Organización de Computadoras 2023

Práctica 3 -Explicación

Lógica y compuertas (Parte 1): Funciones lógicas elementales. Puertas lógicas.

Objetivos de la práctica: que el alumno sea capaz de:

- ***Realizar operaciones lógicas***
- ***Usar máscaras y realizar equivalencias entre operaciones sucesivas.***
- ***Establecer la salida de circuitos combinatorios simples.***
- ***Confeccionar tablas de verdad.***
- ***Describir la relación entre entradas y salidas por ecuaciones.***

Bibliografía:

- ***"Organización y Arquitectura de Computadoras" de W. Stallings, Apéndice A, pág. 645.***
- ***"Principios de Arquitectura de Computadoras" de Miles J. Murdocca, apéndice A, pág. 441.***
- ***Apunte 3 de la cátedra, "Sistemas de Numeración: Operaciones Lógicas".***

Operaciones Lógicas

1. Realizar las siguientes operaciones lógicas:

a. 10011001 **AND** 10101110
b. 01011000 **AND** 11110011
c. 10011001 **OR** 10101110
d. 01011000 **OR** 11110011
e. 10011001 **XOR** 10101110
f. 01011000 **XOR** 11110011
g. **NOT** 010111000
h. **NOT** 111010100
i. 10011001 **NAND** 10101110
j. 01011000 **NAND** 11110011
k. 10111001 **NOR** 11101110
l. 01011010 **NOR** 11010011
m. 10111001 **XNOR** 11101110
n. 01011010 **XNOR** 01011010

Las operaciones lógicas se realizan bit a bit, si tenemos la que hacer una operación lógica (OL) como:

$$\begin{array}{rl} & X_7\,X_6\,X_5\,X_4\,X_3\,X_2\,X_1\,X_0 \\ \text{OL} & \underline{Y_7\,Y_6\,Y_5\,Y_4\,Y_3\,Y_2\,Y_1\,Y_0} \\ & Z_7\,Z_6\,Z_5\,Z_4\,Z_3\,Z_2\,Z_1\,Z_0 \end{array}$$

A cada valor Xi se le aplica la operación lógica OL con el correspondiente Yi' obteniendose un valor Zi

Luego las tablas de verdad para las operaciones lógicas básicas son:

X	Y	X and Y	X or Y	X xor Y
0	0	0	0	0
1	0	0	1	1
0	1	0	1	1
1	1	1	1	0

Ejemplo:

```
        10101100
    AND 11000101
        10000100
```

2. Dado un byte **$X=[X_7,X_6,X_5,X_4,X_3,X_2,X_1,X_0]$** (los X representan bits con valores indeterminados), ¿qué resultado obtendré al aplicarle una operación lógica junto a un valor predeterminado (máscara)?: Analice para cada operación cómo los bits de la 'máscara' condicionan el resultado que se obtendrá. **¿Puede reconocer un patrón para cada mascara?**
En los casos de más de una operación, obtenga el resultado y a ese resultado aplíquele la operación siguiente.Ejemplo:

$$
\begin{array}{ll}
 & X_7X_6X_5X_4X_3X_2X_1X_0 \\
\text{AND} & 0_7 1_6 0_5 0_4 0_3 1_2\ 0_1 0_0 \\
\hline
 & 0_7 X_6 0_5 0_4 0_3 X_2 0_1 0_0 \\
\text{XOR} & 0_7 1_6 0_5 0_4 0_3 0_2 1_1 0_0 \\
\hline
 & 0_7 {\sim}X_6 0_5 0_4 0_3 X_2 1_1 0_0
\end{array}
$$

Nota: ~X = X negado (valor opuesto a X)

a. **X OR 00011000**
b. **X OR 11001100**
c. **X AND 01010101**
d. **X AND 01001100**
e. **X XOR 01010101**
f. **X XOR 11001100**
g. **X OR 10000001**, al resultado **AND 00111001**, y al resultado **XOR 11001111**
h. **X AND 10001110**, al resultado **OR 11001100**, y al resultado **XOR 01010011**
i. **X XOR 10010010**, al resultado **AND 11100110**, y al resultado **OR 00110111**
j. **X XNOR 10011001**, al resultado **NAND 11001100**, y al resultado **NOR 00011000**
k. **X XOR 10100101**, al resultado **NAND 11100111**, y al resultado **NOR 01010110**

Las máscaras sirven para distintos acciones; algunas de ellas son:

1) Forzar un determinado bit a un valor 0 o 1;
2) Mantener un determinado bit con valor igual al actual
3) Invertir el valor de un determinado bit, de 0 a 1 o de 1 a 0, dependiendo del valor anterior.

Para cumplir esto, las máscaras se utilizan en conjunto con las operaciones lógicas.
Tengamos en cuenta las siguientes reglas:

<u>Operación lógica AND</u>

Xi AND 0 = 0, pues

Si Xi = 0 => 0 AND 0 = 0
Si Xi = 1 => 1 AND 0 = 0

Xi AND 1 = Xi, pues

Si Xi = 0 => 0 AND 1 = 0 = Xi
Si Xi = 1 => 1 AND 1 = 1 = Xi

Entonces con la operación lógica AND podemos forzar un valor a 0 o mantener el valor anterior.

<u>Operación lógica OR</u>

Xi OR 0 = Xi, pues

Si Xi = 0 => 0 OR 0 = 0 = Xi
Si Xi = 1 => 1 OR 0 = 1 = Xi

Xi OR 1 = 1, pues

Si Xi = 0 => 0 OR 1 = 1
Si Xi = 1 => 1 OR 1 = 1

Entonces con la operación lógica OR podemos forzar un valor a 1 o mantener el valor anterior.

<u>Operación lógica XOR</u>

Xi XOR 0 = Xi, pues

Si Xi = 0 => 0 XOR 0 = 0 = Xi
Si Xi = 1 => 1 XOR 0 = 1 = Xi

Xi XOR 1 = -Xi, pues

Si Xi = 0 => 0 XOR 1 = 1 = -Xi (valor contrario, pasa de 0 a 1)
Si Xi = 1 => 1 XOR 1 = 0 = -Xi (valor contrario, pasa de 1 a 0)

Entonces con la operación lógica XOR podemos forzar un valor al contrario o mantener el valor anterior.

Ejemplo:

X OR 11111000: XXXXXXXX
OR <u>1 1 1 1 1 0 0 0</u>
1 1 1 1 1 XXX (En base a la explicación anterior)

3. Complete las siguientes líneas punteadas con el operador lógico adecuado (sean AND, OR, XOR, NOT), en las siguientes expresiones de modo tal que se cumpla la igualdad propuesta:

a. **1000 …………. 1101 = 1101**
b. **1111 …………. 0101 = 0101**
c. **1101 …………. 1001 = 0100**
d. **………. (1111 …………. 0011) = 1100**
e. **$X_3 X_2 X_1 X_0$………… 1110 …………. 0101 …………. 0101 = X_3 0 X_1 0**
f. **$X_3 X_2 X_1 X_0$………… 1000 …………. 1011 …………. 1110 = 0 1$\underline{X_1}$ $\underline{X_0}$**
g. **……..($X_3 X_2 X_1 X_0$…………1001)= $\underline{X_3}$11$\underline{X_0}$**

Se entiende que cada X es un bit desconocido que puede ser 1 o 0, debiendo obtenerse el resultado final al combinar diferentes operaciones lógicas, siguiendo el orden correcto.
Ejemplo:

1 0 0 0
Nop <u>1 1 0 1</u>
1 1 0 1

Cual es la operación lógica Nop? Se observa que la operación está forzando valores a 1 y otros se mantienen igual => la Nop es OR

4. Dado un byte **X=[X_7,X_6,X_5,X_4,X_3,X_2,X_1,X_0]** (los X representan bits con valores indeterminados), aplíquele operaciones lógicas (1 o más) con un byte MASK, que deberá también determinar, para lograr los siguientes efectos:

a) Poner en 1 los bits 1,3 y 5 dejando los demás bits iguales.
b) Poner a 1 los bits 4 y 6 dejando los demás iguales.
c) Poner a 0 los bits 1, 3 y 5 dejando los demás iguales.
d) Poner a 0 los bits 4 y 6 dejando los demás iguales.
e) Cambiar los bits 1, 3 y 5 a su complemento dejando los demás iguales.
f) Cambiar los bits 4 y 6a su complemento dejando los demás iguales.
g) Poner en 1 los bits 1 y 5, poner en 0 los bits 7 y 0, cambiar el bit 6 por su complemento y dejar los demás iguales.
h) Poner en 0 los bits 1, 5 y 6, cambiar el bit 4 por su complemento y dejar los demás iguales.

Ejemplo: Poner el bit 3 en 1, el bit 6 en 0 cambiar el bit 2 y dejar los demás inalterados

Para ello tenemos que hacer 3 operaciones lógicas:

1) Sabemos que el OR permite forzar un bit a 1 con mascara 1, y 0 deja inalterado. Entonces para forzar el bit 3 a 1 al bit e le aplicamos máscara 1, y al resto máscara 0:

$$\begin{array}{ll} & X_7\,X_6\,X_5\,X_4\,X_3\,X_2\,X_1\,X_0 \\ \text{OR} & \underline{0\;\;0\;\;0\;\;0\;\;0\;\;1\;\;0\;\;0} \\ & X_7\,X_6\,X_5\,X_4\,X_3\;\;1\;X_1\,X_0 \end{array}$$

2) Al resultado anterior le tenemos que forzar el bit 6 a 0, para eso sabemos que debemos usar la operación lógica AND en el bit 6 con máscara 0, y el resto con máscara 1 queda inalterado; además, al bit 3 forzado a 1 en el paso anterior le aplicamos mascara 1 para que mantenga valor 1 con el AND; entonces:

$$\begin{array}{ll} & X_7\,X_6\,X_5\,X_4\,X_3\;\;1\;X_1\,X_0 \\ \text{AND} & \underline{1\;\;1\;\;0\;\;1\;\;1\;\;1\;\;1\;\;1} \\ & X_7\,X_6\;\;0\;X_4\,X_3\;\;1\;X_1\,X_0 \end{array}$$

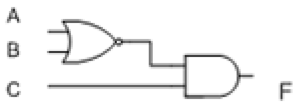
3) Finalmente tenemos que cambiar el valor del bit 2; para ello se utiliza la operación lógica XOR que con máscara 1 invierte valor, y con máscara 0 mantiene valores inalterados:

$$\begin{array}{ll} & X_7\,X_6\;\;0\;X_4\,X_3\;\;1\;X_1\,X_0 \\ \text{XOR} & \underline{0\;\;0\;\;0\;\;0\;\;0\;\;0\;\;1\;\;0} \\ & X_7\,X_6\;\;0\;X_4\,X_3\;\;1\;{-X_1}\;X_0 \end{array}$$ ***(-Xi ,indica valor opuesto a Xi)***

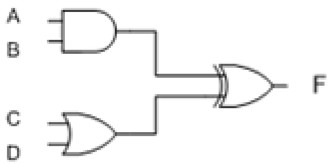
Circuitos Combinatorios

3. Construir la tabla de verdad de los siguientes circuitos. Especifique además la ecuación que describe la relación entre entradas-salidas.

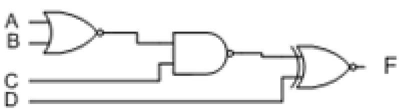
1)



2)



3)



1) Primero debemos obtener la función F asociada al circuito; En el ejemplo 1, la salida F depende de 2 entradas al circuito AND a las que llamaremos X e Y, entonces:

$F = X . Y$

Luego debemos reemplazar X e Y por lo que ellos representan, X está representado por un circuito NOR con entradas A y B, e Y es la entrada C:

$X = \overline{A + B}$
$Y = C$

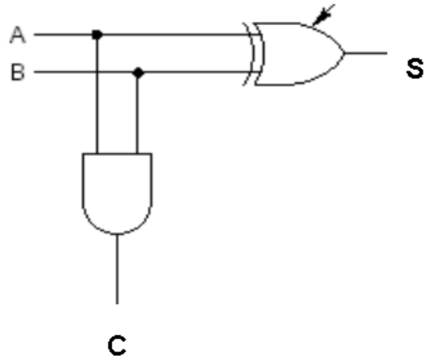
Reemplazando: $F = (\overline{A + B}).C$

Los paréntesis permiten visualizar claramente el orden de precedencia de las operaciones lógicas. Hemos obtenido la función F, ahora podemos hacer la tabla de verdad:

A	B	C	A + B	- (A + B)	F
0	0	0	0	1	0
0	0	1	0	1	1
0	1	0	1	0	0
0	1	1	1	0	0
1	0	0	1	0	0
1	0	1	1	0	0
1	1	0	1	0	0
1	1	1	1	0	0

Recordar generar cada columna en base a las calculadas anteriormente. Porque son 8 filas de valores de entrada? Porque son 3 entradas con 2 valores posibles, las combinaciones son siempre 2 a la N, con N número de entradas.

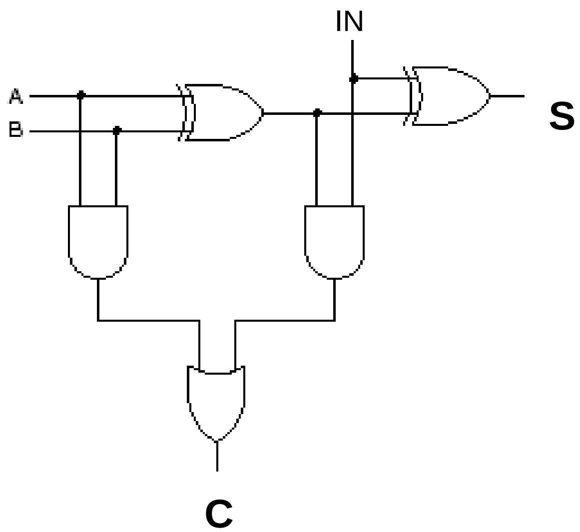
4)



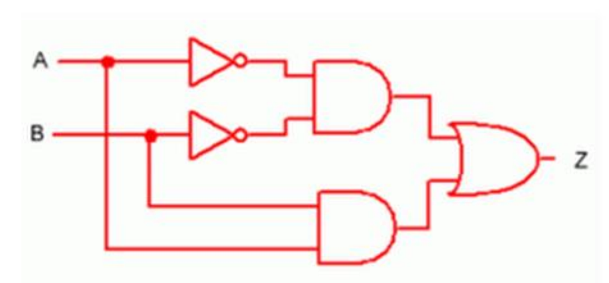
Este ejemplo representa un sumador de un bit.
Las entradas son A y B,
La salida S (XOR) representa la suma y
la salida C (AND) representa el Carry,

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

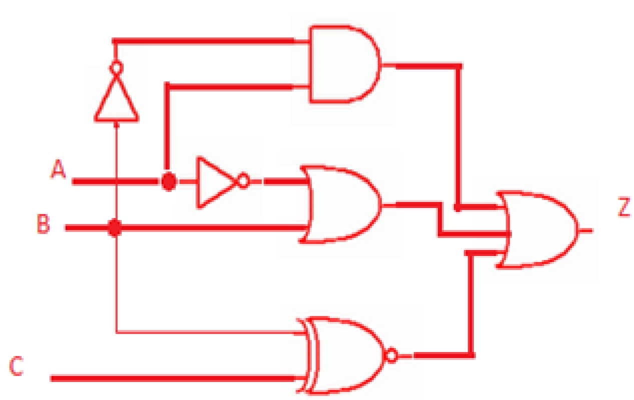
5)



6)



7)



Organización de Computadoras 2023

Lógica y compuertas (Parte 2): Circuitos Combinacionales y Secuenciales.

Objetivos de la práctica: que el alumno domine

- ***Circuitos lógicos y diagramas de compuertas***
- ***Introducción a equivalencias lógicas***
- ***método de sumas de productos.***
- ***Describir el funcionamiento de los distintos tipos de flip flops.***
- ***Comprender el funcionamiento de un circuito secuencial.***

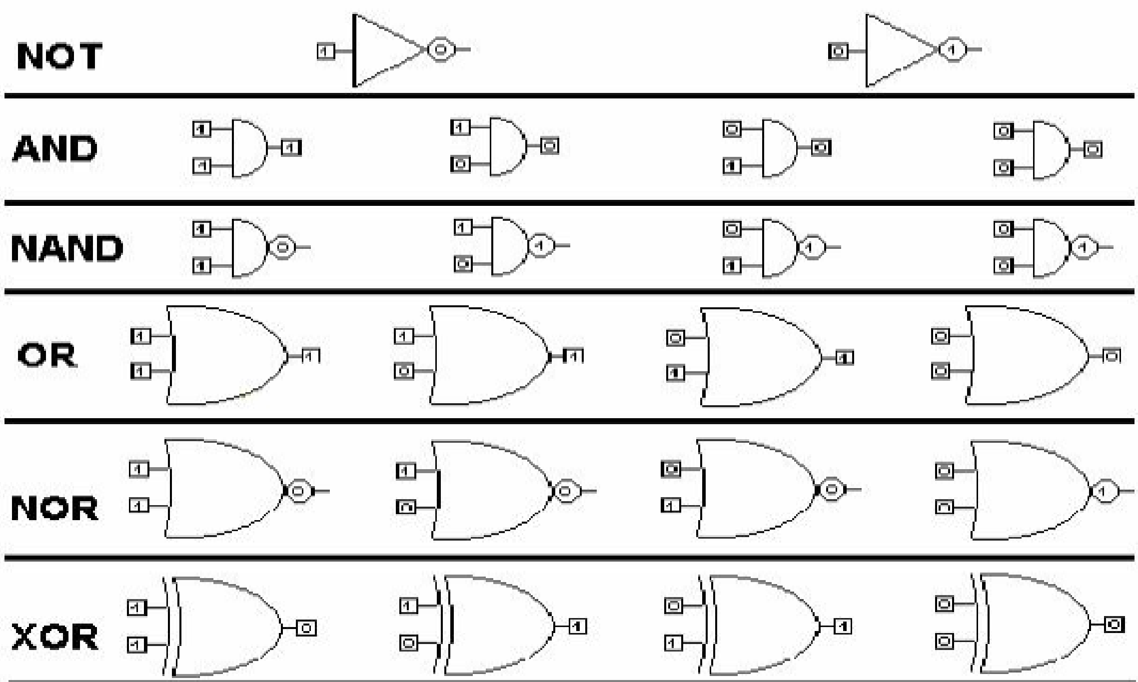
Bibliografía:

- ***"Principios de Arquitectura de Computadoras" de Miles J. Murdocca, apéndice A, pág. 441.***
- ***Apunte 3 de la cátedra, "Sistemas de Numeración: Operaciones Lógicas".***
- ***Apunte 5 de la cátedra, "Circuitos Lógicos Secuenciales".***

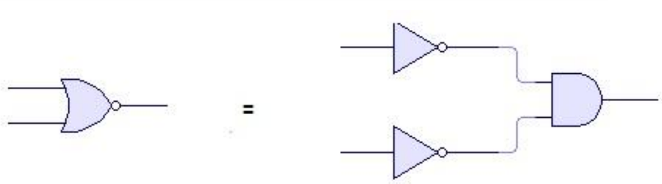
<u>Tener en cuenta para resolución de ejercicios 6 al 10:</u>

Tablas de Verdad: Una tabla de verdad muestra el resultado de una proposición compuesta para cada combinación de valores de verdad que se le puedan asignar a sus componentes de entrada.

Tengamos en cuenta las respuestas de los distintos conectivos lógicos/Compuertas:



Equivalencias lógicas mediante tablas de verdad: Es posible demostrar que dos circuitos son equivalentes si ante iguales entradas responden con el mismo valor de salida. Para llevar a cabo esta demostración, alcanza con construir la tabla de verdad de ambos circuitos y validar que las respuestas coinciden para iguales entradas.
Ejemplo: (La conocida Ley de De Morgan, donde se puede verificar que ante iguales combinaciones de valores de entrada para A y B, la respuesta del circuito es igual en ambos casos)



$\overline{A+B}$

A	B	A+B	$\overline{A+B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

$\overline{A}°\overline{B}$

A	B	$\overline{A}$	$\overline{B}$	$\overline{A}°\overline{B}$
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0

Otras equivalencias lógicas:

Conjunto cerrado de operaciones lógicas usando sólo compuertas Nand o Nor:
Es posible (su justificación excede el objetivo de este curso) reescribir cualquier expresión lógica compuesta, como una expresión equivalente utilizando EXCLUSIVAMENTE compuertas Nand o Nor. Esto favorece el diseño de circuitos al resolver cualquier lógica con un único tipo de compuertas.

Equivalencias lógicas para representar cualquier conectivo lógico como compuertas Nand:

- $\overline{A} \cong \overline{A} + \overline{A} \cong \overline{A.A}$ (Aplico 2 equivalencias lógicas, la última es la ley de De Morgan).
- $A + B \cong \overline{\overline{A + B}} \cong \overline{\overline{A}.\overline{B}} \cong \overline{(\overline{A.A})(\overline{B.B})}$ (doble negación, De Morgan, equivalencia anterior para la negación).
- $A.B \cong \overline{\overline{A.B}} \cong \overline{(\overline{A.B})(\overline{A.B})}$ (doble negación, 1er equivalencia para la negación).
- $A \otimes B \cong (A.\overline{B}) + (\overline{A}.B)$..... (definición del or exclusivo, resta aplicar las equivalencias previas para producto, suma y negación para llegar a utilizar sólo compuertas Nand).

El resto de las compuertas pueden reescribirse sólo con compuertas Nand basándose en las equivalencias previas.

---

6. Demostrar mediante tabla de verdad si se cumplen o no las siguientes equivalencias:
→ INCISOS d), e) y f) A RESOLVER

a) $\overline{(A.B)} = \overline{A} + \overline{B}$ (La segunda ley de De Morgan)

A	B	$\overline{(A.B)}$	$\overline{A}$	$\overline{B}$	$\overline{A} + \overline{B}$
0	0	1	1	1	1
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	0	0	0

SON EQUIVALENTES

b) $A + B.C = (A + B) + (A + C)$

A	B	C	B . C	A + (B . C)	A + B	A + C	(A + B) + (A + C)
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	1
0	1	0	0	0	1	0	1
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

NO SON EQUIVALENTES

c) $(A + B).C = (A.B) + (A.C)$

A	B	C	A + B	(A + B) . C	A . B	A . C	(A . B) + (A . C)
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	0	0	0
1	0	0	1	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	0	1	0	1
1	1	1	1	1	1	1	1

NO SON EQUIVALENTES

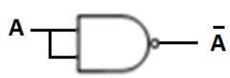
d) $A + A + B = A + B + B$

e) $A + B.C = A.C + B$

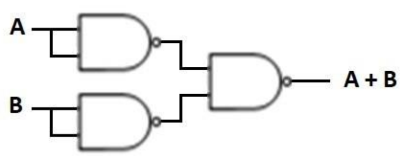
f) $A \oplus B = \overline{A} \oplus \overline{B}$

7. Modifique los siguientes circuitos para que sean todas compuertas **NAND**.

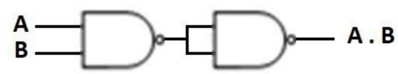
NOT: $\overline{A} = \overline{A . A}$



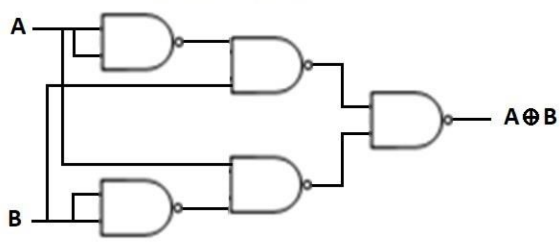
OR: $A + B = \overline{\overline{A + B}} = \overline{\overline{A} . \overline{B}}$



AND: $A . B = \overline{\overline{A . B}}$

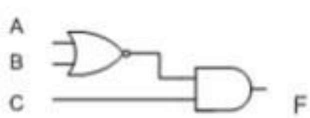


XOR: $A \oplus B = (\overline{A} . B) + (A . \overline{B})$

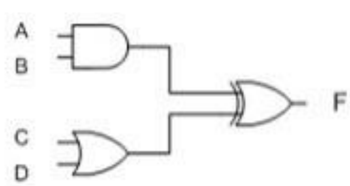


Nota: para lograr este circuito primero se reemplaza con los circuitos con compuertas NAND que corresponden a cada operacion y luego se cancelan las negaciones consecutivas.

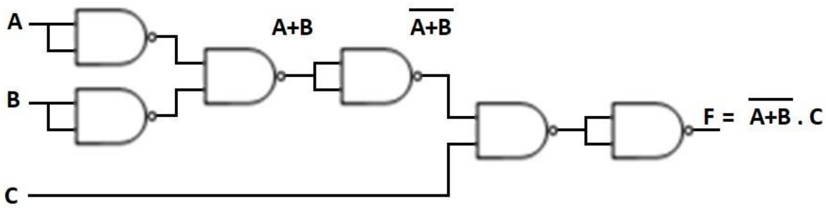
1)



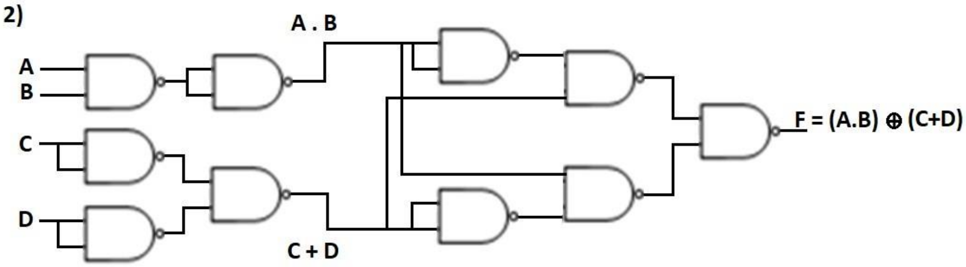
2)



1)

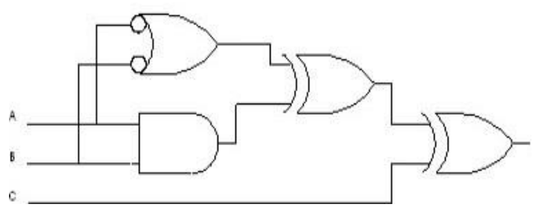


2)



8. Reescriba las compuertas lógicas Not, Or, And y Xor utilizando exclusivamente compuertas **NOR**. (Ver como se resolvió el mismo caso para compuertas Nand, en *Tener en . . .*).
→ EJERCICIO A RESOLVER

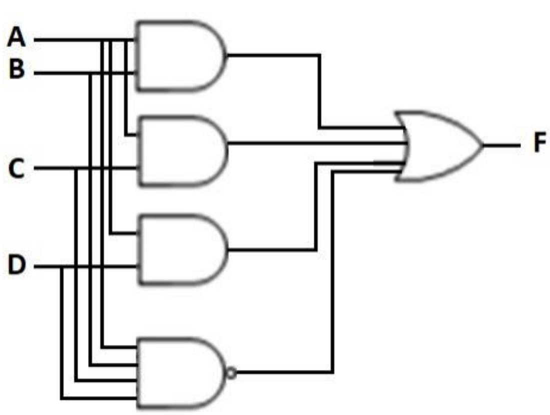
9. Construya la tabla de verdad del siguiente circuito. Analice los valores y basándose en sus conclusiones construya un diagrama más simple que implemente la misma función de salida. Escriba además la ecuación de salida en forma de función.
→ EJERCICIO A RESOLVER



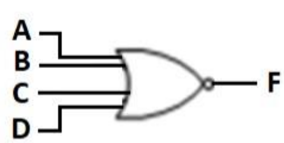
10. Dadas las siguientes relaciones, dibuje los diagramas de compuertas que cumplen con ellas. Modifíquelos utilizando sólo compuertas **NOR**. Modifíquelos utilizando sólo compuertas **NAND**.
→ INCISOS c) y d) A RESOLVER

a) $F = AB + AC + AD + \overline{ABCD}$ b) $F = \overline{A + B + C + D}$ c) $F = \overline{A + B\overline{C} + C}$ d) $F = A\overline{B} + \overline{A}B$

a)

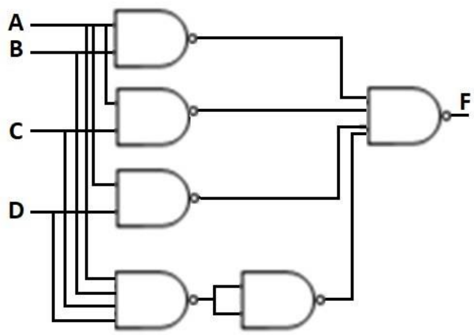


b)



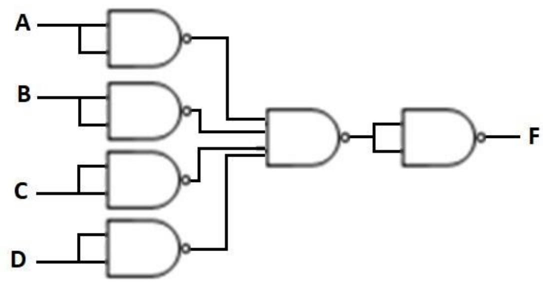
NAND:

a)



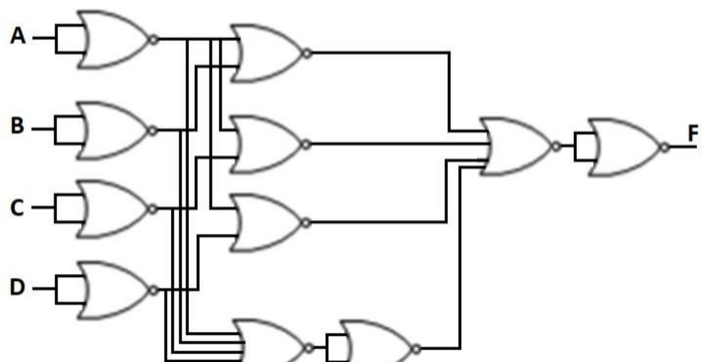
Nota: se cancelaron negaciones consecutivas

b)

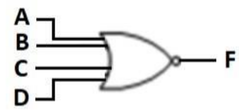


NOR:

a)



b)



Nota: queda igual al original

Tener en cuenta para ejercicios 11 al 13:

Suma de Productos: Es posible inferir la fórmula lógica asociada a una función desconocida de la cual sólo se conoce la respuesta ante todas las combinaciones posibles de entradas….
Ejemplo: Supongamos una función que recibe 2 parámetros A y B, si conocemos la respuesta F de la ecuación en base a los posibles valores de A y B mediante la siguiente tabla de verdad:

A	B	F
0	0	**1**
0	1	**1**
1	0	**0**
1	1	**1**

¿En qué casos la salida F será 1? Rta: Cuando las entradas sean A=0 **y** B=0, **o** A=0 **y** B=1, **o** A=1 **y** B=1.
Dicho de otra manera, podemos interpretar como respuesta válida que F será 1 cuando no ocurra A y no ocurra B, o no ocurra A y sí ocurra B, o cuando ocurran A y B.
Esto que es tan simple de entender en lenguaje cotidiano, se traslada con el mismo concepto a la idea de suma de productos, considerando que estamos haciendo una Disyunción/Suma (con la simbología que deseemos: O, Or, ˅, +) de Conjunciones/Productos (simbología: y, And, ˄, .). En conclusión podemos inferir de la anterior tabla de verdad lo siguiente:

$\boxed{F = \overline{A}.\overline{B} + \overline{A}.B + A.B}$ (Por convención y de manera análoga a las operaciones aritméticas conocidas entendemos que ante la ausencia de paréntesis se calculan primero los productos y luego las sumas con los resultados intermedios de cada producto).

Para validar la veracidad de lo expuesto, se debe armar la tabla de verdad de la proposición compuesta y comprobar que coinciden las salidas para todas las combinaciones posibles de la tabla original.

Imaginemos ahora una función que recibe 4 variables A,B,C,D que representan los 4 dígitos de un número binario (Siendo D el menos significativo hasta A como más significativo)….Respondamos ahora la siguiente pregunta:
¿Cuándo viene representado el número 5? (Sabemos que el 5 se representa en binario como 0101)

Rta: cuando viene A=0 **y** B=1 **y** C=0 **y** D=1. O dicho de otra manera, cuando NO ocurra A y SI ocurra B y NO ocurra C y SI ocurra D.
Conclusión: Se puede representar una ecuación que retorne 1 cuando en las cuatro entradas reciba el número 5, de la siguiente manera: $F_5 = \overline{A}.B.\overline{C}.D$ (Notar que la salida F_5 tomará valor 1 exclusivamente cuando las entradas ABCD sean 0101)
Ahora estamos preparados para determinar una ecuación que, por ejemplo, retorne 1 cuando el número representado en las cuatro entradas sea 5 **o** 7 **o** 9 (es decir 0101 **o** 0111 **o** 1001)

$F = \overline{A}.B.\overline{C}.D + \overline{A}.B.C.D + A.\overline{B}.\overline{C}.D$ (Notar que la salida F tomará el valor 1 exclusivamente cuando las entradas sean alguna de las 3 definidas, en cualquier otra combinación de entrada, la ecuación retornará 0).

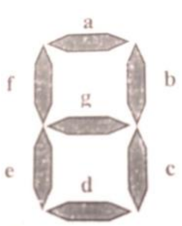
---

11. Para la siguiente tabla de verdad encuentre una fórmula lógica correspondiente (utilizando suma de productos).

A	B	C	F
0	0	0	**0**
0	0	1	**1**
0	1	0	**1**
0	1	1	**0**
1	0	0	**0**
1	0	1	**1**
1	1	0	**0**
1	1	1	**0**

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

12. Diseñe un circuito que tenga como entrada código BCD empaquetado (4 entradas) y 7 salidas para controlar los 7 segmentos de un display numérico, siendo la salida para los segmentos '0' para apagado y '1' para prendido. Construya la tabla de verdad y la ecuación de la salida correspondiente a los segmentos **a, b, c, d, e, f** y **g**.
→ COMPLETAR LA TABLA DE VERDAD Y EL RESTO DE LAS ECUACIONES

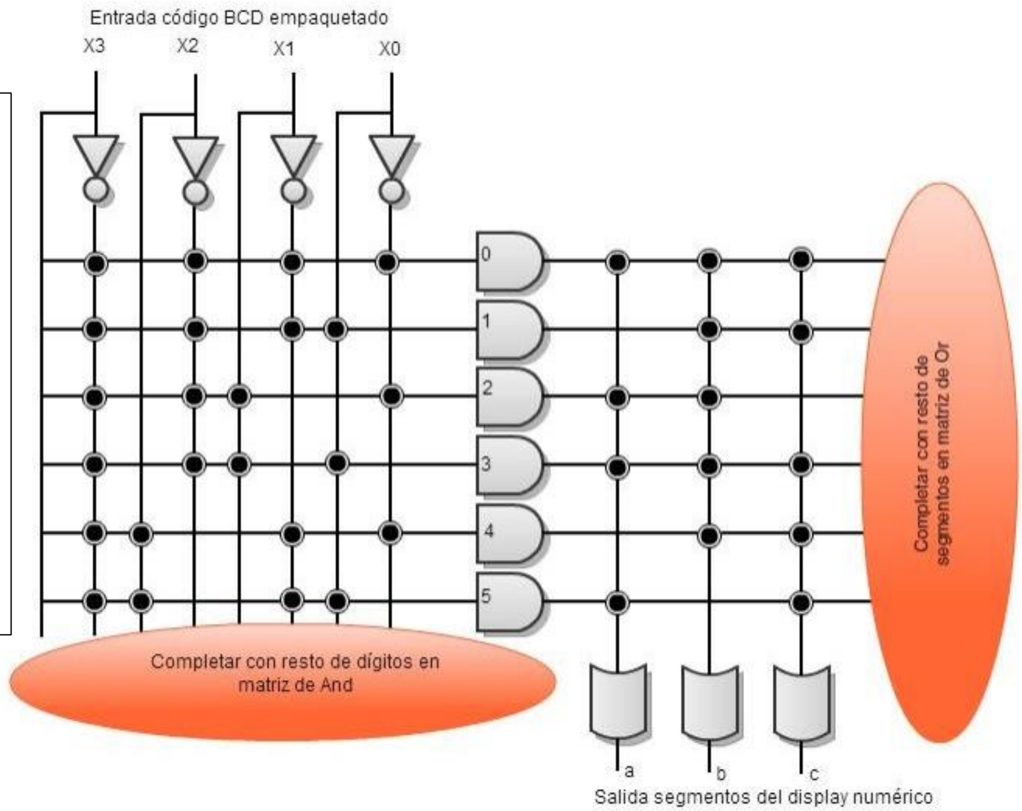


Ayuda 1: Cada segmento se considera como una salida distinta, y cada uno se debe activar (poner en 1) dependiendo del número recibido en las entradas que representan los 4 bits de un BCD empaquetado.
Ejemplo: El segmento **b** se debe activar cuando se recibe un 1 (0001), o un 2 (0010), o un 3 (00110, o un 4 (0100), o un 7 (0111), o un 8 (1000), o un 9 (1001). Se aplica la misma idea con el resto de las salidas.

	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1					0		
4	0	1	0	0					0		
5	0	1	0	1					0		
6	0	1	1	0					1		
7	0	1	1	1					1		
8	1	0	0	0					1		
9	1	0	0	1					0		

$$e = \overline{A}.\overline{B}.\overline{C}.\overline{D} + \overline{A}.\overline{B}.C.\overline{D} + \overline{A}.B.C.\overline{D} + \overline{A}.B.C.D + A.\overline{B}.\overline{C}.\overline{D}$$

<u>Ayuda 2:</u> Gráficamente, el circuito con las 4 entradas y las 7 salidas conviene diseñarlo como una matriz de compuertas And, seguida de la matriz de compuertas Or (basarse en la siguiente gráfica parcial del circuito:



13. Un controlador de proceso industrial recibe como entrada tres señales de temperatura T1, T2, T3 (T1<T2<T3) que adoptan el valor lógico '1' cuando la temperatura es mayor que t1, t2 y t3 respectivamente. Diseñar un circuito que genere una señal F cuando la temperatura esté comprendida entre t1 y t2 o cuando la temperatura sea mayor que t3.
→ EJERCICIO A RESOLVER

<u>Tener en cuenta para ejercicios 14 al 18:</u>

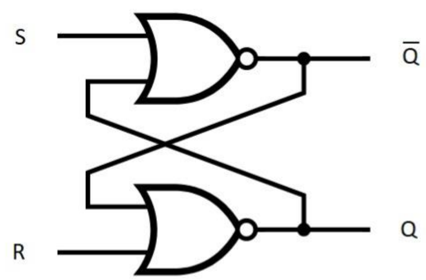
Circuitos Secuenciales: <u>(repasar apuntes de la cátedra y teoría)</u>

- Flip flop S-R asincrónico:
  - Problemas de sincronismo ante cambios de entrada durante el cálculo.
  - Reacción frente a doble entrada de 1's.
- Flip flop S-R sincrónico:
  - Resuelve problema de sincronismo, pero mantiene problema ante doble entrada de 1's.
- Flip flop D:
  - Pequeña variante del S-R que resuelve el problema de la doble entrada de 1's.
- Flip flop J-K:
  - Incorpora posibilidad de alterar el valor previo (complemento lógico).
- Flip flop T:
  - Pequeña variante del J-K, que sólo se dedica a invertir su valor ante cada orden del clock.

---

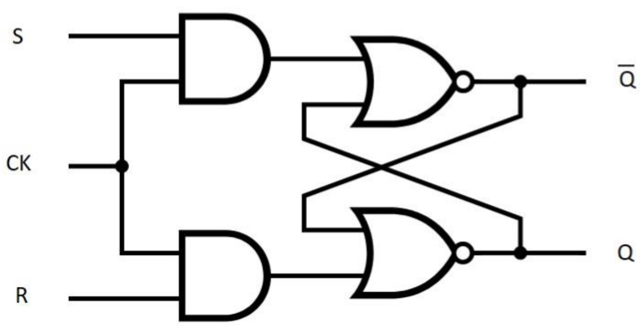
14. Dibuje el esquema de compuertas que componen un flip flop S-R. Describa a través de una tabla los estados en función de las entradas. Modifique el esquema anterior para hacerlo sincrónico. Describa gráficamente su respuesta temporal.

SR



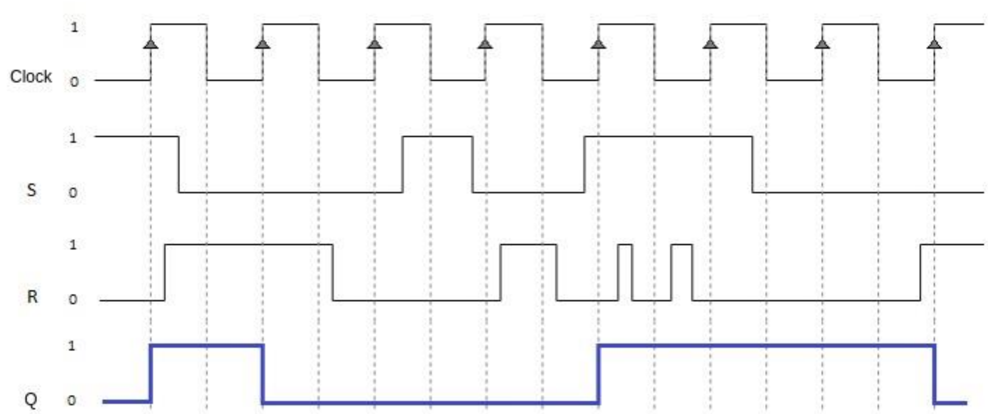
S	R	Q_{N+1}
0	0	Q_N
0	1	0
1	0	1
1	1	PROHIBIDO

SR SINCRONICO:

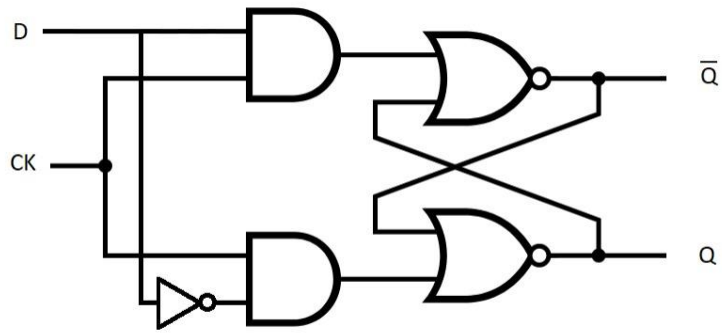


Clock	S	R	Q_{N+1}
1↑	0	0	Q_N
1↑	0	1	0
1↑	1	0	1
1↑	1	1	PROHIBIDO
0	-	-	Q_N

SR SINCRONICO: RESPUESTA TEMPORAL.
ACTIVADO POR FLANCO ASCENDENTE (CLOCK ↑)

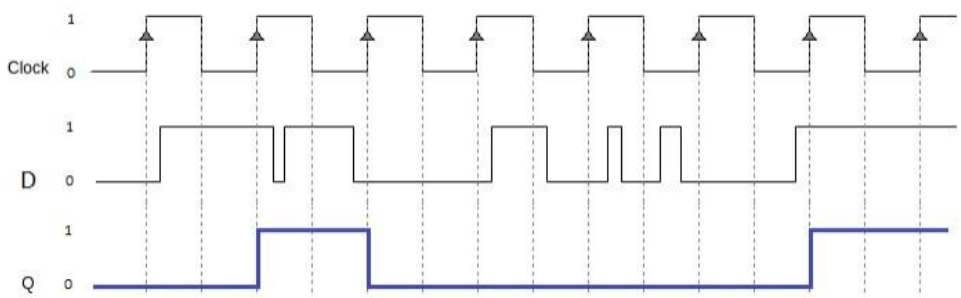


15. Dibuje el esquema de un flip flop D. Detalle en su respuesta temporal como resuelve el problema de la doble entrada de 1’s que se presentaba en el S-R.

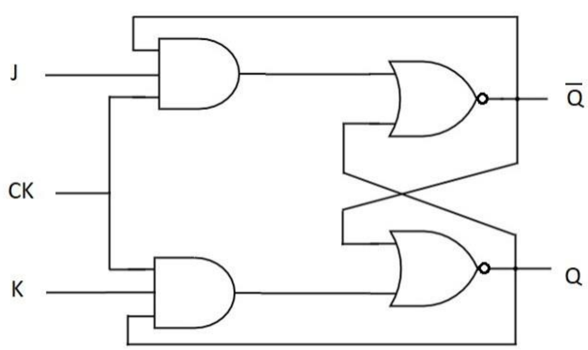


Clock	D	Q_{N+1}
1↑	0	0
1↑	1	1
0	-	Q_N

RESPUESTA TEMPORAL. ACTIVADO POR FLANCO ASCENDENTE (CLOCK ↑)

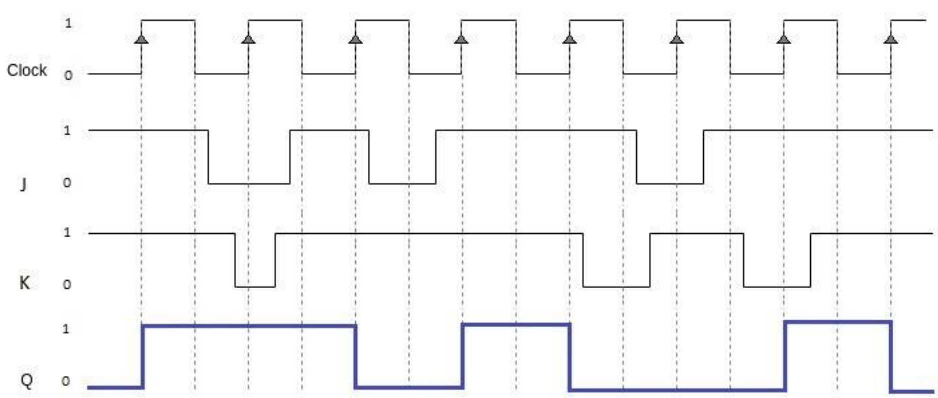


16. Dibuje el esquema de un flip flop J-K, describiendo su respuesta temporal.

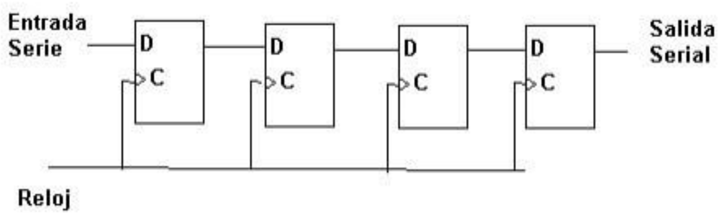


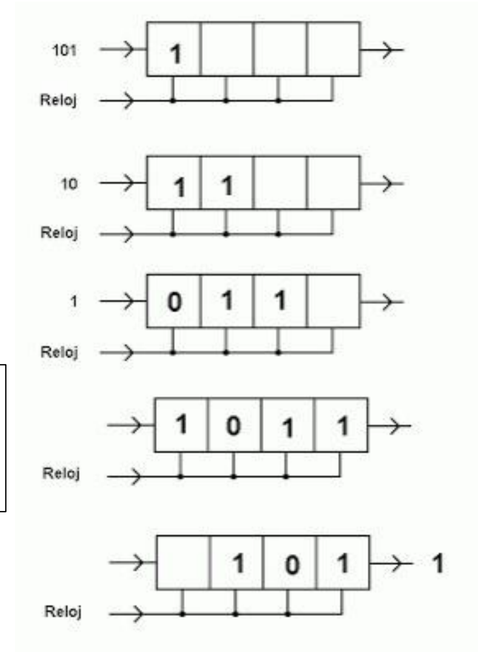
Clock	J	K	Q_{N+1}
1↑	0	0	Q_N
1↑	0	1	0
1↑	1	0	1
1↑	1	1	$\overline{Q_N}$
0	-	-	Q_N

RESPUESTA TEMPORAL. ACTIVADO POR FLANCO ASCENDENTE (CLOCK ↑)



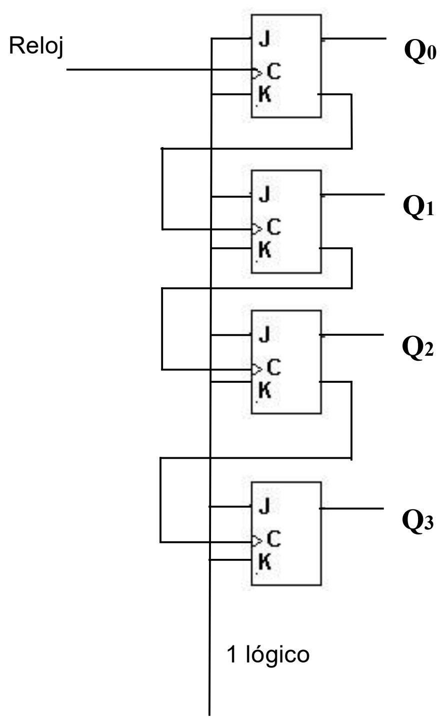
17. Dibuje el diagrama de tiempos del registro de la figura, implementado con flip flops D. Modifíquelo para desplazamiento izquierda derecha y derecha izquierda. **→ EJERCICIO A RESOLVER**



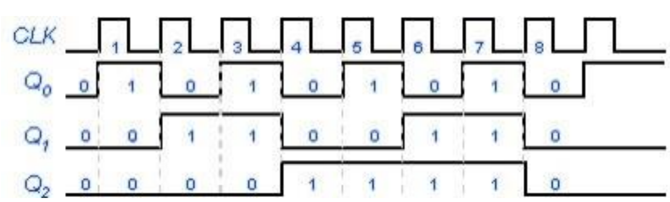


Ayuda: Ejemplo de respuesta temporal para interpretar como responde el registro previo ante la entrada serial del número binario 1011:

18. Describa gráficamente la respuesta temporal de cada flip flop ante una señal de unos y ceros entrando por Reloj. **→ EJERCICIO A RESOLVER**



Ayuda: El diagrama correspondiente considerando sólo los primeros 3 flip-Flops es el siguiente:



Se observa como la respuesta de cada flip-flop emite una onda a la mitad de frecuencia que su clock de entrada.