

Tema I

Lógica Combinacional

1.1. Sistema de numeración

La rama Digital de la Electrónica utiliza el sistema de numeración binario, en el cual únicamente existen dos dígitos: '0' y '1', denominados bits. Esto es debido a que los dispositivos van a ser modelados como interruptores, los cuales pueden tener dos valores: *conducción* y *en corte*. Dichos interruptores serán controlados para poder variar entre los dos estados que pueden tomar, como se observa en la figura 1.1.

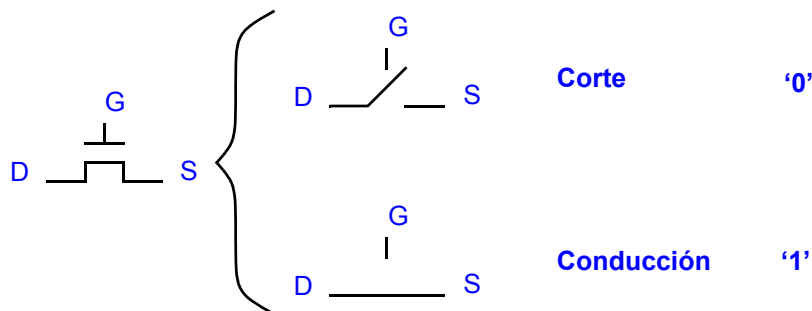


Figura 1.1.- Sistema numérico de Electrónica Digital.

No obstante, la mayoría de las personas (y por tanto potenciales usuarios de sistemas digitales) utilizan básicamente el sistema numérico decimal, y desconocen el sistema binario. Luego, es necesario que los diseñadores (no los usuarios) conozcan la manera de compatibilizar ambos sistemas numéricos, es decir, algunas reglas para pasar de un sistema a otro. En principio disponemos de dos métodos, uno de cuyas diferencias se encuentra en el sistema en el que se realizarán las operaciones: método polinómico, en el que las operaciones se realizan en el sistema destino; y método iterativo, en el que las operaciones se realizan en el sistema fuente. Por lo tanto, y tomando la premisa de realizar todas las operaciones en el sistema decimal, utilizaremos el método polinómico para pasar de binario a decimal y el método iterativo para pasar de decimal a binario.

1.1.1. Método polinómico

El método polinómico se basa en la notación posicional y el valor numérico.

La notación posicional es aquella en la que un número se representa por una serie de dígitos, cuya posición determina el peso que tendrá en su valor numérico.

Por lo tanto, en el sistema decimal, el valor numérico de un número será el resultado de calcular la siguiente serie.

$$(N_3N_2N_1N_0) = \sum_{i=0}^3 N_i 10^i$$

Luego, si estamos en un sistema binario, únicamente hay que cambiar la base por la que se realizan las multiplicaciones. Luego, el resultado es:

$$(N_3N_2N_1N_0) = \sum_{i=0}^3 N_i 2^i$$

Una vez que conocemos la manera de obtener el valor numérico, si dicho valor numérico es calculado en el sistema decimal, tendremos su valor numérico en el sistema decimal. Por lo tanto, para pasar un número del sistema binario al sistema decimal únicamente hay que evaluar dicho sumatorio. En el caso que tengamos números decimales, el procedimiento es el mismo pero con exponentes negativos. A modo de ejemplo presentamos varias conversiones en la figura 1.2.

$$\begin{aligned} 100101 &= 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 35 \\ 1111 &= 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 15 \\ 011010 &= 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 26 \\ 110.001 &= 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 + 0 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} = 6.125 \\ 1000.10 &= 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} = 8.5 \end{aligned}$$

Figura 1.2.- Ejemplos de conversión entre sistemas binario y decimal.

1.1.2. Método iterativo

Observando el valor numérico de un número, utilizando el teorema de la división (el cual nos dice que el dividendo es igual al divisor por el cociente más el resto, de tal forma que el resto siempre es menor que el divisor), y sabiendo que los dígitos de un sistema numérico son menores al valor de su base podemos obtener las siguientes igualdades:

$$\frac{\text{Dividendo}}{\text{divisor}} = \text{cociente} + \frac{\text{resto}}{\text{divisor}}$$

$$\frac{(N_n \dots N_1 N_0)}{2} = \sum_{i=1}^n N_i 2^i + \frac{N_0}{2}$$

Por lo tanto, el resto de la división por dos (la base) es el bit menos significativo. Si volvemos a hacer la división con el cociente que nos ha quedado, el nuevo resto será el bit menos significativo del cociente, es decir, el bit N_1 en la división del primer cociente.

A modo de ejemplo, en la figura 1.3 mostramos varias conversiones de decimal a binario. Como se ha comentado previamente, el primer resto se corresponde con el bit menos significativo, mientras que el último resto (o el cociente de la última división) se corresponde con el bit más significativo.

El método anterior únicamente sirve para números enteros, ya que al utilizar números decimales, el resto no sólo sería un bit sino que se correspondería con varios (más concreta-

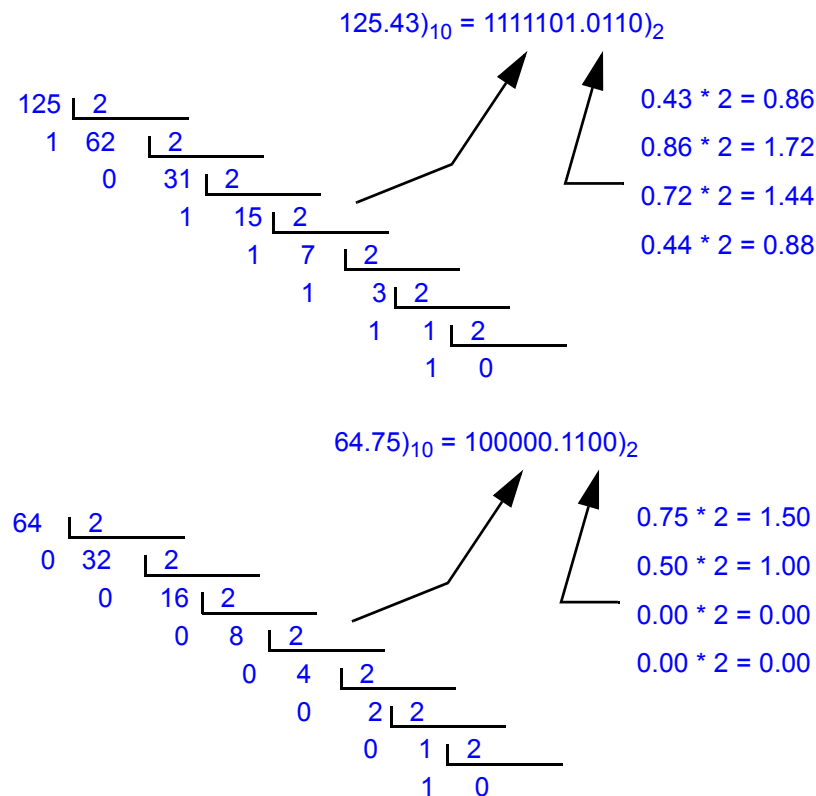


Figura 1.5.- Ejemplos de conversión

1.1.3. Otros sistemas numéricos

A pesar de que el sistema numérico que gobierna la Electrónica Digital es el sistema binario, dicho sistema no suele ser muy utilizado cuando el valor numérico que queremos representar es grande. Así por ejemplo, el número 100 decimal se representa con tres dígitos, mientras que su representación en el sistema binario tiene siete bits.

Para estos casos se utiliza el sistema hexadecimal (aunque también se utiliza el sistema octal en menor medida). El sistema hexadecimal (octal) es de base 16 (8), y los dígitos serán {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F} ({0, 1, 2, 3, 4, 5, 6, 7}), donde los dígitos A, B, C, D, E y F tienen el valor numérico 10, 11, 12, 13, 14 y 15. Siempre podemos pasar de un sistema a otro utilizando los métodos mencionados anteriormente; si las dos bases son diferentes de la decimal, por ejemplo de hexadecimal a octal, se recomienda como paso intermedio convertir los números a decimal.

No obstante, la importancia de los sistemas hexadecimales y octales en Electrónica Digital radica en que sus bases se pueden expresar como potencias de dos (la base del sistema binario). Por lo tanto, veamos la correspondencia en la representación polinómica del sistema binario y hexadecimal.

$$N_7N_6N_5N_4N_3N_2N_1N_0|_2 = N_72^7 + N_62^6 + N_52^5 + N_42^4 + N_32^3 + N_22^2 + N_12^1 + N_02^0$$

El cual se pueden agrupar en potencias de 16 de la siguiente forma:

$$N_7N_6N_5N_4N_3N_2N_1N_0|_2 = (N_72^3 + N_62^2 + N_52^1 + N_42^0)16^1 + (N_32^3 + N_22^2 + N_12^1 + N_02^0)16^0$$

Ahora, si en lugar de bits, utilizamos dígitos hexadecimales, nos quedaría la siguiente expresión:

$$N_7N_6N_5N_4N_3N_2N_1N_0|_2 = (H_1)16^1 + (H_0)16^0$$

Luego, un dígito hexadecimal (octal) se correspondería con cuatro (tres) bits. Además, la conversión de hexadecimal (octal) a binario o viceversa no debe realizarse sobre el número completo sino por grupos de cuatro (tres) bits.

Luego, el método para convertir números binarios a hexadecimales (octal) se deben agrupar los bits en grupos de cuatro (tres), y convertir cada uno de los grupos por separado. En el caso de que los grupos no sean completos, se completarán con 0's a la izquierda (si el grupo está a la izquierda de la coma decimal) o a la derecha (si el grupo está a la derecha de la coma decimal). En el caso de que la conversión sea de hexadecimal (octal) a binario, se convertirá dígito a dígito, en la cual se realizarán en grupos de cuatro (tres) bits.

En la figura 1.6 mostramos algunos ejemplos de conversión directa entre sistemas binario y hexadecimal.

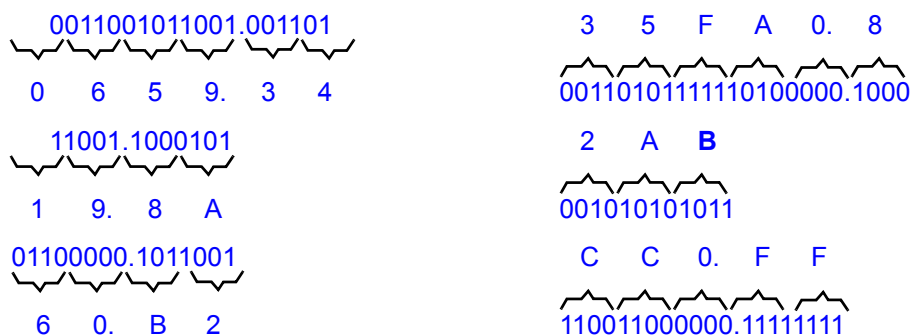


Figura 1.6.- Ejemplos de conversión directa.

1.2. Leyes del álgebra de Boole

Hasta ahora, hemos indicado el sistema numérico que gobierna la Electrónica Digital. Ahora debemos dar sus leyes matemáticas. Estas leyes coinciden con las de un álgebra de Boole, en el cual determinado un conjunto B y dos operaciones, suma y producto, se cumplen los cuatro siguientes postulados:

- P1.- las operaciones tienen la propiedad conmutativa.
 $a+b = b+a$
 $a \cdot b = b \cdot a$
- P2.- las operaciones son distributivas entre sí
 $a \cdot (b+c) = a \cdot b + a \cdot c$
 $a+(b \cdot c) = (a+b) \cdot (a+c)$

- P3.- las operaciones tienen elementos identidad diferentes dentro de B. Estos elementos son definidos como 0 para (+) y 1 para (\cdot).
 $a+0 = a$
 $a \cdot 1 = a$
- P4.- para cada elemento, a, del conjunto B, existe otro elemento denominado complemento, \bar{a} también del conjunto B, tal que se cumple:
 $a+\bar{a} = 1$
 $a \cdot \bar{a} = 0$

Cuando el conjunto es limitado al conjunto binario $\{0, 1\}$ y las operaciones están definidas según la tabla 1.1 se habla de álgebra de conmutación.

Tabla 1.1. Operaciones del álgebra de conmutación.

| a | b | $\bar{a} = a'$ | $a \cdot b$ | $a+b$ |
|---|---|----------------|-------------|-------|
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |

Una operación estará determinada por la combinación de entradas en la que su valor de salida sea único, es decir, el producto estará determinado por la combinación que hace que valga '1' (y por lo tanto, todas sus entradas sean '1'), y la suma estará determinada por la combinación que hace que valga '0' (y por lo tanto, todas sus entradas sean '0').

A continuación se muestra las principales leyes que se cumplen en cualquier álgebra de Boole, y por lo tanto, el álgebra de conmutación.

Principio de dualidad.- Cualquier teorema o identidad algebraica deducible de los postulados anteriores puede transformarse en un segundo teorema o identidad válida sin mas que intercambiar las operaciones binarias y los elementos identidad.

Teorema 1.1.- El elemento \bar{a} del 4º postulado (denominado complemento o negación de a) está unívocamente determinado, es decir, es único.

Teorema 1.2.- (o Teorema de elementos nulos) Para cada cualquier elemento a, se verifican las siguientes igualdades
 $a+1 = 1$
 $a \cdot 0 = 0$

Teorema 1.3.- Cada uno de los elementos identidad es el complemento del otro, es decir, $1' = 0$ y $0' = 1$

Teorema 1.4.- (o Teorema de idempotencia) Para cada elemento a, se verifican las siguientes igualdades:
 $a + a = a$
 $a \cdot a = a$

Teorema 1.5.- (o Teorema de involución) Para cada elemento de a , se verifica que el complemento del complemento de a es a , es decir,
 $(a')' = a$

Teorema 1.6.- (o Teorema de absorción) Para cada par de elementos, a y b , se verifica:

$$a + a \cdot b = a$$

$$a \cdot (a + b) = a$$

Teorema 1.7.- Para cada par de elementos, a y b , se verifica:

$$a + a' \cdot b = a + b$$

$$a \cdot (a' + b) = a \cdot b$$

Teorema 1.8.- (o Leyes de DeMorgan) Para cada par de elementos, a y b , se verifica

$$(a + b)' = a' \cdot b'$$

$$(a \cdot b)' = a' + b'$$

Teorema 1.9.- (o Leyes de DeMorgan generalizadas) Para cualquier conjunto de elementos se verifica:

$$\overline{(X_0 + X_1 + \dots + X_n)} = \overline{X_0} \cdot \overline{X_1} \cdot \dots \cdot \overline{X_n}$$

$$\overline{(X_0 \cdot X_1 \cdot \dots \cdot X_n)} = \overline{X_0} + \overline{X_1} + \dots + \overline{X_n}$$

Teorema 1.10.- (o Teorema de asociatividad) Cada uno de los operadores binarios (+) y (\cdot) cumple la propiedad asociativa, es decir, para cada tres elementos, a , b y c , se verifica

$$(a + b) + c = a + (b + c)$$

$$(a \cdot b) \cdot c = a \cdot (b \cdot c)$$

1.3. Puertas lógicas

Una de las principales ventajas de utilizar el álgebra de conmutación radica en que las operaciones básicas de este álgebra (operación AND, OR y NOT) tienen un equivalente directo en términos de circuitos. Estos circuitos equivalentes a estas operaciones reciben el nombre de **puertas lógicas**. No obstante, el resto de circuitos lógicos básicos también reciben el nombre de puertas, aunque su equivalencia se produce hacia una composición de las operaciones lógicas básicas.

Las tres puertas fundamentales reciben el mismo nombre que los operadores, es decir, existen las **puertas AND**, **puertas OR** y **puertas NOT**. La última puerta recibe el nombre más usual de **inversor**. En la figura 1.7 mostramos los símbolos de estas puertas tanto tradicionales como internacionales, aunque usaremos preferentemente los símbolos tradicionales.

A pesar de tener ya disponibles las puertas que realizan las tres operaciones básicas del álgebra de conmutación, también existen otras puertas las cuales son muy utilizadas (a pesar de ser combinaciones de las anteriores). Estas puertas son las puertas NAND y NOR (que se corresponden con las operaciones AND y OR complementadas, respectivamente), y XOR y XNOR (que se corresponden con las funciones de paridad y paridad complementada, respectivamente, muy utilizadas en sistemas aritméticos). Estas puertas se muestran en la figura 1.8.

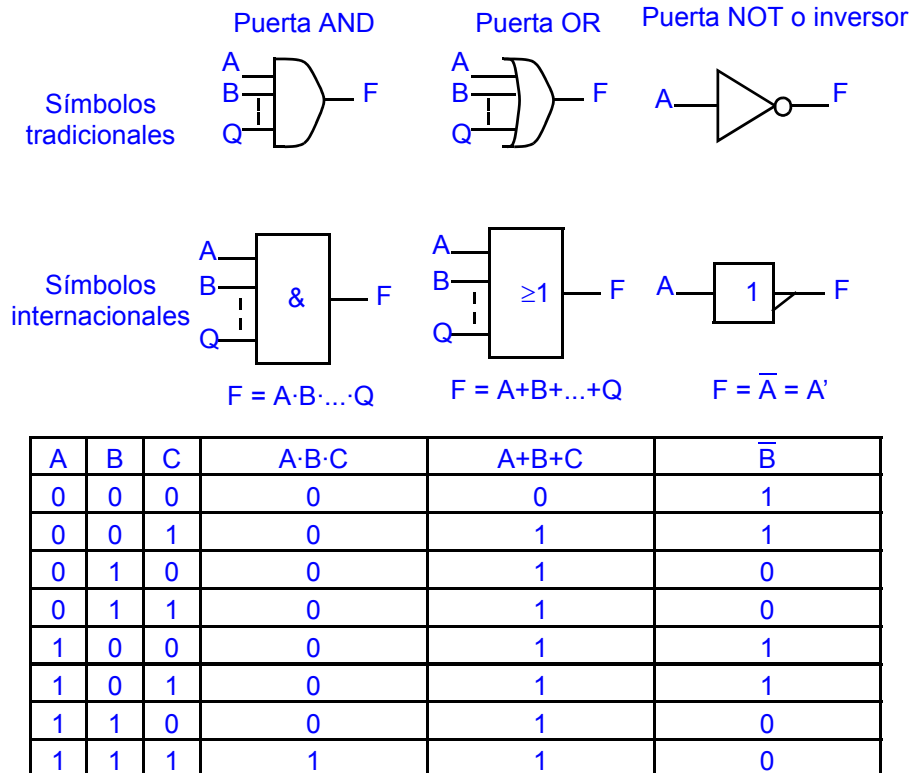


Figura 1.7.- Simbología tradicional e internacional y tabla de combinaciones correspondientes a las puertas lógicas básicas.

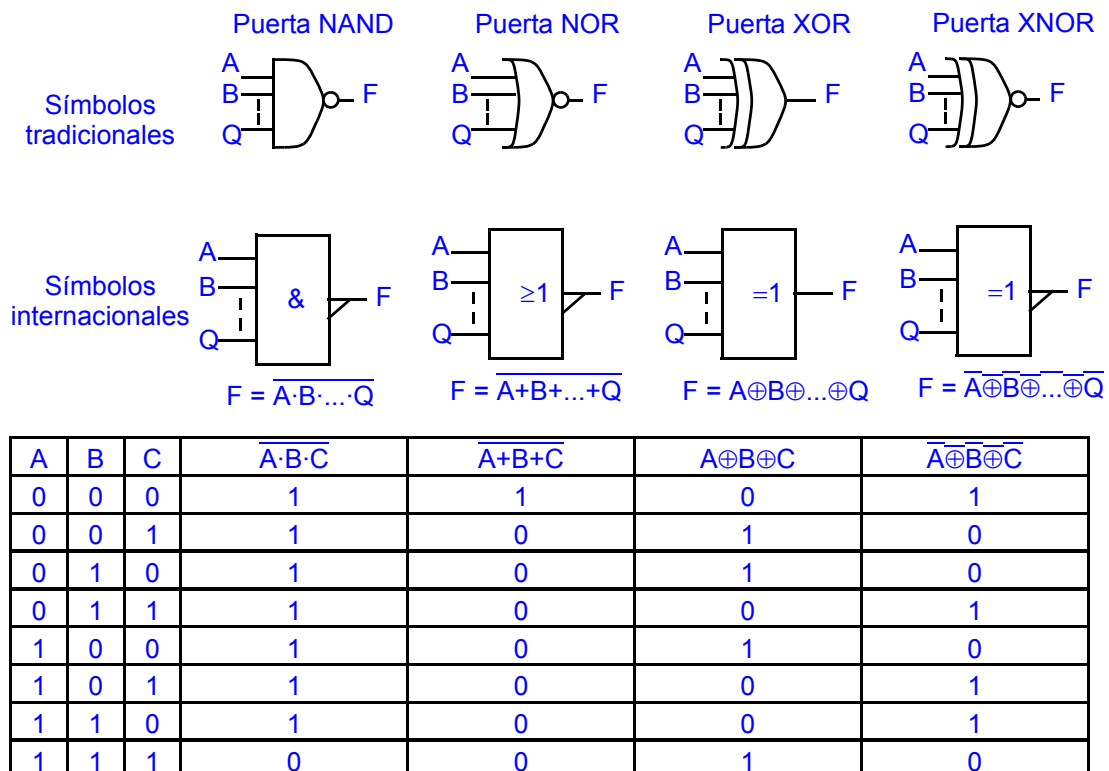
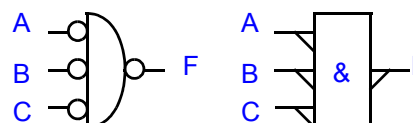


Figura 1.8.- Simbología tradicional e internacional y tabla de combinaciones correspondientes a las puertas lógicas compuestas.

En el caso de la simbología, cabe destacar que la referencia a la negación es un círculo (en el caso de la simbología tradicional) y una línea oblicua (en el caso de la simbología internacional), independientemente de donde se encuentre (entrada o salida), como se muestra en la figura 1.9.



| A | B | C | $F = \overline{(A \cdot B \cdot C)}$ |
|---|---|---|--------------------------------------|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

Figura 1.9.- Ejemplo de la simbología de la negación.

En esta asignatura se utilizará la simbología más utilizada, la cual es la simbología tradicional.

1.4. Análisis y síntesis combinacional

Como se ha podido comprobar del apartado anterior, existe una relación directa entre las puertas lógicas y las operaciones lógicas, lo cual limita la actuación a determinar la función lógica que se realiza. A la hora de representar una función combinacional, podemos optar por una fórmula lógica y/o una tabla de verdad (o de combinaciones).

La fórmula lógica es una expresión que contiene variables y operadores lógicos dispuesto de una forma *lógica*.

Una tabla de verdad (o de combinaciones) de una función de N variables es una tabla con (al menos) N+1 columnas (N variables de entrada y 1 variable de salida), con tantas filas como combinaciones diferentes puedan tener las N variables de entrada (2^N en el sistema binario). Los valores de las columnas de las variables de entrada son tales que estén representadas todas las combinaciones (por lo general en orden creciente), y los valores de la columna de la variable de salida serán los correspondientes a la combinación de las variables de entrada de dicha fila.

El problema de análisis se puede definir de la siguiente forma:

Dado un circuito electrónico, determina el comportamiento y la funcionalidad que presenta dicho circuito.

Es decir, para obtener la funcionalidad del circuito, sólo se tiene que sustituir cada una de las puertas por su funcionalidad según el flujo de señal (jerarquía de operaciones), como podemos apreciar en el ejemplo de la figura 1.10.

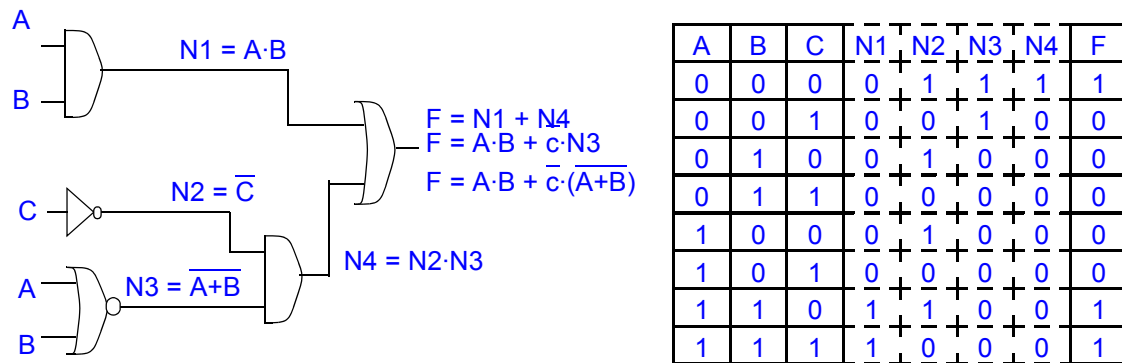


Figura 1.10.- Ejemplo de análisis de un circuito combinacional.

El problema de diseño se puede definir de la siguiente forma:

Dados un comportamiento y una funcionalidad, determina el circuito electrónico que los presenta.

Dicho problema se suele dividir en dos partes:

- Obtención de la función mediante su tabla de combinaciones.
- Obtención de la fórmula lógica a partir de su tabla.

1.4.1. Obtención de la tabla de combinaciones

Para llevar a cabo esta tarea, no existe ningún método. De hecho, únicamente hay que transcribir el comportamiento a valores lógicos de las diferentes señales (entrada y salida) del sistema. Veamos un ejemplo:

Se dispone de un campo de cultivo inteligente que automáticamente abre un toldo para permitir que los rayos solares lleguen al cultivo y abre y/o cierra las válvulas de riego. Para ello se dispone de tres sensores: humedad (H), temperatura (T) y luminosidad (L), programados con sus valores correspondientes tal que estarán activados cuando se sobrepasen sus valores. El toldo (A) debe estar abierto en las condiciones siguientes:

- La temperatura y la humedad, ambos, no sobrepasen sus valores, o.
- La luminosidad sobrepase su valor, y la temperatura no.

En caso contrario, el toldo debe estar cerrado. En cambio, las válvulas de riego deben estar abiertas cuando se den las siguientes condiciones:

- La humedad no esté activada.
- La temperatura no sobrepase su valor.

Para realizar la transcripción, debemos generar la tabla de combinaciones (con sus entradas y salidas) e identificar cada una de las combinaciones con su significado en el sistema que nos ocupa, mostrada en la tabla 1.2. Una vez que tengamos el significado de cada combinación, sólo tenemos que determinar las condiciones expresadas en el problema y darle el valor de salida apropiado.

Tabla 1.2. Tabla de combinaciones del sistema inteligente de cultivo

| Significado | Temp. | Hum. | Lumin. | Toldo | Riego |
|--|-------|------|--------|-------|-------|
| Ningún sensor alcanza su valor | 0 | 0 | 0 | 1 | 1 |
| Solamente alcanza su valor la luminosidad | 0 | 0 | 1 | 1 | 1 |
| Solamente alcanza su valor la humedad | 0 | 1 | 0 | 0 | 1 |
| Solamente no alcanza su valor la temperatura | 0 | 1 | 1 | 1 | 1 |
| Solamente alcanza su valor la temperatura | 1 | 0 | 0 | 0 | 1 |
| Solamente no alcanza su valor la humedad | 1 | 0 | 1 | 0 | 1 |
| Solamente no alcanza su valor la luminosidad | 1 | 1 | 0 | 0 | 0 |
| Todos alcanzan sus valores | 1 | 1 | 1 | 0 | 0 |

También podríamos obtener directamente una fórmula transcribiendo directamente las especificaciones con las siguientes premisas:

- Una negación de una variable se interpreta como el complemento de la variable.
- El “y” se interpreta como la operación AND.
- El “o” se interpreta como la operación OR.

Por lo tanto las fórmulas quedarían de la siguiente forma:

$$\text{TOLDO} = \overline{\text{TEMPERATURA}} \cdot \overline{\text{HUMEDAD}} + \text{LUMINOSIDAD} \cdot \text{TEMPERATURA}$$

$$\text{RIEGO} = \overline{\text{HUMEDAD}} + \overline{\text{TEMPERATURA}}$$

No obstante, se suele dar la tabla de combinaciones ya que es el punto de partida para la tarea siguiente.

1.4.2. Obtención de la fórmula lógica a partir de su tabla

Una forma fácil de obtener una fórmula lógica es identificar los términos directamente de la tabla de combinaciones. Dichos términos pueden ser sumas o productos, en función del valor de salida que consideremos.

Si consideramos los 1's (valor que identifica a un producto, como se vio anteriormente), cada combinación que dé dicho valor será un término producto, los cuales se sumarán. En dicho término, todos sus operandos deben tomar el valor '1', por lo que si la variable de entrada vale '1' aparecerá tal cual; pero si vale '0' aparecerá complementada para obtener el operador igual a '1'. Luego, la señal de salida TOLDO tomará la siguiente fórmula:

$$\text{TOLDO} = \overline{\text{TEMP}} \cdot \overline{\text{HUM}} \cdot \overline{\text{LUM}} + \overline{\text{TEMP}} \cdot \overline{\text{HUM}} \cdot \text{LUM} + \overline{\text{TEMP}} \cdot \text{HUM} \cdot \text{LUM}$$

Si consideramos los 0's (valor que identifica a una suma, como se vio anteriormente), cada combinación que dé dicho valor será un término suma, los cuales se multiplicarán. En dicho término, todos sus operandos deben tomar el valor '0', por lo que si la variable de entrada vale '1' aparecerá complementada para obtener el operador igual a '0'; pero si vale '0' aparecerá tal cual. Luego, la señal de salida RIEGO tomará la siguiente fórmula:

$$\text{RIEGO} = (\overline{\text{TEMP}} + \overline{\text{HUM}} + \overline{\text{LUM}}) \cdot (\overline{\text{TEMP}} + \overline{\text{HUM}} + \overline{\text{LUM}})$$

Así, en la tabla 1.3 se muestran los términos producto y suma correspondiente a las diferentes combinaciones de señales de entrada.

Tabla 1.3. Términos producto y suma

| A | B | C | Término producto | Término suma |
|---|---|---|--|--|
| 0 | 0 | 0 | $\overline{A} \cdot \overline{B} \cdot \overline{C}$ | $A+B+C$ |
| 0 | 0 | 1 | $\overline{A} \cdot \overline{B} \cdot C$ | $A+B+\overline{C}$ |
| 0 | 1 | 0 | $\overline{A} \cdot B \cdot \overline{C}$ | $A+\overline{B}+C$ |
| 0 | 1 | 1 | $\overline{A} \cdot B \cdot C$ | $A+\overline{B}+\overline{C}$ |
| 1 | 0 | 0 | $A \cdot \overline{B} \cdot \overline{C}$ | $\overline{A}+B+C$ |
| 1 | 0 | 1 | $A \cdot \overline{B} \cdot C$ | $\overline{A}+B+\overline{C}$ |
| 1 | 1 | 0 | $A \cdot B \cdot \overline{C}$ | $\overline{A}+\overline{B}+C$ |
| 1 | 1 | 1 | $A \cdot B \cdot C$ | $\overline{A}+\overline{B}+\overline{C}$ |

No obstante, la fórmula obtenida de la forma anterior no es mínima, y por lo tanto, el circuito asociado puede ser minimizado en el número de puertas y/o en el número de entradas de cada puerta. De hecho, si partimos de la fórmula de la señal TOLDO, y aplicando las diferentes leyes del álgebra de Boole podemos llegar a la siguiente expresión:

$$\text{TOLDO} = \overline{\text{TEMP}} \cdot \overline{\text{HUM}} \cdot \overline{\text{LUM}} + \overline{\text{TEMP}} \cdot \overline{\text{HUM}} \cdot \text{LUM} + \overline{\text{TEMP}} \cdot \text{HUM} \cdot \text{LUM}.$$

Aplicando la propiedad distributiva en los dos primeros términos obtenemos

$$\text{TOLDO} = \overline{\text{TEMP}} \cdot \overline{\text{HUM}} \cdot (\overline{\text{LUM}} + \text{LUM}) + \overline{\text{TEMP}} \cdot \text{HUM} \cdot \text{LUM}.$$

Aplicando una de las formas del cuarto postulado al paréntesis obtenemos

$$\text{TOLDO} = \overline{\text{TEMP}} \cdot \overline{\text{HUM}} + \overline{\text{TEMP}} \cdot \text{HUM} \cdot \text{LUM}.$$

Aplicando la propiedad distributiva obtenemos

$$\text{TOLDO} = \overline{\text{TEMP}} \cdot (\overline{\text{HUM}} + \text{HUM} \cdot \text{LUM})$$

Aplicando el teorema 1.7 al paréntesis obtenemos

$$\text{TOLDO} = \overline{\text{TEMP}} \cdot (\overline{\text{HUM}} + \text{LUM})$$

Luego, aplicando las diferentes leyes del álgebra de Boole podemos reducir la fórmula lógica, y por lo tanto, el circuito asociado. El problema de utilizar este método es que al utilizar más leyes, existe más probabilidad de equivocación en su tratamiento.

Una forma de minimización que no conlleva la aplicación de tantas leyes es la utilización del mapa de Karnaugh.

Un mapa de Karnaugh es una tabla de combinaciones con una disposición especial, de tal forma que los valores de las variables de entrada correspondiente a una celda y sus adyacentes sólo se diferencian en uno de ellos.

En la figura 1.11 mostramos la estructura de los mapas de Karnaugh de dos, tres y cuatro variables, así como los mapas de Karnaugh correspondientes a las señales RIEGO y TOLDO.

El método para obtener la fórmula a partir del mapa de Karnaugh consiste en seguir los siguientes pasos:

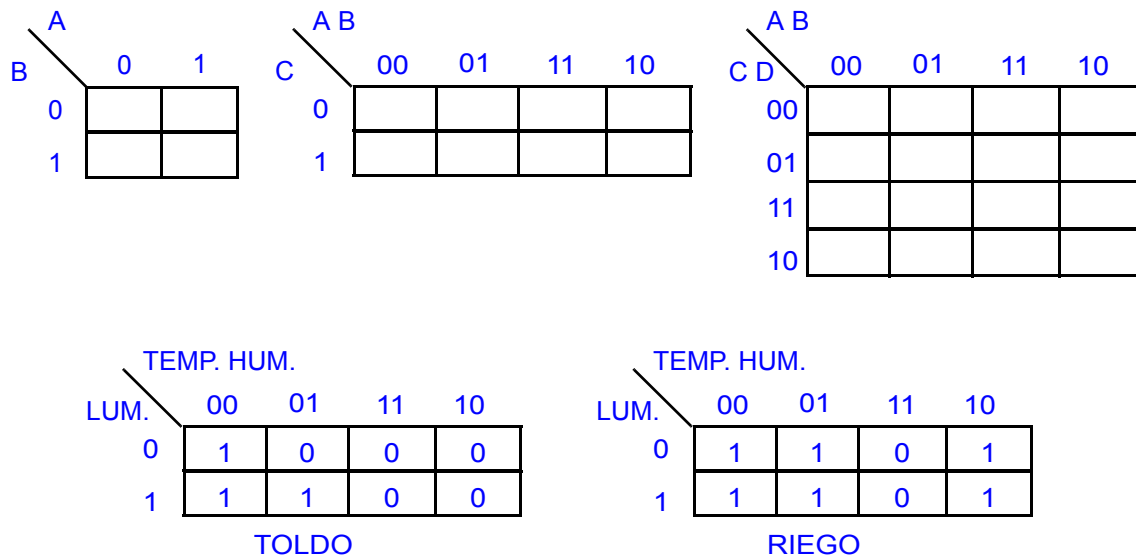


Figura 1.11.- Estructura y ejemplos del mapa de Karnaugh.

- Obtener los grupos de 1's (0's) adyacentes cuyo número sea igual a una potencia de dos.
- Elegir el número mínimo de grupos que engloban a todos los 1's (0's).
- Por cada grupo de 1's (0's) hay un término producto (suma). En dicho término aparecerán únicamente las variables que no cambien, de tal forma que la variables que tomen el valor '1' ('0') aparecerán sin complementar y si toman el valor '0' ('1') aparecerán complementadas.

En la figura 1.12a mostramos el ejemplo de obtener las señales TOLDO y RIEGO utilizando términos producto; mientras que en la figura 1.12b mostramos el mismo ejemplo utilizando términos sumas.

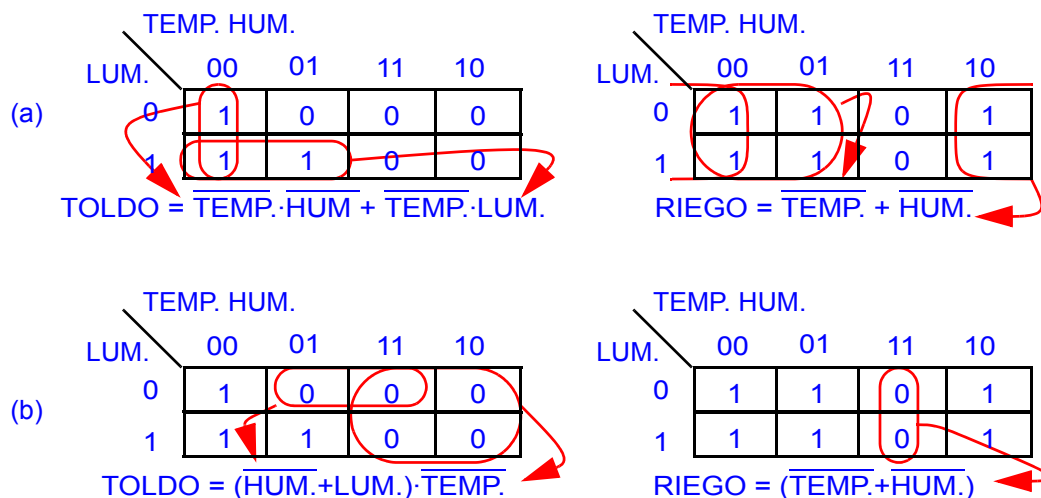


Figura 1.12.- Ejemplos del mapa de Karnaugh: (a) utilizando términos producto, y (b) utilizando términos sumas.

Tema II

Lógica secuencial

2.1. Comportamiento secuencial

Hasta ahora, únicamente hemos visto circuitos combinacionales, es decir, circuitos en los que las salidas dependen única y exclusivamente de las combinaciones de entradas, y no de la historia pasada del sistema. Vamos a ver cuáles serían las formas de onda del siguiente circuito.

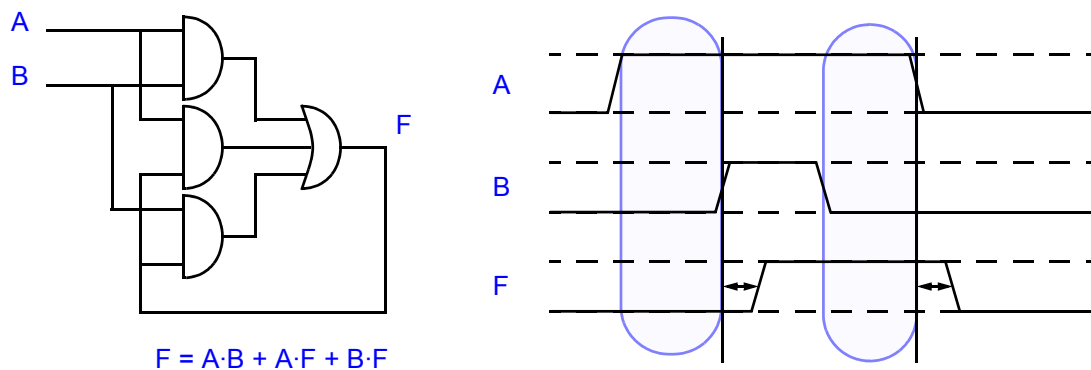


Figura 2.1.- Ejemplo de circuito digital junto con sus formas de ondas.

En las formas de onda podemos comprobar que la señal de salida F cambiará de valor (cuando sea necesario) después del retraso de la puerta AND y OR. Otra característica de este circuito que también podemos observar es que la salida no depende únicamente de las señales de entrada, A y B, ya que para una misma combinación de entradas, $AB = "10"$, obtenemos dos valores de salida diferentes, '0' y '1' respectivamente, mostradas en la figura 2.1. Si nos fijamos en las formas de onda, podemos observar que el valor de salida en dicha combinación de entrada coincide con el valor que tenía la señal de salida previamente. Este hecho implica que la salida depende de la historia pasada del circuito, además de las señales de entrada.

A estos sistemas que muestran dependencia con la historia, se les denomina sistemas secuenciales, más formalmente

Un circuito de conmutación secuencial se define como un circuito bivaluado en el cual, la salida en cualquier instante depende de las entradas en dicho instante y de la historia pasada (o secuencia) de entradas.

Estos sistemas están más extendidos que los sistemas puramente combinacionales, ya que presenta un aumento considerable en la funcionalidad.

La dependencia de esta historia puede ser ventajosa e incluso necesaria para algunas aplicaciones en las que es necesario recordar una determinada situación. Algunos ejemplos de esta ventaja (necesidad) pueden ser:

- La creación de un reloj, que se modela con la afirmación "la salida será el valor complementario de su valor anterior",
- La creación de un contador, que se modela con la afirmación "la salida será el resultado de sumar uno a su valor anterior",...

- Un sistema de control de un paso a nivel, el cual debe saber cuando el tren está dentro del paso, está entrando, saliendo o está fuera.

Por lo tanto, se ve cuando menos interesante incluir la dependencia del tiempo en los sistemas y en especial, en los sistemas digitales.

Según lo visto anteriormente, podemos dar un modelo de un sistema secuencial como el mostrado en la figura 2.2. En dicho modelo podemos observar la lógica combinacional que genera el procesado del sistema, y la realimentación (a través de elementos de memoria) que genera la dependencia con la historia del sistema.

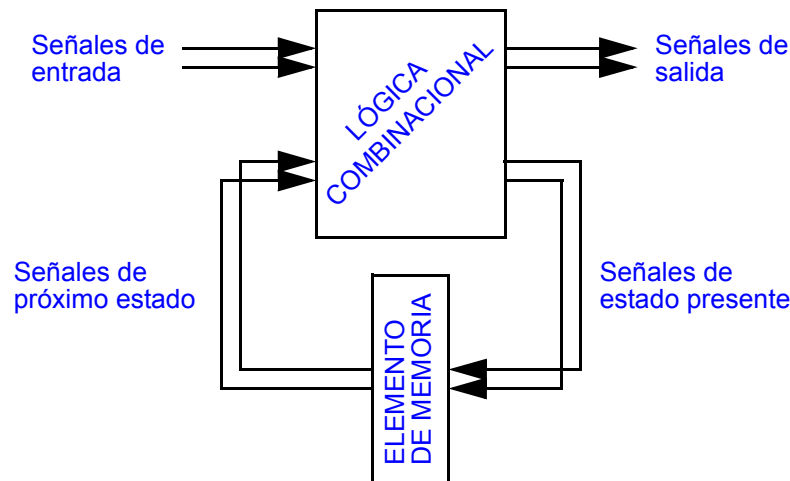


Figura 2.2.- Modelo genérico de un sistema secuencial.

Por lo tanto, los elementos de memoria son los encargados de temporizar la historia del circuito, es decir, indicar el momento en el que cambia la historia. Según este cambio, podemos clasificar esta temporización en tres categorías:

- Temporización por realimentación directa. Los cambios son considerados instantáneamente (despreciando los retrasos de las conexiones).
- Temporización por elementos de retraso. Los cambios son considerados después de que transcurra un tiempo estipulado, el cual viene determinado por el elemento de retraso utilizado.
- Temporización por elementos de memoria. Los cambios son considerados por un elemento de memoria, el cual suele estar controlado por una señal externa (denominada generalmente reloj).

La opción más utilizada es la tercera, es decir, la utilización de elementos de memoria. Esta situación es debida a que el control externo es mucho más versátil que la realimentación directa o a través de un elemento de retraso en las cuales no existe ningún control directo sobre el cambio de la historia del sistema.

Por lo tanto, vamos a considerar el estudio de los elementos de memoria, cuya importancia viene determinada por el comentario anterior. Además, dichos elementos son básicos para la generación de memoria de semiconductores (muy utilizadas en sistemas informáticos).

2.2. Elementos de memoria

Entre las muchas definiciones que podemos encontrar de elemento de memoria, vamos a elegir la siguiente:

Un **elemento de memoria** es aquel elemento capaz de almacenar un estado durante un tiempo determinado.

Los dos elementos, mostrados en la figura 2.3, son elementos de memoria.

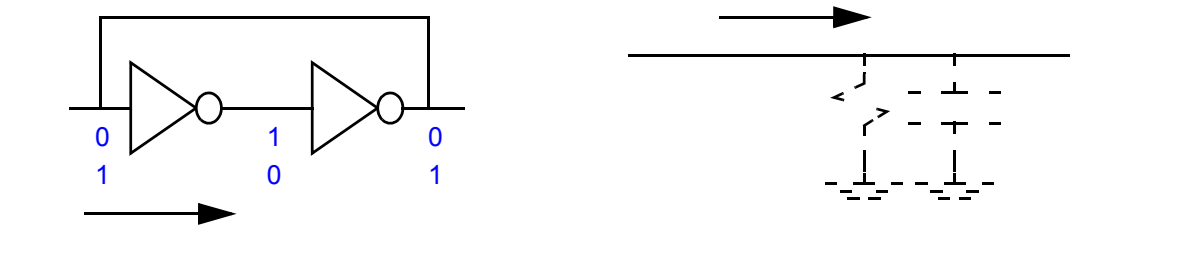


Figura 2.3.- Ejemplos de elementos de memoria

El primer elemento está formado por dos inversores realimentados de tal forma que el valor a la entrada del primer inversor es el mismo que a la salida del segundo inversor, estando de acuerdo con el lazo de realimentación. Por lo tanto, mientras que el dato de entrada no cambie, el dato de salida permanecerá sin cambiar, es decir, quedará almacenado. De igual forma podemos comprobar que una simple línea de conexión muestra el mismo comportamiento, de tal forma que la tensión es almacenada en el condensador parásito asociado a dicha línea.

La diferencia entre ambos elementos se encuentra en el tiempo que permanece almacenado el dato, característica que se suele denominar **duración de la información**. En el primer elemento, la información permanecerá almacenada indefinidamente hasta que el dato de entrada cambie su valor. En cambio, en el segundo caso, de la misma forma que hay un condensador parásito, también existe una resistencia parásita, creando un camino de descarga a través de la resistencia. Como el dato no es regenerado por ningún elemento (como sucede con los inversores en el primer elemento), cuando se sobrepasa un determinado tiempo, que se denomina tiempo de descarga y suele considerarse proporcional al producto RC , la tensión almacenada no es lo suficiente alta como para identificar un nivel lógico o cambia su valor. A este tipo de almacenamiento se denomina **almacenamiento dinámico**; mientras que cuando el dato permanece durante un tiempo indefinido, el almacenamiento se denomina **almacenamiento estático**. En el caso del almacenamiento dinámico, para evitar la pérdida de la información es necesario volver a almacenar la información de forma periódica (antes de superar el tiempo de descarga), lo cual se conoce como **ciclo de refresco**.

Otra propiedad que podemos encontrar en los ejemplos anteriores consiste en un almacenamiento instantáneo. Cuando el dato de entrada cambia, el valor almacenado en el elemento de memoria cambia de forma instantánea (después de que se haya superado el retraso impuesto por el elemento), como podemos ver en la figura 2.4. A esta propiedad se la conoce con el nombre de **transparencia**, diciéndose entonces que estamos considerando un **elemento de memoria transparente**.

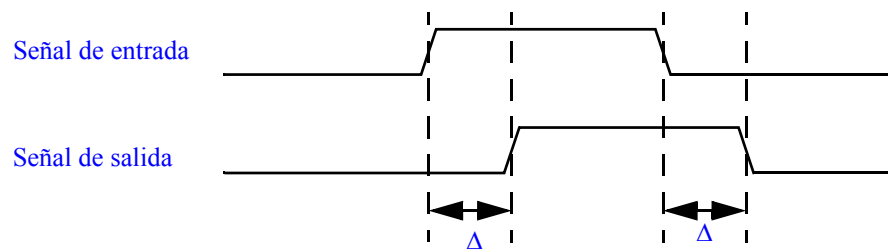


Figura 2.4.- Formas de onda correspondiente a un elemento de memoria transparente

En contraposición a este tipo de elementos podemos encontrar elementos de memoria no transparentes. En este tipo de elementos, los cambios correspondientes a los datos almacenados no obedecen directamente a los cambios de los datos de entrada, sino que solamente se producirán cuando lo indique un señal de control. Así, los elementos mostrados en la figura 2.5(a) pertenecen a este grupo.

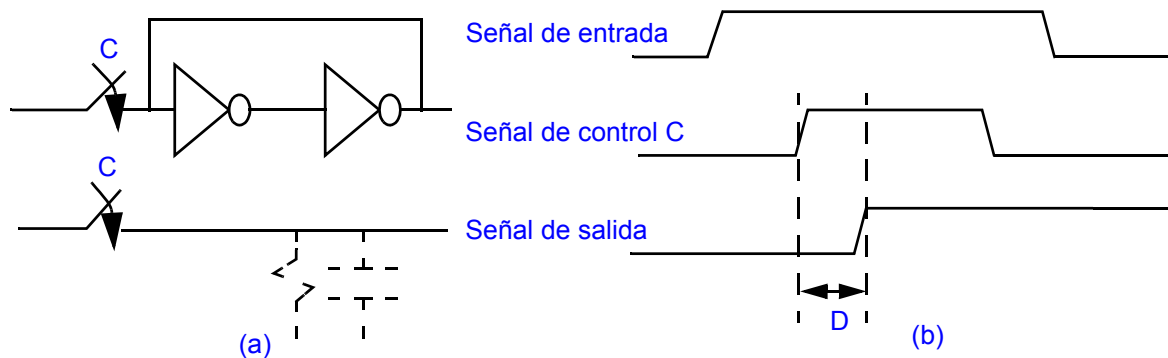


Figura 2.5.- (a) Elementos de memoria controlados o no transparentes y (b) su formas de onda.

Como podemos ver en la figura 2.5(b), la señal de control C controla un conmutador que permite o no el paso del dato de entrada al resto del elemento. Así, mientras C tenga un valor bajo, evitando el paso del dato de entrada, el elemento mantiene almacenado el dato anterior (en el caso de las formas de onda se ha supuesto un nivel bajo). En esta fase de operación se dice que el elemento es opaco o está en su **fase opaca**, de tal forma que se pierde toda influencia respecto a los datos de entrada. En cambio, cuando la señal C permite el paso del dato de entrada, el elemento obedece a este último. En esta fase de operación se dice que el elemento es transparente o está en su **fase transparente**.

En función de esta señal de control o de criterios de transparencia, podemos clasificar a estos elementos de memoria en:

- Elementos sensibles al nivel o **latches**.- Son los elementos en los que la fase de transparencia se corresponde con el intervalo en el que la señal de control tiene su nivel activo.
- Elementos de memoria sensibles a la transición o **flip-flops**.- Son los elementos en los que la fase de transparencia se corresponde con una transición de la señal de control,

por lo que también se dice que carece de esta fase (por ser un instante y no un intervalo) y por tanto de esta propiedad.

Este comportamiento se puede apreciar en la figura 2.6. De ambos tipos de elementos, los flip-flops son los que muestran una mayor independencia con respecto a los datos de entrada, o lo que es lo mismo, una menor ventana de transparencia.

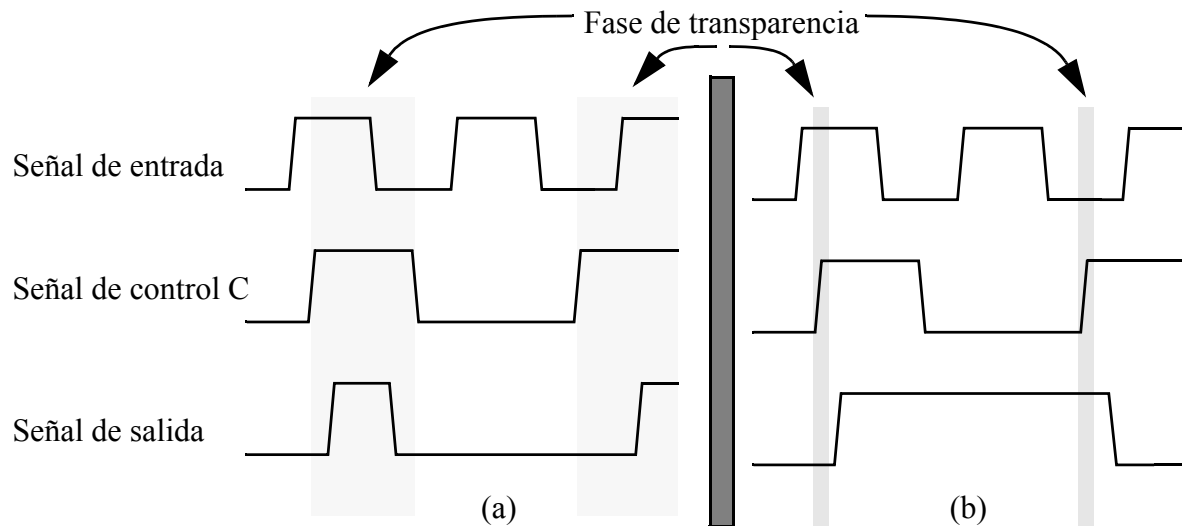


Figura 2.6.- Formas de onda correspondientes a (a) los latches y a (b) los flip-flops.

En la figura 2.7 mostramos la simbología referente a los elementos de memoria.

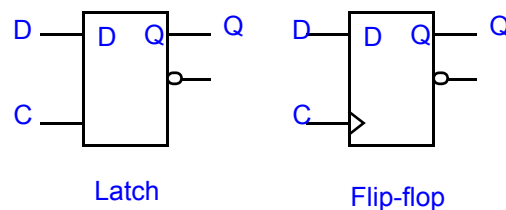


Figura 2.7.- Simbología de los elementos de memoria.

2.3. Diagramas de estado

Al igual que en la lógica combinacional, la forma inicial de representar los sistemas era la tabla de combinaciones; en la lógica secuencial, la forma inicial de representar dichos sistemas es el diagrama y/o tabla de estado. El diagrama será una representación gráfica, mientras que la tabla será tabular (como su propio nombre indica).

En los diagramas de estado podemos encontrar dos elementos: estados y transiciones. Los estados son las letras o símbolos enmarcados (dentro de un círculo generalmente). En cambio, las transiciones son arcos dirigidos que llevan asociadas una/s etiquetas.

Los estados se pueden definir como las posibles situaciones a las que puede llegar el autómatas que estamos describiendo. Dentro de estos estados podemos distinguir entre estados

estables y estados inestables. Cuando existe alguna transición para la cual se llega al mismo estado desde el que se parte, se dice que es un **estado estable**. Mientras que si no existe ninguna transición que cumpla la condición anterior, se dice que es un **estado inestable**.

Las transiciones corresponderán a los eventos en las entradas que producirán los cambios de estado en el sentido de la flecha del arco. El cambio de estado se producirá si la condición de entrada coincide con la etiqueta asociada a dicha transición.

Consideremos el ejemplo de una puerta automática de un garage, como se muestra en la figura 2.8. En dicho sistema se dispone de una puerta y tres sensores: un sensor de posición del coche (S), un sensor para el mando a distancia (M), y un tope para determinar cuando la puerta ha alcanzado el máximo (puerta completamente abierta) o el mínimo (puerta completamente cerrada).

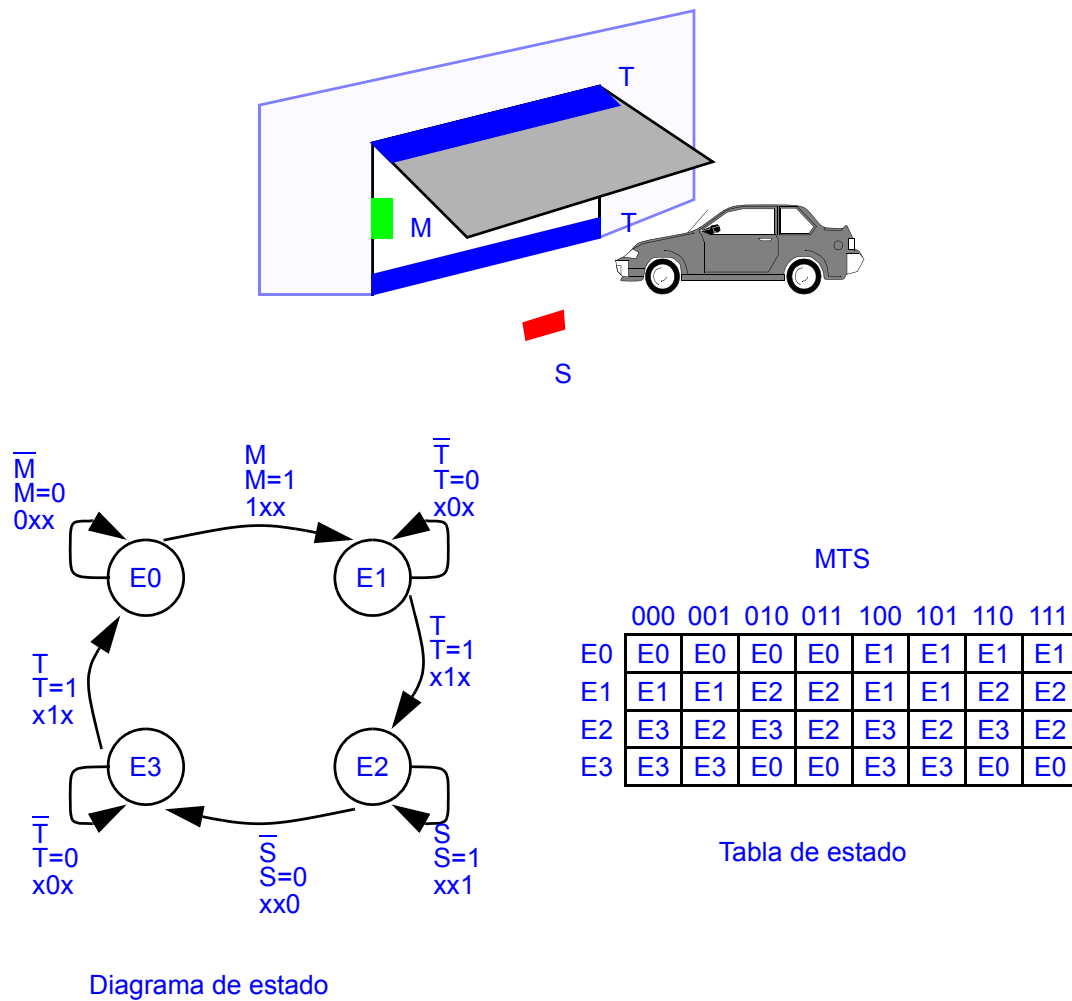


Figura 2.8.- Ejemplo de sistema secuencial.

En este ejemplo podemos determinar cuatro estados:

- un estado en el que la puerta permanece cerrada (E0),
- un estado en el que la puerta está subiendo (E1),

- un estado en el que la puerta está abierta (E2),
- un estado en el que la puerta está bajando (E3).

Es decir, posibles situaciones en las que se pueda encontrar nuestro sistema.

En cambio, las transiciones son los eventos que tienen significado en el sistema y estado. La nomenclatura de estos eventos puede variar según los diferentes autores: podemos encontrarlos referidos a la señal en sí (cuando la señal vale '1' aparecerá tal cual, mientras que si vale '0' aparecerá complementada); o bien indicando explícitamente su valor ($M=0$); o bien dando valores según un orden implícito, el cual ha sido determinado en algún momento). Algunos ejemplos de eventos son:

- No se ha pulsado el mando a distancia ($M = 0$).
- La puerta ha llegado a uno de los topes y el coche está delante de la puerta ($T = 1$ y $S = 1$).

Debido a la existencia de multitud de máquinas equivalentes, no existe ningún procedimiento sistemático que se pueda seguir para realizar esta traducción. Luego la pericia del diseñador será un factor con muy alta influencia en la calidad de la traducción obtenida. No obstante, existen una serie de guías que se pueden seguir. Entre estas guías podemos encontrar las siguientes:

- Pasar de una descripción verbal a una secuencia de entrada/salida.
- Si se conoce alguna máquina válida para nuestro sistema, se suele tomar como máquina de partida.
- Siempre se comienza por un estado conocido; en el caso de que exista un estado inicial, se empezará por él.
- Para cada estado se asigna una transición por cada combinación de entradas, indicando el próximo estado y el valor de las salidas.
- No hay que temer introducir nuevos estados en caso de duda, ya que durante el proceso de reducción se eliminarán todos los estados sobrantes.
- Una vez obtenido el diagrama, se aplica la secuencia de entradas para comprobar que la secuencia de salida es correcta.

2.4. Análisis y síntesis secuencial

Las tareas de análisis y síntesis en lógica secuencial puede definirse de la misma forma que en lógica combinacional.

Los pasos a seguir en la tarea de análisis son los siguientes:

- Obtención de las funciones combinacionales de las señales de próximo estado y de salida. La identificación de las señales de estado (próximo y presente) dependerán de la constitución del circuito; así dichas señales serán:
 - Si el circuito dispone de elementos de memoria, las señales de salida (estado presente) y entrada (próximo estado) de los elementos de memoria.
 - Si el circuito dispone de realimentaciones directas, las señales de las realimenta-

ciones: en las entradas de las puertas (estado presente) y en las salidas de las puertas (próximo estado).

- Derivación de la tabla de transiciones, que será parecido a una tabla de estados.
- Asignación de estados.
- Obtención de la tabla y/o diagrama de estados.

Veamos esta tarea con un par de ejemplos.

En la figura 2.9 mostramos un circuito secuencial con elementos de memoria, el cual se quiere analizar.

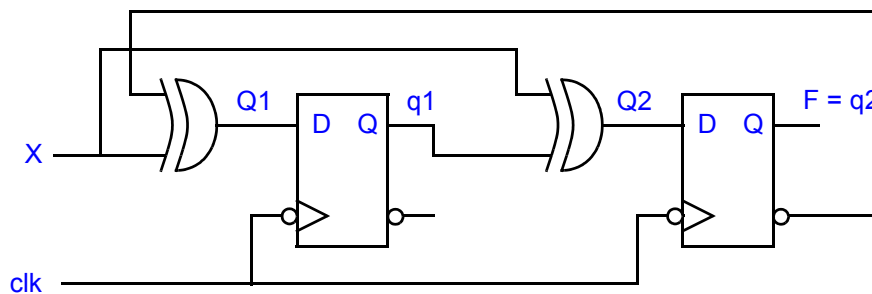


Figura 2.9.- Ejemplo de análisis de un circuito secuencial.

En primer lugar debemos identificar las señales de salida y de próximo estado. Como el circuito está implementado utilizando elementos de memoria, tendremos una señal de estado por cada uno de ellos, es decir, dos señales de estado $Q1$ y $q1$, y $Q2$ y $q2$ (estado próximo y presente respectivamente), y una de ellas, $q2$, coincide con la señal de salida. Luego las funciones serán las siguientes:

- $Q1 = X \oplus q2$
- $Q2 = X \oplus q1$
- $F = q2$

Una vez que tenemos dichas funciones, generaremos la tabla de transición, que será una tabla de estado, pero en lugar de estados aparecerán las combinaciones de las señales de estados. Dicha tabla es mostrada en la figura 2.10.

El siguiente paso consiste en dar a cada combinación de las señales de estados un nombre (que se corresponderá con el estado), y sustituirlo en la tabla de transición, obteniendo de ese modo la tabla de estado.

Si consideramos el circuito de la figura 2.11, el proceso de análisis tendría la siguiente solución.

En primer lugar debemos identificar las señales de salida y de próximo estado. Como el circuito está implementado utilizando realimentación directa, tendremos una señal de estado por cada uno de dichas realimentaciones, es decir, una única señal de estado $Q1$ y $q1$ (estado próximo y presente respectivamente), que coincide con la señal de salida. Luego las funciones serán las siguientes:

- $Q1 = x \cdot y + (\bar{y} + z) \cdot q1$
- $F = x \cdot y + (\bar{y} + z) \cdot q1$

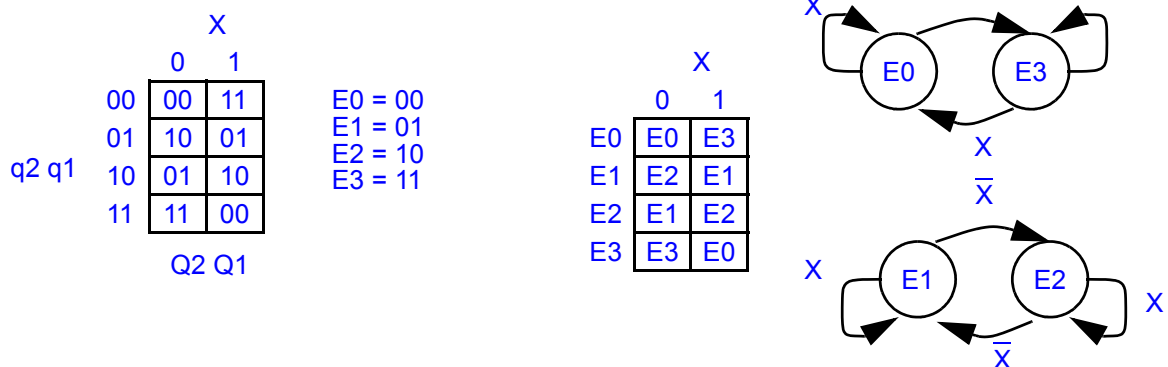


Figura 2.10.- Tablas de transición y de estados correspondiente al ejemplo de la figura 2.9

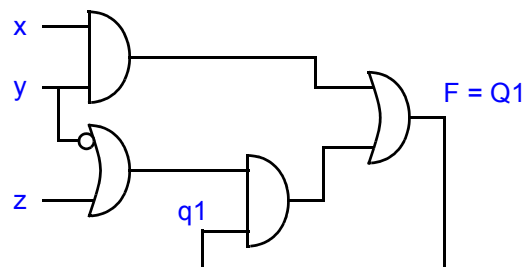


Figura 2.11.- Ejemplo de análisis de un circuito secuencial.

Una vez que tenemos dichas funciones, generaremos la tabla de transición, que será una tabla de estado, pero en lugar de estados aparecerán las combinaciones de las señales de estados. Dicha tabla es mostrada en la figura 2.12.

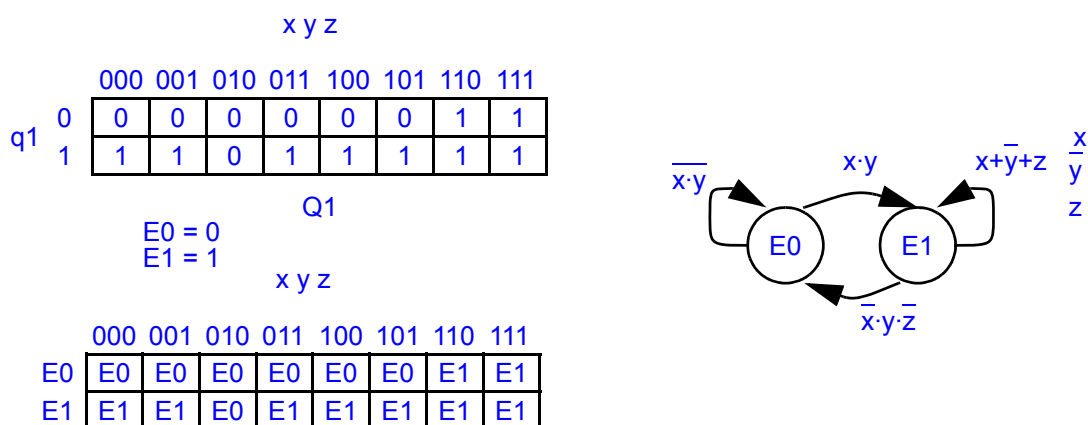


Figura 2.12.- Tablas de transición y de estados correspondiente al ejemplo de la figura 2.11

La tarea de síntesis es el proceso contrario, para lo cual debemos seguir los pasos anteriores de forma inversa:

- Obtención del diagrama y/o tabla de estados
- Asignamiento de estados. Para lo cual vamos a utilizar una codificación *one-hot*, que consiste en disponer de una señal de estado por cada uno de los estados.
- Determinación de las funciones combinaciones de las señales de próximo estado y de salida.
- Implementación de las funciones combinacionales anteriores con puertas e inserción de elementos de memoria donde corresponda.

Veamos esta tarea con un par de ejemplos.

Se desea obtener el circuito secuencial que gobierna la puerta automática de garage mostrada en la figura 2.8

Como ya se ha comentado, obtener el diagrama de estados no es una tarea metódica sino que depende de la pericia del diseñador. El diagrama de estados ya fue obtenido y se mostró en la figura 2.8.

Debido a la codificación *one-hot*, vamos a tener una señal de estado por cada estado, por lo que las funciones podemos extraerlas directamente del diagrama. La señal de próximo estado vendrá determinada por todos los arcos que lleguen al estado en cuestión; mientras que la señal de estado presente vendrá determinada por los arcos que salgan del estado, al cual habrá que adjuntar la condición del arco. Así, en nuestro ejemplo, las funciones serán las siguientes:

- $E0 = e0 \cdot \overline{M} + e3 \cdot T$
- $E1 = e0 \cdot M + e1 \cdot \overline{T}$
- $E2 = e1 \cdot T + e2 \cdot S$
- $E3 = e2 \cdot \overline{S} + e3 \cdot \overline{T}$

Podemos comprobar que solamente una de las señales de próximo estado anteriores tomará el valor '1' mientras que el resto serán iguales a '0'.

Una vez que tenemos las fórmulas, las pasamos directamente a circuitos teniendo en cuenta que la señal de próximo estado será la entrada del elemento de memoria, y la señal de estado presente será la salida del elemento de memoria correspondiente, según se muestra en la figura 2.13.

Veamos otro ejemplo. Se dispone de una máquina trituradora, en la cual hay dos sensores de presencia y dos motores de trituración como se muestra en la FIGURA. Se pretende diseñar un circuito secuencial que controle los motores de tal forma que:

- Cuando no haya nada que triturar, los dos motores estén parados.
- Cuando el tanque esté lleno, los dos motores deben funcionar simultáneamente.
- Cuando el tanque esté medio lleno, sólo funcionará uno de los motores, concretamente el que estaba parado en la última vez que se produjo esta misma situación.

Lo primero que tenemos que hacer es generar el diagrama de estado. Para ello, en primer lugar debemos identificar las posibles situaciones (estados) en los que la máquina se puede encontrar:

- Esté funcionando únicamente el motor 0 (E01), por lo tanto $M0=1$ y $M1=0$.
- Esté funcionando únicamente el motor 1 (E11), por lo tanto $M0=0$ y $M1=1$.

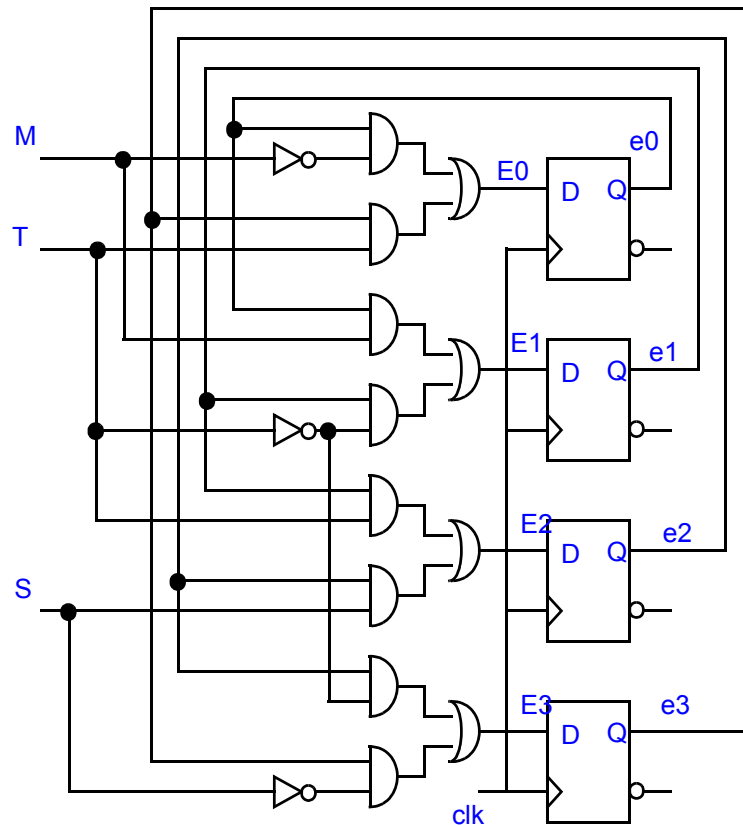


Figura 2.13.- Circuito correspondiente al diagrama de estados de la figura 2.8.

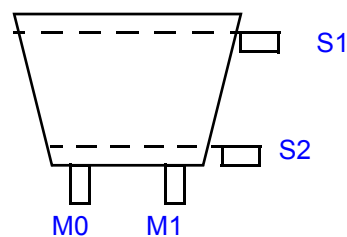


Figura 2.14.- Esquema de la máquina trituradora.

- Se hayan parado los motores después de estar funcionando el motor 0 (E00), por lo tanto $M0=0$ y $M1=0$.
- Se hayan parado los motores después de estar funcionando el motor 1(E10), por lo tanto $M0=0$ y $M1=0$.
- Estén funcionando los dos motores después de funcionar sólo el 0 (E02), por lo tanto $M0=1$ y $M1=1$.
- Estén funcionando los dos motores después de funcionar sólo el 1(E12), por lo tanto $M0=1$ y $M1=1$.

El desdoble de los estados de que estén funcionando o parados los dos motores es necesario para recordar cuál de ellos era el que había estado parado la última vez.

Una vez que tenemos los estados, tenemos que poner las transiciones, para lo cual

tenemos que identificar los eventos que tengan sentido en cada uno de los estados. Por ejemplo,

- si estamos en el estado E00 y el tanque está por la mitad ($\overline{S1} \cdot S2$), debemos ir al estado E11 (en el que sólo funciona el motor que estaba previamente ocioso);
- si estamos en el estado E11 y el tanque se vacía ($S1 \cdot \overline{S2}$), debemos ir al estado E10 (en el que no funciona ningún motor, pero se indica que el motor ocioso era el 0).

Luego, el diagrama de estados correspondiente se muestra en la figura 2.15.

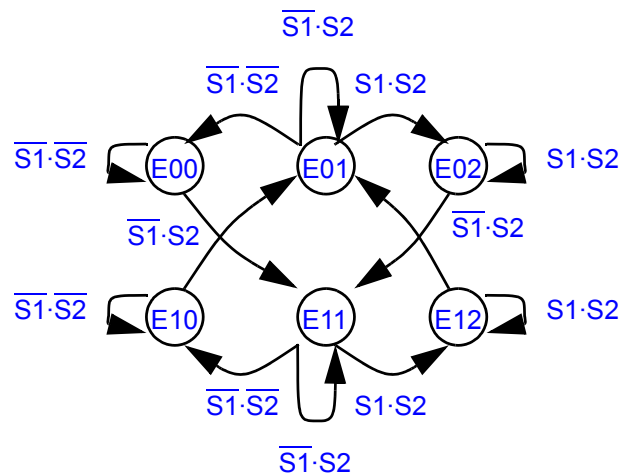


Figura 2.15.- Diagrama de estados correspondiente al controlador de la máquina trituradora.

El siguiente paso consiste en determinar las ecuaciones del sistema, tanto las de próximo estado como las de salida. Dichas ecuaciones serán las siguientes:

- $E00 = e00 \cdot \overline{S1} \cdot \overline{S2} + e01 \cdot \overline{S1} \cdot \overline{S2} = (e00 + e01) \cdot \overline{S1} \cdot \overline{S2}$
- $E01 = e10 \cdot \overline{S1} \cdot S2 + e12 \cdot \overline{S1} \cdot S2 + e01 \cdot \overline{S1} \cdot S2 = (e10 + e12 + e01) \cdot \overline{S1} \cdot S2$
- $E02 = e01 \cdot S1 \cdot S2 + e02 \cdot S1 \cdot S2 = (e01 + e02) \cdot S1 \cdot S2$
- $E10 = e10 \cdot \overline{S1} \cdot \overline{S2} + e11 \cdot \overline{S1} \cdot \overline{S2} = (e10 + e11) \cdot \overline{S1} \cdot \overline{S2}$
- $E11 = e00 \cdot \overline{S1} \cdot S2 + e02 \cdot \overline{S1} \cdot S2 + e11 \cdot \overline{S1} \cdot S2 = (e00 + e02 + e11) \cdot \overline{S1} \cdot S2$
- $E12 = e11 \cdot S1 \cdot S2 + e12 \cdot S1 \cdot S2 = (e11 + e12) \cdot S1 \cdot S2$
- $M0 = e02 + e12 + e01$
- $M1 = e02 + e12 + e11$

Luego, el esquema del circuito será el mostrado en la figura 2.16.

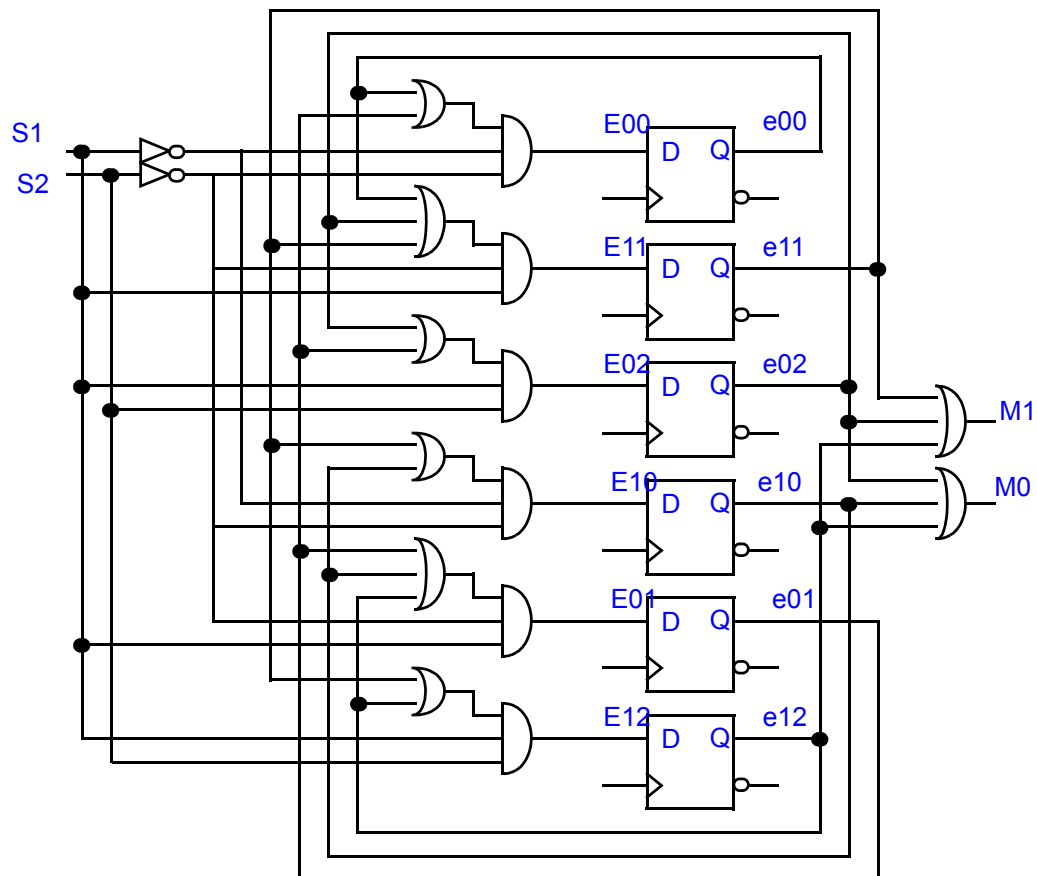


Figura 2.16.- Circuito de control correspondiente a la máquina trituradora.