FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

PLD-MEMORIAS
DISPOSITIVOS LÓGICOS PROGRAMABLES

Prof. L. F. Caram

PLD-MEMORIAS

PLD: Programmable Logic Device (Dispositivo Lógico Programables)

Variables o Líneas de Entrada

Líneas de Datos

Matriz de OR's

Datos

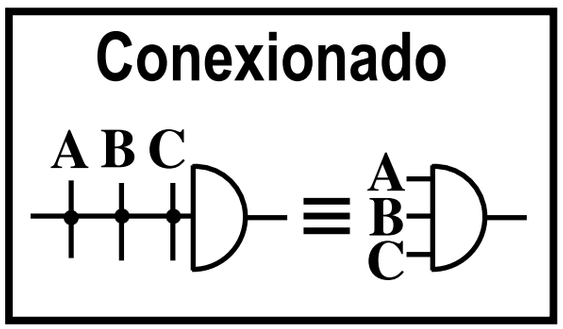
Palabra

Direcciones o Líneas de Direcciones

Matriz de AND's

A B C

Conexionado



Cantidad de direcciones
= Número de **AND'S**

Ancho de palabra
= Número de **OR'S**

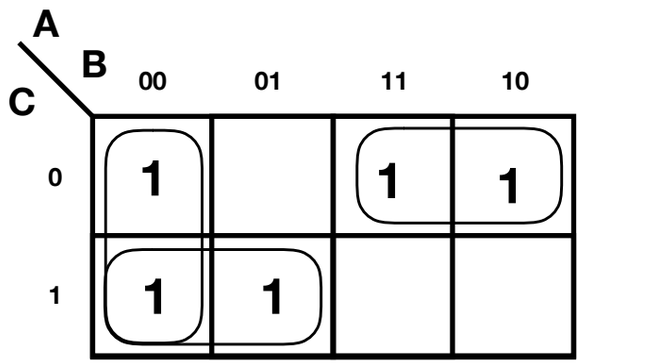
CAPACIDAD =
Número de AND'S x Número de OR'S

Variables o Líneas de Salida → F_0 F_1 F_n

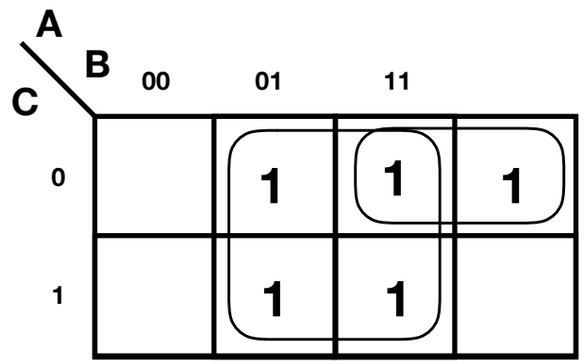
PLD-MEMORIAS

EJEMPLO: Implementar las siguientes funciones: $F_0 = \sum_m (0, 1, 3, 4, 6)$; $F_1 = \sum_m (0, 1, 3, 5, 7)$; $F_2 = \sum_m (2, 3, 4, 6, 7)$ con ROM, PAL y PLA.

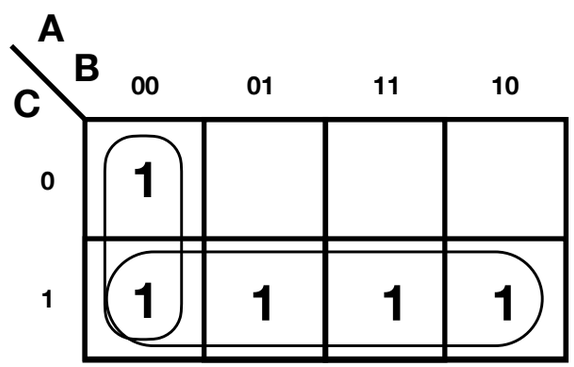
A	B	C	F_0	F_1	F_2
0	0	0	1	1	0
0	0	1	1	1	0
0	1	0	0	0	1
0	1	1	1	1	1
1	0	0	1	0	1
1	0	1	0	1	0
1	1	0	1	0	1
1	1	1	0	1	1



$$F_0 = \overline{A}.\overline{B} + \overline{A}.C + A.\overline{C}$$



$$F_2 = A.\overline{C} + B$$



$$F_1 = \overline{A}.\overline{B} + C$$

PLD-MEMORIAS

ROM: Read Only Memory (Memoria de Solo Lectura)

Relación entre Cantidad de Direcciones y Variables de Entradas

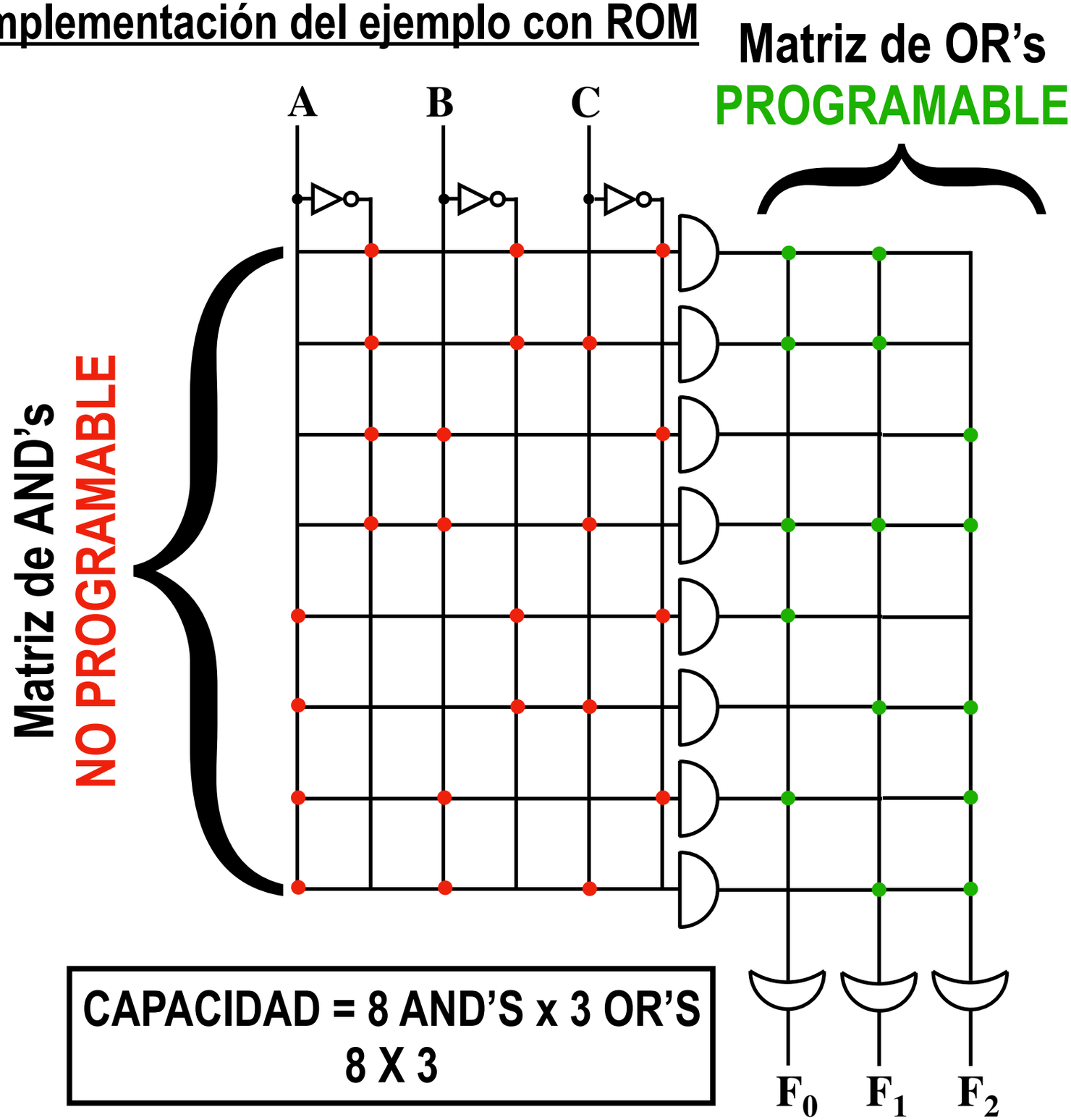
 # de Direcciones = $2^{\text{\# de Entradas}}$

En la **ROM**, la matriz de **AND's**, **NO** se puede Programar, viene de fabrica grabada por defecto con todos los minitérminos, como en un decodificador. Por lo tanto el número de variables de entrada determina el número de direcciones (cantidad de **AND's**).

La matriz de **OR's** se puede programar sin ninguna restricción.

Por lo tanto la implementación es directa desde la tabla de verdad colocaré un punto (conexión) donde la función valga 1 (minitérmino). Conviene tener la función expresada como suma de minitérminos y no simplificarla.

Implementación del ejemplo con ROM



CAPACIDAD = 8 AND'S x 3 OR'S
8 X 3

PLD-MEMORIAS

PAL: Programable Array Logic (Lógica de Arreglo Programable)

En la **PAL**, la matriz de **OR's**, **NO** se puede Programar, viene de fabrica grabada por defecto con una cantidad determinada de direcciones por función de salida y **NO** se pueden reutilizar los términos grabados en otras direcciones que no pertenezcan a la función que estamos implementando. Es decir no se podrá poner mas de un punto rojo (conexión) en cada fila de la matriz de **OR's**.

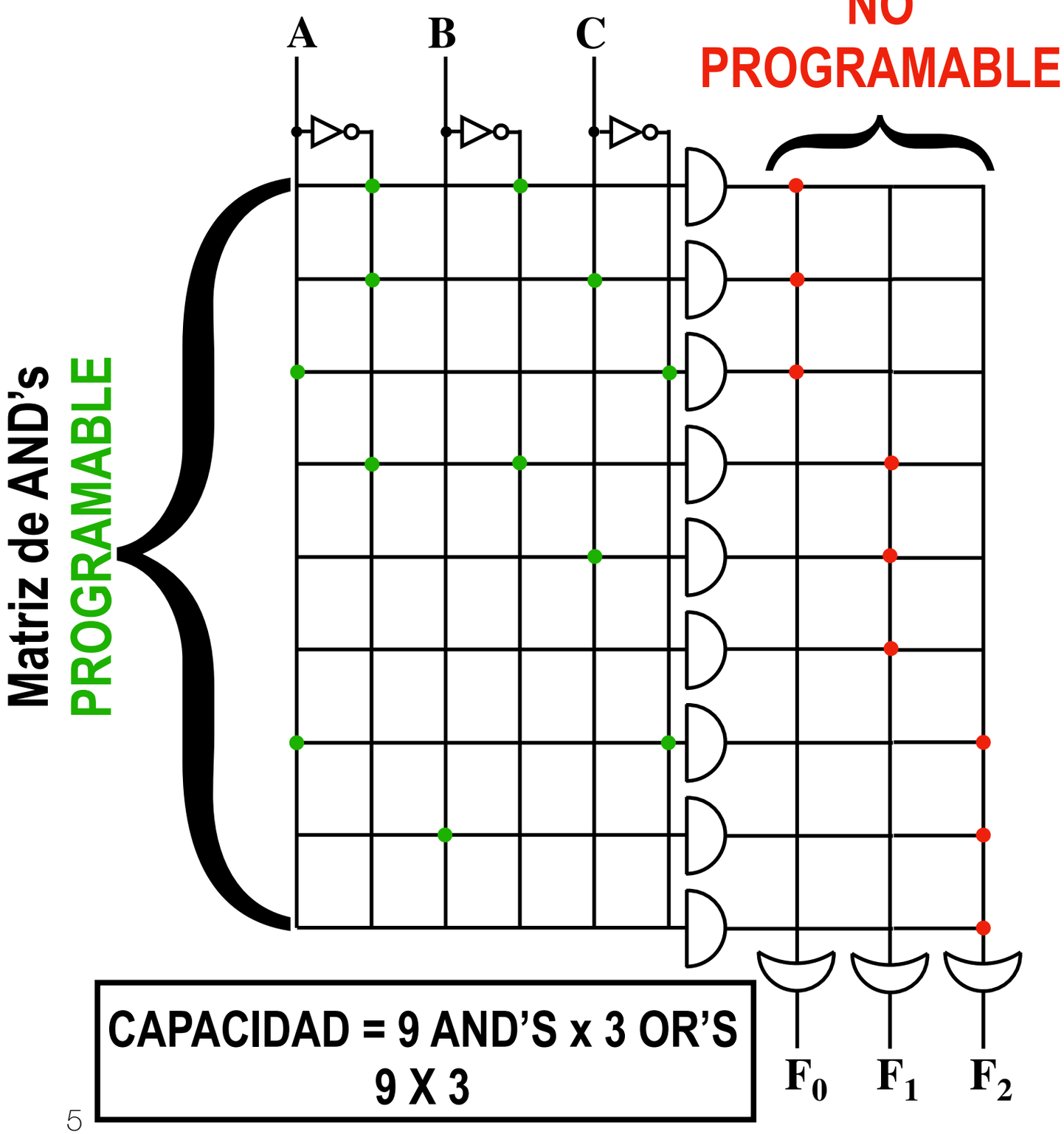
En la matriz de **AND's** se pueden programar sin ninguna restricción los términos simplificados, respetando los grupos de direcciones de cada función de salida.

Por lo tanto para la implementación conviene simplificar la función.

Debo elegir una memoria que tenga tantas direcciones por función de salida como la función del problema a implementar tenga la mayor cantidad de términos una vez simplificada.

Observación: no se utilizan algunas direcciones, ya que hay funciones que tienen menos términos que otras a implementar.

Implementación del ejemplo con PAL



CAPACIDAD = 9 AND'S x 3 OR'S
9 X 3

PLD-MEMORIAS

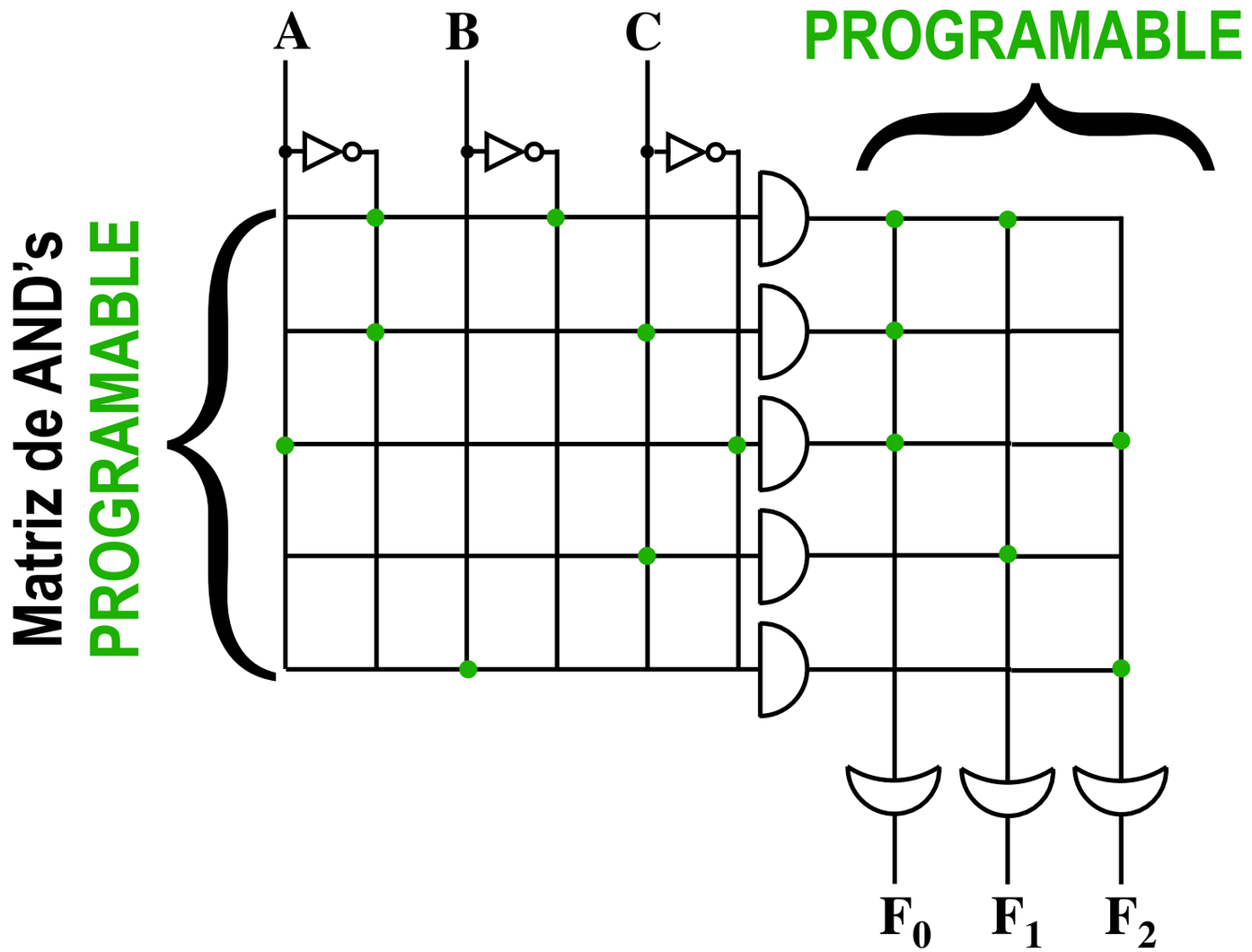
PLA: Programable Logic Array (Arreglo Lógico Programable)

En la **PLA**, ambas matrices de **OR's** y de **AND's** se pueden Programar sin ninguna restricción y se pueden reutilizar los términos grabados en otras direcciones que no pertenezcan a la función que estamos implementando. La cantidad de direcciones (cantidad de AND's) se calcula sumando todos los términos distintos de las funciones simplificadas.

Por lo tanto para la implementación conviene simplificar la función.

Observación: **NO** se dejan direcciones sin utilizar, ya que hay funciones que tienen términos iguales entonces se comparten en la implementación.

Implementación del ejemplo con PLA



CAPACIDAD = 5 AND'S x 3 OR'S
5 X 3

PLD-MEMORIAS

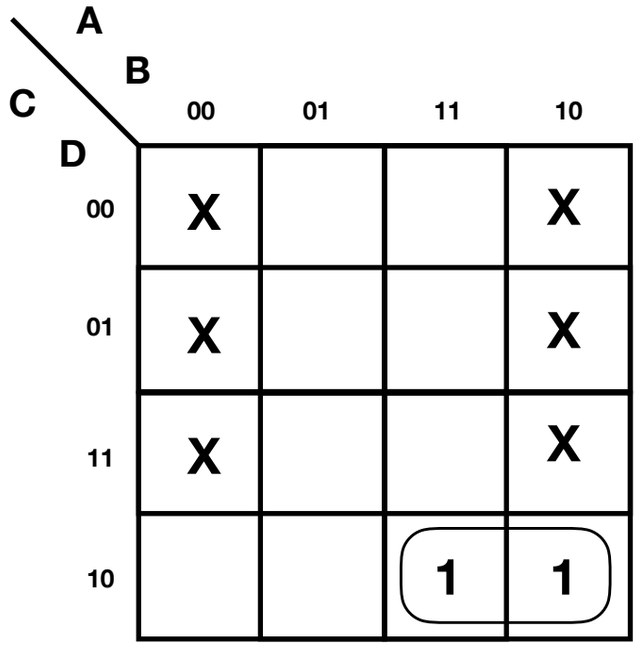
CONCLUSIONES: ROM - PAL - PLA

PLD / CARACTERÍSTICAS	ROM	PAL	PLA
MATRIZ DE AND's	NO PROGRAMABLE	PROGRAMABLE	PROGRAMABLE
MATRIZ DE OR's	PROGRAMABLE	NO PROGRAMABLE	PROGRAMABLE
FUNCIÓN SIMPLIFICADA	NO	SI	SI
NÚMERO DE DIRECCIONES (AND's)	NºAND's = 2 NºENTRADAS	NºAND's = CANTIDAD DE FUNCIONES A IMPLEMENTAR X CANTIDAD DE TÉRMINOS DE LA FUNCIÓN SIMPLIFICADA CON MAS TÉRMINOS.	NºAND's = CANTIDAD DE TÉRMINOS DISTINTOS DE TODAS LAS FUNCIONES SIMPLIFICADAS.
OBSERVACIONES		GRUPO DE DIRECCIONES EXCLUSIVO DE CADA FUNCIÓN DE SALIDA. NO SE PUEDEN COMPARTIR CON OTRA FUNCIÓN. NO MAS DE UN PUNTO POR FILA EN LA MATRIZ DE OR's.	SE USAN TODAS LAS DIRECCIONES.

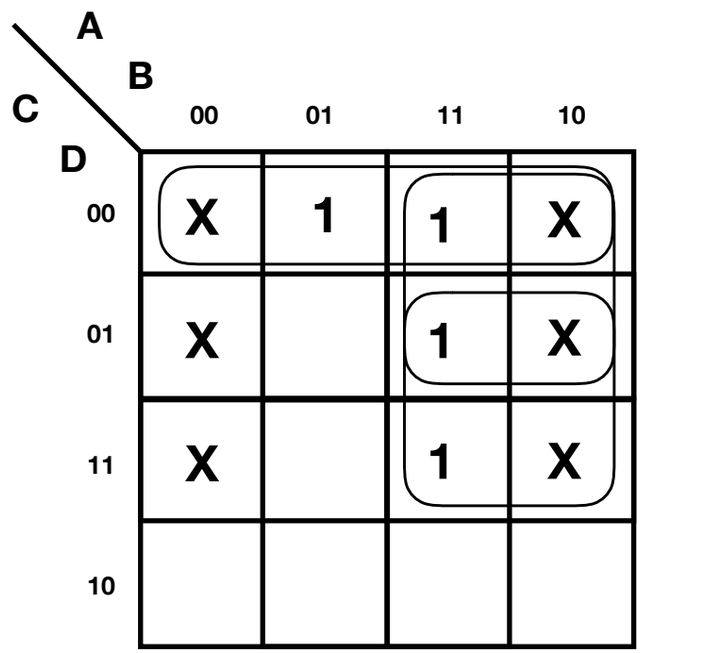
PLD-MEMORIAS

Ejercicio 7: Convertir el código BCD exceso 3 Gray a BCD 8421 utilizando, alternativamente, ROM, PAL y PLA. Comparar los resultados obtenidos.

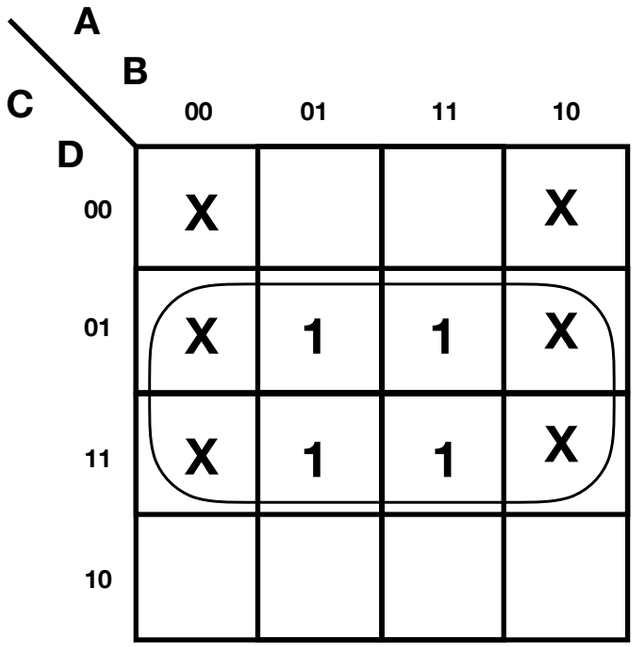
	BCD GRAY EX – 3				BCD 8 – 4 – 2 – 1			
N°	A	B	C	D	A′	B′	C′	D′
	0	0	0	0	X	X	X	X
	0	0	0	1	X	X	X	X
	0	0	1	1	X	X	X	X
0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	0	1
2	0	1	1	1	0	0	1	0
3	0	1	0	1	0	0	1	1
4	0	1	0	0	0	1	0	0
5	1	1	0	0	0	1	0	1
6	1	1	0	1	0	1	1	0
7	1	1	1	1	0	1	1	1
8	1	1	1	0	1	0	0	0
9	1	0	1	0	1	0	0	1
	1	0	1	1	X	X	X	X
	1	0	0	1	X	X	X	X
	1	0	0	0	X	X	X	X



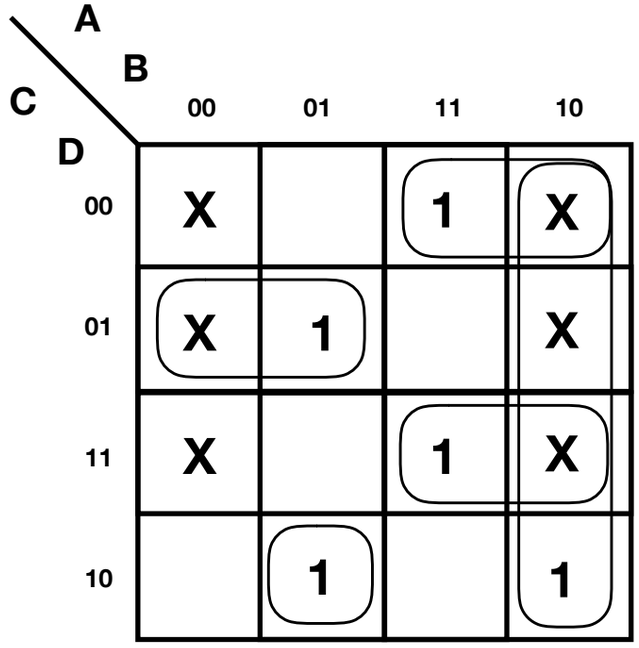
$$A' = AC\overline{D}$$



$$B' = AD + A\overline{C} + \overline{C}\overline{D}$$



$$C' = D$$

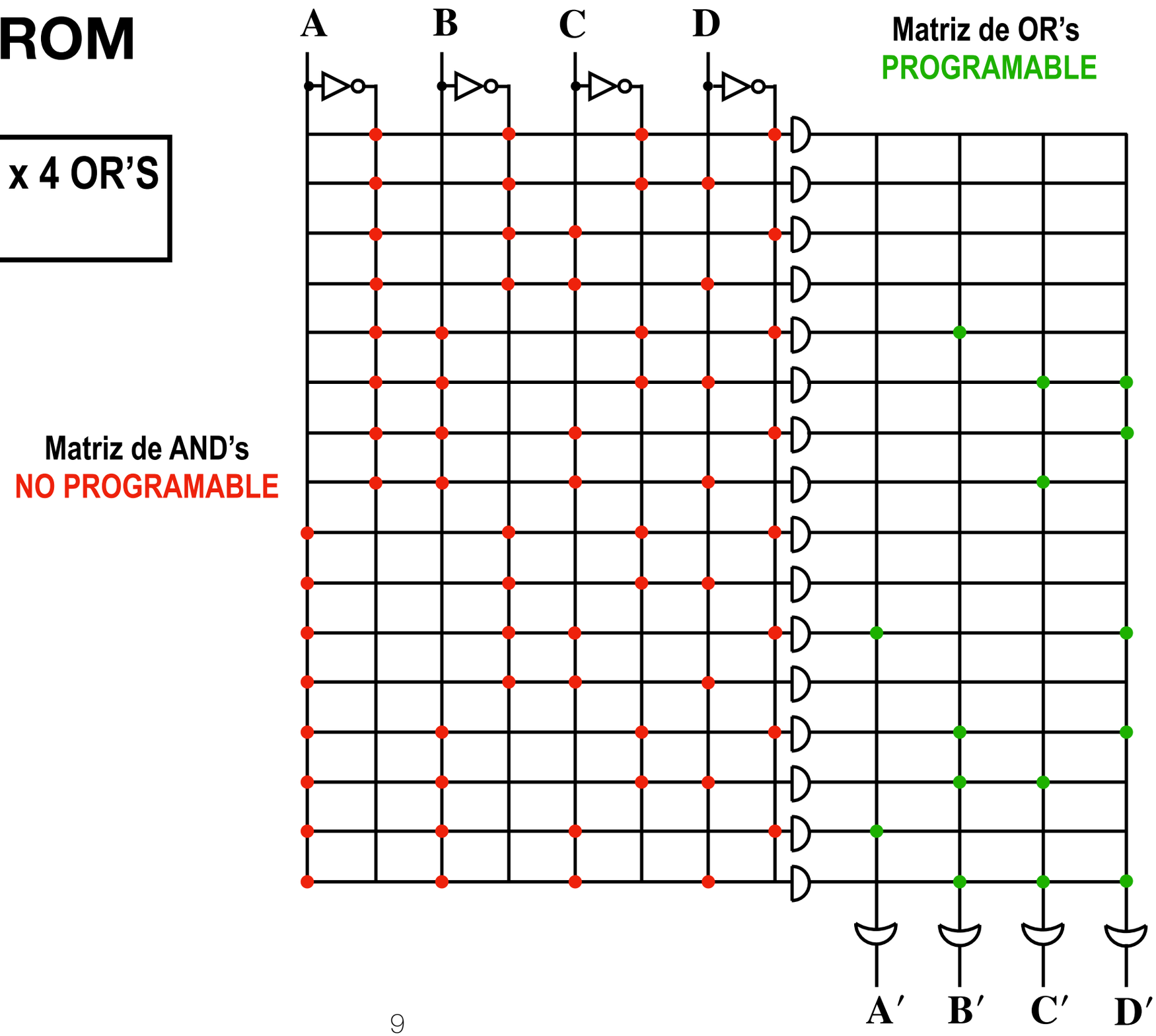


$$D' = A\overline{B} + A\overline{C}\overline{D} + \overline{A}\overline{C}D + ACD + \overline{A}BC\overline{D}$$

PLD-MEMORIAS

Implementación del ejercicio 7 con **ROM**

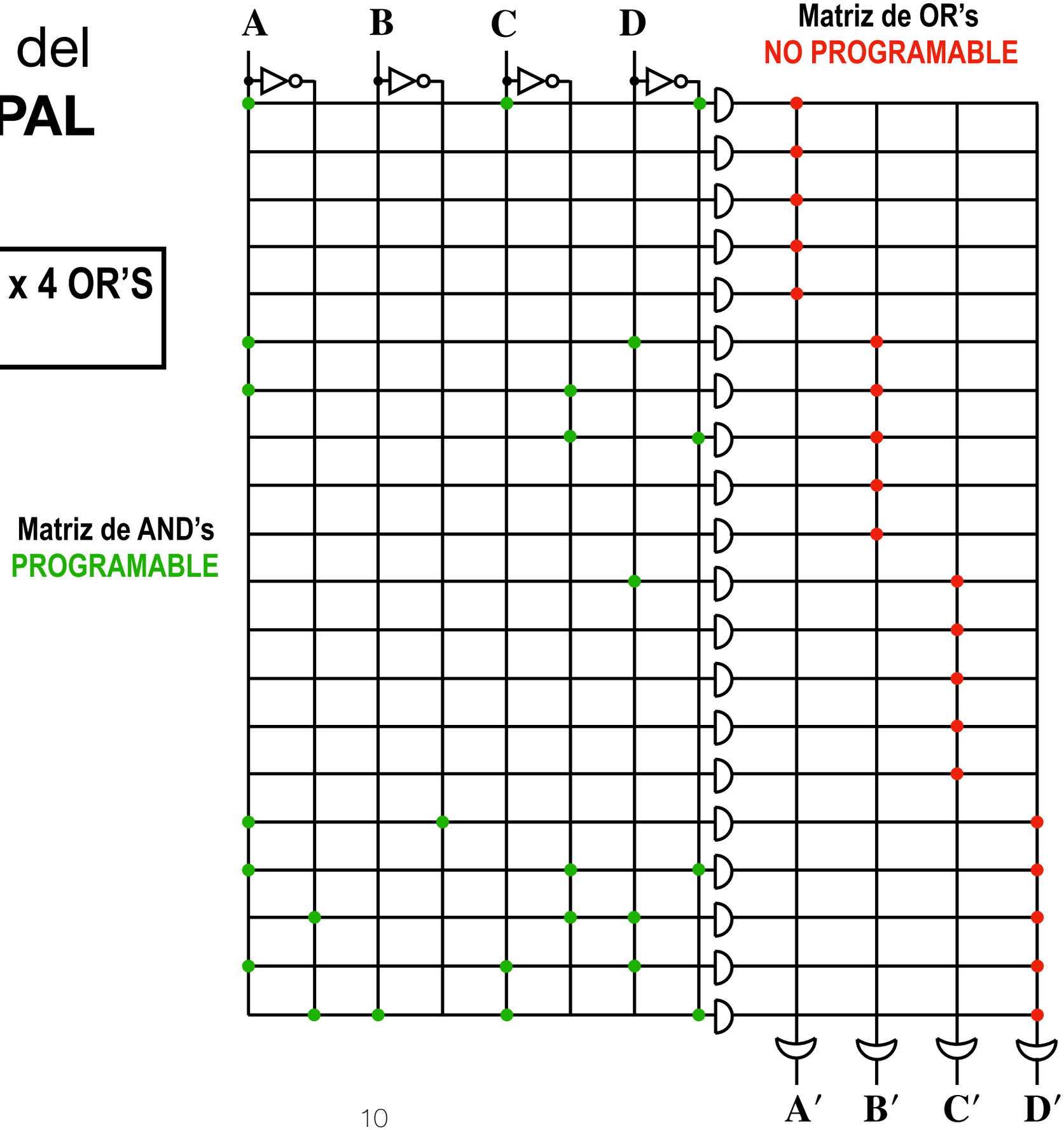
CAPACIDAD = 16 AND'S x 4 OR'S
16 X 4



PLD-MEMORIAS

Implementación del ejercicio 7 con **PAL**

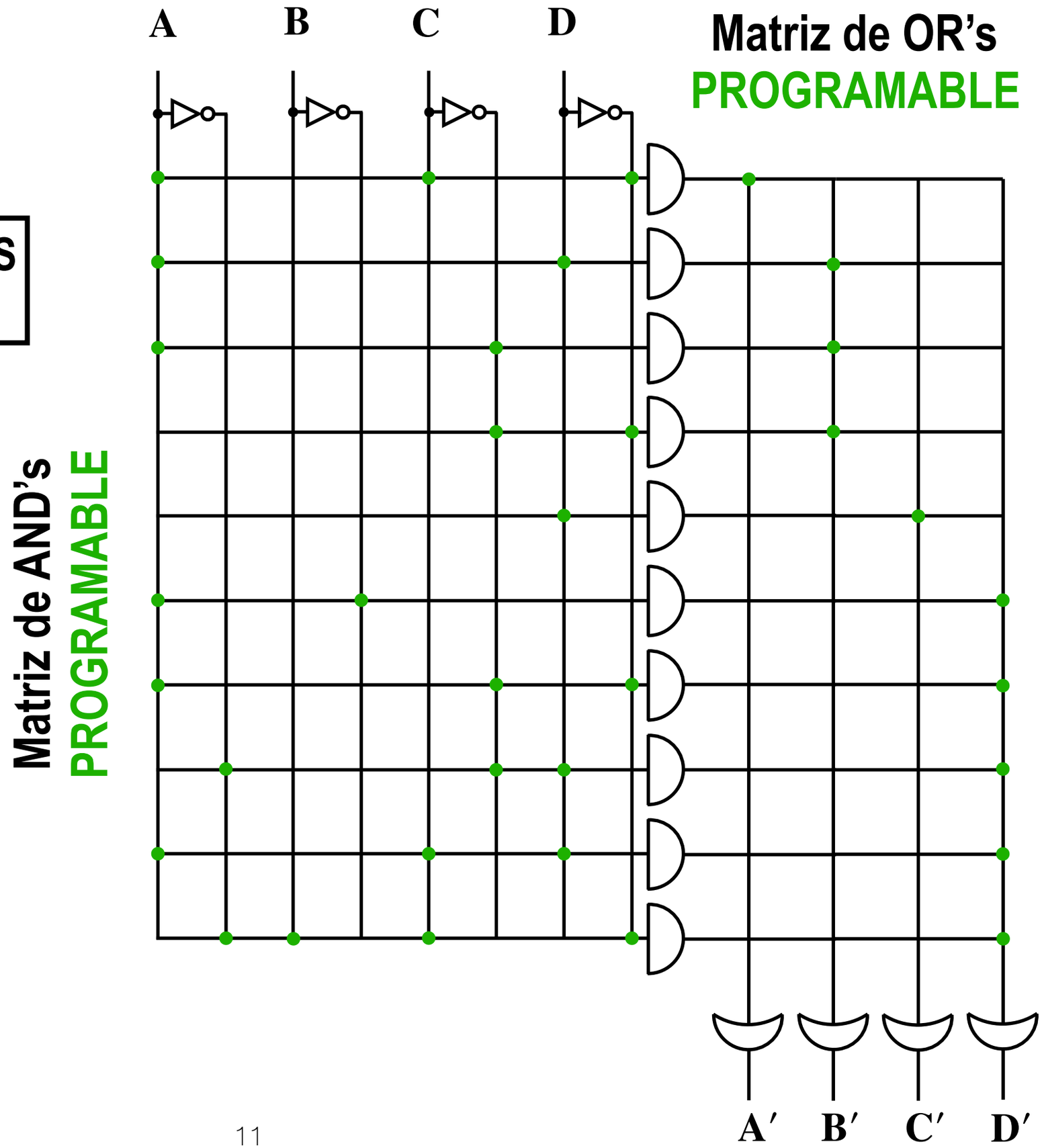
CAPACIDAD = 20 AND'S x 4 OR'S
20 X 4



PLD-MEMORIAS

Implementación del ejercicio 7 con **PLA**

CAPACIDAD = 10 AND'S x 4 OR'S
10 X 4

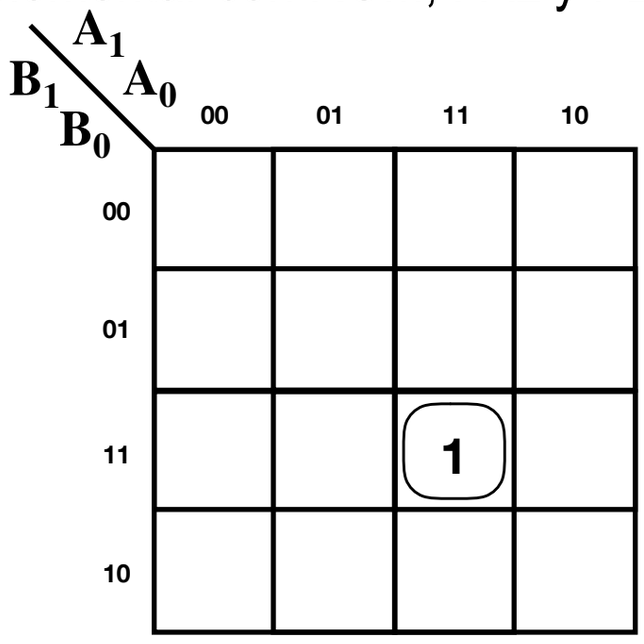


PLD-MEMORIAS

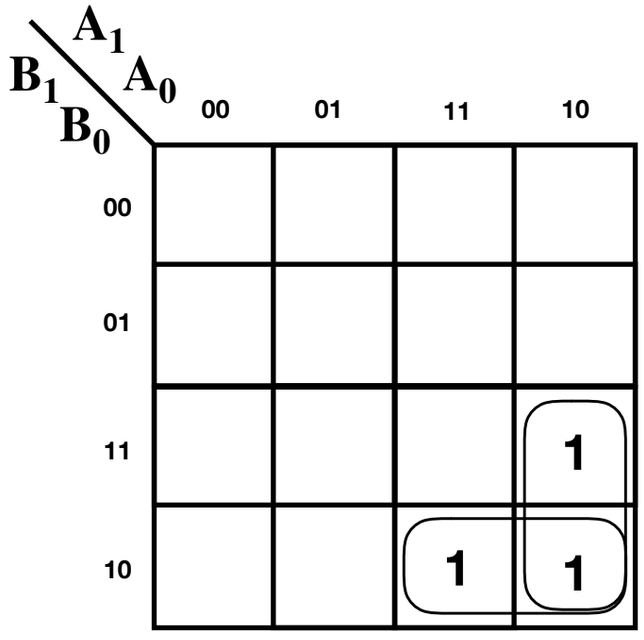
Ejercicio: Realizar un circuito que multiplique dos palabras de 2 bits, siendo $\mathbf{A}$ y $\mathbf{B}$ dos palabras de 2 bits cada una, es decir $\mathbf{A} = \mathbf{A_1A_0}$ y $\mathbf{B} = \mathbf{B_1B_0}$. Implementar con ROM, PAL y PLA.

$$
\begin{array}{cccccc}
 & & & \times & A_1 & A_0 \\
 & & & & B_1 & B_0 \\
\hline
 & & & & A_1 B_0 & A_0 B_0 \\
+ & & A_1 B_1 & & A_0 B_1 & \\
\hline
 & S_3 & S_2 & & S_1 & S_0
\end{array}
$$

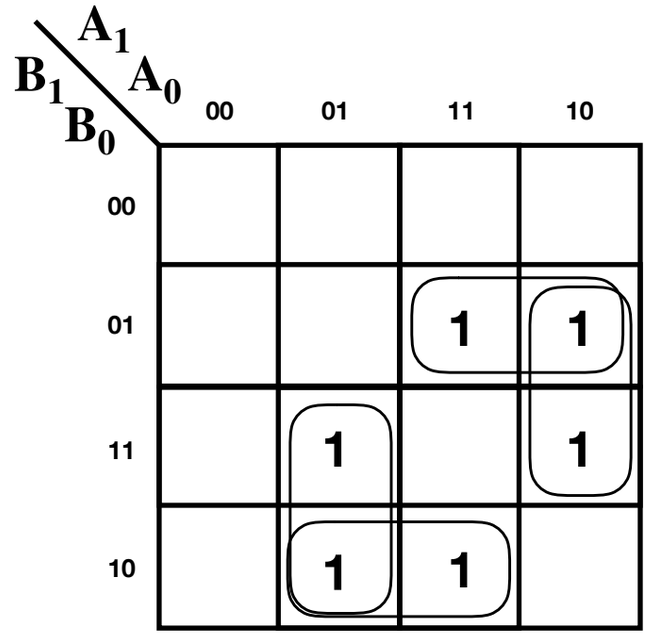
A_1	A_0	B_1	B_0	S_3	S_2	S_1	S_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1



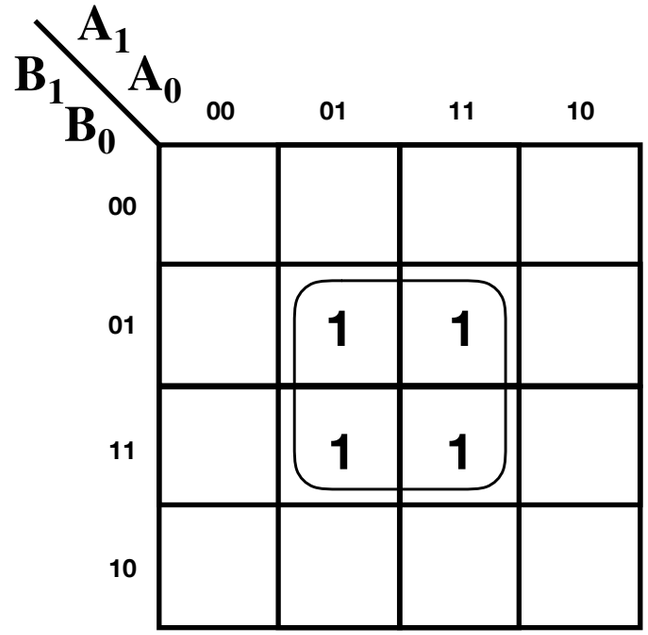
$$S_3 = A_1A_0B_1B_0$$



$$S_2 = A_1\overline{A}_0B_1 + A_1B_1\overline{B}_0$$



$$S_1 = A_1\overline{B}_1B_0 + A_0B_1\overline{B}_0 + A_1\overline{A}_0B_0 + \overline{A}_1A_0B_1$$



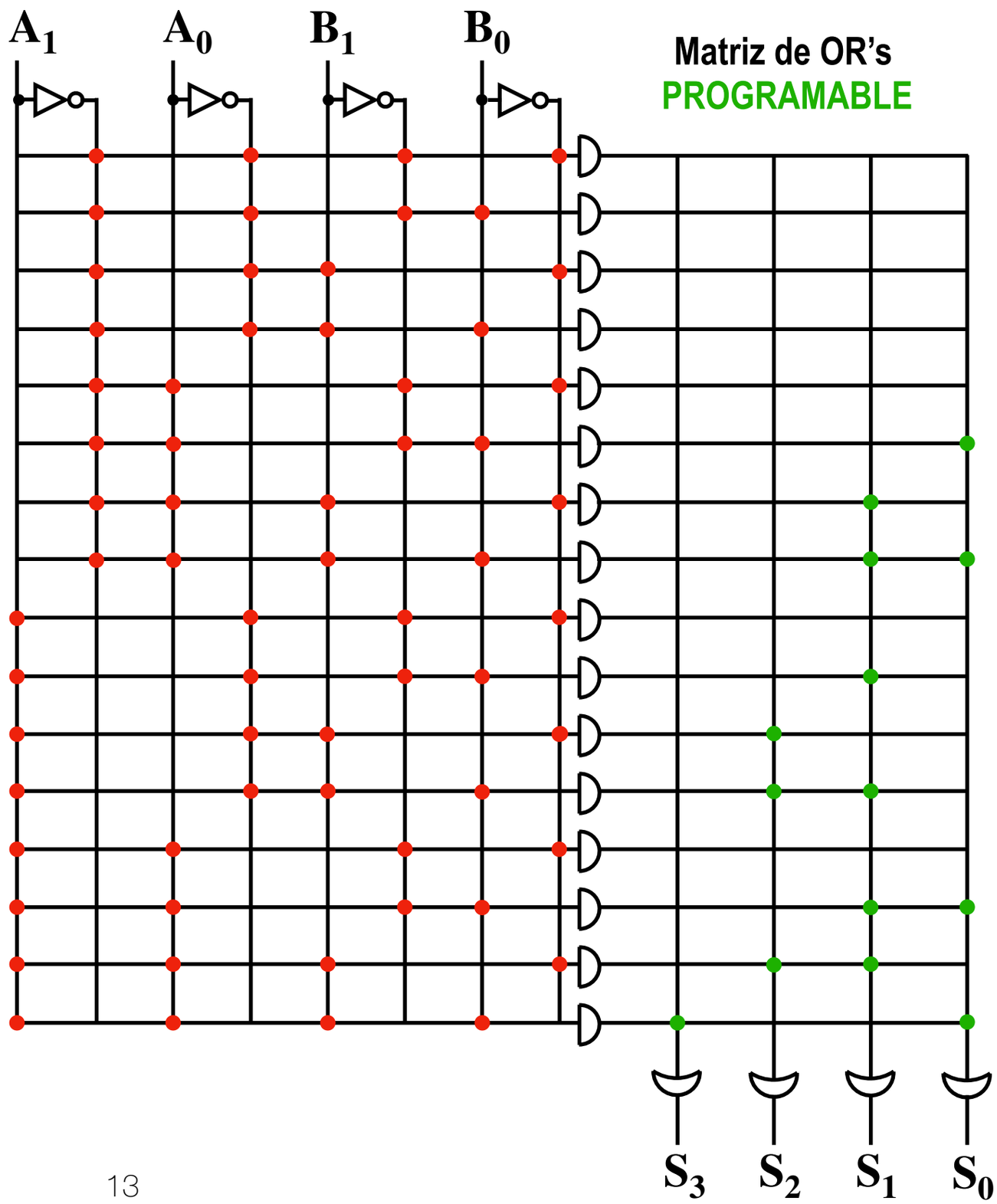
$$S_0 = A_0B_0$$

PLD-MEMORIAS

Implementación del ejercicio del multiplicador con **ROM**

CAPACIDAD = 16 AND'S x 4 OR'S
16 X 4

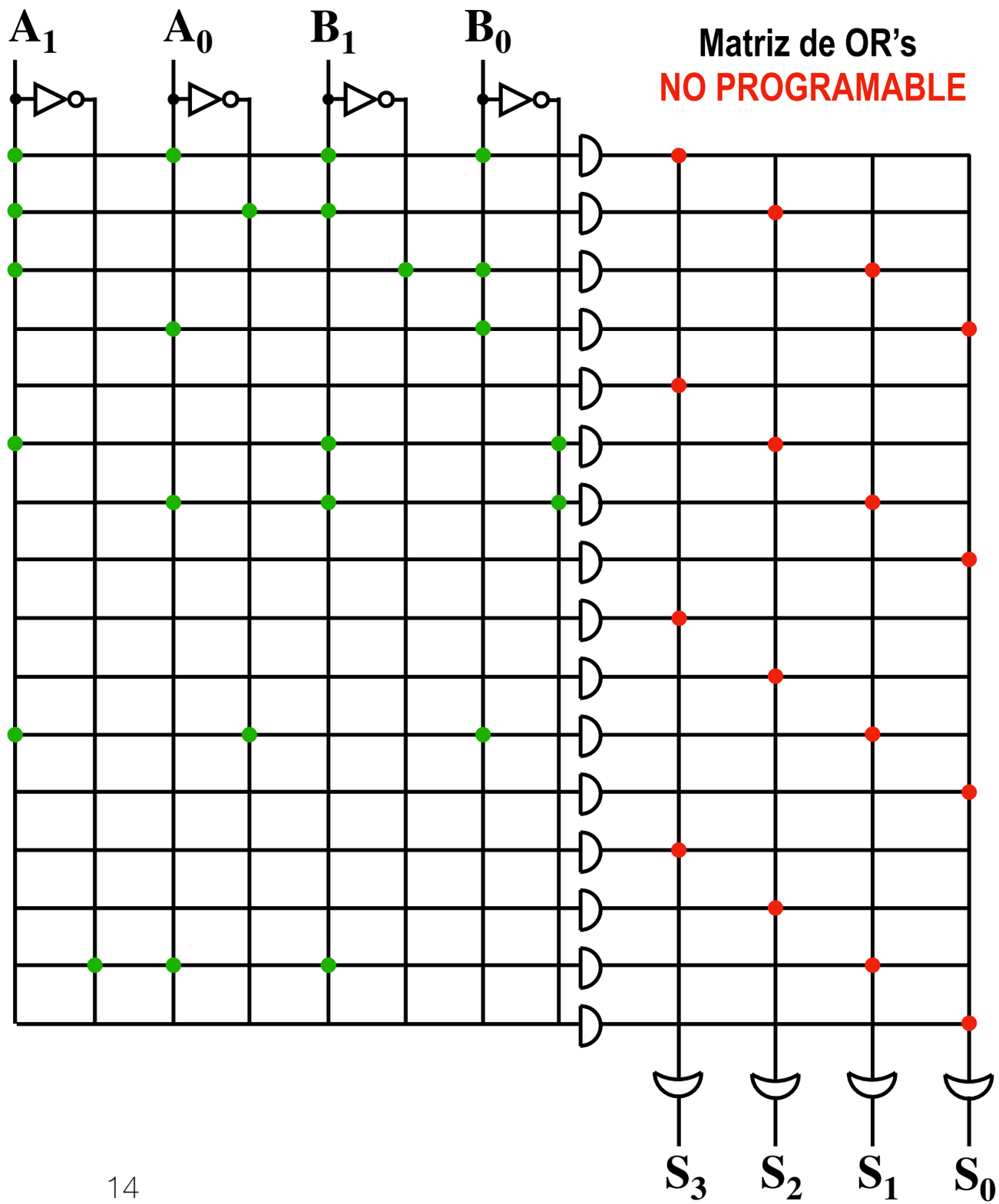
Matriz de AND's
NO PROGRAMABLE



PLD-MEMORIAS

Implementación del ejercicio del multiplicador con **PAL**

CAPACIDAD = 16 AND'S x 4 OR'S
16 X 4

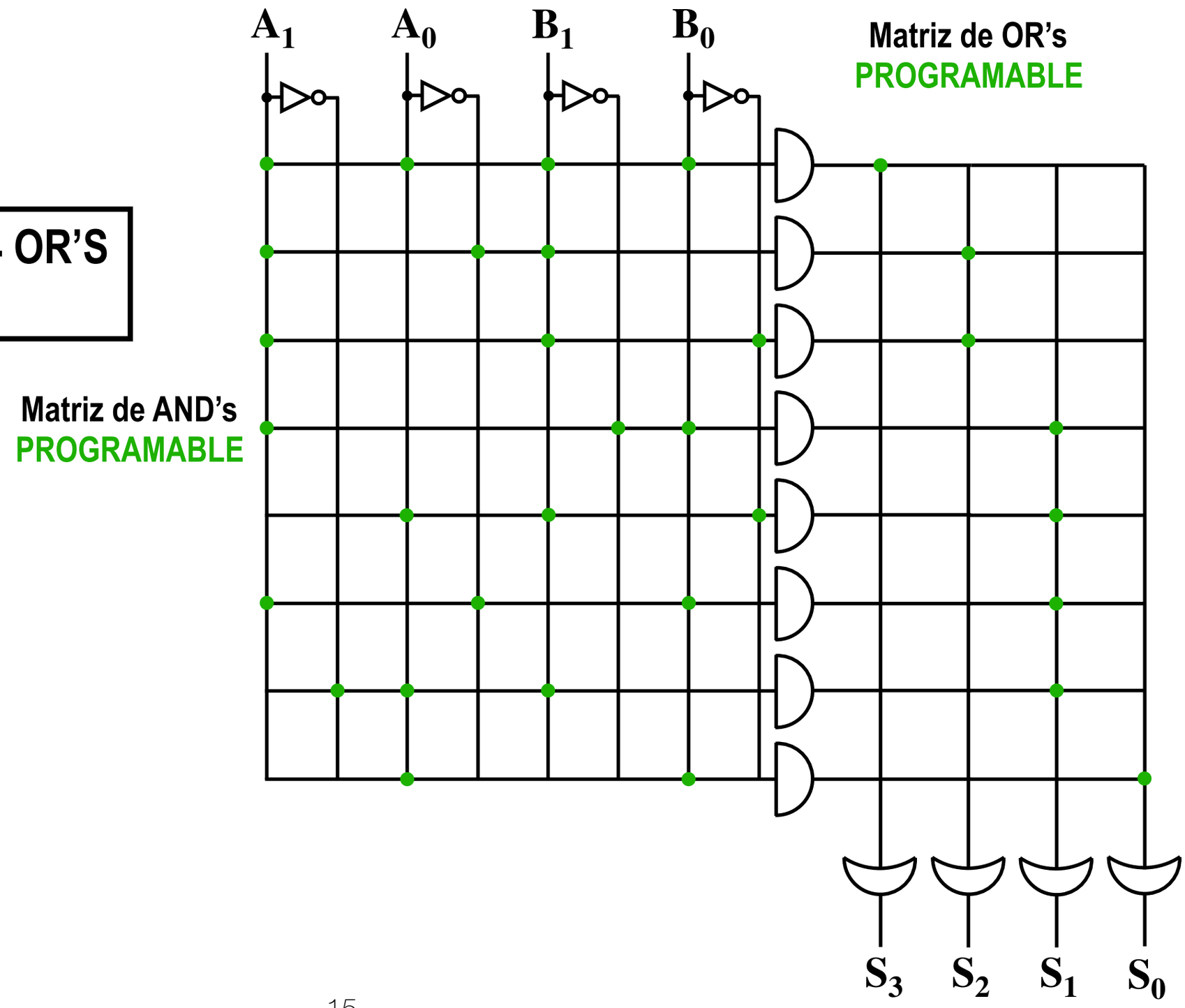


Observación: Se dejan direcciones sin utilizar, ya que las funciones S_3, S_2 y S_0 que tienen términos S_1. En este modelo las direcciones de cada función de salida, no son correlativas, por ejemplo las que corresponden a S_1 son las **2**, **6**, **10** y **14**.

PLD-MEMORIAS

Implementación del ejercicio
del multiplicador con **PLA**

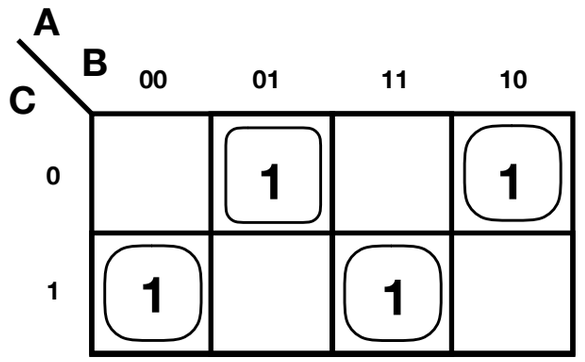
CAPACIDAD = 8 AND'S x 4 OR'S
8 X 4



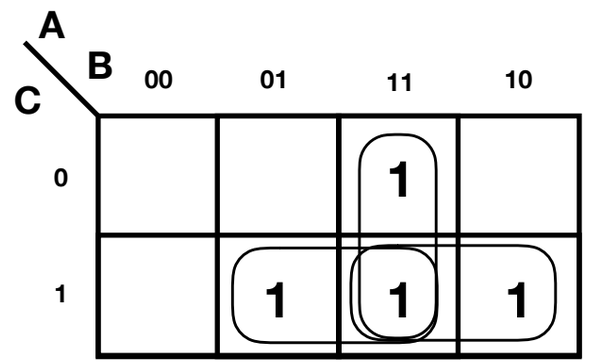
PLD-MEMORIAS

Ejercicio: Implementar el circuito de un sumador y de un restador completo en diferentes dispositivos PLD utilizando ROM, PAL y PLA.

A	B	C	S	C_A
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

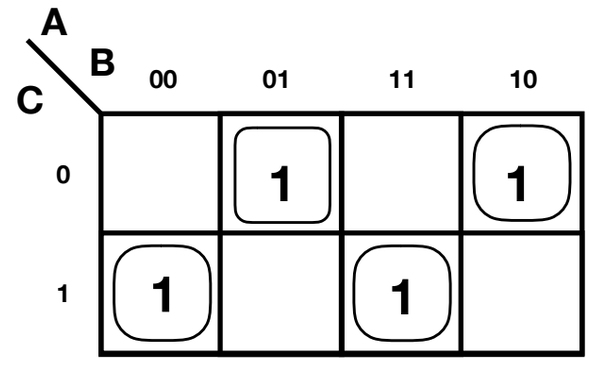


$$S = A \oplus B \oplus C$$

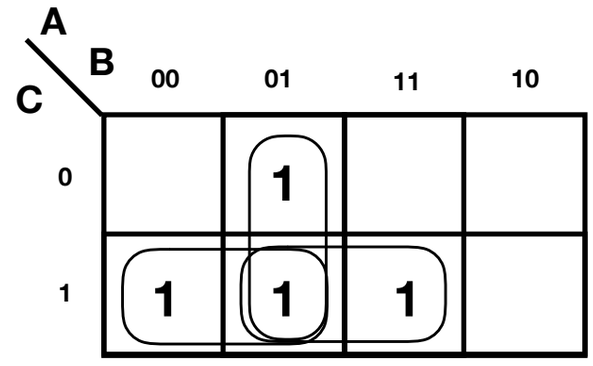


$$C_A = AB + AC + BC$$

A	B	C	R	B_W
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



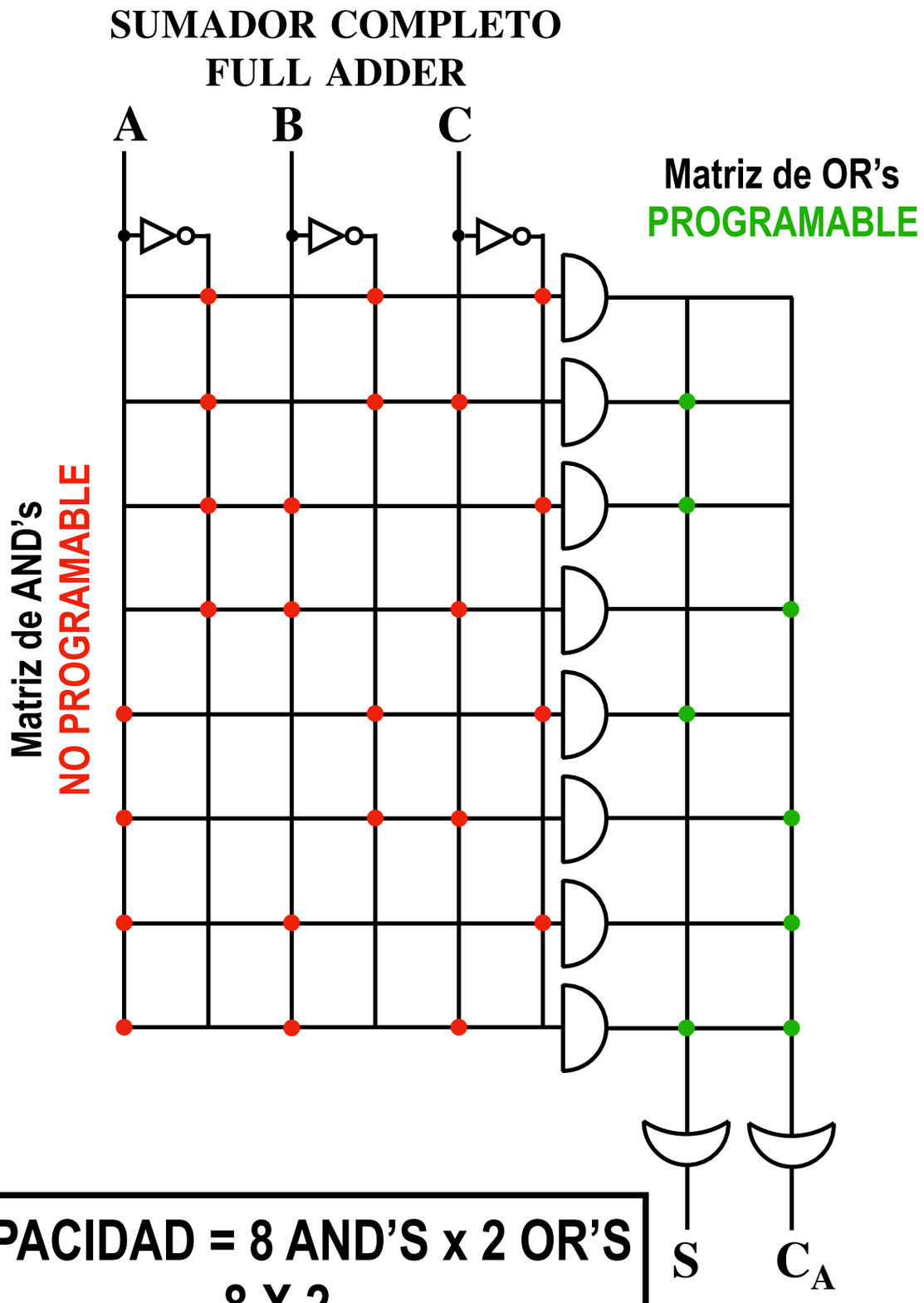
$$R = A \oplus B \oplus C$$



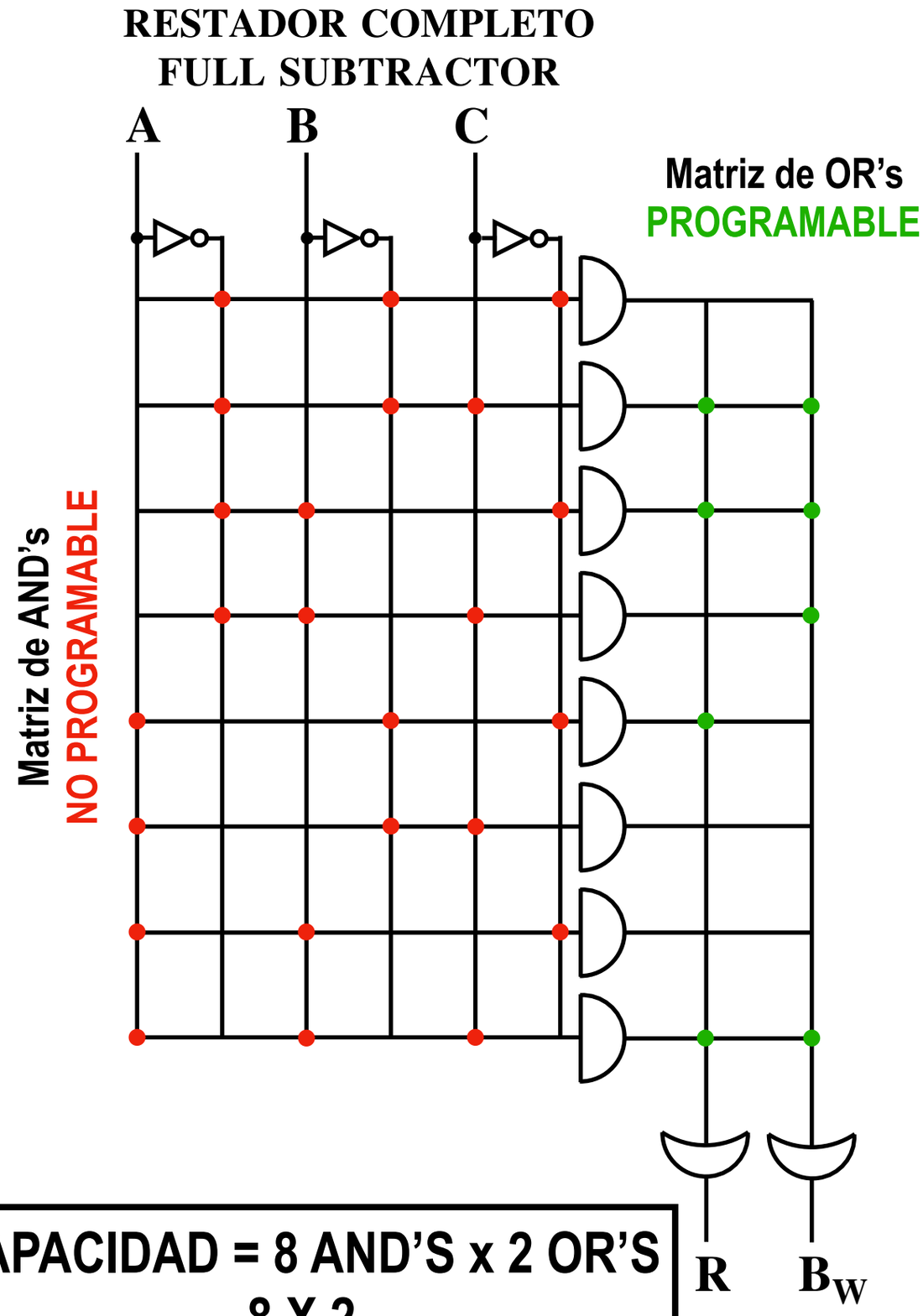
$$B_W = \overline{A}B + \overline{A}C + BC$$

PLD-MEMORIAS

Implementación del ejercicio con **ROM**



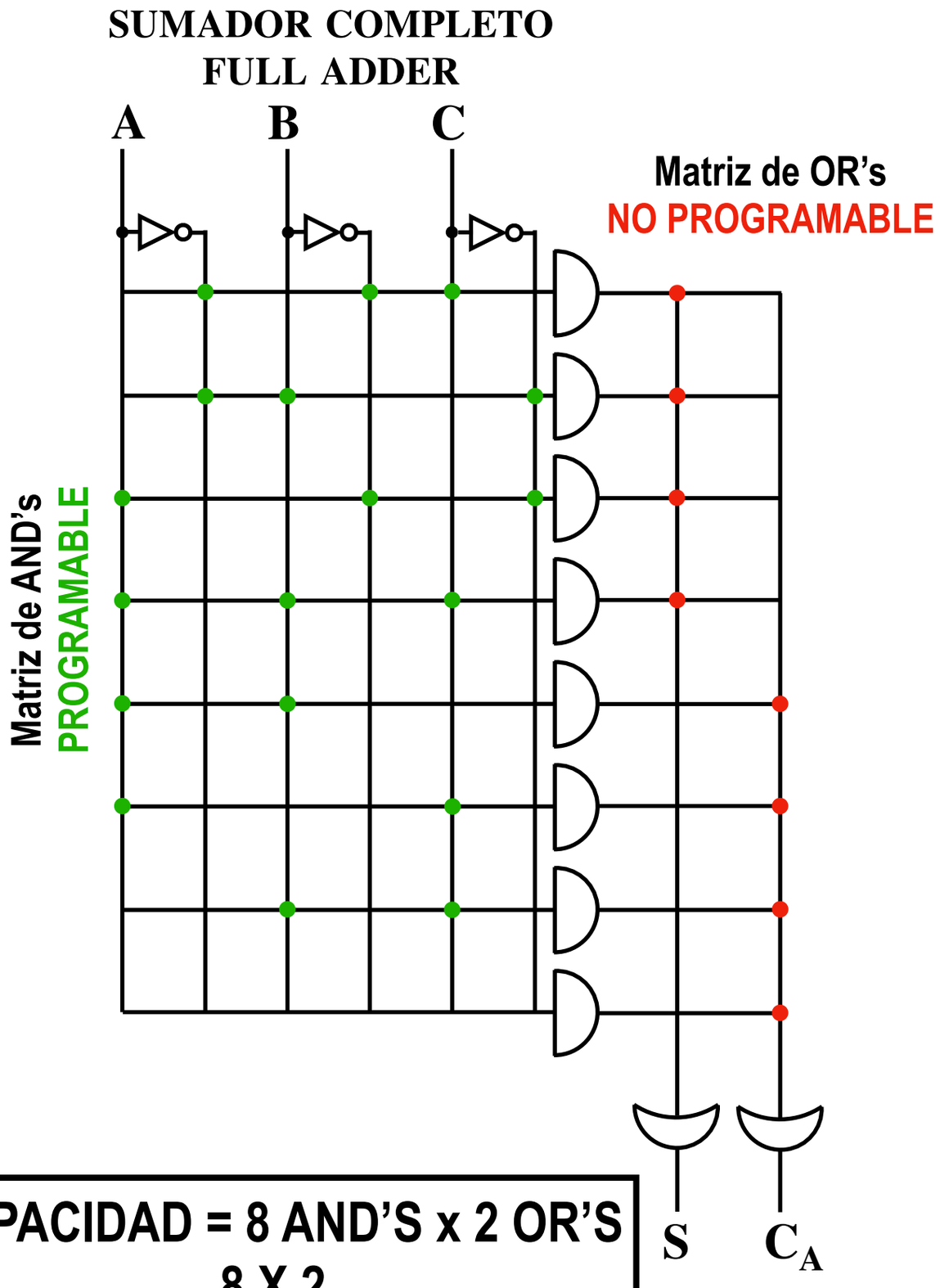
CAPACIDAD = 8 AND'S x 2 OR'S
8 X 2



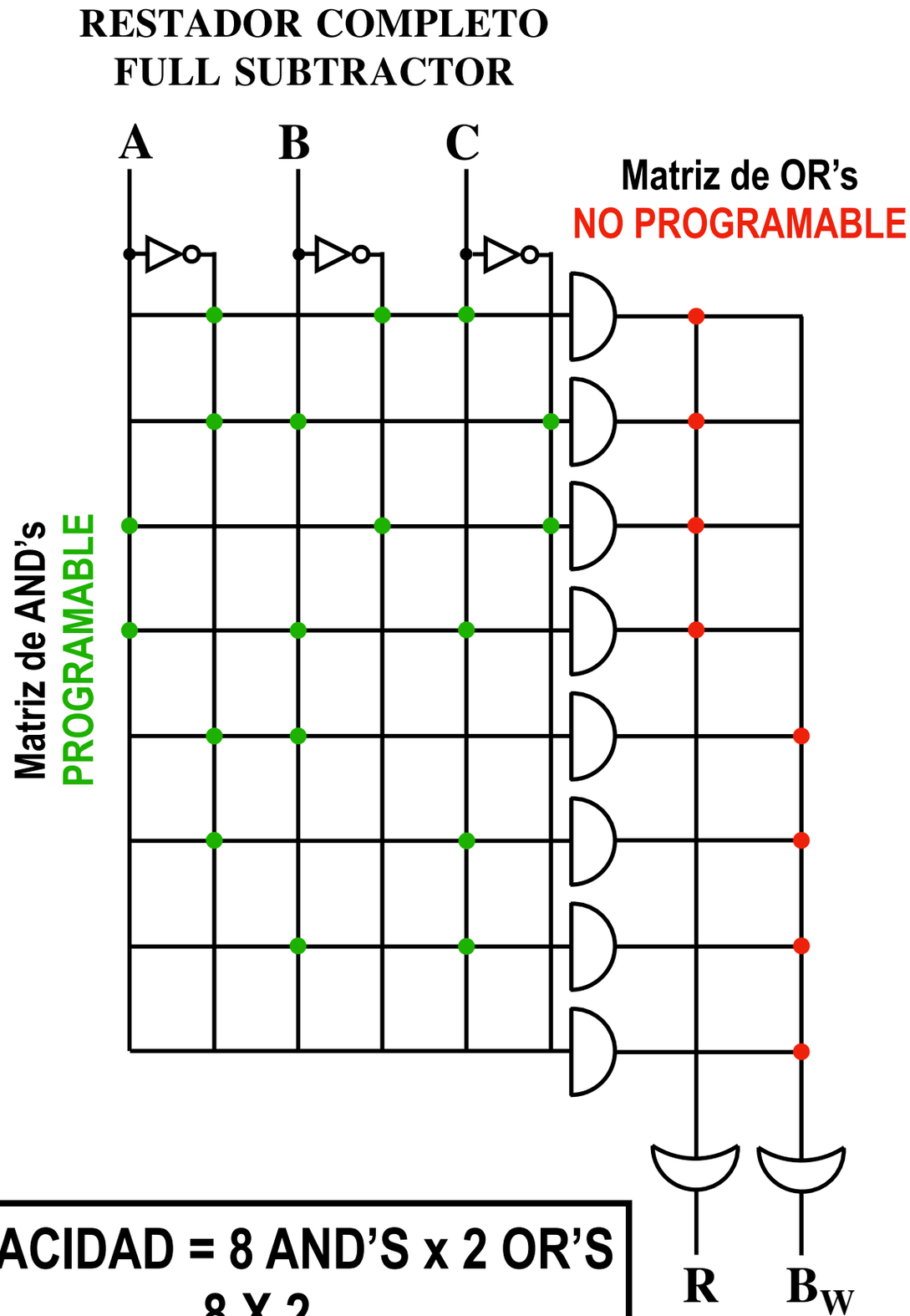
CAPACIDAD = 8 AND'S x 2 OR'S
8 X 2

PLD-MEMORIAS

Implementación del ejercicio con **PAL**



CAPACIDAD = 8 AND'S x 2 OR'S
8 X 2

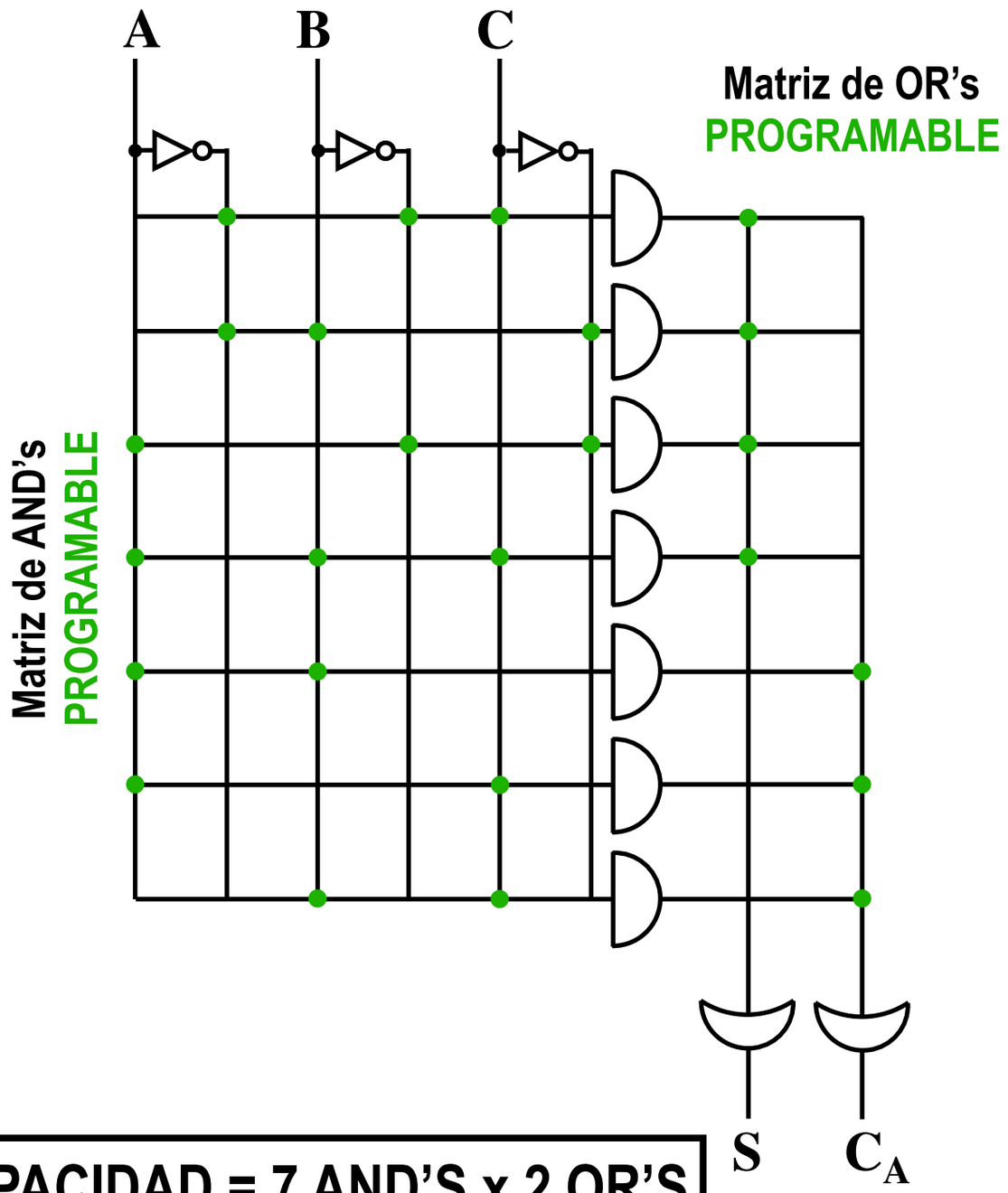


CAPACIDAD = 8 AND'S x 2 OR'S
8 X 2

PLD-MEMORIAS

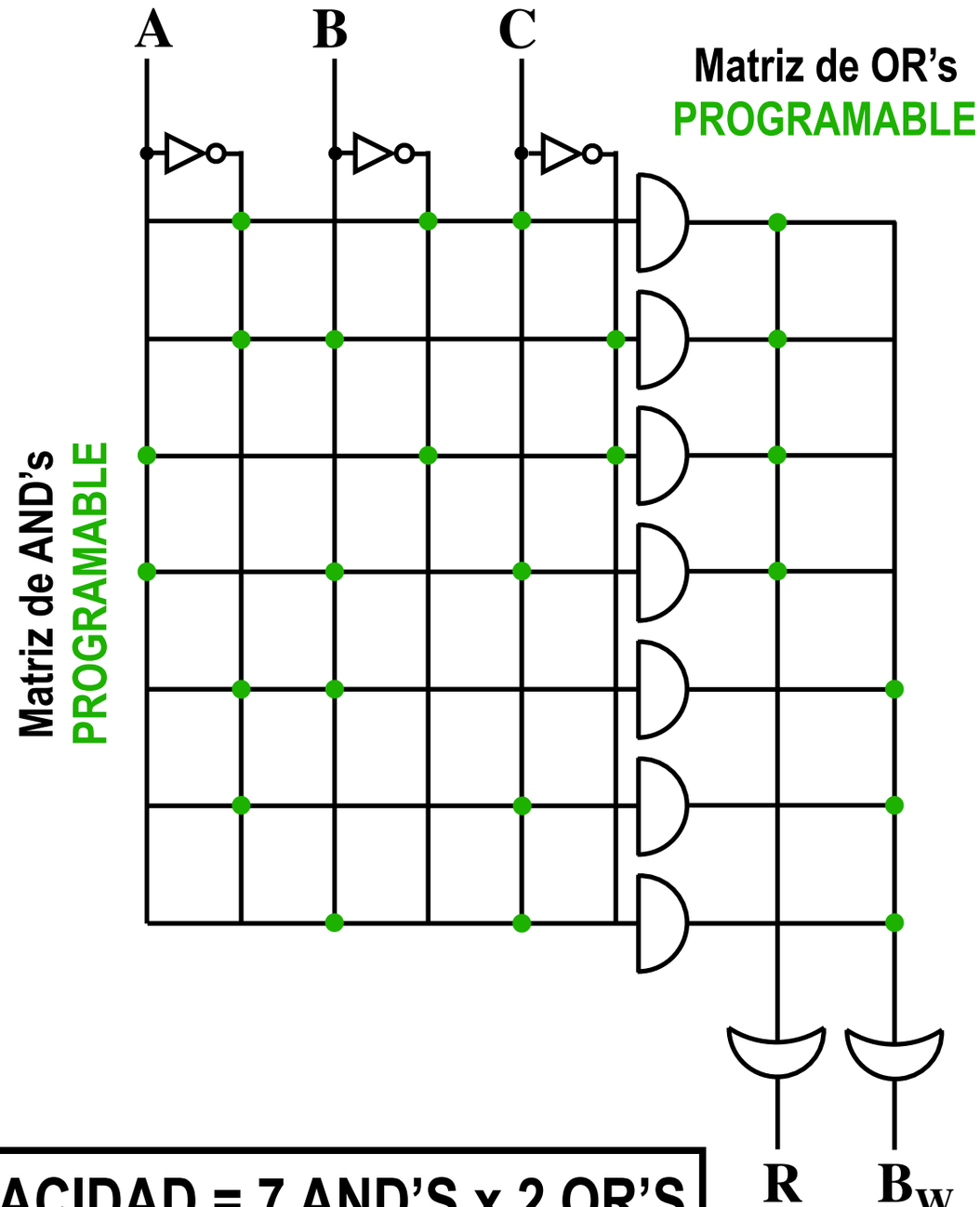
Implementación del ejercicio con **PLA**

SUMADOR COMPLETO
FULL ADDER



CAPACIDAD = 7 AND'S x 2 OR'S
7 X 2

RESTADOR COMPLETO
FULL SUBTRACTOR

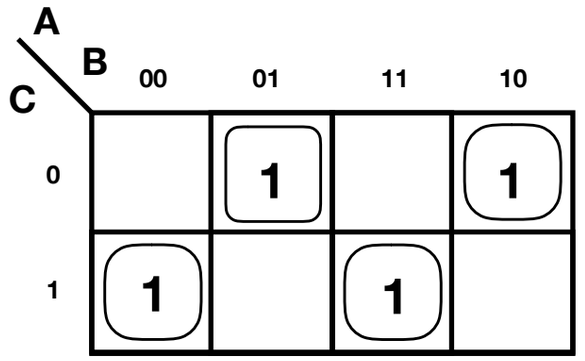


CAPACIDAD = 7 AND'S x 2 OR'S
7 X 2

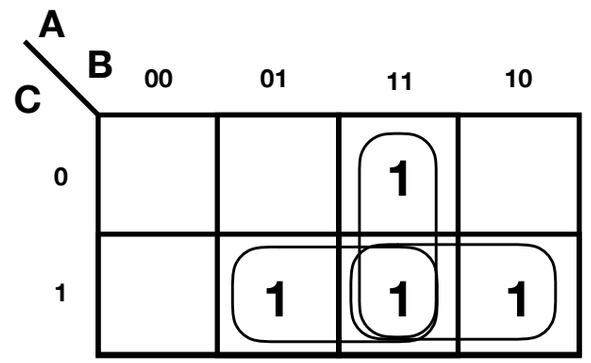
PLD-MEMORIAS

Ejercicio: Implementar el circuito de un sumador y de un restador completo en un mismo dispositivo PLD utilizando ROM, PAL y PLA.

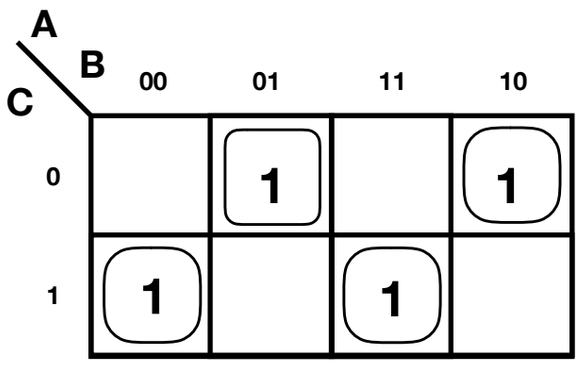
A	B	C	S	C_A	R	B_W
0	0	0	0	0	0	0
0	0	1	1	0	1	1
0	1	0	1	0	1	1
0	1	1	0	1	0	1
1	0	0	1	0	1	0
1	0	1	0	1	0	0
1	1	0	0	1	0	0
1	1	1	1	1	1	1



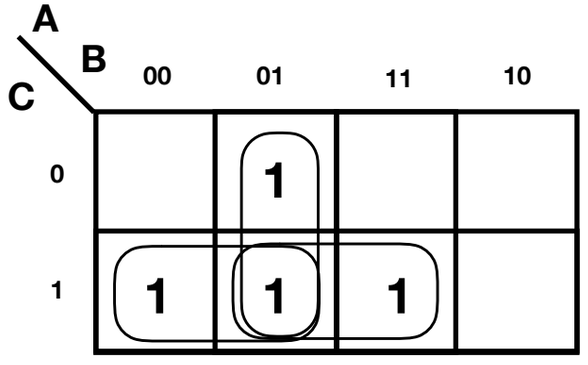
$$S = A \oplus B \oplus C$$



$$C_A = AB + AC + BC$$



$$R = A \oplus B \oplus C$$

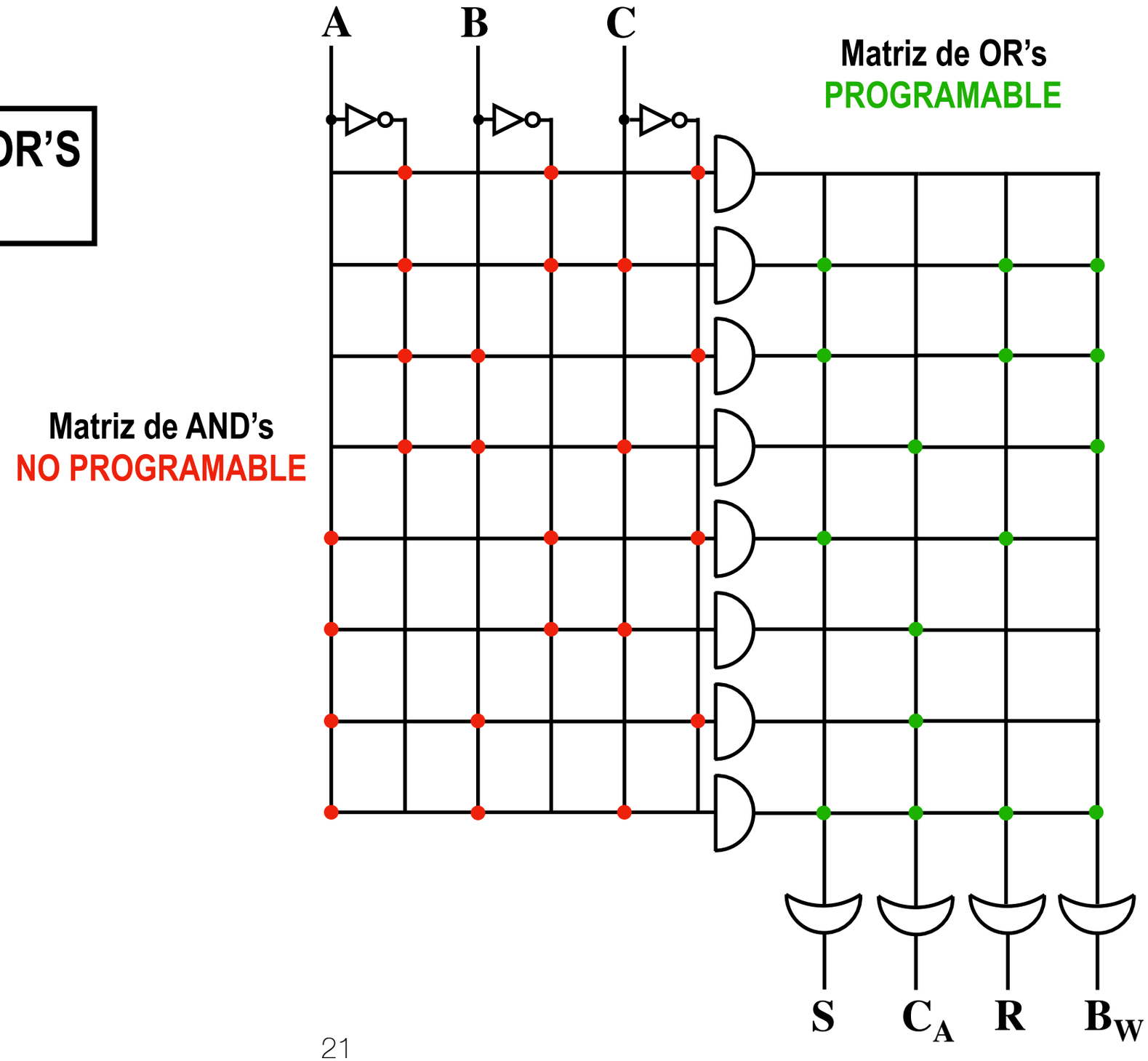


$$B_W = \overline{A}B + \overline{A}C + BC$$

PLD-MEMORIAS

Implementación del ejercicio del sumador y restador con **ROM**

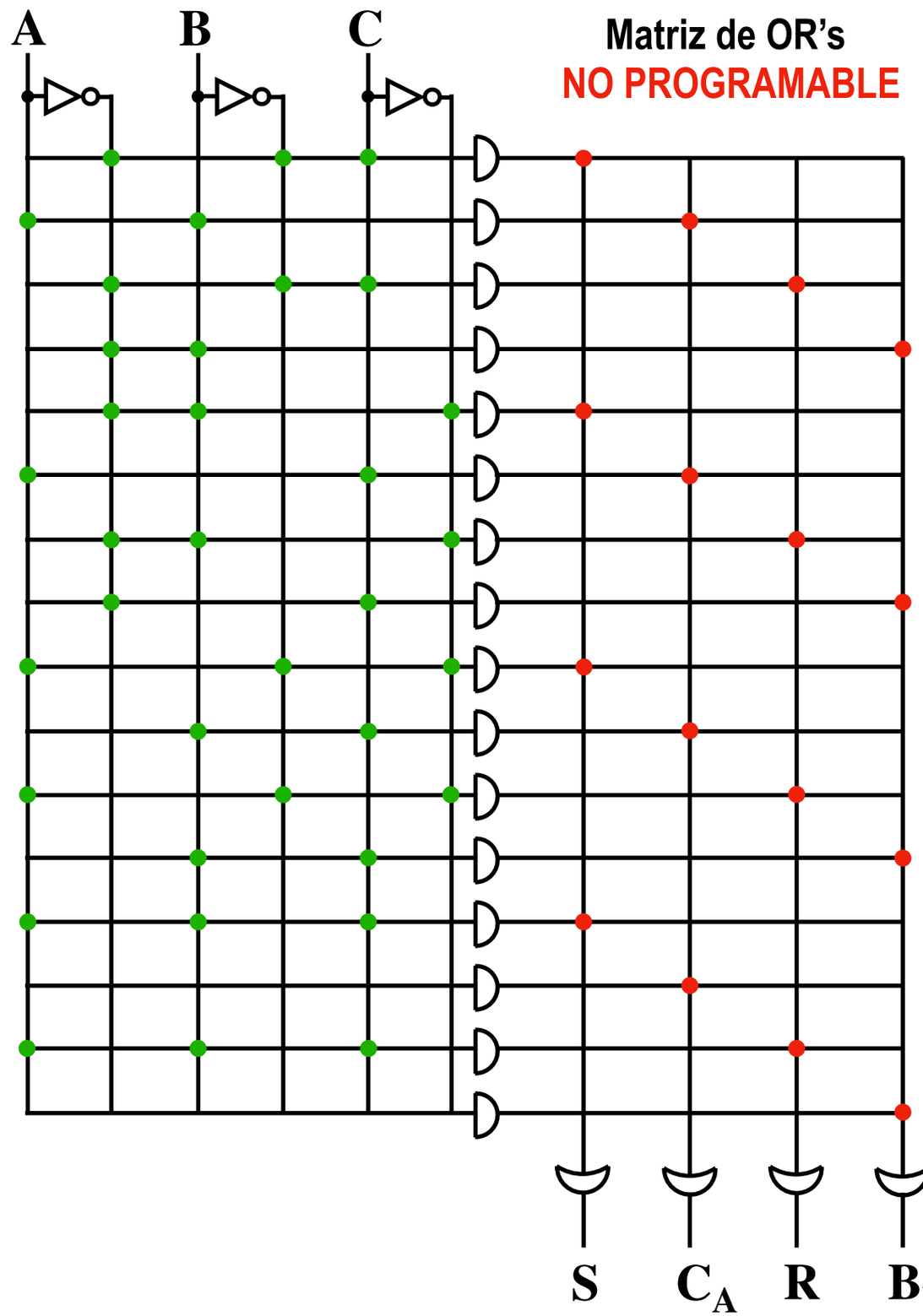
CAPACIDAD = 8 AND'S x 4 OR'S
8 X 4



PLD-MEMORIAS

Implementación del ejercicio del sumador y restador con **PAL**

CAPACIDAD = 16 AND'S x 4 OR'S
16 X 4



Observación: Se dejan direcciones sin utilizar, ya que las funciones C_A y B_W que tienen un término menos que **S** y **R**. En este modelo las direcciones de cada función de salida, no son correlativas, por ejemplo las que corresponden a **S** son las **0**, **4**, **8** y **12**, por lo tanto se debe ser muy cuidadoso al momento de implementar.

PLD-MEMORIAS

Implementación del ejercicio del sumador y restador con **PLA**

CAPACIDAD = 9 AND'S x 4 OR'S
9 X 4