de grupos, pero que contengan el mayor número de 'ceros', de manera que todos ellos queden cubiertos.**
4. **Para obtener la expresión, cada grupo representa una suma. La variable que cambie de valor dentro del grupo queda eliminada. El producto se obtiene asignando la variable sin negar al 0 y la negada al 1.**
5. **La expresión mínima es el producto de las sumas resultantes de cada grupo.**

Ejemplo:

- Simplificar la misma función F(D,C,B,A) = Π_4(3,7,10,11,15).
  - Hemos formado 2 grupos:
    - **{(3),(11),(15),(7)} ⇒ (B' + A')**
    - **{(10),(11)} ⇒ (D' + C + B')**

Por tanto:

$$F = (\overline{D} + C + \overline{B})(\overline{B} + \overline{A})$$

- Se puede comprobar que es la misma función:

CD \ AB	00	01	11	10
00	(0)	(2)	0 (3)	(1)
01	(8)	0 (10)	0 (11)	(9)
11	(12)	(14)	0 (15)	(13)
10	(4)	(6)	0 (7)	(5)

$$F = (\overline{D} + C + \overline{B})(\overline{B} + \overline{A}) = \overline{D}\overline{B} + \overline{D}\overline{A} + C\overline{B} + C\overline{A} + \overline{B} + \overline{B}\overline{A} = \overline{B} + \overline{D}\overline{A} + C\overline{A}$$

Método de Karnaugh (VI)

MAPAS COMPACTOS DE 5 VARIABLES

Es posible construir un mapa de 5 variables a partir de un mapa de 4 variables, según se observa a continuación. En este mapa, cada casilla representa dos términos producto. Para el primero de ellos, la quinta variable de la función vale 0 (términos del 0 al 15) y para el segundo término, vale 1 (términos del 16 al 31).

***Para representar la función en el mapa se siguen estas reglas*:**

1. **Si los dos términos de una casilla forman parte de la función, ponemos un 1 en esa casilla (suponemos que estamos obteniendo suma de productos).**
2. **Si sólo el primer término de una casilla pertenece a la función, ponemos el nombre de la quinta variable negada en esa casilla.**
3. **Si sólo el segundo término de una casilla pertenece a la función, ponemos el nombre de la quinta variable sin negar en esa casilla.**
4. **Las condiciones no importa son comodines que nos sirven para poner en las casillas lo más beneficioso para la simplificación.**

***Para obtener los grupos se siguen las siguientes reglas*:**

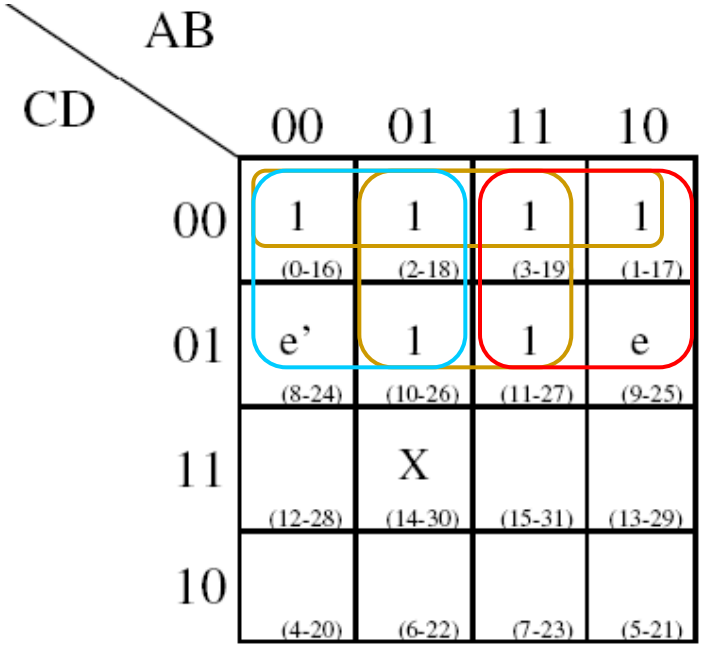
1. **Se cubren todos los unos que formen parte de la función aprovechando, si es posible, las condiciones no importa (todos ellos con los grupos necesarios)**
2. **Se cubre la quinta variable negada aprovechando, si es posible, los unos y condiciones no importa para realizar los mayores grupos posibles.**
3. **Se cubre la quinta variable sin negar aprovechando, si es posible, los unos y condiciones no importa para realizar los mayores grupos posibles.**

La expresión algebraica de los grupos se obtiene igual, con la excepción de que aquellos grupos con la quinta variable directa o negada dentro de ellos deben añadir a su expresión algebraica esa quinta variable tal y como aparece en su interior (directa o negada).

Método de Karnaugh (VII)

Ejemplo:

Simplificar mediante Karnaugh la función: $f = \sum_{5}(0,1,2,3,8,10,16,17,18,19,25,27) + \sum_{\phi}(11,14,26,30)$



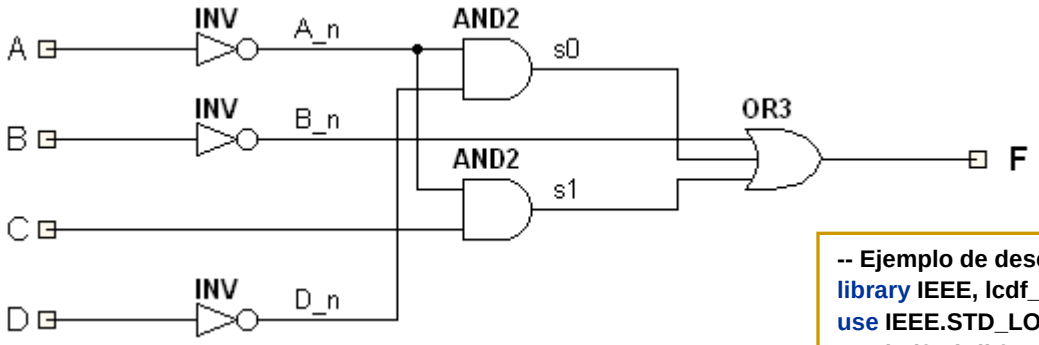
La función simplificada es por tanto: $f = \overline{c}\,\overline{d} + b\overline{c} + \overline{a}\,c\,\overline{e} + a\overline{c}e$

Una pequeña introducción a los Lenguajes de Descripción de Hardware (I)

- **Son lenguajes parecidos a los lenguajes de programación, pero están orientados a describir estructuras hardware y sus comportamientos.**
- **Se distinguen principalmente porque describen operaciones en paralelo.**
- **Una de sus aplicaciones es proporcionar al diseñador una alternativa a los esquemáticos.**
- **El lenguaje que veremos durante la asignatura es VHDL (*VHSIC Hardware Description Language*, o más correctamente *Versatile Hardware Description Language*). Este lenguaje fue desarrollado inicialmente por el Departamento de Defensa de EEUU, convirtiéndose después en un lenguaje estándar del *IEEE (Instituto de Ingenieros Eléctricos y Electrónicos)*. VHSIC son las iniciales de *Very High Speed Integrated Circuits.***
- **En VHDL se puede describir un circuito de tres formas diferentes:**
  - ***Descripción estructural*: Cuando se emplea para describir interconexiones de componentes. Una descripción de este tipo refleja por tanto un netlist del circuito.**
  - ***Descripción de flujo de datos*: En este caso se describe su función en vez de su estructura y se lleva a cabo mediante sentencias concurrentes. Un ejemplo de este tipo de descripción son las ecuaciones booleanas.**
  - ***Descripción algorítmica*: En este caso, se describe el circuito mediante algoritmos.**
- **Existen otros lenguajes de descripción de hardware, como por ejemplo *Verilog*.**
- **Se puede emplear una estructura HDL denominada *testbench (banco de test)*, que sirve para verificar el correcto funcionamiento de un circuito.**

Una pequeña introducción a los Lenguajes de Descripción de Hardware (II)

Ejemplo: Descripción VHDL ***estructural*** del circuito de la figura



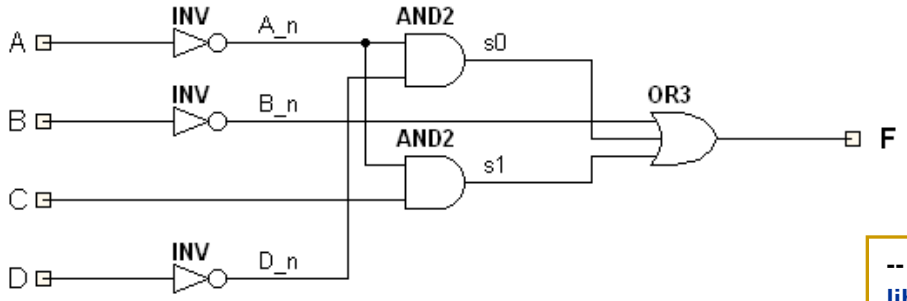

```
-- Ejemplo de descripción estructural
library IEEE, lcdf_vhdl;
use IEEE.STD_LOGIC_1164.ALL;
use lcdf_vhdl.func_prims.all

entity funcion_1_vhdl_est is
   port ( A : in  STD_LOGIC;
          B : in  STD_LOGIC;
          C : in  STD_LOGIC;
          D : in  STD_LOGIC;
          F : out  STD_LOGIC);
end funcion_1_vhdl_est;

architecture estructura_1 of funcion_1_vhdl_est is
      signal A_n, B_n, D_n, s0, s1: std_logic;
begin
      p0: INV port map (in1 => A, out1 => A_n);
      p1: INV port map (in1 => B, out1 => B_n);
      p2: INV port map (in1 => D, out1 => D_n);
      p3: AND2 port map (in1 => A_n, in2 => D_n, out1 => s0);
      p4: AND2 port map (in1 => A_n, in2 => C, out1 => s1);
      p5: OR3 port map (in1 => B_n, in2 => s0, in3 => s1, out1 => F);
end estructura_1;
```

Una pequeña introducción a los Lenguajes de Descripción de Hardware (III)

Ejemplo: Descripción VHDL como ***flujo de datos*** del circuito de la figura.




```
-- Ejemplo de descripción de flujo de datos
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity funcion_1_vhdl_flujo is
   port ( A : in  STD_LOGIC;
          B : in  STD_LOGIC;
          C : in  STD_LOGIC;
          D : in  STD_LOGIC;
          F : out  STD_LOGIC);
end funcion_1_vhdl_flujo;

architecture flujo_de_datos_1 of funcion_1_vhdl_flujo is
      signal A_n, B_n, D_n: std_logic;
      begin
      A_n <= not A;
      B_n <= not B;
      D_n <= not D;
      F <= (A_n and D_n) or (A_n and C) or B_n;
end flujo_de_datos_1;
```