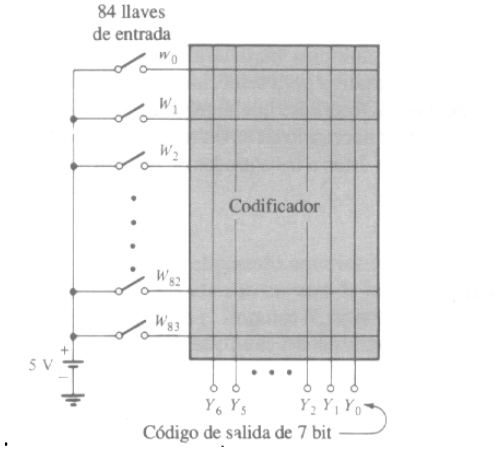
Clase 9

Codificador

Un decodificador es un sistema que acepta una información de **M** bit y establece el estado 1 en una (y sólo una) de 2^{M} líneas de salida. En otras palabras, un decodificador identifica un código particular. Al proceso inverso se lo denomina codificador. Un codificador tiene un número de entradas, de las que sólo una está en estado 1 y se forma un código de N bit dependiente de cual de las entradas sea la excitada.

Supongamos que se quiera transmitir un código binario con cada pulsación de una tecla (de una máquina de escribir o un teletipo). Necesitaremos tantos bit como los que sea necesarios para transmitir las 26 letras minúsculas del teclado americano (las 27 del español sin la ñ ni las letras acentuadas por supuesto, así como la diéresis) otras 26 para las mayúsculas, los 22 caracteres especiales ¡ ” # $ % ’ & * ( ) - _ = + : ; < , > . ? / y los diez símbolos de los números decimales (0 a 9) tenemos un total de 84 códigos para lo que se necesitarán al menos 7 bit (con seis no alcanza pues 2^6=64 < 84 < 2^7=128). Modificamos un teclado clásico (de máquina de escribir vieja) de manera tal que al presionar una tecla se cierra un interruptor que le entrega una tensión de 5V a una línea de entrada un determinado codificador como se muestra en la figura siguiente:



En el interior de la parte sombreada hay una serie de conductores cruzados que se conectarán de una manera determinada para formar los códigos deseados. A manera de muestra buscaremos la codificación para los números decimales para los que necesitaremos sólo cuatro bit en vez de 7 y elegiremos el sistema BCD ya visto en clases anteriores cuya tabla de verdad es (como ya vimos):

Entradas										Salidas			
W_9	W_8	W_7	W_6	W_5	W_4	W_3	W_2	W_1	W_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

Las entradas W_i (i= 0,1,2,3,....,9) representan las teclas, cuando la tecla número 6 está apretada la representamos diciendo que W_6=1, En principio suponemos que no se activa mas de una tecla por vez; de la tabla se deduce que para que Y_0 = 1 hace falta que W_1=1 si no, W_3=1 o que W_5=1, o W_7=1 o W_9=1 de donde se obtiene en notación de Boole que:

$$Y_0 = W_1 + W_3 + W_5 + W_7 + W_9$$

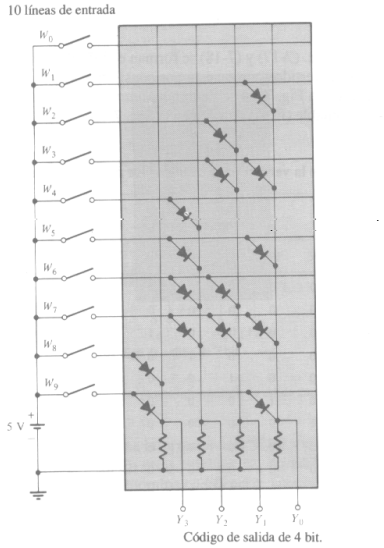
y de forma análoga obtenemos

$$Y_1 = W_2 + W_3 + W_5 + W_7$$

$$Y_2 = W_4 + W_5 + W_5 + W_7$$

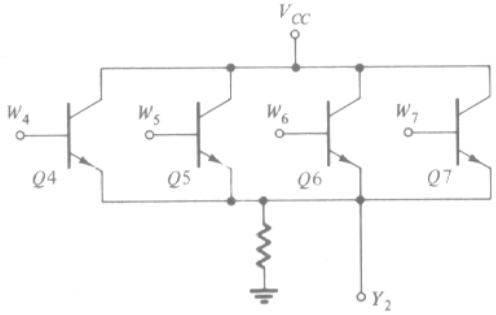
$$Y_3 = W_8 + W_9$$

Para armar con puertas OR estas ecuaciones sólo necesitamos diodos, como se ve en la figura siguiente



Matriz codificadora BCD hecha con diodos.

Cada diodo de la figura anterior podría reemplazarse por el diodo

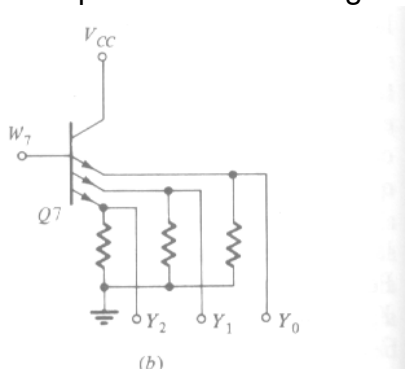


base-emisor de un transistor con la ventaja que la llave interruptora (que estaba ligada al teclado) maneja ahora la base del transistor (y por lo tanto disminuye el consumo de potencia). Si al colector se le suministra una tensión de alimentación V_{CC}, queda una puerta OR hecha con transistores (en modo seguidor por emisor) como se ve en la figura que sigue

Puerta OR hecha con transistores para la ecuación: $Y_1 = W_2 + W_3 + W_5 + W_7$

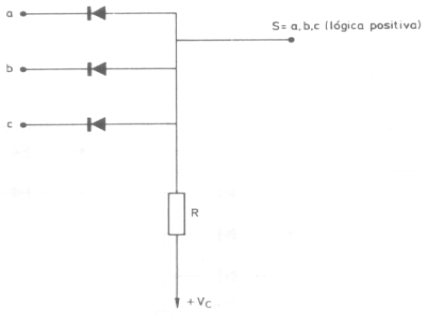
El esquema anterior necesita en total para las diez llaves interruptoras 15 transistores, como se puede ver pues, para la ecuación de Boole que representa a Y_1 necesitamos cuatro transistores, así como para la ecuación Y_2, para la ecuación que representa a Y_3 necesitamos otros dos y para Y_0 necesitamos cinco, todos ellos con el colector conectado a la tensión de alimentación V_{CC} (mostrar en ejemplo los 15 transistores).

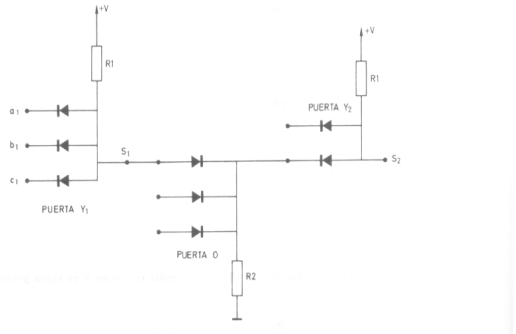
Pero dado que algunos de estos tendrían la base conectada a la misma tensión por ejemplo cuando se activa W_7 estaríamos activando las bases de tres transistores (estos tres transistores ya tenían el colector conectado entre sí a la tensión de alimentación), en tecnologías anteriores hemos visto que estos se reemplazan por transistores de emisor múltiple con lo que reducimos todo a nueve transistores de emisor múltiple como el de la figura que sigue:

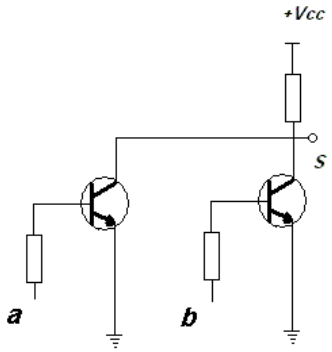


En realidad el transistor que tiene la base conectada a W_1 sólo necesita un emisor, así como cada uno de los transistores cuyas bases están conectadas respectivamente a las entradas W_2, W_4 y W_8; los que están conectados a las entradas W_3 W_5 W_6 y W_9 necesitan al menos de dos emisores en tanto que el transistor conectado a la entrada W_7 tendrá un emisor múltiple de tres salidas. Lógicamente la entrada W_0 no activa ningún transistor. (Mostrar como queda al final).

Implementación de compuertas

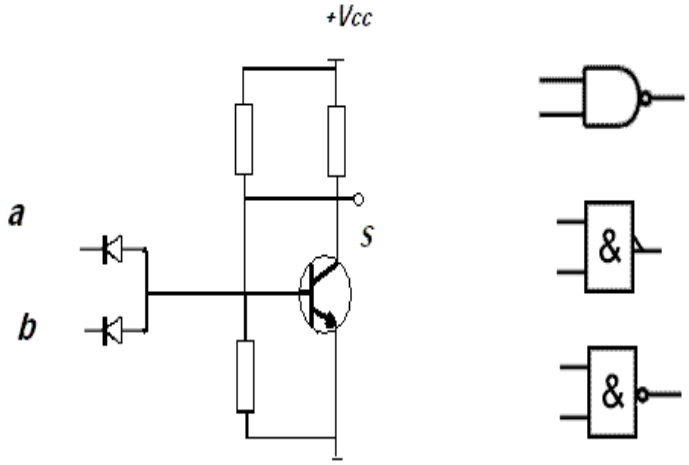


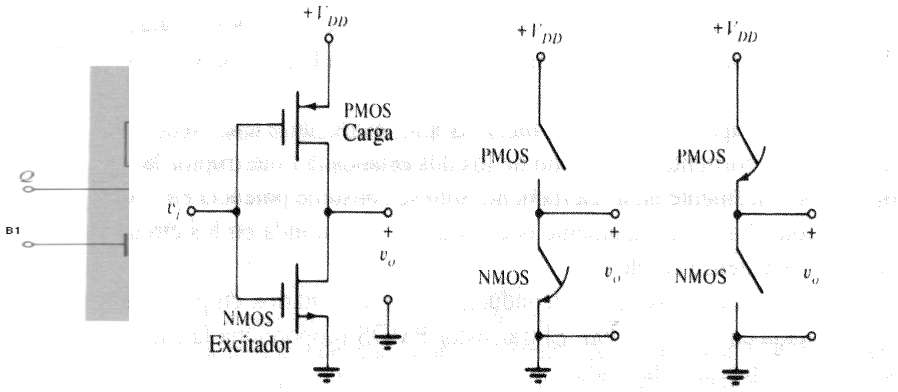


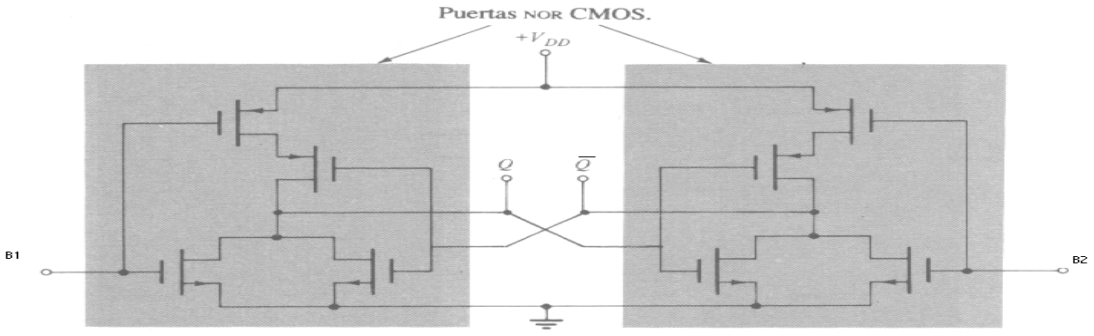


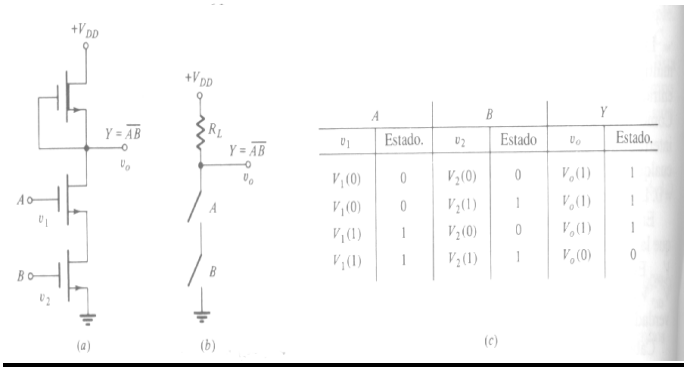
a	B	s	š = (a + b)
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	1

Figura 2.2 esquema del circuito de una puerta NOR hecho con dos transistores. Tabla de verdad de dicho circuito



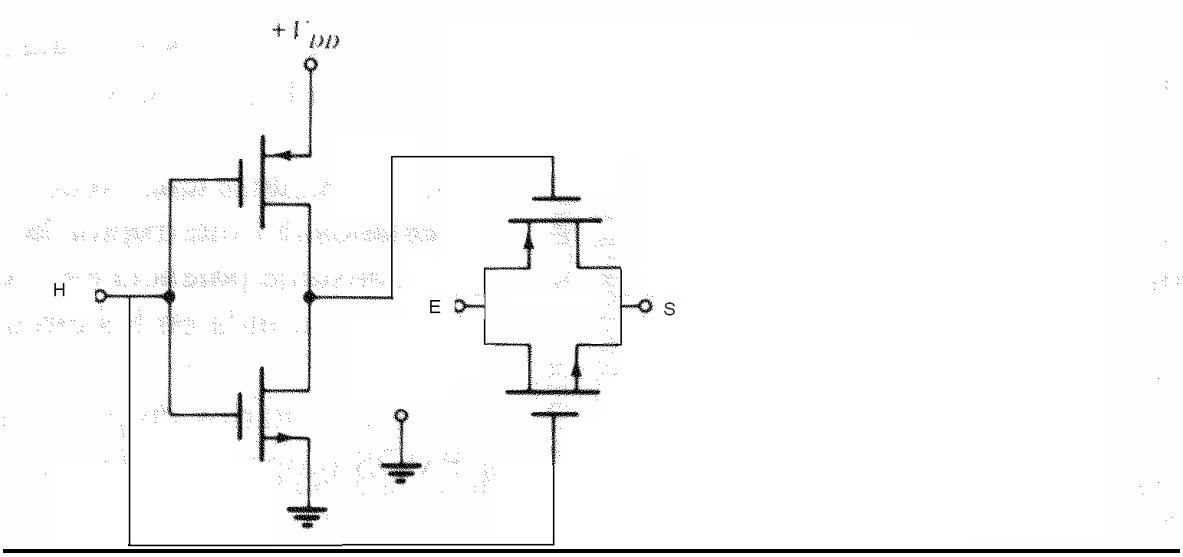






A		B		Y	
v_1	Estado.	v_2	Estado	v_o	Estado.
$V_1(0)$	0	$V_2(0)$	0	$V_o(1)$	1
$V_1(0)$	0	$V_2(1)$	1	$V_o(1)$	1
$V_1(1)$	1	$V_2(0)$	0	$V_o(1)$	1
$V_1(1)$	1	$V_2(1)$	1	$V_o(0)$	0

(c)



Codificador con prioridad

En el análisis del codificador anterior se presupuso que no existía la posibilidad que se apretaran dos teclas al mismo tiempo, esto no es lo que ocurre en la realidad, vamos entonces a analizar un codificador en el que ante la posibilidad que se presionen dos teclas simultáneamente por lo menos establezca una prioridad, para esto tendremos que modificar la tabla de verdad correspondiente; en la tabla siguiente se ha representado esta situación asignando el valor x para el caso en que no importe el estado de esa variable, de tal modo que si se aprietan simultáneamente dos teclas (x ej 5 y 6) o 4 y 1 como puede ocurrir dependiendo el teclado que uno use, se activará sólo el mayor de cada par de esos números (el 6 o el 4 según el par que se trate).

Entradas										Salidas			
W_9	W_8	W_7	W_6	W_5	W_4	W_3	W_2	W_1	W_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	x	0	0	0	1
0	0	0	0	0	0	0	1	x	x	0	0	1	0
0	0	0	0	0	0	1	x	x	x	0	0	1	1
0	0	0	0	0	1	x	x	x	x	0	1	0	0
0	0	0	0	1	x	x	x	x	x	0	1	0	1
0	0	0	1	x	x	x	x	x	x	0	1	1	0
0	0	1	x	x	x	x	x	x	x	0	1	1	1
0	1	x	x	x	x	x	x	x	x	1	0	0	0
1	x	x	x	x	x	x	x	x	x	1	0	0	1

La expresión de Boole para cada una de las salidas será

$$Y_0 = \sim W_9 \sim W_8 \sim W_7 \sim W_6 \sim W_5 \sim W_4 \sim W_3 \sim W_2 W_1$$
$$+ \sim W_9 \sim W_8 \sim W_7 \sim W_6 \sim W_5 \sim W_4 W_3$$
$$+ \sim W_9 \sim W_8 \sim W_7 \sim W_6 W_5 + \sim W_9 \sim W_8 W_7 + W_9$$

$$Y_1 = \sim W_9 \sim W_8 \sim W_7 \sim W_6 \sim W_5 \sim W_4 \sim W_3 W_2$$

$$+\sim W_9\ \sim W_8\ \sim W_7\ \sim W_6\ \sim W_5\ \sim W_4\ W_3 + \sim W_9\ \sim W_8\ \sim W_7\ W_6 + \sim W_9\ \sim W_8\ W_7$$

$$Y_2 = \sim W_9\ \sim W_8\ \sim W_7\ \sim W_6\ \sim W_5\ W_4 + \sim W_9\ \sim W_8\ \sim W_7\ \sim W_6\ W_5 + \sim W_9\ \sim W_8\ \sim W_7\ W_6 + \sim W_9\ \sim W_8\ W_7$$

$$Y_3 = \sim W_9\ W_8 + W_9$$

A modo de ejemplo simplificaremos la ecuación para

$$Y_1 = \sim W_9\ \sim W_8\ (\sim W_7\ B + W_7)$$

en la que

$$B = \sim W_6\ \sim W_5\ \sim W_4\ \sim W_3\ W_2 + \sim W_6\ \sim W_5\ \sim W_4\ W_3 + W_6$$

Aprovechando la relación A+~A*B = (A+B)

$$A+\sim A*B = (A+\sim A)*(A+B)$$

por el III-2 postulado del álgebra de Boole

$$(A+\sim A)*(A+B) = 1*(A+B)$$

$$Y_1 = \sim W_9\ \sim W_8\ (B + W_7)$$

De la definición de B y aprovechando la misma relación obtenemos que

$$B = \sim W_6\ C + W_6 = W_6 + C \qquad \text{con}$$

$$C = \sim W_5\ \sim W_4\ \sim W_3\ W_2 + \sim W_5\ \sim W_4\ W_3 = \sim W_5\ \sim W_4\ (\sim W_3\ W_2 + W_3)$$
$$= \sim W_5\ \sim W_4\ (W_2 + W_3)$$

Resumiendo

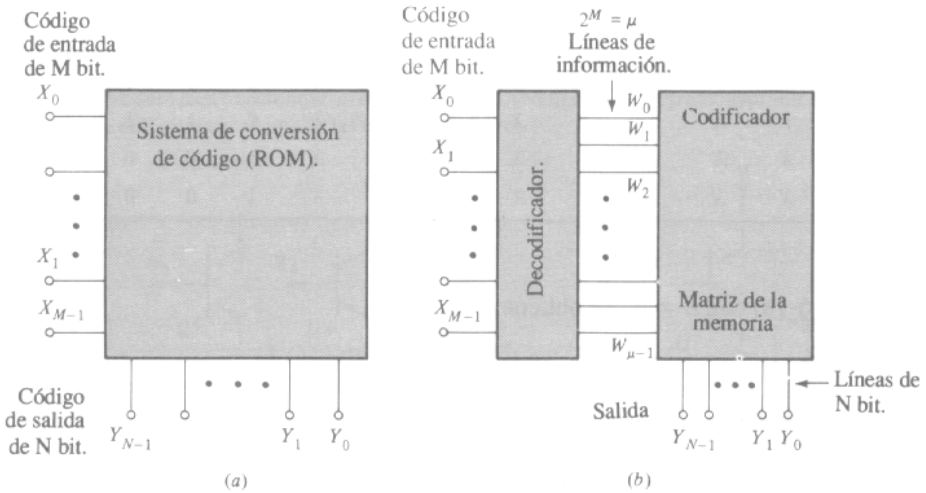
$$Y_1 = \sim W_9\ \sim W_8\ (W_7 + W_6 + \sim W_5\ \sim W_4\ W_3 + \sim W_5\ \sim W_4\ W_2)$$

Para generar $\sim W_9\ \sim W_8 = \sim(W_9 + W_8)$ se emplea una puerta NOR, y para generar Y_1 una puerta AND-OR de 2 -2 - 4 - 4 entradas. Procediendo de forma análoga hallamos los valores para Y_0 Y_2 e Y_3

Esta lógica se fabrica en un chip integrado a media escala (el 74LS147) y codifica con la prioridad mencionada 10 líneas decimal en 4 líneas BCD. Entre sus aplicaciones están la de codificación de teclados pequeños.

Memoria de sólo lectura

Consideremos el problema de convertir un código binario a otro. Partimos entonces de tomar M entradas (X_0, X_1, X_2,....,X_{M-1}), con un sistema de conversión como se muestra en la figura y que me genera N salidas (Y_0, Y_1, Y_2, Y_3,....,Y_{N-1}) pudiendo se N mayor menor o igual a M. Un sistema tal se llama ROM y está representado en la figura siguiente. En el proceso las M entradas (X_0, X_1, X_2,....,X_{M-1}) (código binario de M bit) me producen $2^M = \mu$ líneas de información distintas (W_0, W_1,



W_2, W_3,....,$W_{\mu-1}$) cada línea de información se deberá codificar en otro de N bit produciendo una salida (Y_0, Y_1, Y_2,....,Y_{N-1}) con N mayor, menor o igual que M (es posible que mas de una entrada conduzca al mismo código de salida).

La relación funcional entre las informaciones de salida y de entrada se realizan en el bloque codificador de la figura. Sea cual fuere la forma en que dicha información quede almacenada permanentemente se dice entonces que el sistema tiene memoria no disipable (no volátil) . Los elementos de la memoria son los diodos de la figura anterior o los de los transistores multiemisores ya mencionados. La información de salida se puede obtener cuantas veces se quiera con tan solo colocar los código de entrada y leer la de salida. No obstante, ya que la relación almacenada entre los códigos de salida y de entrada no se puede modificar sin añadir o eliminar elementos memorizadores, a este sistema se lo llama memoria de sólo lectura (Read Only Memory ROM)

Convertidores de código

A manera de ejemplo veamos la transformación de código binario a código Gray. Al pasar de una línea a la siguiente del Gray, sólo se puede cambiar un bit, y sólo uno, de 0 a 1 o viceversa. (Esta propiedad no define unívocamente un código y por lo tanto se pueden formar varios códigos Gray). Los bit de entrada de la tabla siguiente se codifican en un ROM formando las líneas de información W_0, W_1, W_2, W_3,...., W_{15} codificando luego al código Gray deseado Y_3, Y_2, Y_1, Y_0

Entrada	Inf decod.	Salida codigo Gray

X_3	X_2	X_1	X_0	W_n	Y_3	Y_2	Y_1	Y_0
0	0	0	0	W_0	0	0	0	0
0	0	0	1	W_1	0	0	0	1
0	0	1	0	W_2	0	0	1	1
0	0	1	1	W_3	0	0	1	0
0	1	0	0	W_4	0	1	1	0
0	1	0	1	W_5	0	1	1	1
0	1	1	0	W_6	0	1	0	1
0	1	1	1	W_7	0	1	0	0
1	0	0	0	W_8	1	1	0	0
1	0	0	1	W_9	1	1	0	1
1	0	1	0	W_{10}	1	1	1	1
1	0	1	1	W_{11}	1	1	1	0
1	1	0	0	W_{12}	1	0	1	0
1	1	0	1	W_{13}	1	0	1	1
1	1	1	0	W_{14}	1	0	0	1
1	1	1	1	W_{15}	1	0	0	0

Las W son las salidas del decodificador, por ejemplo

$W_0 = \sim X_3 \sim X_2 \sim X_1 \sim X_0 \quad W_5 = \sim X_3 X_2 \sim X_1 X_0$

Se observa que la salida

$Y_0 = W_1 + W_2 + W_5 + W_6 + W_9 + W_{10} + W_{13} + W_{14}$

Esta ecuación se cumple conectando ocho diodos con sus cátodos unidos todos a Y_0 y sus ánodos conectados a las líneas $W_1\ W_2\ W_5\ W_6\ W_9\ W_{10}\ W_{13}$ y W_{14} del decodificador. (También se pueden usar los diodos base-emisor de los transistores como se indicó en el codificador del teclado)

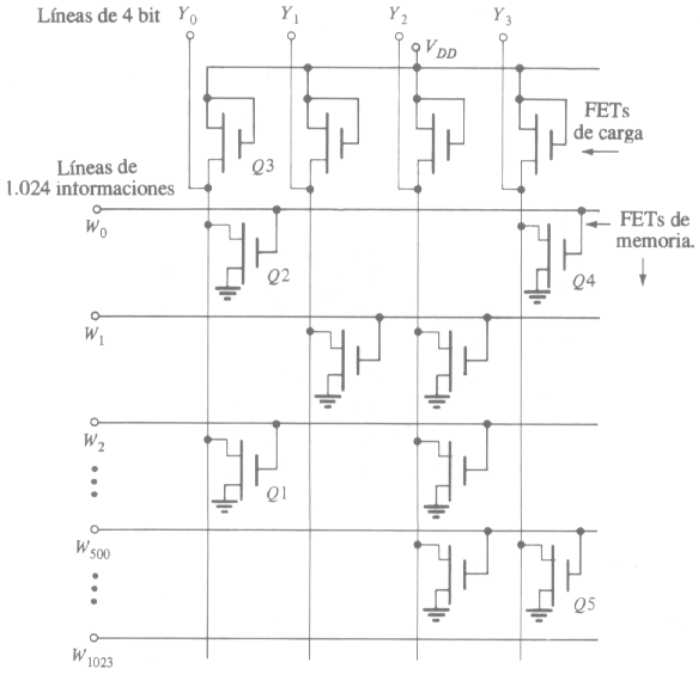
Programación del ROM

Consideremos la memoria de sólo lectura (ROM) bipolar de 256 bit dispuesta en 32 informaciones de 8 bit cada una. La entrada del decodificador es un código binario de 5 bit, y sus salidas son las 32 líneas de información. El codificador está formado por 32 transisores (estando cada una de las bases conectada a una línea distinta) y con 8 emisores cada transistor. El cliente rellena la tabla de verdad que desea que satisfaga su ROM y entonces el constructor prepara una máscara para la metalización de forma que quede conectado un emisor de cada transistor a la línea de salida apropiada o en su caso dejarlo flotante. Por ejemplo en la conversión binario Gray anterior los transistores conectados a las puertas $W_1\ W_2\ W_5\ W_6\ W_9\ W_{10}\ W_{13}$ y W_{14} tendrán un emisor de cada uno de ellos conectado a la salida Y_0

ROMS NMOS

Las memorias de sólo lectura se forman corrientemente con la tecnología NMOS, frecuentemente como chips de integración a gran escala. Frecuentemente los ROM se fabrican como partes de un sistema más complejo en un solo chip tal como el microprocesador. Consideremos por ejemplo un código de entrada de 10 bit, resultando 2^{10}=1024 líneas de información y con 4 bit para el código de salida. La matriz de la memoria para este sistema consta de 1024 x4= 4096 intersecciones como se indica esquemáticamente en la figura siguiente.

Este es un ROM de 4-Kilobit (4-kb) organizado como 1kb x 4. Esta

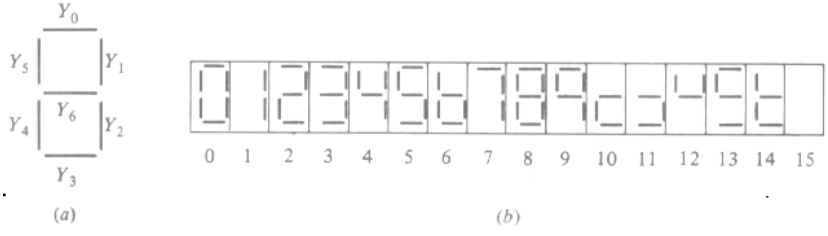


designación proviene del hecho que 2^{10} es aproximadamente 10^3. Así el ROM de 64 kb tiene 2^6 x 2^{10} = 64 x 2^{10} bit. La conversión de código a desarrollar por la ROM se programa permanentemente durante el proceso de fabricación utilizando una máscara diseñada para incluir u omitir un transistor MOS en cada intersección de la matriz. La figura anterior representa un codificador de este tipo en el que se puede ver cómo se conectan los FET de memoria entre las líneas de información y de bit.

Aplicación de los ROM Imagen visible de siete segmentos

Como se ha visto el ROM es una unidad de conversión de código. Sin embargo muchos sistemas prácticos distintos representan una traducción de uno a otro código como ejemplo veamos el indicador luminoso de siete segmentos.

Es muy común hacer visible la lectura de un aparato digital por medio de un indicador luminoso de siete segmentos como el de la figura. El indicador de estado sólido cuyos segmentos reciben su luminosidad de unos diodos emisores de luz de arseniuro de galio. Trabajan a baja tensión y poca potencia y por lo tanto



pueden ser excitados directamente por puertas lógicas integradas.

Entrada				Inf decod.	Salida codigo del indicador de siete segmentos						
X_3	X_2	X_1	X_0	W_n	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0	0	W_0	1	0	0	0	0	0	0
0	0	0	1	W_1	1	1	1	1	0	0	1
0	0	1	0	W_2	0	1	0	0	1	0	0
0	0	1	1	W_3	0	1	1	0	0	0	0
0	1	0	0	W_4	0	0	1	1	0	0	1
0	1	0	1	W_5	0	0	1	0	0	1	0
0	1	1	0	W_6	0	0	0	0	0	1	1
0	1	1	1	W_7	1	1	1	1	0	0	0
1	0	0	0	W_8	0	0	0	0	0	0	0
1	0	0	1	W_9	0	0	1	1	0	0	0
1	0	1	0	W_{10}	0	1	0	0	1	1	1
1	0	1	1	W_{11}	0	1	1	0	0	1	1
1	1	0	0	W_{12}	0	0	1	1	1	0	1
1	1	0	1	W_{13}	0	0	1	0	1	1	0
1	1	1	0	W_{14}	0	0	0	0	1	1	1
1	1	1	1	W_{15}	1	1	1	1	1	1	1

Las diez primeras imágenes de la figura son las cifras del 0 al 9 que en el instrumento digital están representadas en forma BCD. Un código de 4 bit tiene 16 estados posibles y las imágenes del 10 al 15 son los únicos símbolos utilizados para identificar una condición de BCD no válida.

El problema de pasar de la entrada BCD a las salidas de siete segmentos se resuelve fácilmente empleando un ROM. Si un segmento excitado (luminoso) se identifica como estado 0 y uno apagado como estado 1 se obtiene la tabla de verdad ya mostrada.