TEMA: **CIRCUITOS COMBINACIONALES.**

INSTITUTO TECNOLÓGICO SUPERIOR DE SAN ANDRÉS TUXTLA

División Ingeniería Mecatrónica IMCT-2010-229

Periodo: Febrero – Junio 2025 Grupo: 611A

INVESTIGACION UNIDAD 3

ELECTRONICA DIGITAL

Docente:

DR JOSE ANGEL NIEVES VAZQUEZ

Unidad 3:

CIRCUITOS COMBINACIONALES

Presenta:

Juan José Jiménez Reyes	**221U0541**
Juan José Marcial Fiscal	**221U0547**
Miguel de Jesús Polito Cerón	**221U0552**
Perla Joselin Quino Caixba	**221U0555**
Rocio Teoba Herrera	**221U0562**

San Andrés Tuxtla Veracruz **07 de abril de 2025**

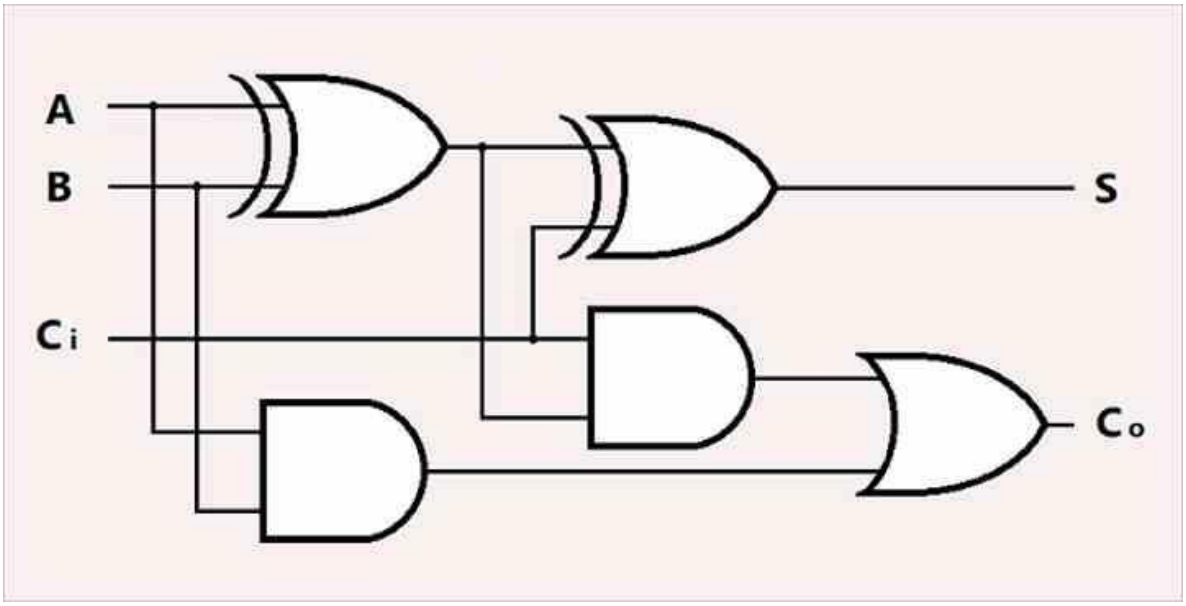
ÍNDICE

INTRODUCCIÓN

En este documento abordamos de manera integral el estudio de los circuitos combinacionales, elementos fundamentales en el campo de la electrónica digital. Explicamos la metodología de diseño, desde la definición del problema y la construcción de la tabla de verdad, hasta la implementación y simulación de los circuitos. A través del análisis de componentes esenciales como sumadores, multiplexores, decodificadores y codificadores, demostramos cómo estas estructuras nos permiten transformar datos en tiempo real, facilitando la realización de operaciones aritméticas, el enrutamiento de señales y la conversión de códigos. Asimismo, resaltamos la relevancia de los dispositivos lógicos programables (PLD) y de los lenguajes de descripción de hardware (HDL) como herramientas indispensables para validar y sintetizar diseños en plataformas como FPGA o ASIC, lo que contribuye a la eficiencia y optimización en el desarrollo de sistemas digitales.

3. CIRCUITOS COMBINACIONALES

LOS CIRCUITOS COMBINACIONALES SON SISTEMAS LÓGICOS FUNDAMENTALES EN ELECTRÓNICA DIGITAL, DONDE LAS SALIDAS DEPENDEN EXCLUSIVAMENTE DE LAS COMBINACIONES ACTUALES DE LAS ENTRADAS, SIN INFLUENCIA DE ESTADOS PREVIOS. A DIFERENCIA DE LOS CIRCUITOS SECUENCIALES (QUE INCLUYEN MEMORIA), ESTOS CARECEN DE CAPACIDAD PARA ALMACENAR INFORMACIÓN, LO QUE LOS HACE IDEALES PARA TRANSFORMAR DATOS EN TIEMPO REAL MEDIANTE OPERACIONES LÓGICAS.



CARACTERÍSTICAS CLAVE:

- Relación entrada-salida directa: Cada combinación de *n* entradas genera una salida única, definida por su tabla de verdad (2^n posibles combinaciones).
- Representación booleana: Se modelan mediante funciones lógicas usando operadores básicos (AND, OR, NOT) o compuestos (XOR, NAND, NOR). Estas pueden expresarse en formas canónicas, como *suma de minitérminos* o *producto de maxitérminos*, facilitando su optimización.
- Retardo predecible: El tiempo de respuesta depende del retardo acumulado de las puertas lógicas en la ruta crítica, crucial en sistemas de alta velocidad.

APLICACIONES PRINCIPALES:

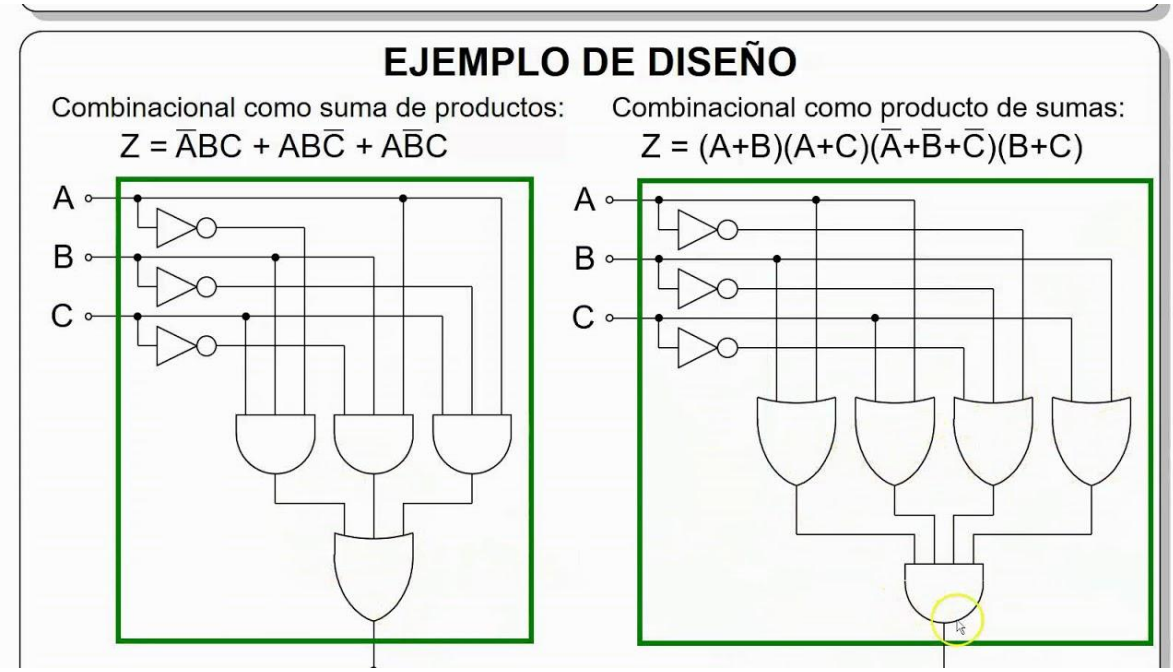
- Operaciones aritméticas: Sumadores, restadores.
- Control y comparación: Comparadores de magnitudes binarias.
- Conversión de datos: Codificadores (ej.: decimal a binario) y decodificadores.
- Enrutamiento de señales: Multiplexores (seleccionan entradas) y demultiplexores (distribuyen salidas).

VENTAJAS Y CONSIDERACIONES:

- Simplicidad estructural: Al no requerir elementos de memoria, su diseño es más sencillo.
- Optimización crítica: La simplificación reduce el número de puertas, mejorando velocidad y eficiencia energética.
- Limitaciones: No son aptos para tareas que requieran almacenamiento o secuenciación (ej.: contadores).

3.1 PROCEDIMIENTO DE DISEÑO DE CIRCUITOS COMBINACIONALES

El diseño de circuitos combinacionales es un proceso estructurado que convierte especificaciones funcionales en circuitos lógicos implementables, garantizando precisión y eficiencia. Su metodología incluye etapas iterativas que abarcan desde la conceptualización hasta la validación final.



ETAPAS DEL DISEÑO

1. Definición del Problema:
    - Establecer claramente la función requerida (ej.: sumar dos números binarios, comparar señales).
    - Identificar restricciones técnicas (velocidad, consumo energético, complejidad).
2. Identificación de Variables:
    - Asignar entradas (variables independientes) y salidas (resultados esperados).

- Ejemplo: En un sumador, entradas = bits a sumar (A, B); salidas = resultado (S) y acarreo (C).

3. Construcción de la Tabla de Verdad:
   - Listar todas las combinaciones posibles de entradas (2^n para *n* variables) y definir la salida correspondiente para cada caso.
   - Base para derivar la función lógica.
4. Derivación de la Función Booleana:
   - Expresar las salidas en términos algebraicos usando formas canónicas:
     - *Suma de minitérminos* (OR de combinaciones donde la salida es 1).
     - *Producto de maxitérminos* (AND de combinaciones donde la salida es 0).
5. Simplificación Lógica:
   - Aplicar técnicas para reducir la complejidad de la función:
     - Mapas de Karnaugh: Agrupar términos adyacentes gráficamente.
     - Método de Quine-McCluskey: Algoritmo para minimización sistemática.
   - Objetivo: Minimizar el número de puertas lógicas y retardos.
6. Implementación Gráfica:
   - Diagrama Lógico: Representar la función simplificada mediante puertas lógicas (AND, OR, NOT, etc.) y sus interconexiones.
   - Diagrama Eléctrico (opcional): Detallar componentes físicos (ej.: CI 7400 para NAND) y conexiones en protoboard o PCB.
7. Simulación y Verificación:
   - Usar herramientas como *Logisim* o *Multisim* para validar el comportamiento del circuito antes de su fabricación.
   - Detectar errores de diseño (ej.: retardos críticos, ruido en señales).

CONSIDERACIONES CLAVE

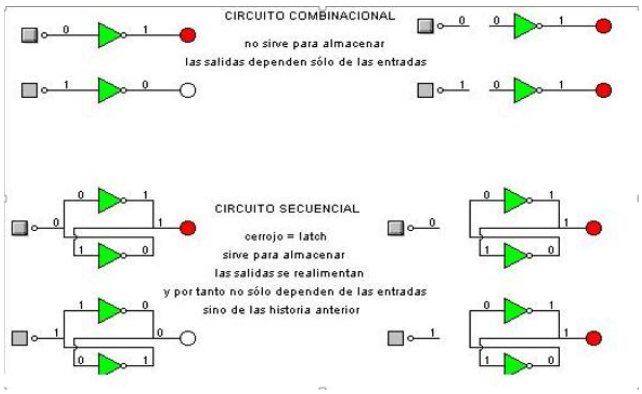
- Iteración Continua: El proceso no es lineal; simplificaciones pueden requerir ajustar la tabla de verdad o el diagrama.
- Optimización: Equilibrar entre complejidad (número de puertas) y rendimiento (retardo total).
- Herramientas Digitales: Software de simulación acelera la depuración y reduce costos en prototipos.

APLICACIONES PRÁCTICAS

- Sistemas Educativos: Enseñanza de lógica digital mediante proyectos como semáforos o calculadoras básicas.
- Desarrollo Profesional: Diseño de circuitos integrados, sistemas embebidos o módulos en FPGA.

VENTAJAS DEL ENFOQUE METODOLÓGICO

- Precisión: Elimina ambigüedades al seguir pasos estructurados.
- Eficiencia: Reduce errores y retrabajos mediante simulaciones tempranas.
- Escalabilidad: Permite adaptar el diseño a requerimientos más complejos (ej.: añadir entradas adicionales).

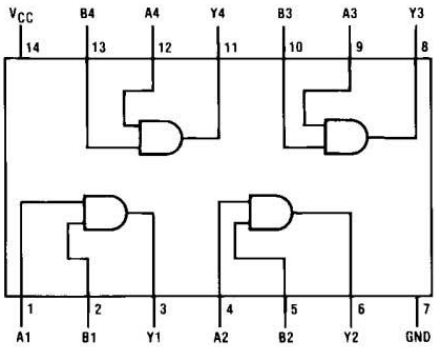


3.2 Circuitos Combinacionales Básicos

Los circuitos combinacionales básicos son componentes esenciales en la electrónica digital, diseñados para realizar operaciones específicas que forman la base de sistemas más complejos. Estos módulos, aunque simples en su funcionamiento, permiten ejecutar tareas críticas como operaciones aritméticas, enrutamiento de señales, conversión de códigos y comparación de datos. Su correcto entendimiento y aplicación son fundamentales para construir sistemas digitales robustos y eficientes.

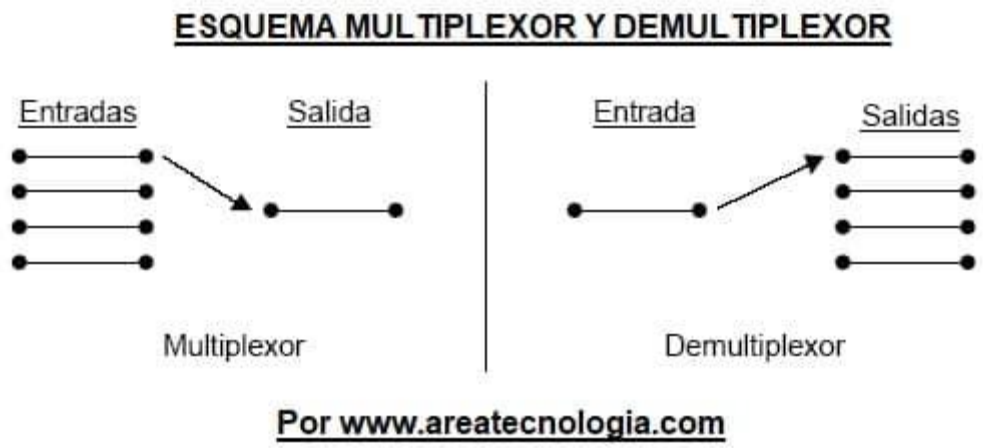
CIRCUITOS ARITMÉTICOS: LA BASE DEL CÁLCULO DIGITAL

Entre los más destacados se encuentran los sumadores y restadores, pilares de las operaciones matemáticas en sistemas digitales. Un semisumador, por ejemplo, utiliza una puerta XOR para generar la suma de dos bits (A y B) y una puerta AND para producir el acarreo. Este circuito, aunque limitado a dos bits, sienta las bases para diseños más avanzados. El sumador completo expande esta funcionalidad al incorporar un acarreo de entrada ($\text{Cin}$$C_{in}$), permitiendo sumar números de múltiples bits mediante conexiones en cascada. Las ecuaciones que lo rigen — $S=A\oplus B\oplus \text{Cin}$$S=A\oplus B\oplus C_{in}$ para la suma y $\text{Cout}=(A\cdot B)+(\text{Cin}\cdot(A\oplus B))$$C_{out}=(A\cdot B)+(C_{in}\cdot(A\oplus B))$ para el acarreo de salida— reflejan su capacidad para manejar operaciones secuenciales. Por otro lado, los restadores aprovechan el concepto de *complemento a dos* para transformar la resta en una suma, combinando inversores y sumadores. Estos circuitos son la columna vertebral de unidades aritmético-lógicas (ALU) en procesadores y microcontroladores.



CIRCUITOS DE ENRUTAMIENTO: MULTIPLEXORES Y DEMULTIPLEXORES

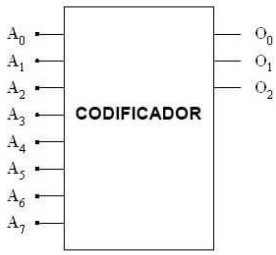
Los multiplexores (MUX) actúan como selectores inteligentes, capaces de dirigir una de varias entradas hacia una única salida mediante señales de control. Por ejemplo, un MUX 4:1 utiliza dos bits de control para elegir entre cuatro entradas, optimizando el uso de buses de datos en sistemas de comunicación. Su contraparte, el demultiplexor (DEMUX), realiza la función inversa: distribuye una única entrada a múltiples salidas según las señales de control. Ambos circuitos son vitales en aplicaciones como la multiplexación de canales en telecomunicaciones o la gestión de periféricos en sistemas embebidos.



CONVERSIÓN DE CÓDIGOS: CODIFICADORES Y DECODIFICADORES

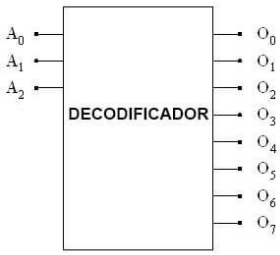
Los codificadores simplifican la representación de datos al convertir entradas activas en códigos binarios compactos. Un ejemplo clásico es el codificador prioritario de 8 a 3 bits, que identifica la entrada de mayor prioridad y genera su equivalente binario, útil en sistemas de interrupciones. Los decodificadores, en cambio, traducen códigos binarios en señales específicas, como en el caso del decodificador BCD a 7 segmentos, que activa los segmentos adecuados de un display numérico. Estos circuitos no solo facilitan la interfaz entre dispositivos, sino que también permiten la reutilización de módulos en diseños modulares.

Codificadores y Decodificadores



Solo una entrada activada a la vez

Salida de código binario



Entrada de código binario

Solo una salida activada a la vez

COMPARADORES: TOMA DE DECISIONES EN TIEMPO REAL

Los comparadores son circuitos especializados en evaluar relaciones entre números binarios. Mediante puertas lógicas jerárquicas, determinan si un número es mayor, menor o igual a otro, generando señales de control que alimentan algoritmos o sistemas de actuación. Su aplicación es crucial en dispositivos como termostatos digitales, donde se comparan valores de temperatura, o en unidades de procesamiento que requieren ordenar datos.

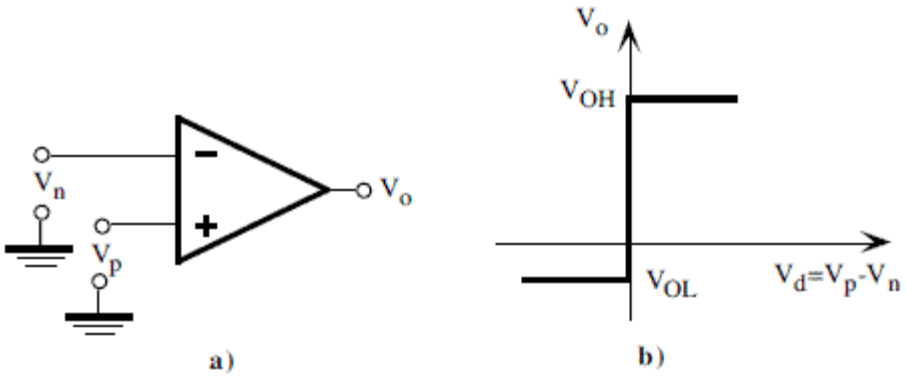


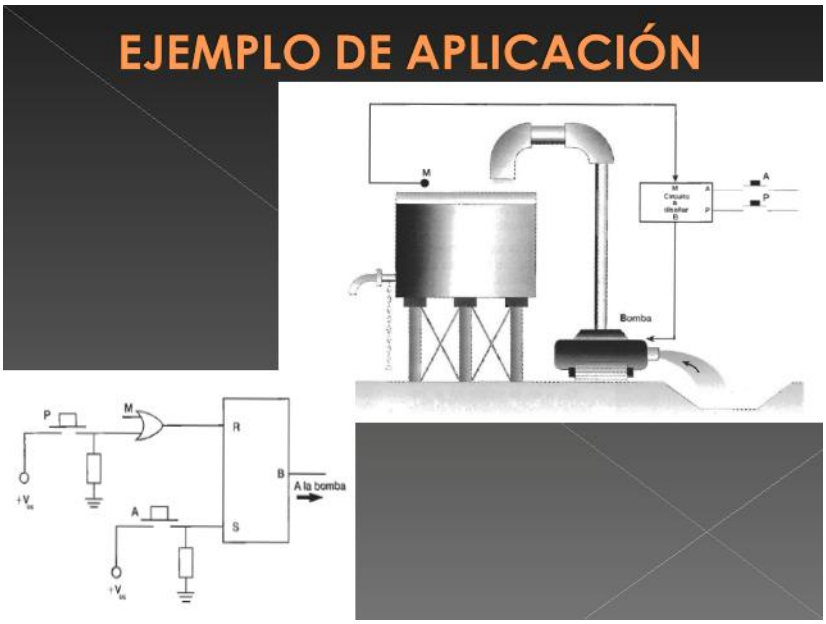
Figura 9.1. Comparador de tensión: **a)** Símbolo, **b)** VTC.

IMPLEMENTACIÓN Y OPTIMIZACIÓN: DEL DISEÑO A LA PRÁCTICA

Estos circuitos suelen implementarse utilizando puertas lógicas estándar (AND, OR, XOR) o mediante circuitos integrados predefinidos de la serie 74XX, como el 7483 (sumador de 4 bits) o el 74151 (multiplexor 8:1). La optimización juega un papel clave: técnicas como el uso de mapas de Karnaugh o algoritmos de minimización (Quine-McCluskey) reducen el número de puertas, mejorando la velocidad y reduciendo el consumo energético. Además, herramientas de simulación como Logisim permiten validar diseños antes de su fabricación, minimizando errores y costos.

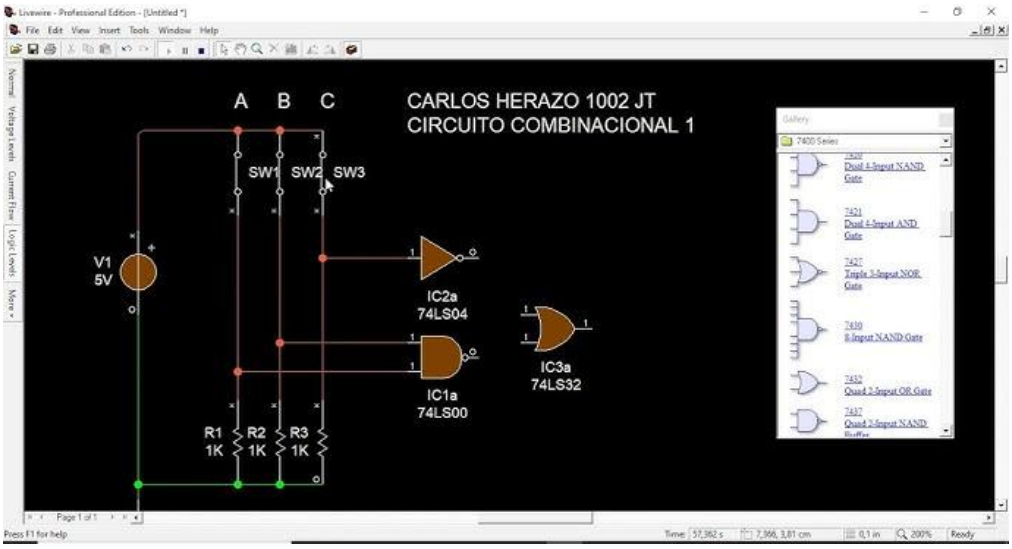
INTEGRACIÓN EN SISTEMAS COMPLEJOS

La verdadera potencia de estos circuitos surge al combinarlos. Por ejemplo, un sumador completo en cascada forma la base de una ALU, mientras que multiplexores y decodificadores se integran en CPUs para gestionar rutas de datos. Incluso sistemas aparentemente sencillos, como una calculadora básica, dependen de la interacción entre comparadores, sumadores y circuitos de conversión. Esta modularidad no solo simplifica el diseño, sino que también permite escalar sistemas para abordar desafíos tecnológicos avanzados, como el procesamiento de señales en tiempo real o la implementación de redes neuronales en hardware.



3.3 Simulación de los circuitos combinacionales

La simulación es un paso crítico en el diseño de sistemas digitales, ya que permite evaluar el comportamiento de un circuito antes de su implementación física. En el caso de los circuitos combinacionales, cuya salida depende únicamente de la combinación actual de sus entradas, la simulación sirve para asegurar que el diseño cumpla con la función lógica deseada en todas las posibles condiciones de entrada.



Importancia de la Simulación

La simulación de circuitos combinacionales tiene varias ventajas:

- **Validación Temprana:** Permite detectar errores de diseño y realizar correcciones sin necesidad de fabricar un prototipo físico, lo que reduce significativamente el tiempo y los costos de desarrollo.

- **Optimización del Diseño:** Al simular el circuito se pueden analizar parámetros críticos como el retardo de propagación, el consumo energético

y la estabilidad de la señal. Esto posibilita la optimización del número de compuertas y la reducción de complejidad lógica.

- **Flexibilidad y Experimentación:** El uso de herramientas de simulación permite experimentar con diferentes configuraciones y simplificaciones. Los diseñadores pueden evaluar el comportamiento del circuito ante variaciones en las condiciones de entrada o cambios en el entorno operativo.

- **Documentación y Análisis:** La simulación genera resultados que pueden ser documentados y analizados, facilitando la comprensión del funcionamiento del circuito y la generación de informes técnicos para fines educativos o de investigación.

Metodología de Simulación

El proceso de simulación de un circuito combinacional suele seguir una serie de pasos estructurados:

1. **Definición del Problema y la Tabla de Verdad:**

   Se parte del enunciado del problema, donde se definen claramente las variables de entrada y salida. Con esta información se elabora una tabla de verdad que describe todas las combinaciones posibles de entradas y la salida correspondiente que el circuito debe generar.

2. **Modelado y Simplificación de la Función Lógica:**

A partir de la tabla de verdad, se obtiene una función booleana que describe el comportamiento del circuito. Esta función puede ser simplificada utilizando técnicas del álgebra de Boole o mapas de Karnaugh, lo que resulta en un diseño más eficiente y económico en términos de hardware.

3. **Implementación en un Lenguaje de Descripción de Hardware (HDL):** Los circuitos combinacionales se modelan comúnmente utilizando lenguajes como VHDL o Verilog. Esto permite describir el comportamiento del circuito a un alto nivel de abstracción y, a la vez, generar código que pueda ser simulado y sintetizado para la implementación en dispositivos FPGA o ASIC. Por ejemplo, se puede definir una entidad que recoja las entradas y salidas, y una arquitectura que implemente la función lógica a través de asignaciones continuas o procesos secuenciales.

4. **Simulación y Verificación con Test Bench:**

   Se diseña un banco de pruebas (test bench) que envíe todas las combinaciones posibles de entrada al circuito. La simulación permite observar en tiempo real cómo se comportan las salidas, comparándolas con las expectativas derivadas de la tabla de verdad. Esta etapa es fundamental para validar que el diseño es correcto y que no existen errores de lógica o problemas de sincronización.

5. **Análisis de Parámetros y Optimización:**

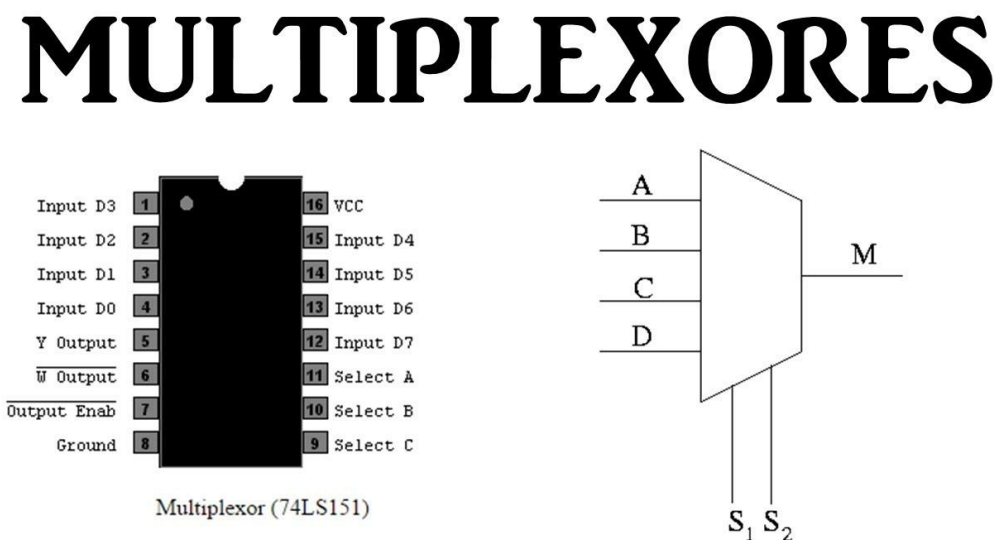
   Durante la simulación se evalúan parámetros como el retardo de propagación (tiempo que tarda un cambio en la entrada en reflejarse en la salida), el consumo de energía y la robustez frente a variaciones de voltaje. Estos datos son esenciales para realizar mejoras y ajustar el diseño a las necesidades específicas del proyecto.

3.3.1 Multiplexores

En la electrónica y las telecomunicaciones, la necesidad de transmitir o procesar múltiples señales a través de medios limitados ha llevado al desarrollo de dispositivos que gestionan el flujo de información de manera eficiente. Los **multiplexores** son circuitos que permiten seleccionar una entre varias señales de entrada y dirigirla a una única salida, lo que facilita la utilización óptima de recursos como cables, frecuencias o incluso componentes internos en circuitos digitales.

¿Qué es un Multiplexor?

Un **multiplexor** (MUX) es un circuito combinacional ya sea digital o analógico que, utilizando un conjunto de entradas de control (también denominadas líneas de selección), dirige la señal de una de sus múltiples entradas hacia una única línea de salida. Esta función se asemeja a la de un interruptor o conmutador, que "elige" cuál fuente de datos se transmite en cada momento.



Por ejemplo, en un multiplexor digital de 2 a 1, se tienen dos entradas (I_0 e I_1), una única salida (Q) y una línea de selección (S). Si S está en un nivel lógico bajo (0), la salida será igual a I_0; si S es alto (1), la salida tomará el valor de I_1.

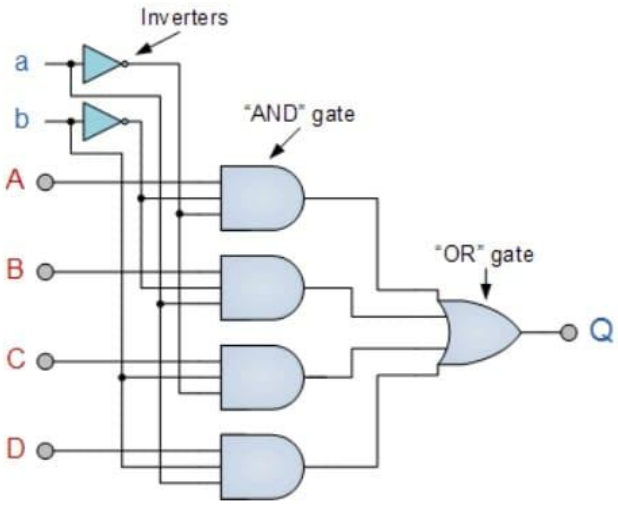
Además, existen multiplexores más complejos, como los de 4 a 1, 8 a 1, etc., en los cuales el número de líneas de selección (k) se relaciona con el número de entradas mediante la fórmula:

$$\boldsymbol{Numero\ de\ netradas} = \boldsymbol{2^k}$$

Principios de Funcionamiento

Lógica de Selección

El funcionamiento de un multiplexor se basa en el uso de puertas lógicas (como AND, OR y NOT) para implementar la función de selección.



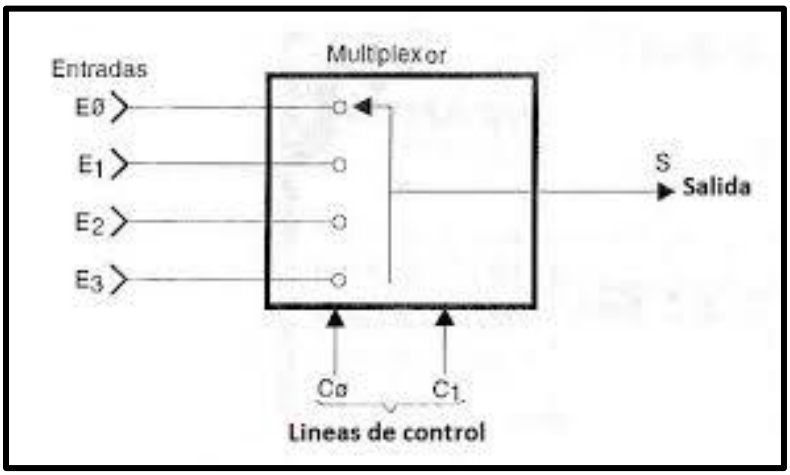
Cada combinación de las señales de control activa únicamente una de las entradas, haciendo que su valor se propague hacia la salida mientras las demás se mantienen bloqueadas. La operación se puede expresar en términos de álgebra booleana; por ejemplo, para un MUX 2 a 1 la función de salida es:

$$Q = \overline{S} * I_0 + S * I_1$$

Este esquema se extiende a configuraciones más complejas, donde se utilizan más líneas de selección para escoger entre 2, 4, 8 o incluso más entradas.

Componentes Internos y Diseño

La estructura básica de un multiplexor incluye:



- **Entradas de datos:** Las señales que se desean transmitir.
- **Líneas de selección:** Entradas de control que determinan qué dato se enviará.

- **Salida:** El canal único por el cual se transmite la señal elegida.

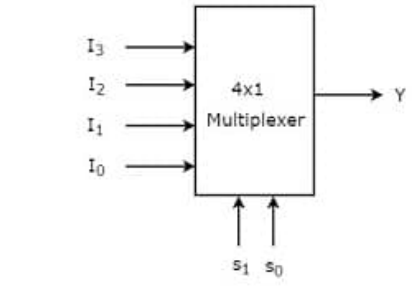
En implementaciones digitales, los multiplexores se pueden construir de forma jerárquica. Por ejemplo, un multiplexor de 4 a 1 puede diseñarse utilizando dos multiplexores de 2 a 1 para la selección primaria y un tercer multiplexor para la selección final de la salida.

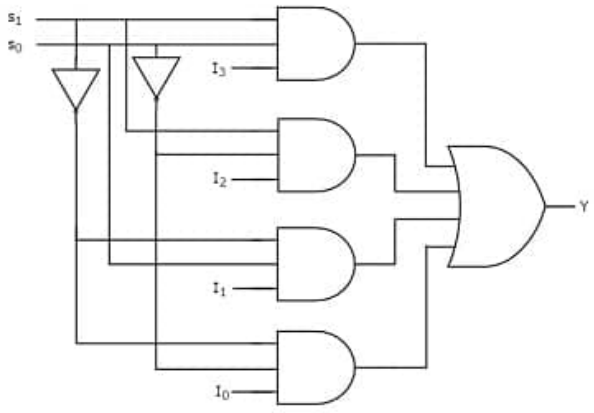
Tipos de Multiplexores

Multiplexores Digitales y Analógicos

- **Digitales:** Diseñados para manejar señales binarias y se implementan usando compuertas lógicas. Ejemplos clásicos incluyen los circuitos integrados de la serie TTL (por ejemplo, 74LS151 para 8 entradas a 1 salida o 74LS153 para 4 a 1) que se usan en aplicaciones de computación y control.

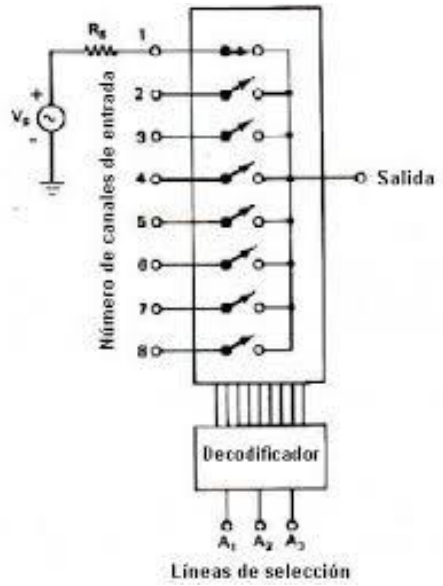
ENTRADAS		SALIDA
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3





MULTIPLEXOR 4 A 1

- **Analógicos:** Emplean interruptores electrónicos, como transistores o MOSFET, para conmutar señales analógicas. Su aplicación es muy extendida en sistemas de adquisición de datos y procesamiento de señales de video o audio.

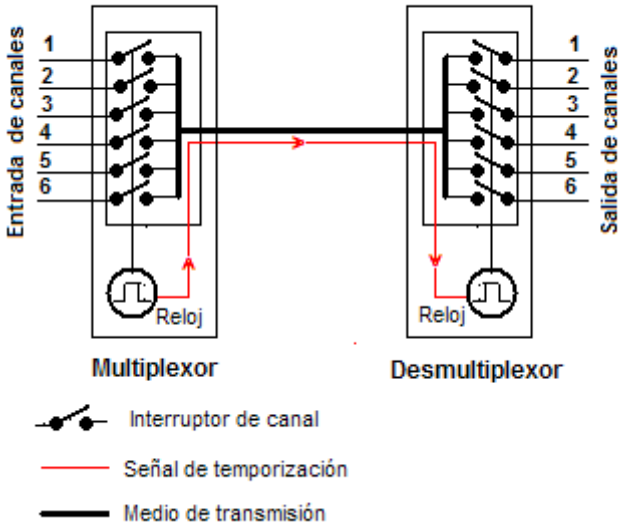


Técnicas de Multiplexación

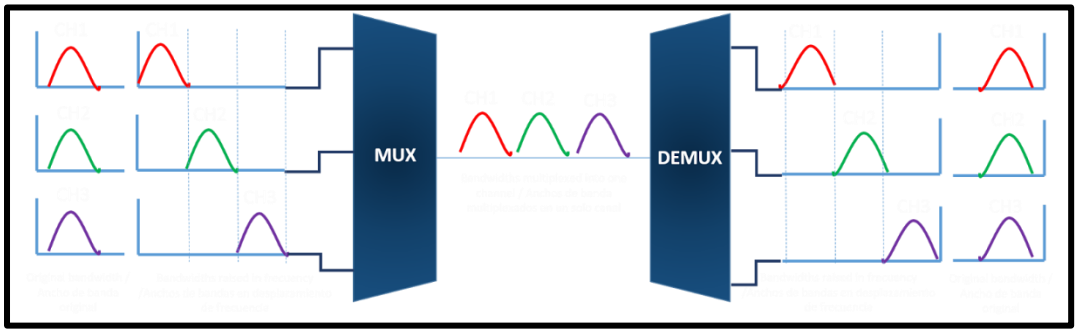
Técnicas de Multiplexación en Comunicaciones

La idea de “multiplexación” se aplica no solo en el diseño de circuitos internos sino también en sistemas de transmisión:

- **Multiplexación por División de Tiempo (TDM):** El canal de transmisión se comparte por intervalos de tiempo asignados a cada señal. Es común en sistemas digitales, como en redes de telefonía o en ciertos protocolos de transmisión de datos.

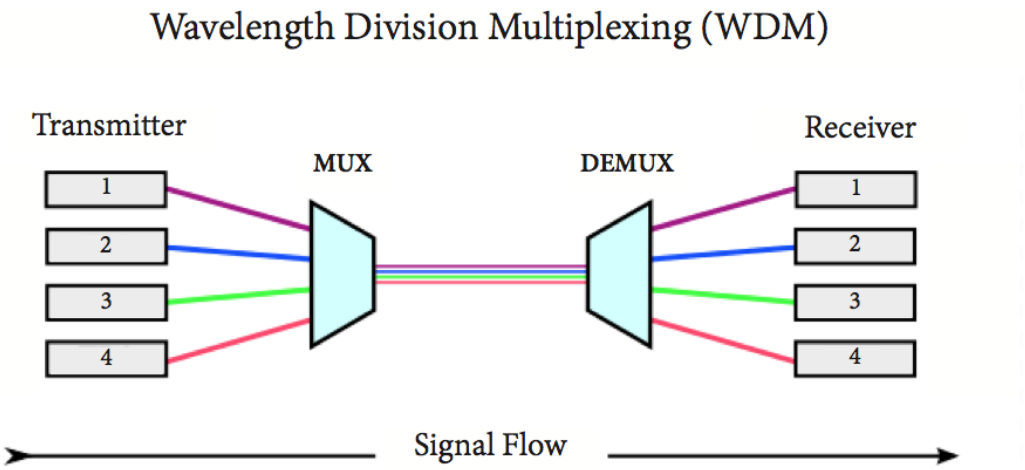


- **Multiplexación por División de Frecuencia (FDM):** Cada señal se modula sobre una portadora de frecuencia diferente y se transmiten simultáneamente por un solo canal; es muy usado en radio y televisión.

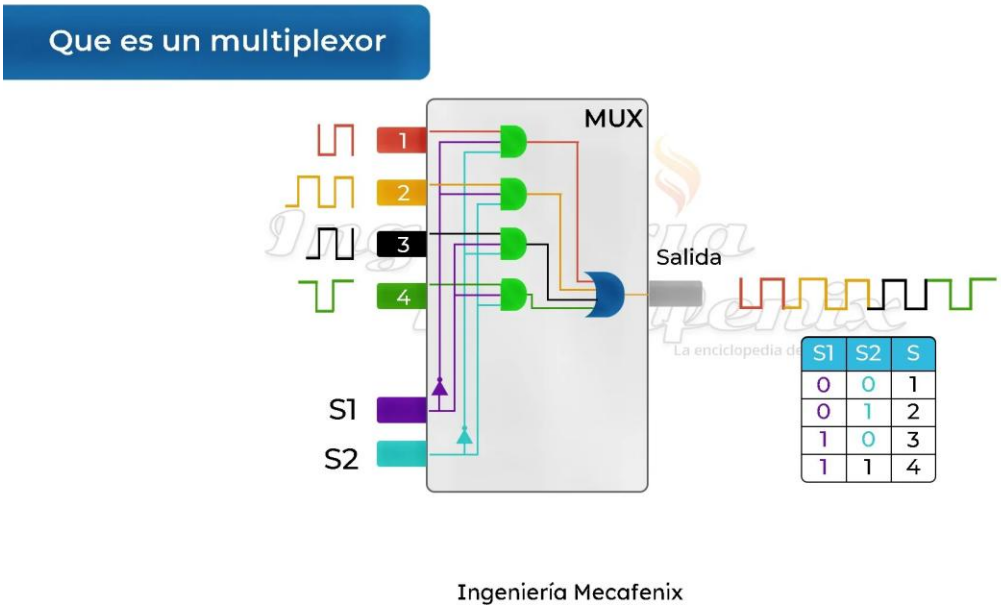


- **Multiplexación por División de Longitud de Onda (WDM):** Utilizada principalmente en sistemas de fibra óptica, combina múltiples longitudes de onda (colores de luz) para transmitir datos, aumentando la capacidad sin necesidad de instalar más cables. En redes ópticas se diferencian tecnologías como CWDM (multiplexación por división de longitud de onda

gruesa) y DWDM (multiplexación por división de longitud de onda densa), que difieren en el espaciado entre canales y en la cantidad de señales que pueden transportar.



Características Técnicas



Ingeniería Mecafenix

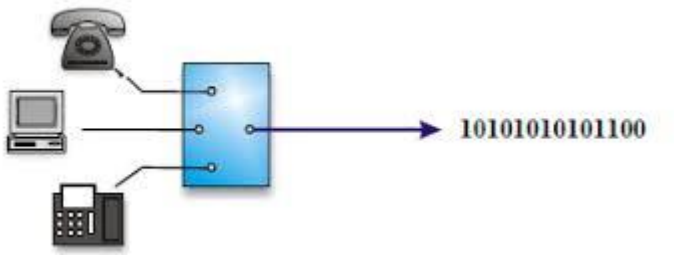
Características

- **Selección Controlada:** Mediante líneas de selección binarias se decide la conexión de una de varias entradas a la salida.
- **Velocidad de Conmutación:** Los multiplexores deben cambiar rápidamente entre entradas sin pérdida significativa de datos, lo que los hace esenciales en sistemas de alta velocidad.
- **Sin Memoria:** Al ser circuitos combinacionales, la salida depende únicamente del estado actual de las entradas y las líneas de selección.
- **Escalabilidad:** Permiten ampliar el número de entradas sin cambiar fundamentalmente la estructura, utilizando diseños jerárquicos.

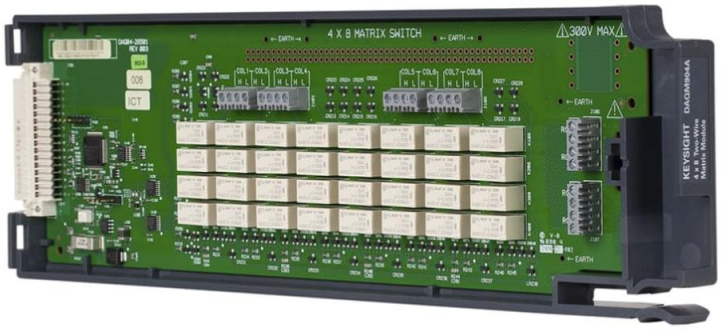
Aplicaciones de los Multiplexores

Los multiplexores tienen una amplia gama de aplicaciones en distintos campos:

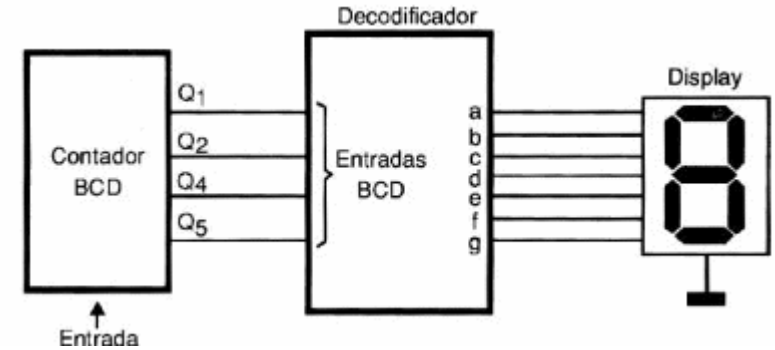
- **Telecomunicaciones y Redes:** Permiten el enrutamiento de múltiples llamadas o flujos de datos a través de una única línea o canal, optimizando el uso del ancho de banda. Por ejemplo, en redes de fibra óptica se emplean multiplexores (y su contraparte demultiplexora) para combinar y separar señales de diferentes longitudes de onda mediante WDM.



- **Sistemas de Adquisición de Datos:**
  Se utiliza un MUX para medir varias señales (de sensores, por ejemplo) con un solo convertidor analógico-digital (ADC), lo que reduce costos y complejidad.

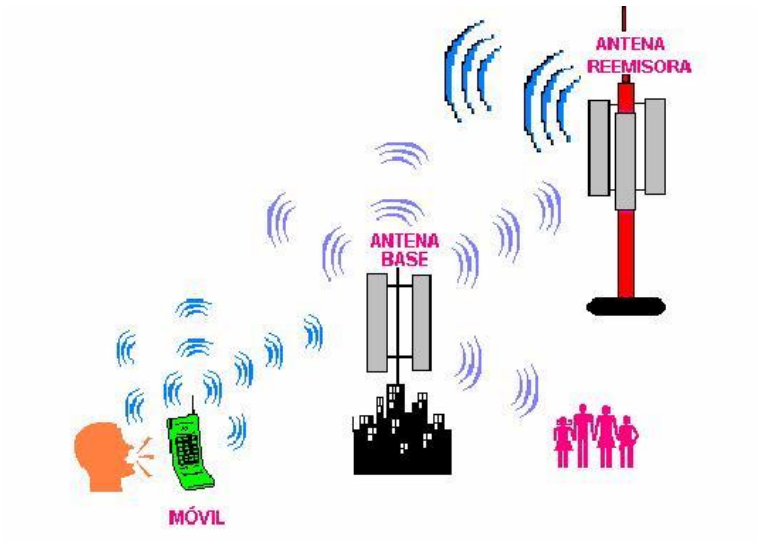


- **Electrónica Digital:**
  En circuitos lógicos y microprocesadores, los multiplexores permiten seleccionar entre diferentes fuentes de datos o instrucción, facilitando la programación y el diseño de sistemas integrados.



- **Aplicaciones de Audio y Video:** Permiten conmutar señales analógicas en sistemas de video y audio, o controlar la ganancia de amplificadores de forma digital, lo que es útil en mezcladores y sistemas de procesamiento de señal audiovisual.

- **Sistemas de Comunicación Móvil:** Tecnologías basadas en acceso múltiple (TDMA, FDMA) utilizan principios similares a los multiplexores para administrar el acceso simultáneo a los recursos de red por múltiples usuarios.

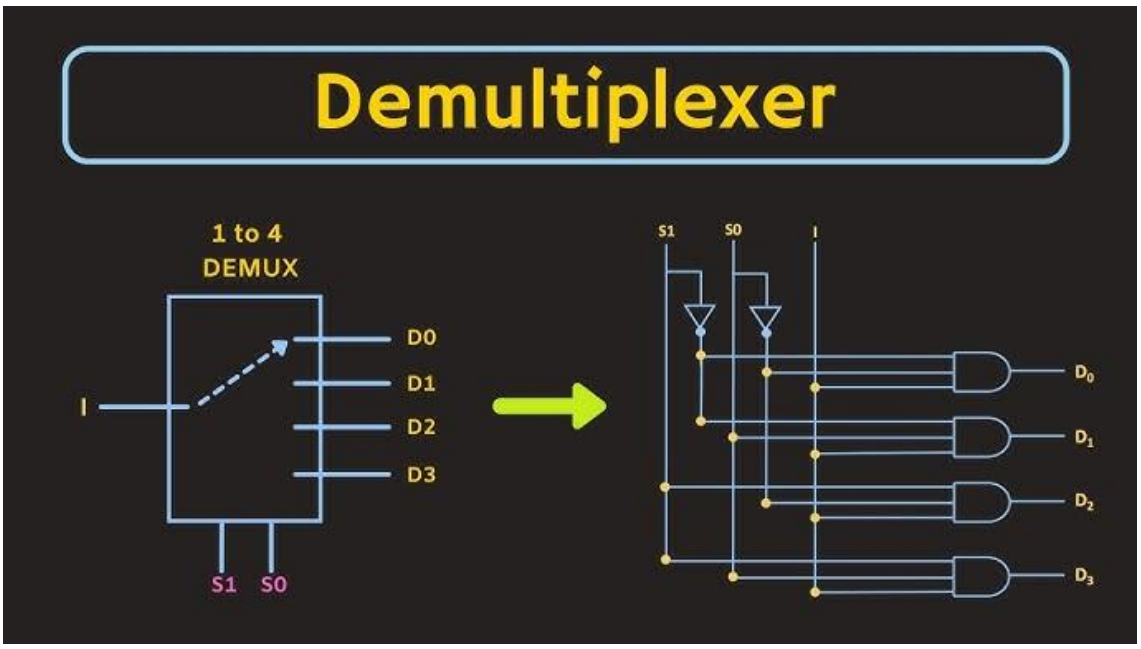


3.3.2 Demultiplexores

En sistemas de comunicaciones y procesamiento de datos es fundamental no solo combinar señales para aprovechar eficientemente el medio disponible (como ocurre con los multiplexores), sino también separar, en el extremo receptor, la señal compuesta en sus componentes originales. Para ello se utiliza el demultiplexor (DEMUX). Este dispositivo es la contraparte del multiplexor y permite distribuir una señal de entrada única a múltiples salidas, de acuerdo con señales de control específicas.

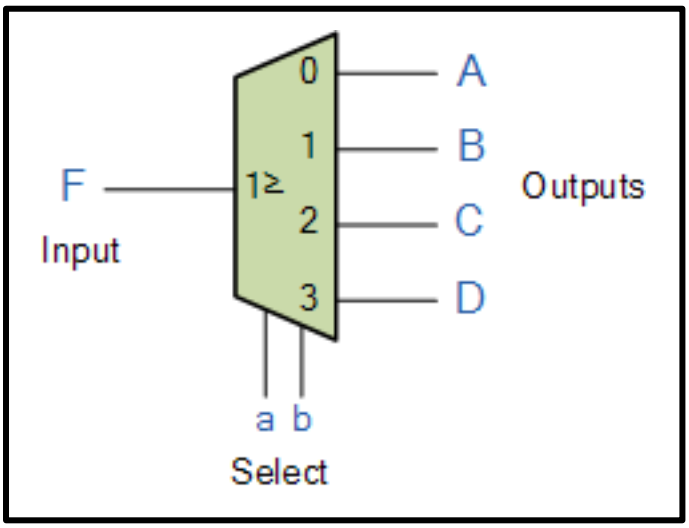
¿Qué es un Demultiplexor?

Un **demultiplexor** (DEMUX) es un circuito combinacional que realiza la función opuesta a la de un multiplexor. Recibe **una sola entrada** de datos y la distribuye a **una de varias salidas**, dependiendo del valor de las **líneas de selección**.



Es decir, actúa como un "enrutador" de señales: **dirige una señal de entrada a una de sus múltiples salidas** basándose en una combinación binaria determinada.

Por ejemplo, un demultiplexor de 1 a 4 tiene:



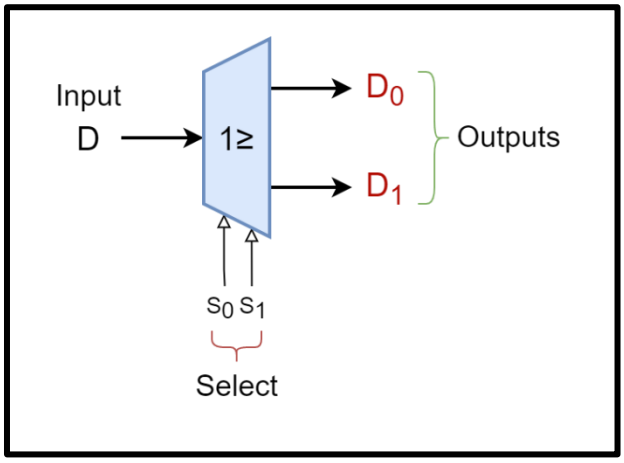
- 1 entrada de datos (F)
- 2 líneas de selección a, b)
- 4 salidas (A, B, C, D)

Dependiendo del valor binario en las líneas de selección, el dato de entrada se enviará a una sola de las salidas, mientras las otras se mantienen inactivas (por lo general en 0).

¿Cómo Funciona un Demultiplexor?

Un demultiplexor utiliza puertas lógicas para enviar la señal de entrada a una de sus salidas, según el valor de las líneas de selección. La lógica que controla esta operación se puede representar con expresiones booleanas y diagramas de compuertas.

Ejemplo de función lógica para un DEMUX 1 a 2:

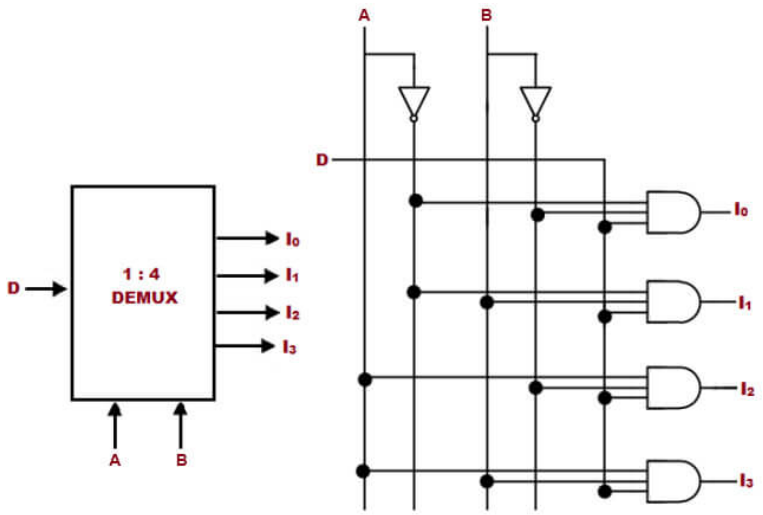


Supongamos que D es la entrada de datos y S1, S0 son las líneas de selección:

- Y0 = D · ¬S1 · ¬S0
- Y1 = D · ¬S1 · S0

Solo una salida se activa dependiendo de la combinación de las señales S1 y S0.

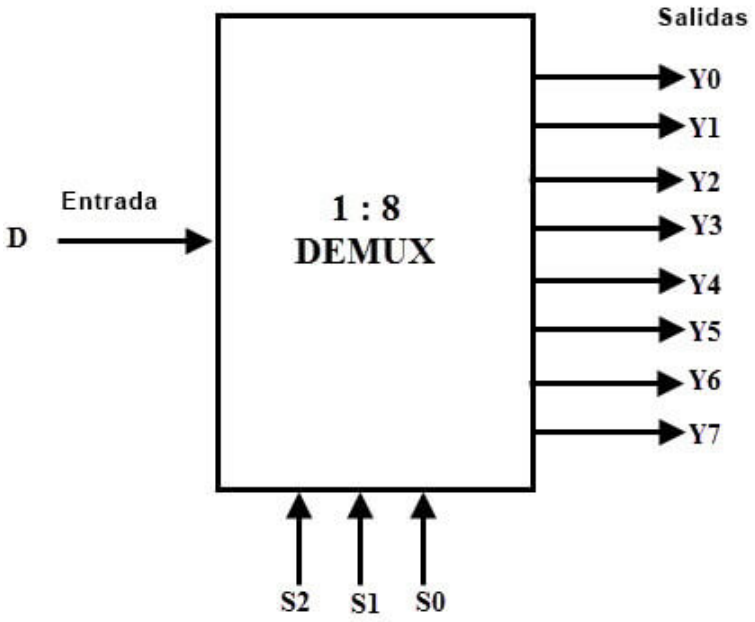
Características de los Demultiplexores



- **Dirección única de señal:** De una sola entrada hacia una de múltiples salidas.
- **Líneas de selección:** Determinan cuál salida recibe el dato.
- **Sin memoria:** Su funcionamiento depende únicamente del estado actual de sus entradas y control.
- **Utilizan compuertas lógicas:** Se implementan con compuertas AND, NOT, etc.
- **Complementarios de los multiplexores:** Muchas veces se usan en conjunto.

Tipos de Demultiplexores

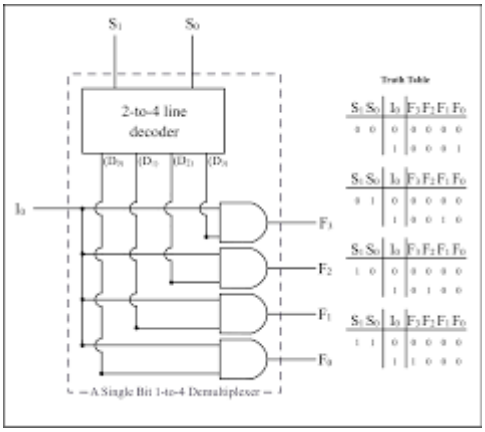
Según la cantidad de salidas:



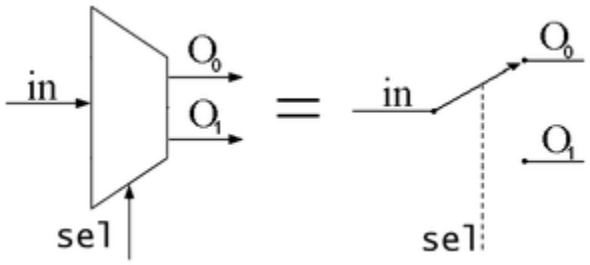
- **1 a 2:** 1 línea de selección (2 salidas posibles)
- **1 a 4:** 2 líneas de selección
- **1 a 8:** 3 líneas de selección
- **1 a 16:** 4 líneas de selección

Según la señal que manejan:

- **Demultiplexores digitales:** Para señales binarias (0 y 1).

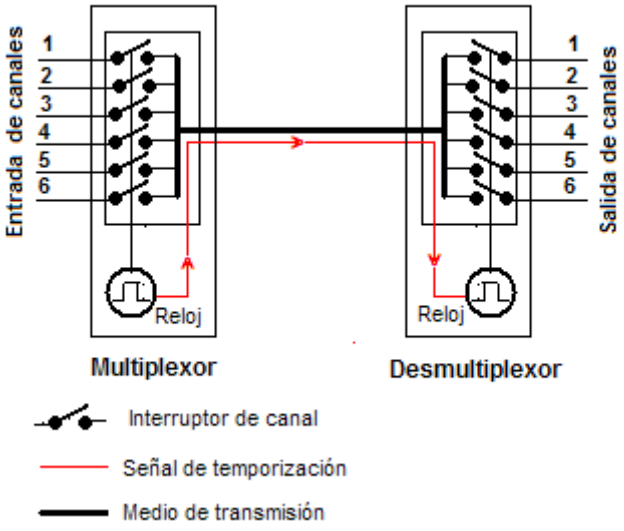


- **Demultiplexores analógicos:** Manejan señales continuas mediante interruptores controlados electrónicamente.



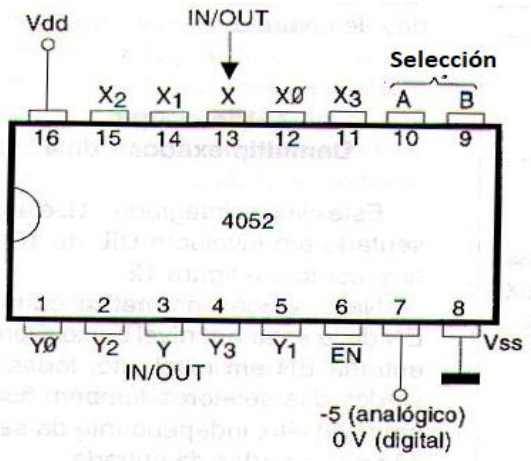
Aplicaciones del Demultiplexor

Redes de comunicación



- Se usan para **separar señales combinadas** en un canal (cuando se usó un multiplexor en el transmisor).
- En sistemas de **multiplexación por tiempo (TDM)**, el demultiplexor recupera las señales originales en el receptor.

Microprocesadores

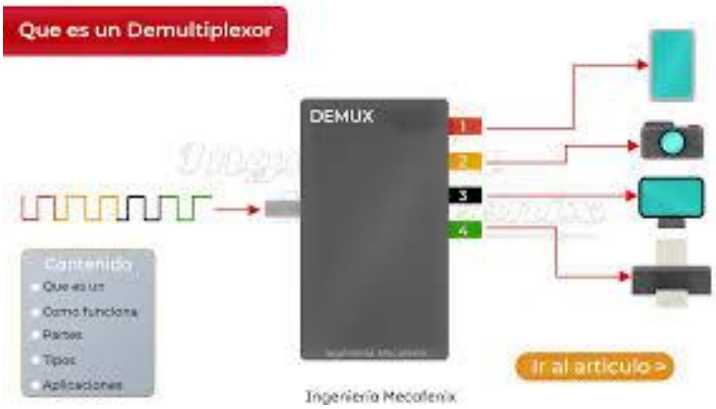


- Para **decodificar instrucciones o direcciones** de memoria.
- Por ejemplo, un demultiplexor puede activar uno entre varios dispositivos conectados (memorias RAM, ROM, puertos, etc.).

Pantallas LED

- En sistemas de matriz de LEDs, se puede usar un demultiplexor para **activar una fila específica**, mientras otro circuito activa columnas, iluminando un LED específico.

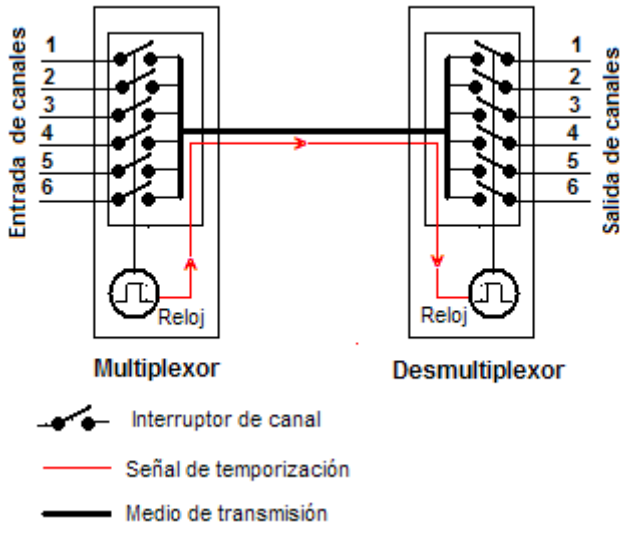
Sistemas de control



- Se utilizan para **dirigir señales de control** a distintos dispositivos desde un único controlador central.

Relación con Multiplexores

- Son complementarios: donde el MUX combina muchas señales en una sola línea, el DEMUX hace lo contrario.



- En sistemas de transmisión, **MUX + línea de comunicación + DEMUX** permite enviar múltiples señales por un solo canal y luego recuperarlas correctamente.

Fig 1 a) Mux/Demux devices

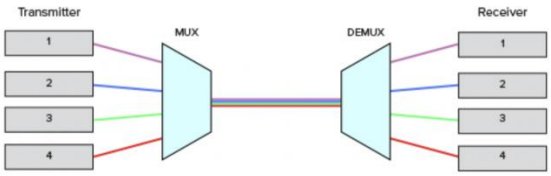
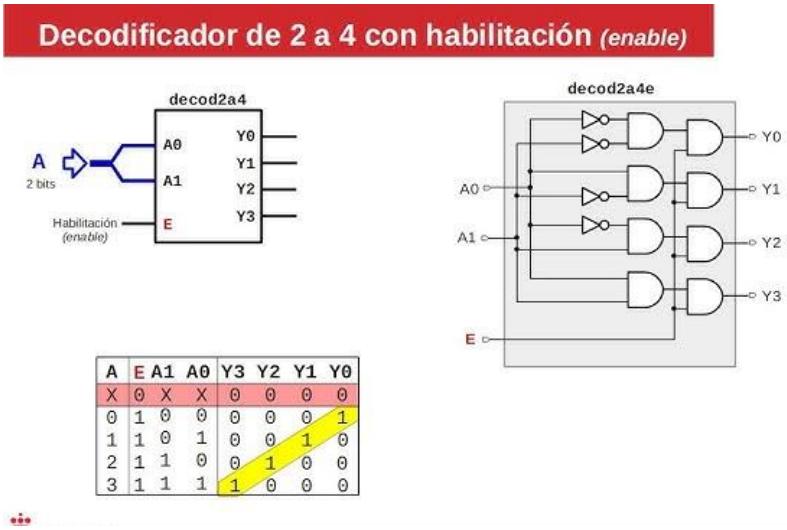


Figure 1b) Wavelength Division Multiplexing/Demultiplexing

3.3.3 Decodificadores

Dentro del amplio mundo de los circuitos combinacionales, los decodificadores ocupan un lugar especial debido a su función esencial de convertir códigos binarios en señales de salida específicas. Estos circuitos son fundamentales en múltiples aplicaciones, desde la selección de direcciones en sistemas de memoria hasta la activación de displays numéricos.

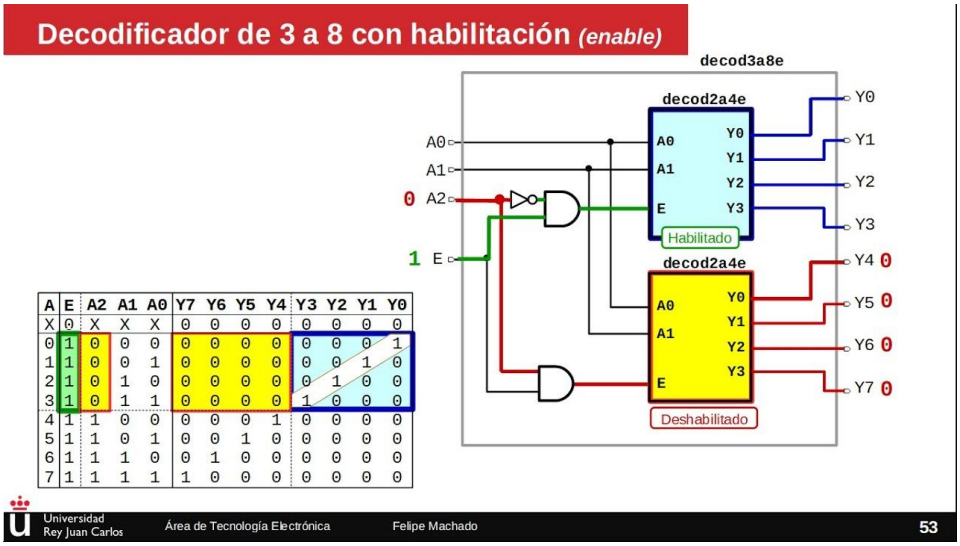


Concepto y Funcionamiento de los Decodificadores

Un decodificador es un circuito que recibe un conjunto de n entradas y genera 2^n salidas, de forma que solo una de ellas se activa para cada combinación única de las entradas. Este comportamiento se logra mediante el diseño de una función lógica que asigna, para cada combinación de entrada, una única salida en estado activo (que puede ser, por ejemplo, un nivel lógico bajo o alto, según el diseño del circuito).

Ejemplo de un Decodificador 3 a 8

En un decodificador de 3 a 8, las tres entradas pueden representar números del 0 al 7 en binario. Por cada combinación, se activa únicamente una de las ocho salidas. Por ejemplo, si las entradas son 0 1 00\,1\,0010 (equivalente al número 2 en decimal), la salida correspondiente al número 2 se activa, mientras que las demás permanecen inactivas.



A	E	A2	A1	A0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
X	0	X	X	X	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	1
1	1	0	0	1	0	0	0	0	0	0	1	0
2	1	0	1	0	0	0	0	0	0	1	0	0
3	1	0	1	1	0	0	0	0	1	0	0	0
4	1	1	0	0	0	0	0	1	0	0	0	0
5	1	1	0	1	0	0	1	0	0	0	0	0
6	1	1	1	0	0	1	0	0	0	0	0	0
7	1	1	1	1	1	0	0	0	0	0	0	0

Línea de Habilitación (Enable)

Muchos decodificadores incluyen una o más entradas de habilitación que controlan la activación global del circuito. Cuando la señal de habilitación no está en el estado adecuado, todas las salidas se mantienen en un estado predeterminado (normalmente desactivadas), lo cual es útil para evitar errores o interferencias en sistemas complejos donde se requieren múltiples decodificadores.

Aplicaciones de los Decodificadores

Los decodificadores tienen una amplia variedad de aplicaciones en el diseño digital:

- **Selección de Memoria:** En sistemas de memoria, los decodificadores se utilizan para seleccionar la celda o bloque de memoria correcto, permitiendo la lectura o escritura de datos en una dirección específica.

- **Displays de 7 Segmentos:** Los decodificadores BCD a 7 segmentos son ampliamente utilizados en calculadoras, relojes digitales y otros dispositivos de visualización. Estos circuitos convierten un código BCD (usualmente de 4 bits) en las señales necesarias para encender los segmentos correctos de un display y representar un dígito decimal.
- **Sistemas de Control:** En interfaces de usuario, teclados y paneles de control, los decodificadores permiten identificar y procesar la combinación exacta de pulsaciones o señales, garantizando que se active la función deseada.
- **Implementación de Funciones Lógicas:** Además de sus aplicaciones directas, los decodificadores pueden emplearse para implementar funciones lógicas complejas. Mediante la conexión de las salidas de un decodificador a compuertas lógicas adicionales (por ejemplo, puertas OR o NAND), es posible generar una función lógica a partir de la suma de términos canónicos.

Proceso de Simulación de Decodificadores

La simulación de un decodificador sigue un proceso similar al de otros circuitos combinacionales, pero con algunas consideraciones específicas:

1. **Definición y Análisis de la Tabla de Verdad:**

   Se debe elaborar la tabla de verdad completa, especificando para cada combinación de entradas cuál es la salida que debe activarse. Este paso es fundamental para entender la lógica subyacente del decodificador.

2. **Modelado en HDL:**

   El decodificador se modela utilizando lenguajes como VHDL o Verilog. Por ejemplo, se define una entidad con las entradas (incluyendo la señal de

habilitación) y las salidas correspondientes. La arquitectura se puede implementar mediante un proceso que utilice una sentencia *CASE* para asignar los valores de salida en función del vector de entrada concatenado.

3. **Desarrollo de Test Bench:**

   Se crea un test bench que aplique a las entradas todas las combinaciones posibles, verificando que para cada caso se active únicamente la salida correcta. Esto permite comprobar de forma exhaustiva la funcionalidad del decodificador.

4. **Análisis del Comportamiento y Optimización:**

   La simulación proporciona información sobre el retardo de propagación, el consumo energético y posibles anomalías en la conmutación de las señales. Con esta información se pueden realizar ajustes para optimizar el diseño, tales como la reducción del número de compuertas o la mejora de la estabilidad en la transición entre estados.

3.3.4 CODIFICADORES.

Los codificadores son los dispositivos encargados de la codificación. El concepto de codificación, tal y como se exploró en la sección de Códigos, está vinculado a la "traducción" de un código a otro. Los codificadores pueden implementarse a partir de la aplicación de diagramas de Karnaugh comparando las tablas de verdad. También existen codificadores integrados de diferentes características.

Tienen dos tipos de entradas: de control y de datos. Se muestra un diagrama de un codificador. Las entradas de datos son las que componen el código a "traducir". Las entradas de control por su parte pueden ser de selección, las cuales se emplean en la codificación, y de habilitación, que existen en muchos tipos de integrados, y se encargan de permitir o no el proceso de trabajo.

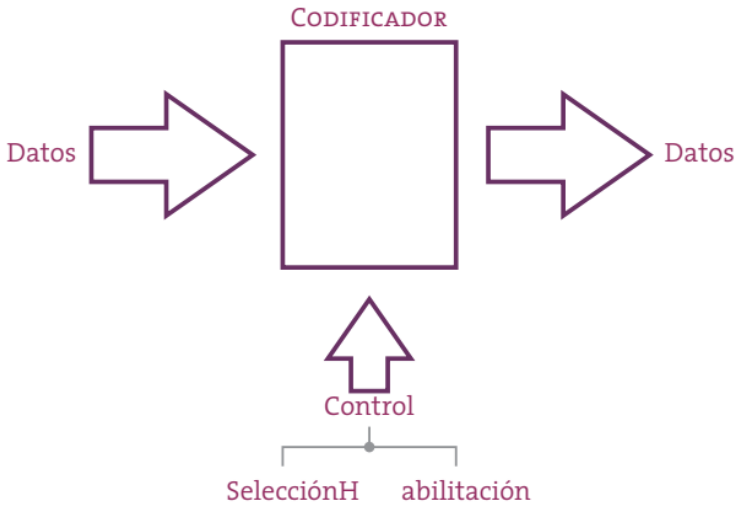


Ilustración 1 Diagrama de un codificador genérico

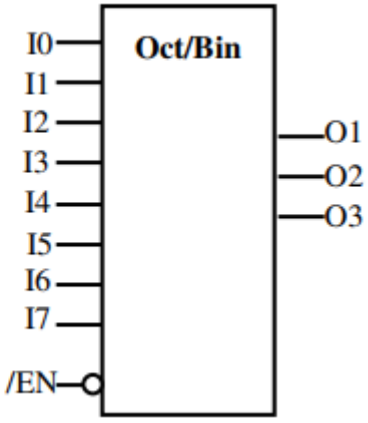
Son los dispositivos MSI que realizan la operación inversa a la realizada por los decodificadores. Generalmente, poseen $\mathbf{2^n}$ entradas y **n** salidas.

Cuando solo una de las entradas está activa para cada combinación de salida, se le denomina **codificador completo**.

Por ejemplo, el siguiente circuito proporciona a la salida la combinación binaria de

la entrada que se encuentra activada. En este caso se trata de un codificador completo de **8** bits, o también llamado codificador de 8 a 3 líneas:

/EN	I0	I1	I2	I3	I4	I5	I6	I7	O1	O2	O3
1	X	X	X	X	X	X	X	X	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	1	0
0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	0	1	1	1	1

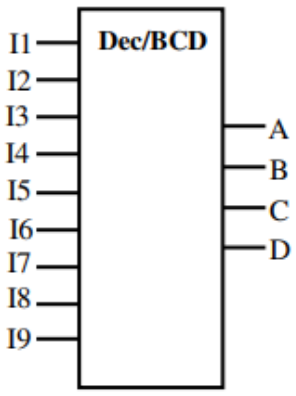


Las salidas codificadas, generalmente se usan para controlar un conjunto de 2^n dispositivos, suponiendo claro está que sólo uno de ellos está activo en cualquier momento. Sin embargo, cuando nos encontremos con que se deben controlar dispositivos que pueden estar activos al mismo tiempo, problema que se suelen encontrar los sistemas microprocesadores, es preciso usar un dispositivo que nos proporcione a la salida el código del dispositivo que tenga más alta prioridad.

Podemos ver una sencilla comparación.

En la siguiente figura se representa el diagrama lógico de un codificador completo de Decimal a BCD natural, junto a su tabla de funcionamiento.

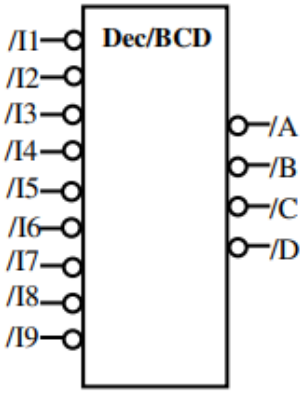
I1	I2	I3	I4	I5	I6	I7	I8	I9	A	B	C	D
0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1
0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	1	0	0	0	0	1	1	0
0	0	0	0	0	0	1	0	0	0	1	1	1
0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	1



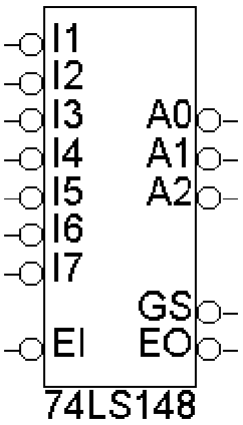
Por otro lado, la figura siguiente representa el diagrama lógico del circuito 74147,

que es un codificador de **prioridad** de Decimal a BCD natural; en la tabla de funcionamiento adjunta se puede notar la diferencia con el anterior.

/I1	/I2	/I3	/I4	/I5	/I6	/I7	/I8	/I9	/A	/B	/C	/D
X	X	X	X	X	X	X	X	0	0	1	1	0
X	X	X	X	X	X	X	0	1	0	1	1	1
X	X	X	X	X	X	0	1	1	1	0	0	0
X	X	X	X	X	0	1	1	1	1	0	0	1
X	X	X	X	0	1	1	1	1	1	0	1	0
X	X	X	0	1	1	1	1	1	1	0	1	1
X	X	0	1	1	1	1	1	1	1	1	0	0
X	0	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1



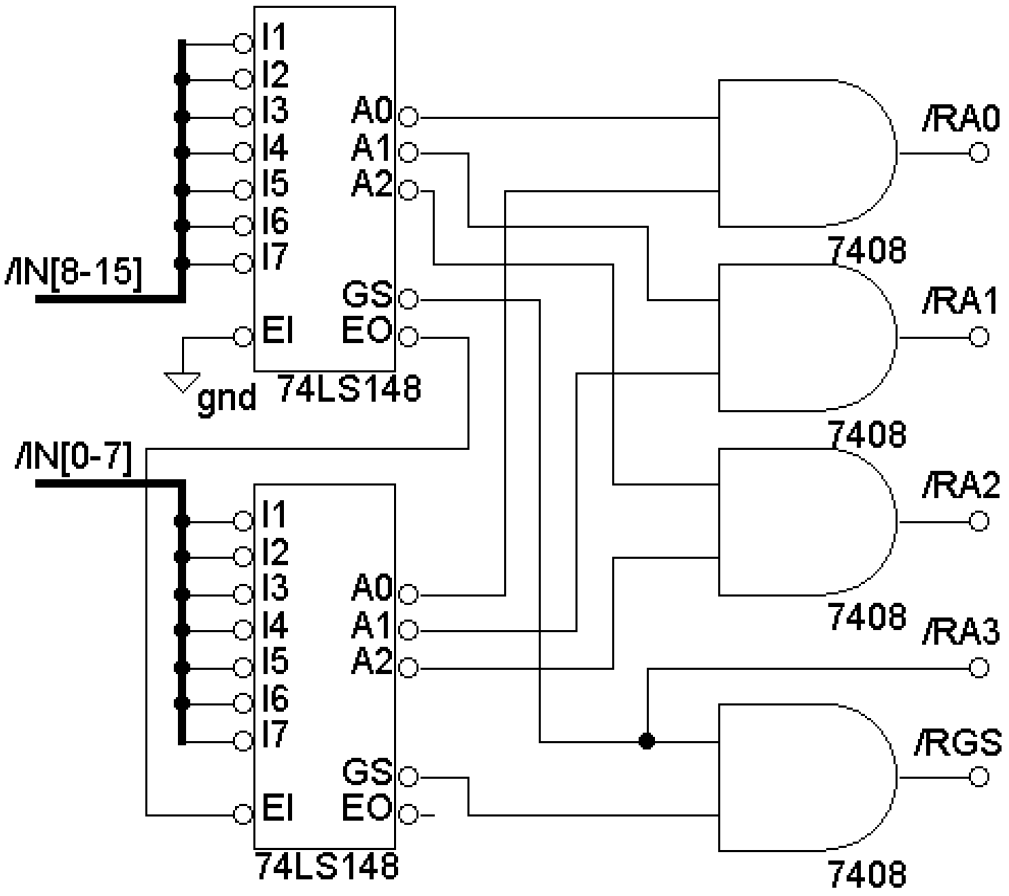
Cuando se trata de establecer la prioridad con mayor número de bits, es preciso recurrir a la asociación de codificadores. El siguiente diagrama muestra un codificador de prioridad de 16 líneas a 4, usando codificadores de prioridad 74148, de 8 a 3 líneas.



/EI: Habilitación

/GS: es 0 cuando el dispositivo está habilitado y una o más de sus entradas está activa

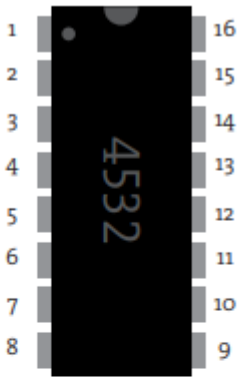
/EO: salida para habilitar otro decodificador de más baja prioridad



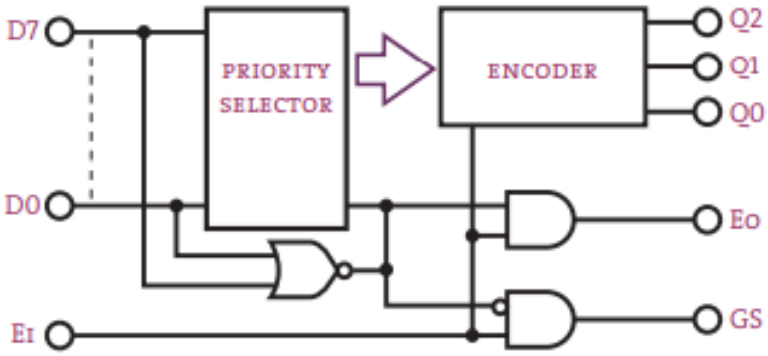
Aplicación a teclados alfanuméricos.

Un teclado alfanumérico se caracteriza, en términos de lógica de circuitos, por generar un código Johnson de tantos bits como teclas existan, y convertirlos en una salida binaria. Los codificadores comerciales muestran una tabla de verdad con el código de prioridad y las respectivas salidas binarias. La relación entradas – salidas es: para N salidas binarias, existen 2N entradas. Se muestra la tabla de verdad de un codificador CMOS de 8 Bits comercial.

CD4532B Diagrama en bloques, y Tabla de verdad.



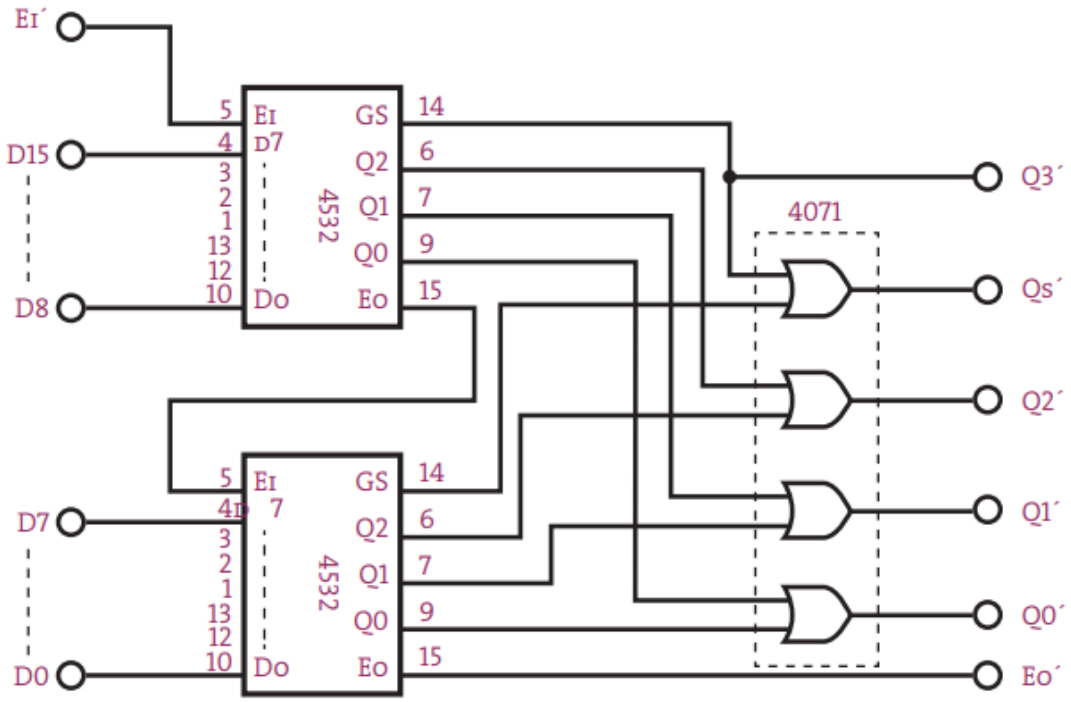
1-D4	16-VDD
2-D5	15-Eo
3-D6	14-Gs
4-D7	13-D3
5-EI	12-D2
6-Q2	11-D1
7-Q1	10-Do
8-Vss	9-Qo



INPUT									OUTPUT				
EI	D7	D6	D5	D4	D3	D2	D1	D0	GS	Q2	Q1	Q0	Eo
0	×	×	×	×	×	×	×	×	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	1
1	1	×	×	×	×	×	×	×	1	1	1	1	0
1	0	1	×	×	×	×	×	×	1	1	1	0	0
1	0	0	1	×	×	×	×	×	1	1	0	1	0
1	0	0	0	1	×	×	×	×	1	1	0	0	0
1	0	0	0	0	1	×	×	×	1	0	1	1	0
1	0	0	0	0	0	1	×	×	1	0	1	0	0
1	0	0	0	0	0	0	1	×	1	0	0	1	0
1	0	0	0	0	0	0	0	1	1	0	0	0	0

Las entradas D7 a D0 pueden corresponder a los botones del teclado. Cuando se desea multiplicar la cantidad de botones pueden conectarse codificadores en cascada respetando la relación entradas – salidas.

CD4532B Conexión en cascada.



Las salidas Q son normalmente conectadas a un microcontrolador que interpreta el valor alfanumérico, o enviadas a un codificador paralelo – serie para ser transmitido como dato a alguna terminal.

Codificador paralelo – serie.

Cuando se envían datos a un módulo de transmisión (Tx) (por ejemplo, un opto-transistor, o un módulo de radiofrecuencia) se lo hace a través de un único cable que contiene toda la información. Por esto, la información en paralelo que se construye al sintetizar un teclado debe nuevamente codificarse para poder ser transmitida.

Los codificadores paralelo – serie, están constituidos por varias etapas. Básicamente, un oscilador, un contador, un sincronizador, y lógicas de habilitación. Existen formas sencillas de implementarlos a través del uso de Flip Flops

mediante registros de desplazamientos. También existen codificadores integrados como los que se muestran en la ilustración. Este tipo de integrados suele contener, adicionalmente, entradas de direccionamiento que permiten construir diferentes dispositivos y controlarlos con uno único. En otras palabras, con un único módulo codificador, puedo controlar muchos módulos decodificadores.

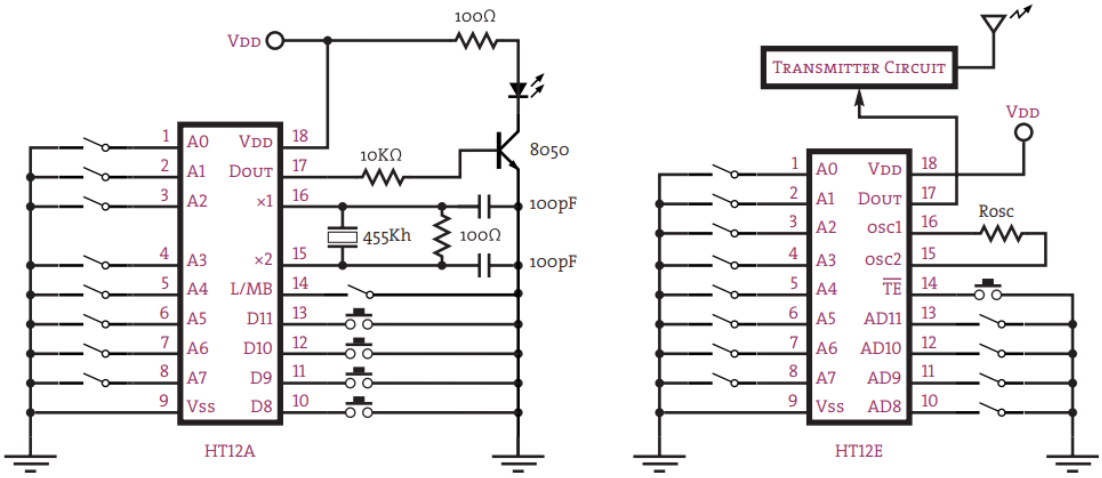


Ilustración 2 HT12A/E PINOUT y Diagrama de conexión.

El decodificador, no es otra cosa, que otro codificador. Si se trata de convertidores paralelo – serie, el decodificador propio será un convertidor serie – paralelo.

Existen ciertos cuidados que hay que tener a la hora de implementar un sistema codificador/decodificador. Los principales problemas aparecen en la sincronización de las frecuencias de reloj que, de no estar correctamente establecidas, la “traducción” no podrá realizarse correctamente. Otro inconveniente suele aparecer en los niveles de tensión: Las normas propias de la familia lógica deben ser respetadas pudiendo ocasionar, en caso contrario, la falta de interpretación del decodificador o la destrucción del mismo.

Se muestra un decodificador comercial complementario

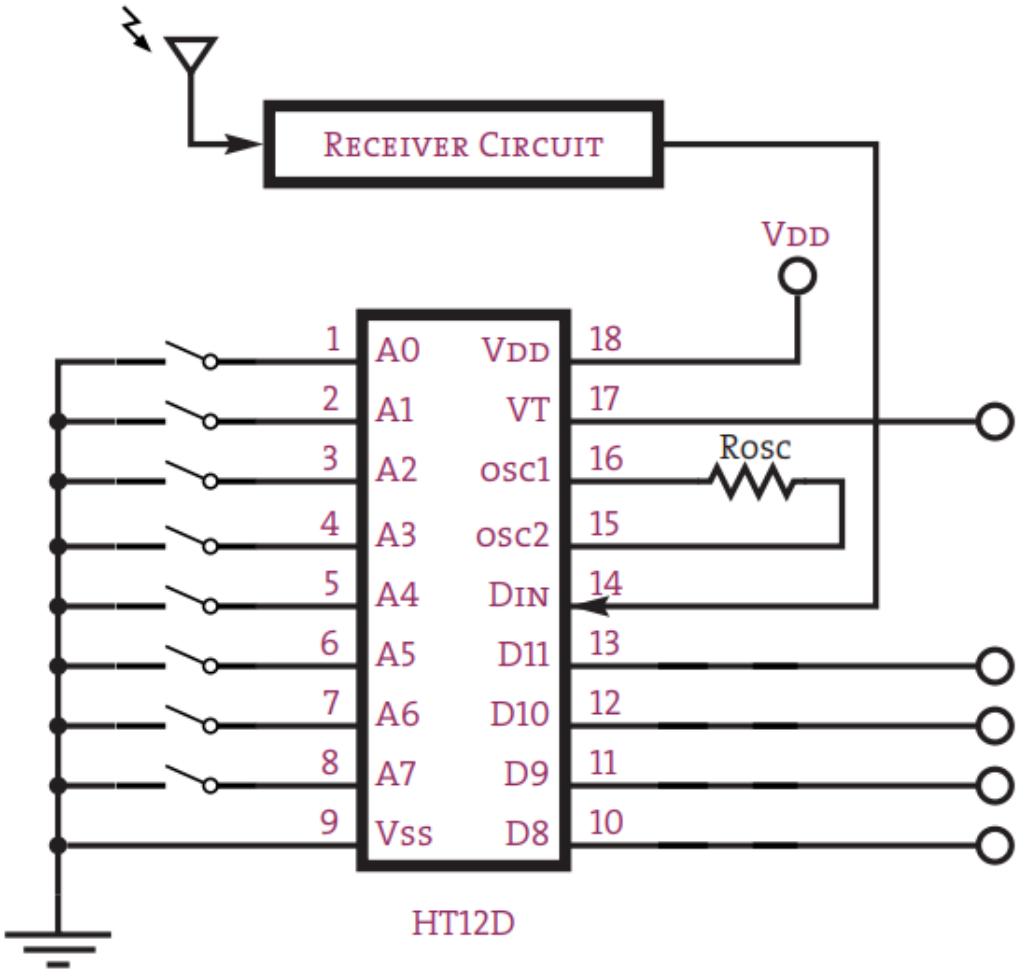


Ilustración 3 HT12D PINOUT y Diagrama de conexión.

Aplicación a control remoto.

Con los conceptos brevemente resumidos puede fácilmente extrapolarse la configuración necesaria para la construcción de un control remoto o control a distancia. Puede verse un diagrama en bloques de un control remoto y un módulo controlado por el mismo.

Ilustración 4 Diagrama en bloques de un sistema controlado a distancia.

Ventajas y desventajas

Ventajas:

- ✓ Reducción de líneas de transmisión.
- ✓ Diseño compacto.
- ✓ Aumento en la eficiencia del sistema.

Desventajas:

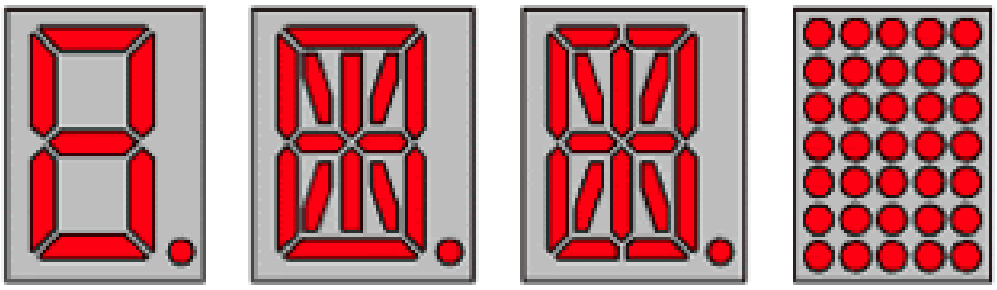
- ❖ Solo una entrada activa a la vez (salvo codificadores por prioridad).
- ❖ Puede requerir circuitos adicionales para manejar entradas inválidas.

Los codificadores son componentes esenciales en la electrónica. Su capacidad para convertir múltiples entradas en un formato binario compacto permite diseñar sistemas eficientes y organizados. El uso de codificadores mejora la comunicación y el control en una gran variedad de aplicaciones electrónicas.

3.3.5 Indicadores numéricos. (Display's).

Los **indicadores numéricos**, también conocidos como **Display's**, son dispositivos utilizados para visualizar información numérica (y en algunos casos alfanumérica) en sistemas electrónicos. Son muy comunes en relojes digitales, calculadoras, paneles de instrumentos y equipos electrónicos.

Es un componente que convierte señales eléctricas en una representación visual, generalmente en forma de dígitos del 0 al 9. Los tipos más comunes son los de **7 segmentos**, **14 segmentos**, **16 segmentos** y **Display's de matriz de puntos**.



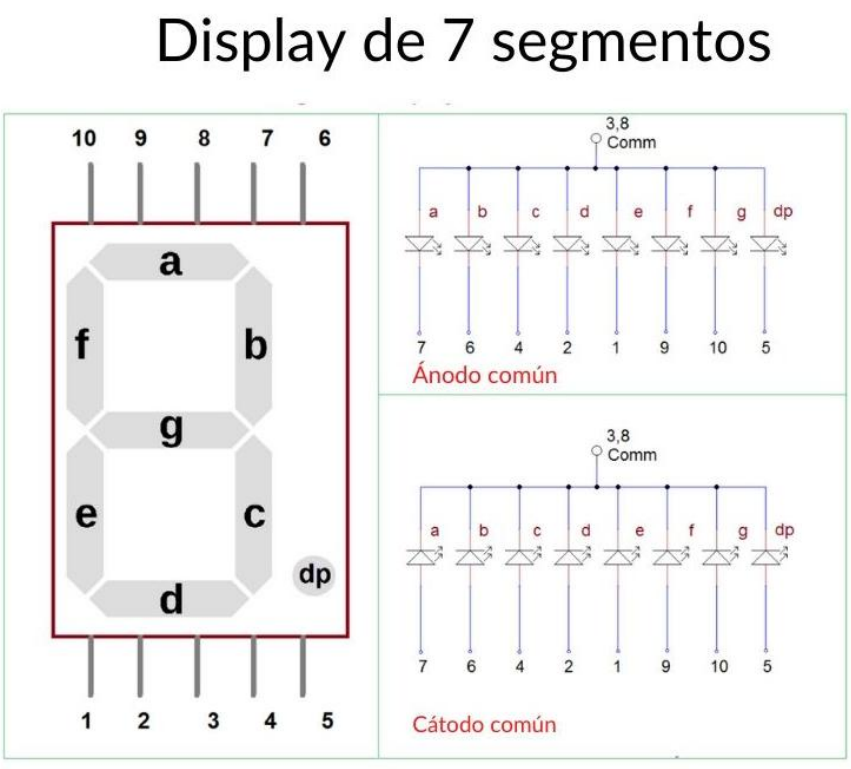
Los indicadores numéricos en electrónica digital son dispositivos que se utilizan para mostrar valores numéricos en formato decimal, binario, hexadecimal u otras bases numéricas.

Display de 7 segmentos

Los display's de 7 segmentos son dispositivos electrónicos de visualización utilizados como una forma fácil de representar numerales decimales y una alternativa a los display's de matriz de puntos más complejos. Los display's de 7 segmentos empezaron a usarse de forma generalizada como una forma popular para visualizar números. Se llaman display's de segmentos porque están compuestos por varios segmentos que se encienden y apagan para dar la apariencia del glifo deseado. Los segmentos generalmente son LED individuales o

cristales líquidos. Los display's de siete segmentos se emplean ampliamente en relojes digitales, medidores electrónicos, calculadoras básicas, pantallas de electrodomésticos, coches, y muchos otros dispositivos que muestran información numérica.

Existen dos tipos diferentes de display's de 7 segmentos: de ánodo común y de cátodo común. En el tipo de ánodo común, todos los ánodos del display están conectados a un pin común, generalmente la fuente de alimentación, y los LED se controlan mediante los cátodos con la conexión a tierra encendida y la potencia pagada. En el tipo de cátodo común, todos los cátodos están conectados a un pin común, en este caso generalmente la conexión a tierra, y los LED los controla el estado de los ánodos con la conexión a tierra apagada y la potencia encendida. Por consiguiente, un paquete de siete segmentos más punto decimal solo requiere nueve pines, aunque los productos comerciales generalmente contienen más pines para corresponderse con el estándar industrial de distribución de pines.



Display de 14 segmentos

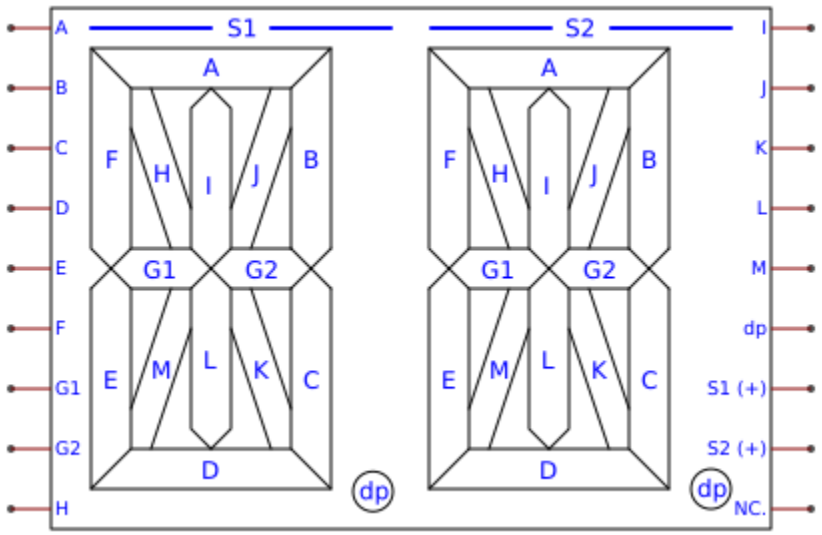
El display de 14 segmentos es un tipo de indicador visual utilizado para mostrar caracteres alfanuméricos. A diferencia del display de 7 segmentos que sólo permite representar dígitos, el de 14 segmentos puede mostrar tanto letras como números con mayor precisión.

Estructura del display de 14 segmentos

Este display está compuesto por 14 segmentos LED individuales, dispuestos de tal manera que permiten representar la mayoría de las letras del alfabeto latino, además de los números del 0 al 9. Los segmentos se etiquetan generalmente como: A, B, C, D, E, F, G1, G2, H, I, J, K, L y M.

Tipos de conexión

- **Ánodo común:** todos los ánodos se conectan a Vcc. Los segmentos se activan con nivel bajo (0).
- **Cátodo común:** todos los cátodos se conectan a tierra. Los segmentos se activan con nivel alto (1).



Display de 16 segmentos

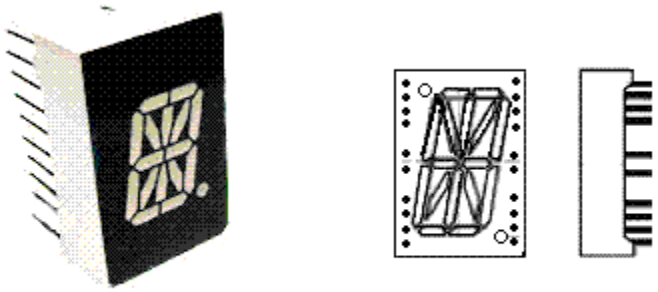
El Display 16 segmentos es una forma de representar caracteres en equipos electrónicos. Está compuesto de siete segmentos que se pueden encender o apagar individualmente. Cada segmento tiene la forma de una pequeña línea. Muchos equipos electrónicos proporcionan información al usuario mediante la utilización de señales luminosas, como la emisora sintonizada en un equipo de radio o la lectura de tensión en un voltímetro digital. Para representar las cifras numéricas se agrupan siete diodos en de segmentos. Estos diodos tienen conectados entre si todos los ánodos o cátodos según el tipo.

Un Display de este tipo está compuesto por siete u ocho leds de diferentes formas especiales y dispuestos sobre una base de manera que puedan representarse todos los símbolos numéricos y algunas letras. Los primeros siete segmentos son los encargados de formar el símbolo y con el octavo podemos encender y apagar el punto decimal. Cada uno de los segmentos que forman la pantalla están marcados con siete primeras letras del alfabeto (‘a’- ‘g’).

Los hay de dos tipos: ánodo y cátodo comunes.

En los de tipo de ánodo común, todos los ánodos de los segmentos están unidos internamente a una patilla común que debe ser conectada a potencial positivo (nivel “1”). El encendido de cada segmento individual se realiza aplicando potencial negativo (nivel “0”) por la patilla correspondiente a través de una resistencia que límite el paso de la corriente.

En los de tipo de cátodo común, todos los cátodos de los segmentos están unidos internamente a una patilla común que debe ser conectada a potencial negativo (nivel “0”). El encendido de cada segmento individual se realiza aplicando potencial positivo (nivel “1”) por la patilla correspondiente a través de una resistencia que límite el paso de la corriente.



(A) (B)

Display de matriz de puntos

El display de matriz de puntos es un dispositivo de salida visual utilizado para mostrar caracteres alfanuméricos, símbolos o gráficos simples. A diferencia de los display's de segmentos, ofrece una mayor flexibilidad en la representación de información visual.

Es una malla rectangular compuesta por puntos luminosos (LEDs) organizados en filas y columnas. Los tamaños comunes incluyen matrices de **5x7**, **8x8**, y **16x16**. Cada punto puede encenderse o apagarse individualmente para formar letras, números o gráficos.

Tipos de display's de matriz

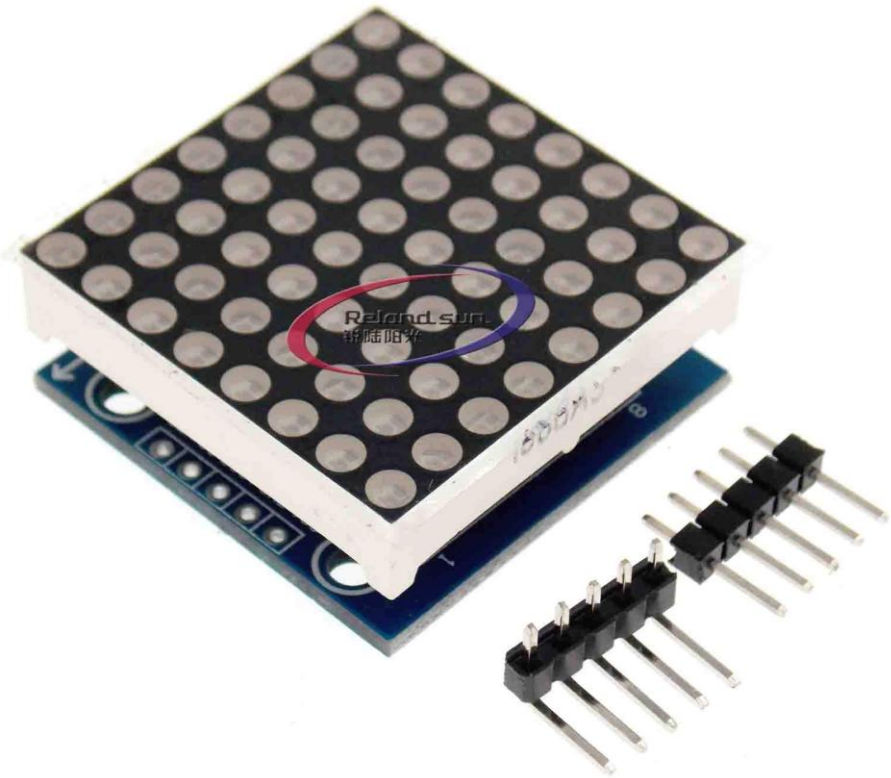
- **Matriz de 5x7:** usada para representar caracteres simples alfanuméricos.
- **Matriz de 8x8:** permite gráficos sencillos, desplazamiento de texto y mayor variedad de caracteres.
- **Matrices mayores (16x16 o más):** usadas en carteles electrónicos, paneles publicitarios y señalización LED.

Funcionamiento

Cada LED de la matriz está conectado en forma de filas y columnas. Para encender un LED específico, se activa su fila y su columna correspondiente.

La activación puede ser:

- **Estática:** todos los LEDs se controlan directamente.
- **Multiplexada:** los LEDs se activan por ciclos rápidamente para aparentar una imagen fija, reduciendo pines de control.
-



3.4 DISPOSITIVOS LÓGICOS PROGRAMABLES

Un PLD (Programmable Logic Device, Dispositivo lógico programable) es un componente electrónico empleado para la fabricación de circuitos digitales. A diferencia de las puertas lógicas un PLD tiene una función indefinida. Antes de que un PLD pueda ser usado en un circuito este puede ser programado.

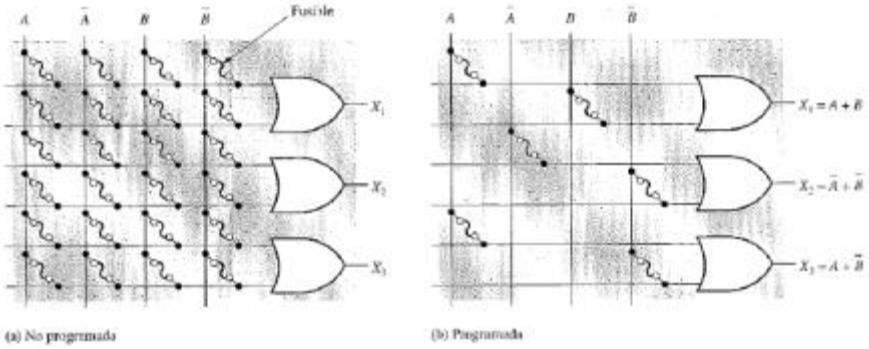
Un PLD está formado por una matriz de compuertas AND y puertas OR, que se pueden programar para conseguir funciones lógicas especificas. Existen cuatro tipos de dispositivos que se clasifican como PLD.

- PROM (Programmable Read Only Memory). Memoria programable de sólo lectura.
- PLA (Programmable Logic Array). Matriz lógica programable.
- PAL (Programmable Array Logic). Matriz lógica programable.
- GAL (Generic Array Logic). Matriz lógica generica.

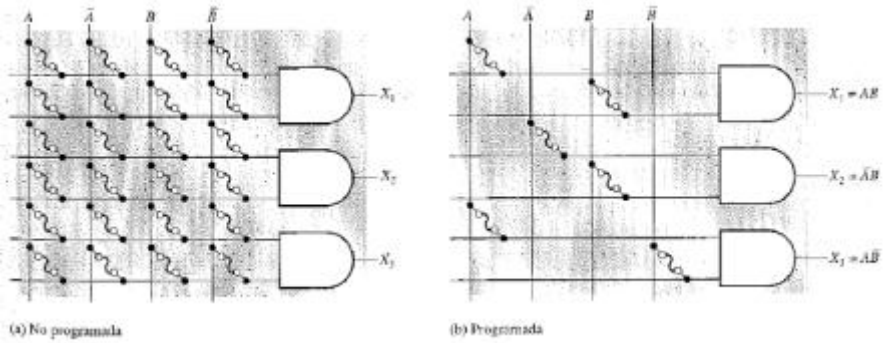
Todos los PLD están formados por matrices programables. Esencialmente, una matriz programable es una red de conductores distribuidos en filas y columnas con un fusible en cada punto de intesección. Las matrices pueden ser fijas o programables.

- Matriz OR. Esta formada por una serie de puertas OR conectadas a una matriz programable con fusibles en cada punto de intersección de una columna y una fila. La matriz se programa fundiendo los fusibles para eliminar las variables seleccionadas de las funciones de salida para un caso especifico. Para cada una de las entradas de una puerta OR sólo queda intacto un fusible que conecta la variable deseada en la entrada de la puerta.

Una vez que el fusible esta fundido, no se puede volver a conectar.



- Matriz AND. Este tipo de matriz esta formado por puertas AND conectadas a una matriz programable con fusibles en cada punto de intersección. al igual que la matriz OR la matriz AND se programa fundiendo los fusibles para eliminar las variables de la función salida. Para cada entrada de una puerta AND sólo queda intacto un fusible que conecta la variable deseada a la entrada de la puerta. Como para la matriz OR la matriz AND con fusibles se puede programar una única vez.



Los PLDs tienen varias ventajas. La primera es la habilidad de integración, que permite integrar una gran cantidad de funcionalidad en un solo chip. Los PLDs eliminan el uso de múltiples chips así como la inconveniencia y desconfianza de usar cableado externo. La segunda ventaja es el hecho de poder cambiar el diseño. Muchos PLDs permiten ser reprogramados o reconfigurados.

Existen dos ramas principales dentro de los dispositivos lógicos programables, la lógica programable de campo y la de fábrica. El término campo en este contexto implica que los dispositivos puedan ser programados en el “campo” del usuario,

mientras que la lógica de fábrica puede ser programada en la misma fábrica donde se construyen, de acuerdo a los requerimientos del cliente. En este caso, la tecnología de programación usa procesos irreversibles, por lo que solo es posible hacerlo una vez.

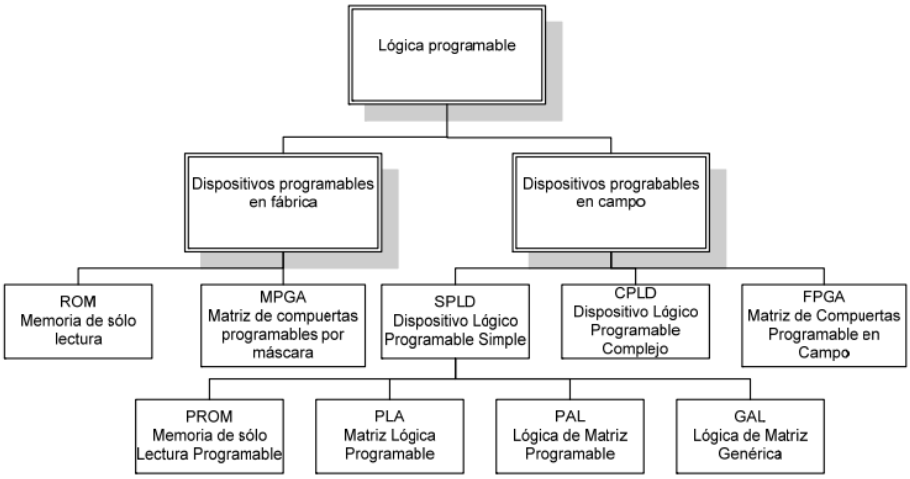


Ilustración 5 Árbol de clasificación de PLDs.

Algunos ejemplos de lógica programable de fábrica son los MPGAs y memorias de sólo lectura (ROMs). Las primeras generaciones de muchos dispositivos programables también fueron programados únicamente en fábrica. Las ROMs son consideradas como lógica programable porque, aunque fueron concebidas como unidades de memoria, también sirven para implementar cualquier circuitería combinacional. Los MPGAs son arreglos de compuertas tradicionales que requieren una máscara para ser diseñados. Los MPGAs son también llamados simplemente gate arrays y han sido la tecnología popular para crear ASICs (Application Specific Integrated Circuits).

Clasificación de los PLDs:

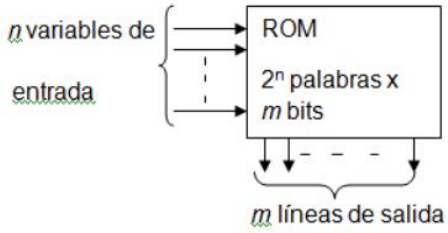
Tipo	*Características*	*Ejemplos*
PROM (Programmable Read-Only Memory)	*Memoria de solo lectura programable, se usa para funciones lógicas simples.*	*27C256*
PLA (Programmable Logic Array)	*Más flexible que la PROM, permite programar tanto la matriz AND como la OR.*	*82S100*
PAL (Programmable Array Logic)	*Solo la matriz AND es programable, la OR es fija. Son más rápidos que los PLA.*	*PAL16L8*
GAL (Generic Array Logic)	*Reprogramables varias veces, versión mejorada de PAL.*	*GAL22V10*
CPLD (Complex Programmable Logic Device)	*Integra múltiples bloques lógicos programables. Ideal para sistemas de lógica media.*	*XC9500 (Xilinx)*
FPGA (Field Programmable Gate Array)	*Dispositivos de alta capacidad y flexibilidad. Contienen miles de puertas lógicas.*	*Spartan-6, Artix-7 (Xilinx), Cyclone (Intel/Altera)*

ROM

Una ROM consiste en un arreglo de dispositivos semiconductores que están interconectados para almacenar de datos binarios. Una vez almacenada la información, puede ser leída cuando se requiera, pero no puede ser modificada bajo condiciones normales de operación.

Las ROMs tienen combinaciones de entradas, que generalmente son llamadas direcciones, y patrones de salidas, llamadas palabras. Una ROM que tiene n líneas de entrada y m líneas de salida contiene un arreglo de 2n palabras, cada una de m

bits de longitud. La dirección sirve para seleccionar una de las 2n palabras, por lo que cuando una combinación de entrada es aplicada a la ROM, el patrón de ceros y unos almacenados en la palabra correspondiente aparece en las líneas de salida.



n variables de entrada	m variables de salida
00...00	100...011
00...01	010...100
00...10	101...010
00...11	011...111
...	...
11...00	001...011
11...01	100...101
11...10	101...011
11...11	111...010

Ilustración 6 Ejemplo de una ROM y su tabla de verdad.

Una ROM consiste básicamente de un decodificador y un arreglo de memoria. Cuando un patrón de entrada se aplica a las entradas del decodificador, una de las 2n salidas de dicho decodificador se activa, seleccionando una de las palabras almacenadas en la memoria y se transfiere a las líneas de salida.

Los tipos básicos de ROM incluyen ROMs programables por máscara, ROMs programables por el usuario (PROMs), ROMs programables borrables (EPROMs), ROMs programables borrables eléctricamente (EEPROMs) y memorias flash. En las ROM programables por máscara, el arreglo de datos se almacena permanentemente a la hora de la manufactura. Este proceso resulta caro debido a que se requiere preparar una máscara especial, usada en la fabricación del dispositivo.

Las EPROMs se programan mediante un programador que provee pulsos de voltaje apropiados para almacenar cargas eléctricas en los lugares de memoria y se borran generalmente usando luz ultravioleta, mientras que las EEPROMs se borran por

medio de pulsos eléctricos. Las EEPROMs tienen un número limitado de veces que pueden ser borradas y reprogramadas, típicamente entre 100 y 1000 veces.

Las memorias flash son similares a las EEPROMs salvo que usan un mecanismo diferente de carga y almacenaje. Una ROM puede implementar cualquier circuito combinacional. Si las salidas de todas las combinaciones de entradas son almacenadas en la ROM, pueden ser buscadas (“looked up” en inglés) en la tabla de verdad almacenada. Por esto, el método que emplea una ROM es también conocido como look-up table (LUT)

PLAs

Un arreglo lógico programable (PLA) realiza la misma función que una ROM. Un PLA con n entradas y m salidas puede realizar m funciones de n variables. La organización interna del PLA difiere de la de la ROM, el decodificador se reemplaza por un arreglo de ANDs que realiza los términos producto seleccionados de las variables de entrada. El arreglo de ORs realiza la operación OR a los términos producto necesarios para formar las funciones de salida.

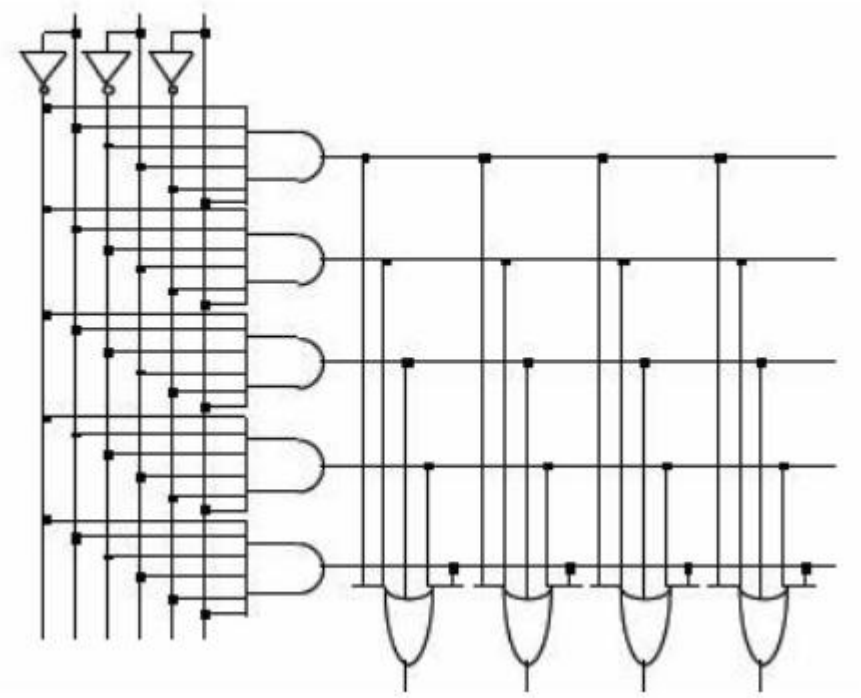


Ilustración 7 Diagrama de un PLA.

Para determinar la información que se graba en un PLA se realiza una tabla para PLA, que es diferente a una tabla de verdad para una ROM. En una tabla de verdad cada fila representa un mintérmino, por lo tanto exactamente una fila se selecciona por cada combinación de valores de entrada, mientras que en cada fila de una tabla para PLA representa un término producto general. Por lo tanto cero, una o más filas se seleccionan por cada combinación de valores de entrada. Para determinar el valor de la función para cierta combinación de entrada, a los valores de la función en las filas seleccionadas de la tabla para PLA se les debe aplicar la operación OR.

término producto	entradas			salidas			
	A	B	C	F0	F1	F2	F3
A'B'	0	0	_	1	0	1	0
AC'	1	_	0	1	1	0	0
B	_	1	_	0	1	0	1
BC'	_	1	0	0	0	1	0
AC	1	_	1	0	0	0	1

Ilustración 8 Ejemplo de una tabla PLA.

PALs

El PAL (Programmable Array Logic) es un caso especial del PLA en el que el arreglo de ANDs es programable y el de ORs es fijo. Sus estructuras son iguales, pero el hecho de que únicamente el arreglo de ANDs sea programable hace más barato y fácil de programar el PAL en comparación con el PLA.

Cuando se diseña con PALs se deben simplificar las ecuaciones lógicas para que quepan en uno (o más) de los PALs existentes. Los términos AND no se pueden compartir entre dos o más compuertas OR, por lo tanto cada función puede ser simplificada por si misma sin importar los otros términos. En cualquier PAL el número de términos AND que alimentan cada compuerta OR es fijo y limitado. Los PALs también pueden contener flip flops D con sus entradas provenientes del arreglo combinacional. Estos se llaman PALs secuenciales. Los PALs fueron desapareciendo con el desarrollo de otros dispositivos, como GALs, CPLDs y FPGAs.

Dispositivos lógicos programables/ Generic Array Logic

Conforme avanzaba la tecnología de circuitos integrados, una gran variedad de dispositivos lógicos programables aparecieron. Los PALs tradicionales no son

reprogramables, sin embargo existen ahora PALs borrables y reprogramables con tecnología flash. A veces, a éstos se les llama PLDs.

El 22CEV10 es un PLD con tecnología CMOS borrable eléctricamente que puede ser usado para hacer tanto circuitos combinacionales como secuenciales.

Además de los arreglos AND y OR, la mayoría te los PLDs tienen algún tipo de macrobloque que contiene multiplexores y otros bloques programables adicionales. Estos PLDs se llaman de acuerdo a sus capacidades de entrada/ salida. Por ejemplo, el 22CEV10 tiene 12 pines de entrada más 10 pines que se pueden programar como entrada o salida (22 en total). Contiene también 10 flop flops D y 10 compuertas OR. Cada compuerta OR dirige una macrocelda lógica de salida. Cada macrocelda contiene uno de los 10 flip flops D, los cuales comparten un reloj común, un reset asícrono de entrada, y un preset síncrono de entrada.

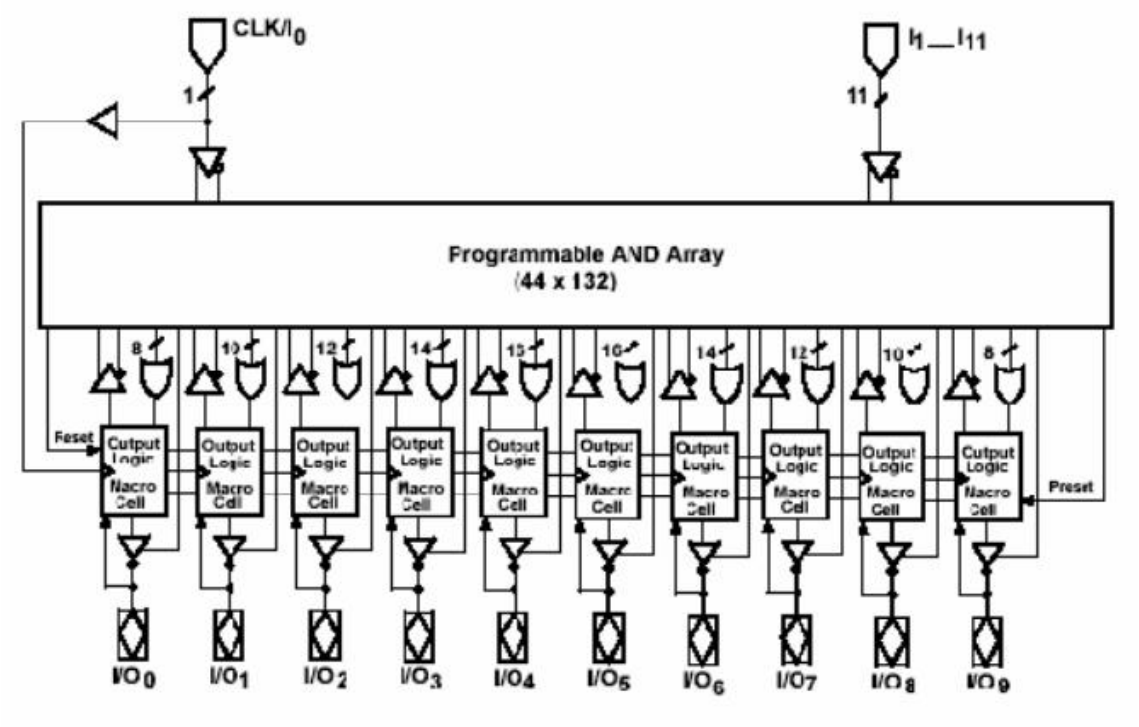


Ilustración 9 Esquema de una GAL 22V10.

CPLDs

Los avances en tecnología han hecho posible la creación de circuitos integrados programables equivalentes a varios PLDs en el mismo chip. A estos circuitos integrados se les llaman dispositivos lógicos programables complejos (CPLDs por sus siglas en inglés).

Un CPLD es un circuito integrado que consiste en un número de bloques lógicos parecidos a un PAL, incluyendo además una matriz programable de interconexiones entre estos bloques. Algunos CPLDs se basan en la arquitectura del PAL, en cuyo caso cada macrocelda contiene un flip flop y una compuerta OR, cuyas entradas están asociadas a un arreglo de compuertas AND fijo, mientras que los CPLDs que se basan en PLAs cada salida de compuertas AND en un bloque se puede conectar a la entrada de cualquier compuerta OR en ese bloque.

FPGAs

Los FPGAs son circuitos integrados que contienen un arreglo de bloques lógicos idénticos con interconexiones programables, en los que el usuario puede programar tanto las funciones realizadas por cada bloque lógico como las conexiones entre bloques.

Los FPGAs tienen varias ventajas con respecto a MPGAs. Un arreglo de compuertas tradicional puede ser usado para implementar cualquier circuito, pero sólo se puede programar en fábrica ya que se requiere hacer una máscara específica para un circuito en particular y el tiempo de diseño para un circuito integrado basado en arreglo de compuertas es de algunos meses. Por otro lado, los FPGAs son productos comerciales, el tiempo de manufactura se puede reducir de meses a algunas horas cambiando de MPGAs a FPGAs. De la misma forma, se vuelve más fácil y más barato corregir errores en los diseños. A volúmenes no tan altos, los FPGAs son más baratos que los MPGAs.

El interior de los FPGAs contiene típicamente tres elementos programables: los bloques lógicos, los bloques de entrada/ salida y las interconexiones. Se considera que los bloques de entrada/ salida se encuentran en la periferia del circuito integrado, éstos conectan las señales lógicas a los pines del chip. Los bloques lógicos se encuentran distribuidos dentro del FPGA y el espacio entre ellos se usa para mandar conexiones entre bloques.

La programabilidad de campo se logra por los elementos que pueden ser reconfigurables por el usuario. Los bloques lógicos se crean usando multiplexores, look-up tables y arreglos de compuertas AND-OR o NAND-NAND, y cualquiera de estas cosas puede ser programada (o configurada) por el usuario.

El trabajo con PLDs proporciona: facilidad de diseño, prestaciones, fiabilidad, economía y seguridad.

- *Facilidad de diseño. Las herramientas de soporte al diseño con PLDs facilitan enormemente este proceso. Estas nuevas herramientas permiten expresar la lógica de los circuitos utilizando formas variadas de entrada tales como; ecuaciones, tablas de verdad, procedimientos para máquinas de estados, esquemas, etc. La simulación digital posibilita la depuración de los diseños antes de la programación de los dispositivos. Todo el equipo de diseño se reduce a un software de bajo coste que corre en un PC, y a un programador.*
- Prestaciones. *Los PLDs TTL que hay en el mercado tienen tiempos de conmutación tan rápidos como los circuitos integrados de función fija más veloces. Los PLDs ECL son todavía más rápidos. Sin embargo, el incremento de velocidad obtenido con los dispositivos CMOS, que ya han igualado o superado en prestaciones a los dispositivos TTL, está provocando el abandono de la tecnología bipolar por parte de los fabricantes. En cuanto al consumo de potencia, los PLDs generalmente consumen menos que el conjunto de chips a los que reemplazan.*

- Fiabilidad. *Cuanto más complejo es un circuito, más probabilidades hay de que alguna de sus partes falle. Puesto que los PLDs reducen el número de chips en los sistemas, la probabilidad de un fallo disminuye. Los circuitos impresos con menor densidad de CI son más fáciles de construir y más fiables. Las fuentes de ruido también se reducen.*
- Economía. *En este apartado, hay aspectos que resultan difíciles de cuantificar. Por ejemplo, los costes de pérdida de mercado por una introducción tardía de un producto. Otros son más claros, por ejemplo, la reducción del área de las placas de circuito impreso obtenida gracias a que cada PLD sustituye a varios circuitos integrados de función fija. Muchas veces se consigue reducir el número de placas de circuito impreso economizándose en conectores. La reducción de artículos en almacén también aporta ventajas económicas.*
- Seguridad. *Los PLDs tienen fusibles de seguridad que impiden la lectura de los dispositivos programados, protegiendo los diseños frente a copias.*

3.5 LENGUAJES DE DESCRIPCIÓN DE HARDWARE (HDL)

Un lenguaje de descripción de hardware es un lenguaje de programación que se utiliza para definir la estructura, diseño y operación de circuitos electrónicos digitales. Los lenguajes de descripción de hardware hacen posible una descripción formal de un circuito digital posibilitando su análisis y su simulación.

Los HDL pueden ser usados para describir especificaciones de hardware, es decir, un programa escrito en un HDL hace posible que el diseñador de hardware pueda modelar y simular un componente electrónico antes de que este sea construido físicamente. Es esta caracteristica lo que hace que a veces los HDL se vean como lenguajes de programación convencionales, cuando en realidad se deberían clasificar más precisamente como lenguajes de modelado.

Desde el punto de vista práctico una gran ventaja de los HDLs está en que es posible inferir el conjunto de operaciones lógicas y el circuito equivalente necesarios para realizar la función del programa. Esto permite saltar desde el ámbito de la simulación de software al de la implementación real del hardware sobre circuitos lógicos tales como los ASIC o las FPGA.

Los lenguajes de descripción de hardware más populares son Verilog y VHDL . Se utilizan ampliamente junto con FPGAs , dispositivos digitales diseñados específicamente para facilitar la creación de circuitos digitales personalizados.

Los lenguajes de descripción de hardware permiten describir un circuito utilizando palabras y símbolos, y luego el software de desarrollo puede convertir esa descripción textual en datos de configuración que se cargan en el FPGA para implementar la funcionalidad deseada.

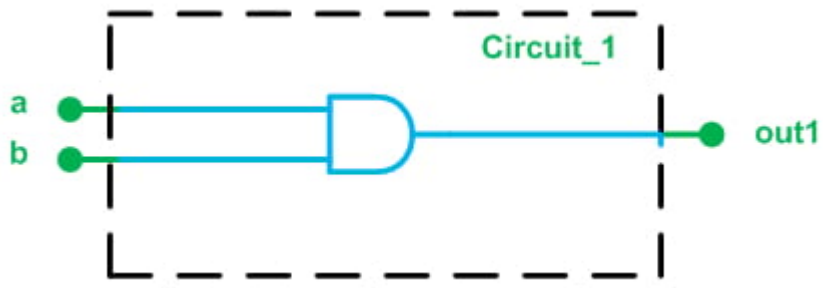
¿Qué se puede hacer con HDL?

- Describir **comportamiento** (cómo debe funcionar un sistema digital).
- Describir **estructura** (cómo se conectan los componentes).
- Simular y verificar el diseño antes de implementarlo en hardware.
- Sintetizar el código para convertirlo en un diseño físico para PLDs.

ejemplo de código HDL:

```
1 entity Circuit_1 is

2    Port ( a : in  STD_LOGIC;

3         b : in STD_LOGIC;

4         out1 : out STD_LOGIC);

5 end Circuit_1;

----------------------------------------------------

6 architecture Behavioral of Circuit_1 is

8 begin

9 out1 <= ( a and b );

10 end Behavioral;
```

El comportamiento eléctrico descrito por este código se puede representar visualmente de la siguiente manera:



Estos diferentes niveles de abstracción añaden etapas en el ciclo de desarrollo utilizando HDLs.

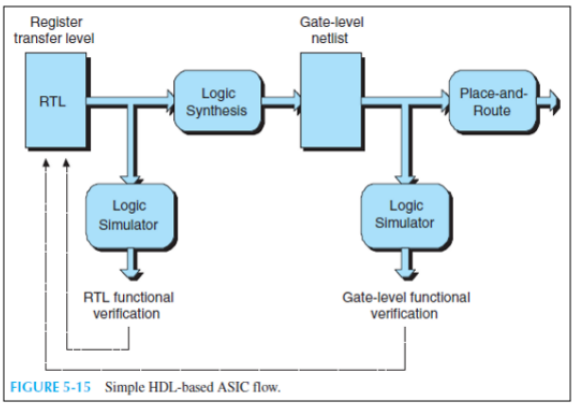


FIGURE 5-15 Simple HDL-based ASIC flow.

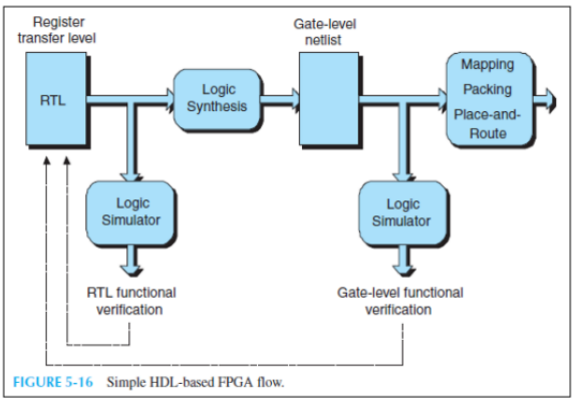


FIGURE 5-16 Simple HDL-based FPGA flow.

Principales lenguajes HDL:

Lenguaje	Características	Usos comunes
VHDL (VHSIC Hardware Description Language)	Lenguaje muy estructurado y fuertemente tipado. Orientado a sistemas grandes y críticos.	Aeronáutica, defensa, telecomunicaciones.
Verilog	Más simple y similar a C. Muy usado en la industria por su sintaxis más accesible.	Diseño de chips, sistemas embebidos.
SystemVerilog	Extensión de Verilog con mejoras en modelado y verificación.	Simulaciones complejas, verificación formal.

Los principales lenguajes de descripción de hardware utilizados hoy en día son:

- Verilog
- VHDL

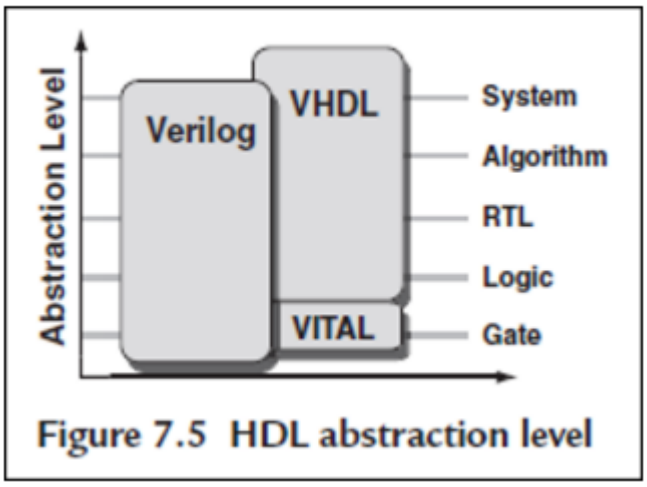


Figure 7.5 HDL abstraction level

- Ponen énfasis en la simulación y utilizan una semántica de eventos discretos que ignora las porciones ociosas del circuito para incrementar la performance de la simulación.
- Describen a los sistemas como una jerarquía de bloques cuyas conexiones son modeladas explícitamente.
- Describen procesos concurrentes de manera procedural.

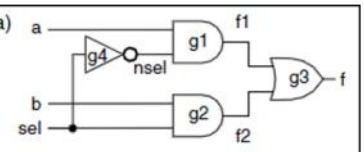
Verilog

- Verilog se desarrolló como un lenguaje para la simulación de hardware usando eventos discretos.
- Es un lenguaje de modelado y de especificación.
- No todos los constructores de Verilog pueden trasladarse al circuito (semántica no determinística en función de un simulador orientado a eventos).
- Tres estilos o formas de modelado:

  - Estructural
  - Funcional

- de Comportamiento

(a)



(b)

```
module mux(f,a,b,sel);
output f;
input a, b, sel;

and g1(f1, a, nsel),
    g2(f2, b, sel);
or  g3(f, f1, f2);
not g4(nsel, sel);

endmodule
```

(c)

```
module mux(f,a,b,sel);
output f;
input a, b, sel;

assign f = sel ? a : b;

endmodule
```

(d)

```
primitive
   mux(f,a,b,sel);
output f;
input a, b, sel;

table
  1?0 : 1;
  0?0 : 0;
  ?11 : 1;
  ?01 : 0;
  11? : 1;
  00? : 0;
endtable
endprimitive
```

FIGURE 7.8 Verilog examples. (a) A multiplexer circuit, (b) the multiplexer described as a Verilog structural model, (c) the multiplexer described using a continuous assignment, (d) a user-defined primitive for the multiplexer, (e) the multiplexer described with imperative code, (f) a testbench for the multiplexer.

(e)
```
module mux(f,a,b,sel);
output f;
input a, b, sel;
reg f;

always @(a or b or sel)
  if (sel) f = a;
  else f = b;

endmodule
```

(f)
```
module testbench;
  reg a, b, sel;
  wire f;

  mux dut(f, a, b, sel);

  initial begin
    $display("a,b,sel -> f");
    $monitor($time,,
             "%b%b%b -> ",
             a, b, sel, f);
    a = 0; b = 0 ; sel = 0;
    #10 a = 1;
    #10 sel = 1;
    #10 b = 1;
    #10 sel = 0;
  end
endmodule
```

FIGURE 7.8 Verilog examples. (a) A multiplexer circuit, (b) the multiplexer described as a Verilog structural model, (c) the multiplexer described using a continuous assignment, (d) a user-defined primitive for the multiplexer, (e) the multiplexer described with imperative code, (f) a testbench for the multiplexer.

Un programa Verilog se compone de módulos (en el sentido de los lenguajes de programación imperativos), cada uno de los cuales:

- Define interfaces con entradas y salidas
- Contiene una o más

- Instancias de otros módulos
- Asignaciones continuas

Comunicación entre módulos:

- Vía wires (modelo estructural): se calcula su valor continuamente.
- Vía registros (modelo de comportamiento): son elementos de memoria.

Cuatro valores: 0, 1, X (don’t care) y Z (tri-state) (tipado insuficiente en ciertas aplicaciones) Es un lenguaje amplio con algunas características poco usadas hoy en día:

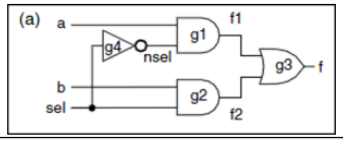
- Modelado a nivel de conmutación de transistores: existen simuladores continuos más precisos para estos casos (como SPICE).
- Delays: se utilizan poco a raíz de los métodos de análisis temporales estáticos.

- System verilog es un estándar que incorpora mejoras para modelado a nivel de sistemas.

VHDL

- VHDL significa VHSIC (Very High Speed Integrated Circuit) Hardware Description Language.
- Fue diseñado como un lenguaje flexible de modelado de sistemas digitales.
- Clara separación entre interfaces (entitites) e implementaciones (architectures). Admite distintas implementaciones para una misma interface (distintos niveles de abstracción).

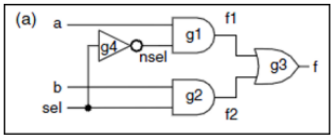
```
(b) architecture arch1 of mux2 is
      signal cc, ai, bi : Bit; -- internal signals

      component Inverter -- component interface
        port (a:in Bit; y: out Bit);
      end component;

      component AndGate
        port (a1, a2:in Bit; y: out Bit);
      end component;

      component OrGate
        port (a1, a2:in Bit; y: out Bit);
      end component;

    begin
      I1: Inverter port map(a => c, y => cc); -- by name
      A1: AndGate port map(a, cc, ai); -- by position
      A2: AndGate port map(a1 => b, a2 => c, y => bi);
      O1: OrGate port map(a1 => ai, a2 => bi, y => d);
    end;
```




```
(a) entity mux2 is
      port (a: in Bit; b: in Bit; c: in Bit; d: out Bit);
    end mux2;
```

FIGURE 7.9 VHDL examples. Compare with Figure 7.8. (a) The entity declaration for the multiplexer, which defines its interface, (b) a structural description of the multiplexer from Figure 7.8(a), (c) a dataflow description with one equation per gate, (d) an imperative behavioral description.


```
(a) entity mux2 is
      port (a: in Bit; b: in Bit; c: in Bit; d: out Bit);
    end mux2;
```




```
(c) architecture arch2 of mux2 is
      signal cc, ai, bi : Bit;
    begin
      cc <= not c;
      ai <= a and c;
      bi <= b and cc;
      d <= ai or bi;
    end;
```

```
(d) architecture arch3 of mux2 is
    begin
      process(a, b, c) -- sensitivity list
      begin
        if c = '1' then
          d <= a;
        else
          d <= b;
        end if;
      end process;
    end;
```

FIGURE 7.9 VHDL examples. Compare with Figure 7.8. (a) The entity declaration for the multiplexer, which defines its interface, (b) a structural description of the multiplexer from Figure 7.8(a), (c) a dataflow description with one equation per gate, (d) an imperative behavioral description.

VHDL soporta

- Modelado estructural
- Modelado de flujo de datos (funcional)
- Modelado de comportamiento
- Algunas características de modelado de sistemas
- Tiene un sistema de tipos más elaborado que Verilog.

```
type address is range 16#0000# to 16#FFFF#;

type Bit is ('0', '1');
type FourV is ('0', '1', 'X', 'Z');
type State is (Reset, Running, Halted);
```

Diseño con HDL

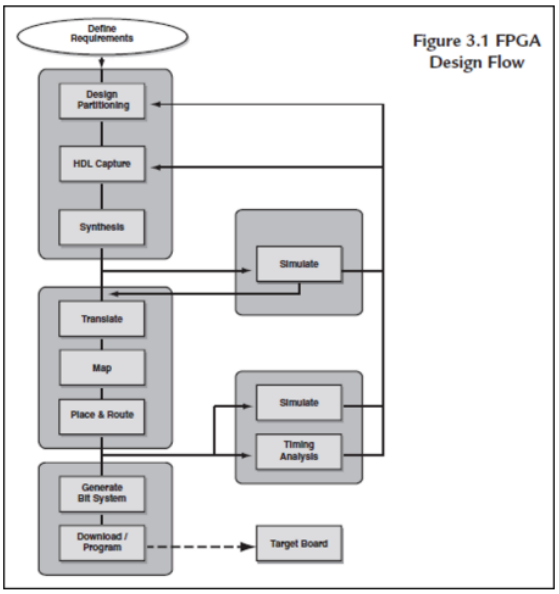


Figure 3.1 FPGA Design Flow

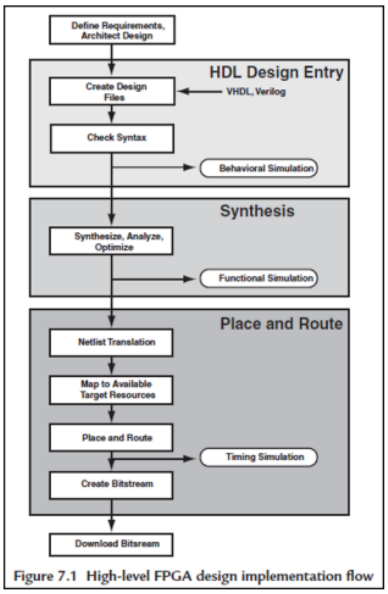


Figure 7.1 High-level FPGA design implementation flow

- Inicialmente se captura el dise˜no: HDL (Verilog o VHDL), Matlab/Simulink, C/C++, etc.
- Usando las representaciones algorítmicas de los HDL, se puede simular el comportamiento del sistema de manera independiente de cómo será implementado.
- Luego se sintetizan las especificaciones a RTL.
- La representación funcional (RTL) permite simular de manera independiente del dispositivo sobre el que se implementará el diseño.
- El proceso de síntesis, a partir de la especificación en RTL, genera una netlist (a nivel de compuertas o LUTs).

Síntesis de comportamiento

El proceso de síntesis se da a varios niveles y considera información cada vez más cercana a la implementación.

```
Xr = (A * C) – (B * D);
Xi = (A * D) + (B * C);
```

```
if rising_edge(CLK) then
  PROD1 = A * C;
  PROD2 = B * D;
  PROD3 = A * D;
  PROD4 = B * C;
  Xr = PROD1 – PROD2;
  Xi = PROD3 + PROD4;
end if;
```

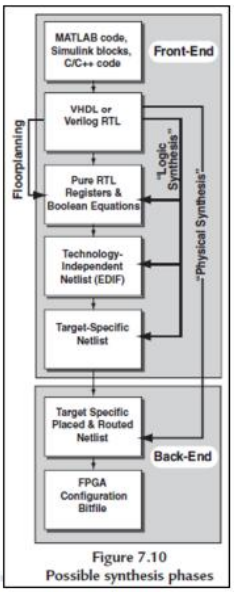


Figure 7.10
Possible synthesis phases

Síntesis lógica

```
if rising_edge(CLK) then
  PROD1 = A * C;
  PROD2 = B * D;
  PROD3 = A * D;
  PROD4 = B * C;
  Xr = PROD1 – PROD2;
  Xi = PROD3 + PROD4;
end if;
```

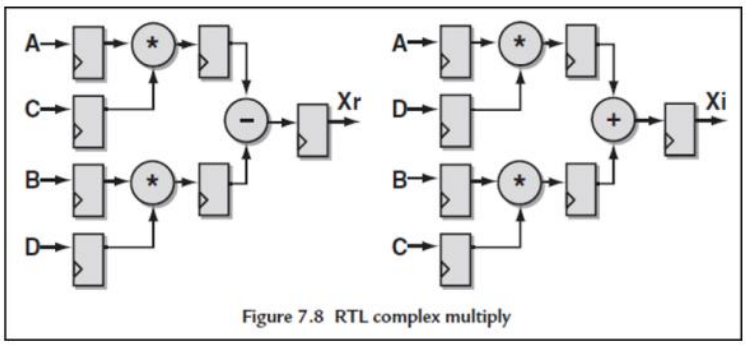
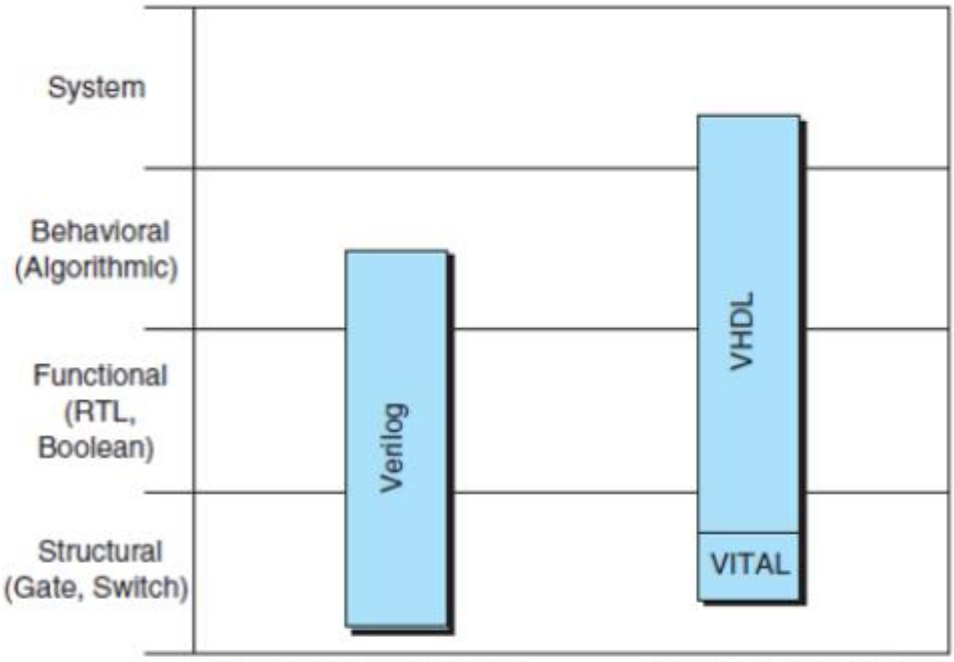


Figure 7.8 RTL complex multiply

Verilog vs VHDL



– Relatively easy to learn	– Relatively difficult to learn
– Fixed data types	– Abstract data types
– Interpreted constructs	– Compiled constructs
– Good gate-level timing	– Less good gate-level timing
– Limited design reusability	– Good design reusability
– Limited design management	– Good design management
– No structure replication	– Supports structure replication

FIGURE 5-19 Levels of abstraction (Verilog versus VHDL).

CONCLUSIÓN

En síntesis, hemos evidenciado que el estudio y diseño de circuitos combinacionales son fundamentales para el avance de la electrónica digital, ya que sientan las bases para implementar funciones complejas a partir de componentes lógicos sencillos. La aplicación de técnicas de simplificación, tales como el uso de mapas de Karnaugh o el método de Quine-McCluskey, junto con el uso de herramientas de simulación, nos permite mejorar la eficiencia y reducir errores en el proceso de diseño. De igual forma, la incorporación de PLD y HDL no solo facilita la realización de prototipos, sino que también nos abre las puertas hacia la integración de sistemas más robustos y adaptables en diversas aplicaciones tecnológicas. Este enfoque metodológico refuerza la importancia del rigor en la ingeniería electrónica y nos incentiva a continuar explorando y optimizando los diseños en el ámbito del circuito digital.

BIBLIOGRAFÍAS

[1] “Dispositivos lógicos programables (PLD)”. Electrónica Digital. Accedido el 7 de abril de 2025. [En línea]. Disponible: https://ecadigitaliiequipo7.wordpress.com/2010/03/08/dispositivos-logicos-programables-pld/

[2] “Dispositivos lógicos programables (PLD)”. Electrónica Digital. Accedido el 7 de abril de 2025. [En línea]. Disponible: https://ecadigitaliiequipo7.wordpress.com/2010/03/08/dispositivos-logicos-programables-pld/

[3] “Â¿QuÃ© es un HDL?” Tu Mejor Maestro. Accedido el 7 de abril de 2025. [En línea]. Disponible: https://tumejormaestro.com/pag/intro/HDL.html

[4] Accedido el 7 de abril de 2025. [En línea]. Disponible: https://mexico.newark.com/display-seven-segment-display-technology?srsltid=AfmBOor54wbw3cF3GP1sPAakTpo51JhQVThDM1MnxTcph6oaULfqXV4m

[5] “Que es un multiplexor, como funciona y que tipos existen”. Ingeniería Mecafenix. Accedido el 7 de abril de 2025. [En línea]. Disponible: https://www.ingmecafenix.com/electronica/componentes/multiplexor/#google_vignette

[6] “La guía definitiva para Mux y Demux: comprensión de los multiplexores y demultiplexores”. fibermall.com. Accedido el 7 de abril de 2025. [En línea]. Disponible: https://www.fibermall.com/es/blog/demux-mux.htm?srsltid=AfmBOorgLGfEbwn1CrlCNEBeRM_i0-UVd1qYA-x93IrwZUugRpHYA4fI

[7] M. M. Mano, "Fundamentos de diseño lógico y de computadoras," 3ª ed., Prentice Hall, 2003. [En línea]. Disponible en: https://gc.scalahed.com/recursos/files/r161r/w24792w/Morris.pdf. [Accedido: 07-abr-2025].

[8] L. Entrena, C. López, M. García, y E. San Millán, "Circuitos combinacionales," Universidad Carlos III de Madrid, 2008. [En línea]. Disponible en:

https://ocw.uc3m.es/pluginfile.php/2302/mod_page/content/18/Tema3_Circuitos_Combinacionales.pdf. [Accedido: 07-abr-2025].

LISTA DE COTEJO INVESTIGACION

ELECTRÓNICA DIGITAL.

Nombre del estudiante: QUINO CAIXBA PERLA JOSELIN.

Tema: Circuitos Combinacionales.

Desarrollo de temas	20	20
Entrega en tiempo y forma	10	10
Claridad en la información	10	10
Total	40 %	40 %

TEMA: CIRCUITOS COMBINACIONALES

INSTITUTO TECNOLÓGICO SUPERIOR DE SAN ANDRÉS TUXTLA

División Ingeniería Mecatrónica IMCT-2010-229

Periodo: Febrero – Junio 2025 Grupo: 611A

PRACTICA DE LA UNIDAD

ELECTRÓNICA DIGITAL

Docente:

DR. JOSÉ ÁNGEL NIEVES VASQUEZ

UNIDAD III

CIRCUITOS COMBINACIONALES

Presenta:

Juan José Jiménez Reyes	**221U0541**
Juan José Marcial Fiscal	**221U0547**
Miguel de Jesús Polito Cerón	**221U0552**
Perla Joselin Quino Caixba	**221U0555**
Rocio Teoba Herrera	**221U0562**

San Andrés Tuxtla Veracruz **10 de abril de 2025**

ÍNDICE

INTRODUCCIÓN

El álgebra de Boole es una herramienta fundamental para el análisis y diseño de sistemas digitales. Esta rama de las matemáticas permite trabajar con variables binarias que únicamente pueden adoptar los valores de 0 y 1, lo cual resulta esencial para la representación de estados lógicos en circuitos electrónicos. El desarrollo de esta teoría fue realizado por George Boole en el año 1854, y posteriormente fue aplicada al diseño de circuitos por Claude Shannon en 1938, lo que marcó un avance importante en el campo de la electrónica digital.

La presente práctica tiene como finalidad comprobar experimentalmente algunos de los postulados principales del álgebra de Boole mediante el uso de compuertas OR. Para ello, se utilizan componentes básicos como una protoboard, interruptores, LEDs, resistencias y circuitos integrados del tipo 74LS32 (compuertas OR) y 74LS04 (compuertas NOT). Se analiza el comportamiento lógico de configuraciones específicas de entrada y se observa el resultado en la salida del circuito, verificando así la validez de los postulados mediante la elaboración de tablas de verdad.

OBJETIVO

El objetivo de esta práctica es verificar experimentalmente los postulados fundamentales del álgebra de Boole mediante la implementación y análisis de circuitos digitales con compuertas OR. A través del uso de componentes electrónicos como protoboard, interruptores, LEDs, resistencias y circuitos integrados 74LS32 (compuertas OR) y 74LS04 (compuertas NOT), se analizará el comportamiento lógico de distintas configuraciones de entrada y se validará la corrección de los resultados obtenidos mediante la elaboración de tablas de verdad.

MATERIALES UTILIZADOS

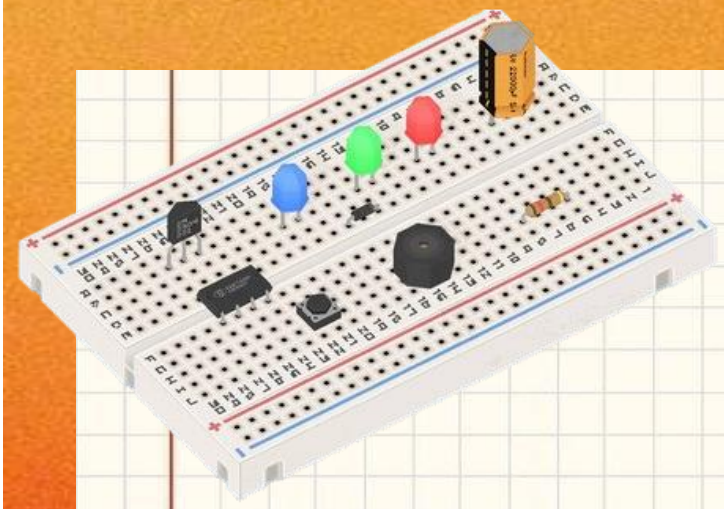
Materiales utilizados para llevar a cabo la PRACTICA 3 y 4 de Electrónica Digital:

1. Tablilla de pruebas.
2. Fuente de alimentación de 6 volts.
3. Switch deslizable de 4 posiciones.
4. Alambre para protoboards.
5. Circuito integrado TTL 4 compuertas OR de 2 entradas cada una.
6. Resistencia de 220 Ω.
7. LED de 5 mm.

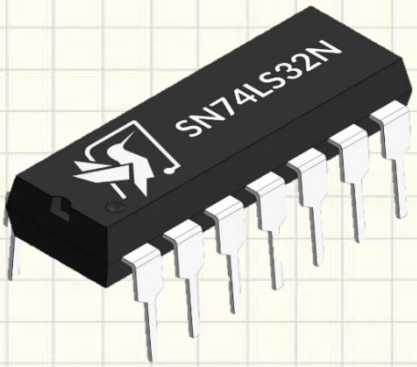
SOFTWARE

Software utilizado para realizar una simulación de prueba:

1. Proteus 8 Professional



PRÁCTICA 3



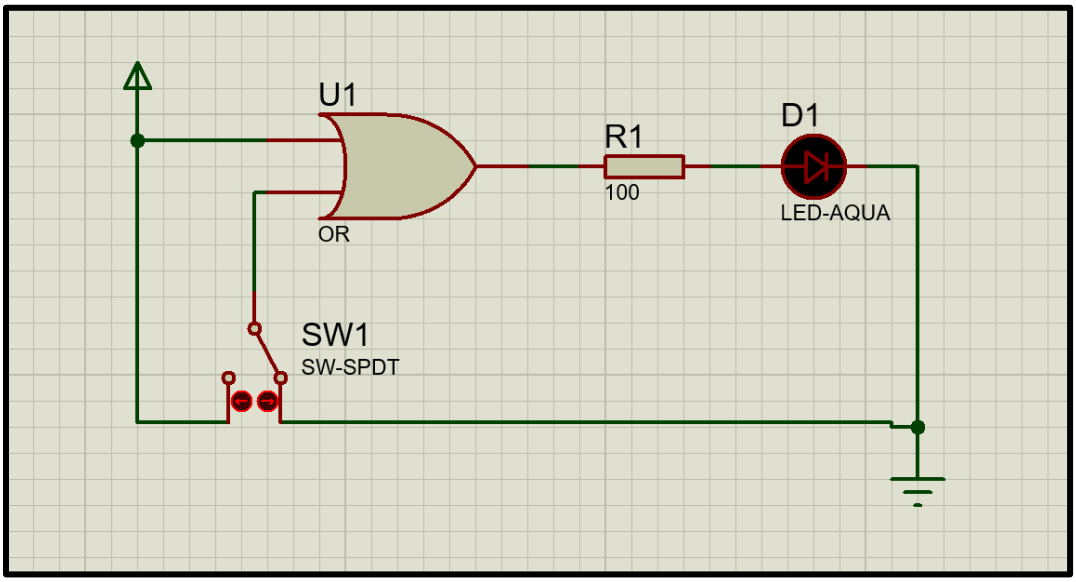
SIMULACIÓN DE LA PRACTICA

Se llevó a cabo la simulación de la PRACTICA 3 mediante el uso del software "Proteus 8 Professional", con el propósito de contrastar sus resultados con los obtenidos en la práctica experimental.

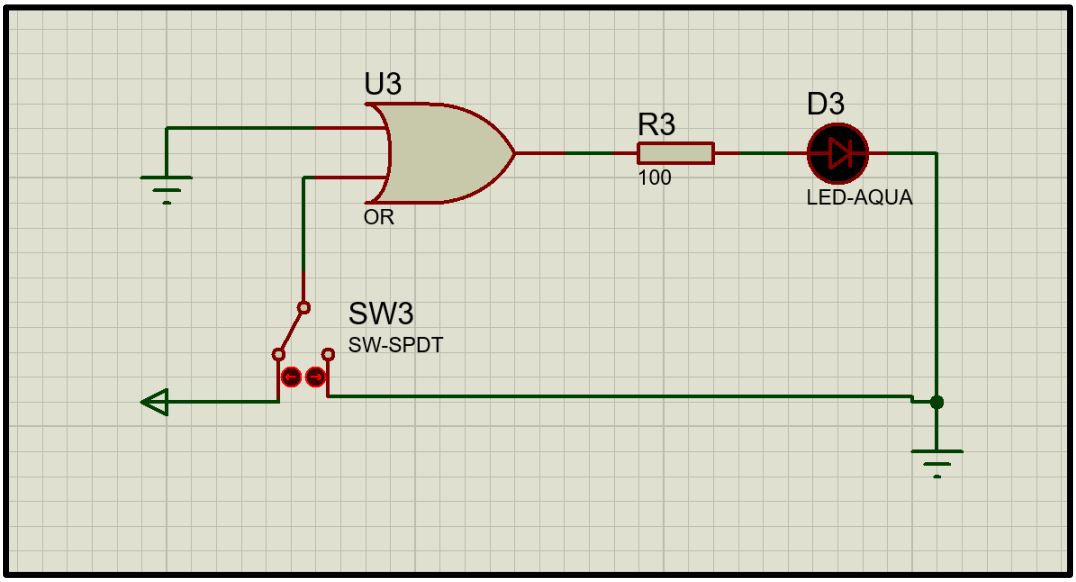
I. REALIZAR CONEXIONES.

Se establecen las conexiones necesarias para que los componentes funcionen correctamente.

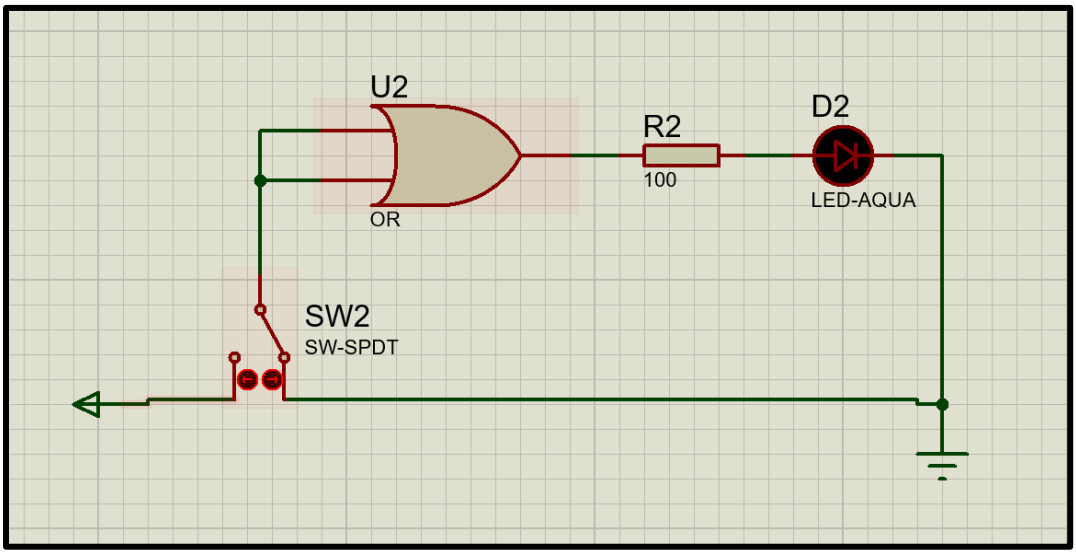
Conexones del primer postulado del algebra de Boole:



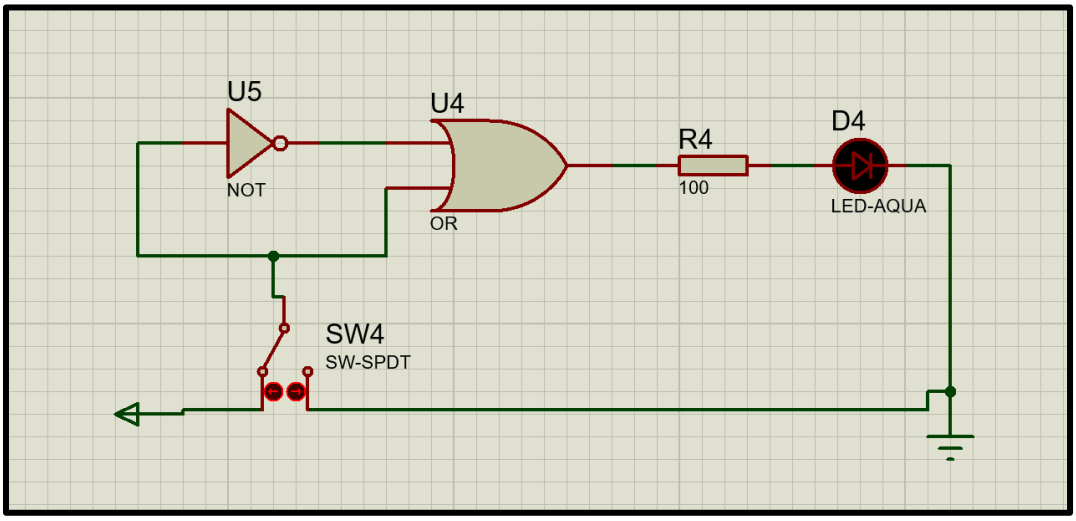
Conexones del segundo postulado del algebra de Boole:



Conexones del tercer postulado del algebra de Boole:



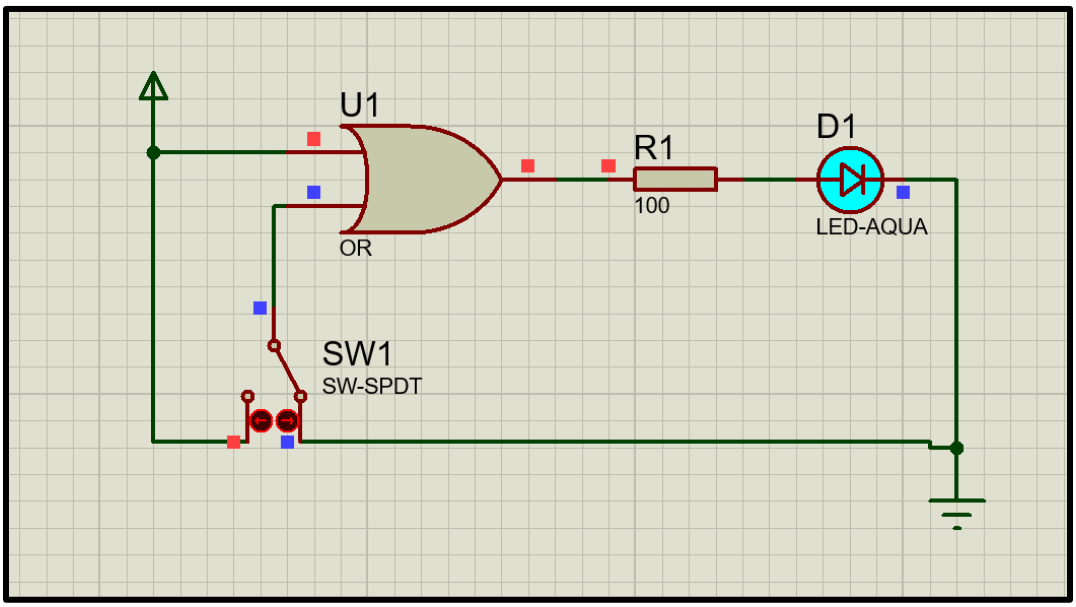
Conexones del cuarto postulado del algebra de Boole:



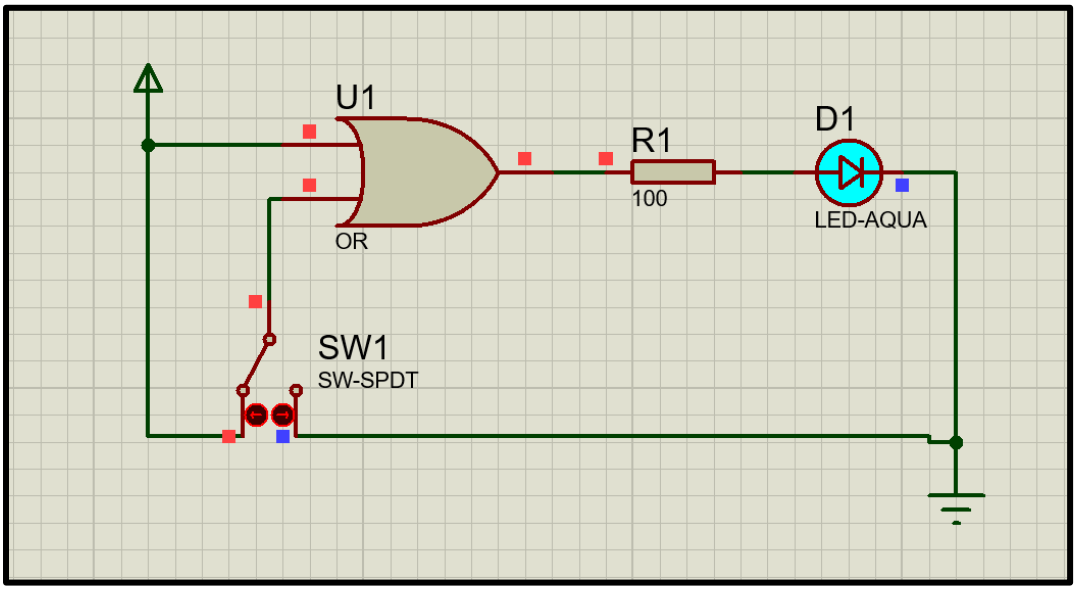
II. COMPROBACIÓN DE LOS POSTULADOS PRINCIPALES DEL ALGEBRA BOOLE.

Primer Postulado:

El circuito incorpora una compuerta OR, que tiene la propiedad de que su salida es 1 (alta) si al menos una de sus entradas es 1.



Esto se relaciona directamente con a + 1 = 1: si una de las entradas de la compuerta es 1 (en este caso, conectado directamente a un voltaje alto o por medio de una configuración que lo garantice), entonces la salida de la OR siempre será 1, independientemente del valor de la otra entrada a.

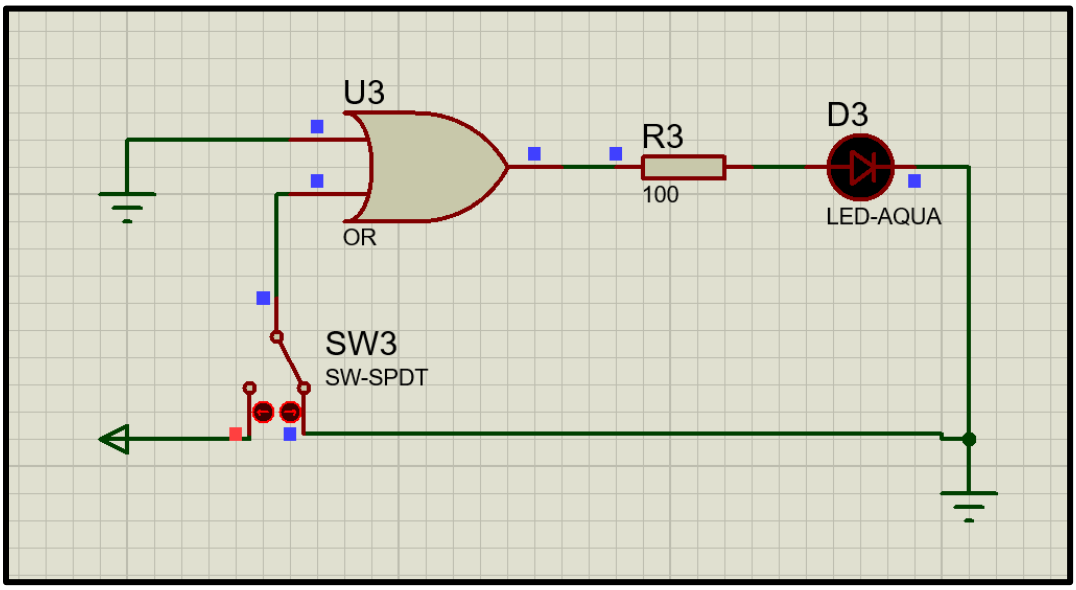


Al utilizar la propiedad a + 1 = 1, el diseño asegura que la salida no cambiará a 0 si al menos un terminal está forzado a un estado lógico alto, lo cual es útil para sistemas de control donde se desea una respuesta predecible (por ejemplo,

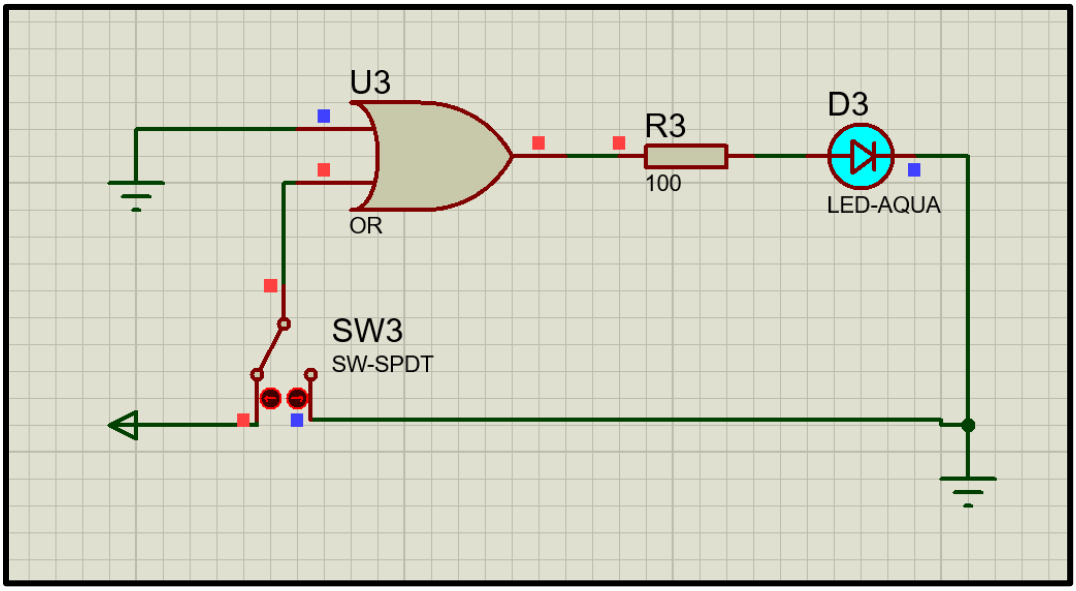
mantener el LED encendido cuando una condición de "encendido" se debe mantener activa).

Segundo Postulado:

Una de las entradas de la OR se conecta a una fuente de voltaje alto (1), la cual puede provenir de un terminal forzado o de una configuración interna que garantice el nivel alto. La otra entrada es la variable a, la cual puede cambiar entre 0 y 1 según las condiciones del sistema o las señales de control.



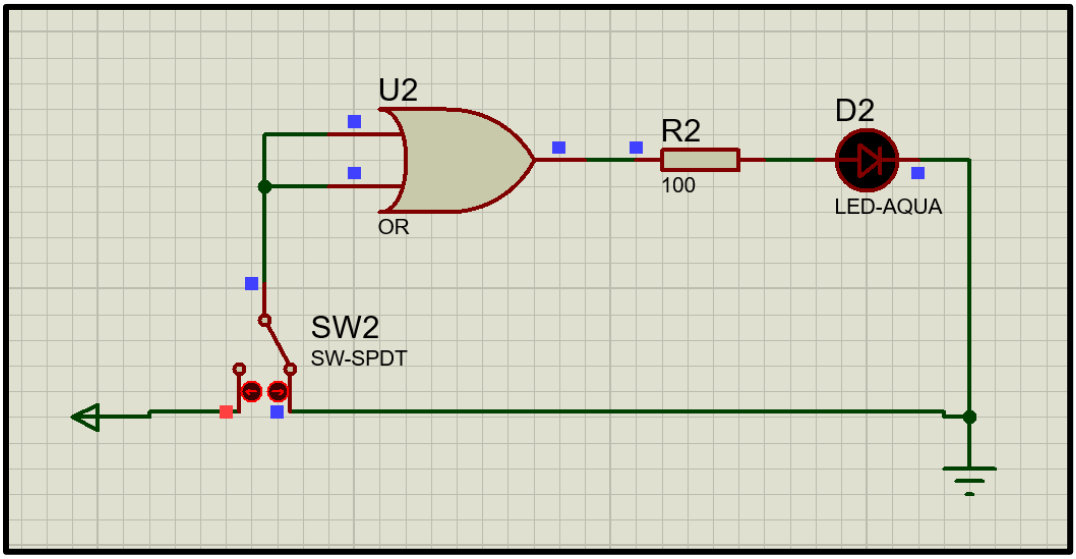
Debido a la propiedad a + 1 = 1, independientemente del valor de aaa, la salida de la compuerta siempre será 1 cuando al menos una entrada esté conectada a un nivel alto.

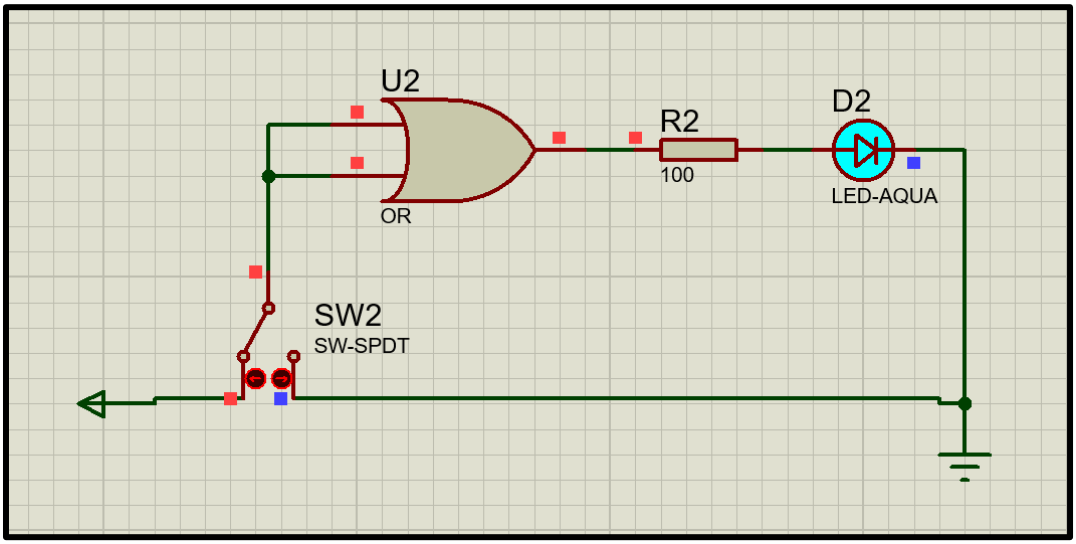


Esto significa que el circuito asegura que la salida se mantenga alta, lo que resulta en un comportamiento predecible y consistente, muy útil en aplicaciones de control.

Tercer Postulado:

Según el álgebra de Boole, cuando se hace la operación OR de una misma variable consigo misma (a + a), el resultado siempre es igual a dicha variable (a). En otras palabras, la compuerta OR no “suma” lógicamente nada adicional, pues las dos entradas son idénticas.

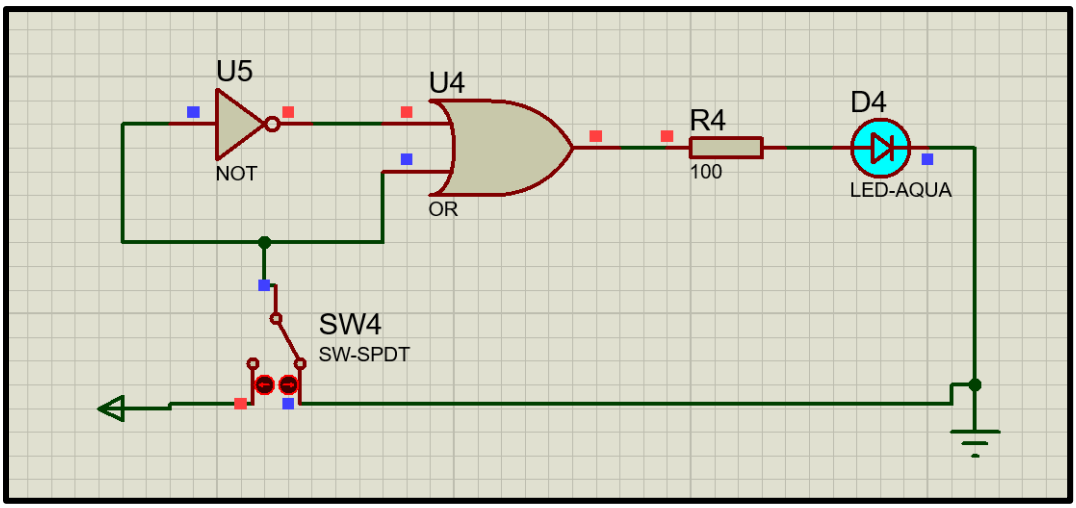




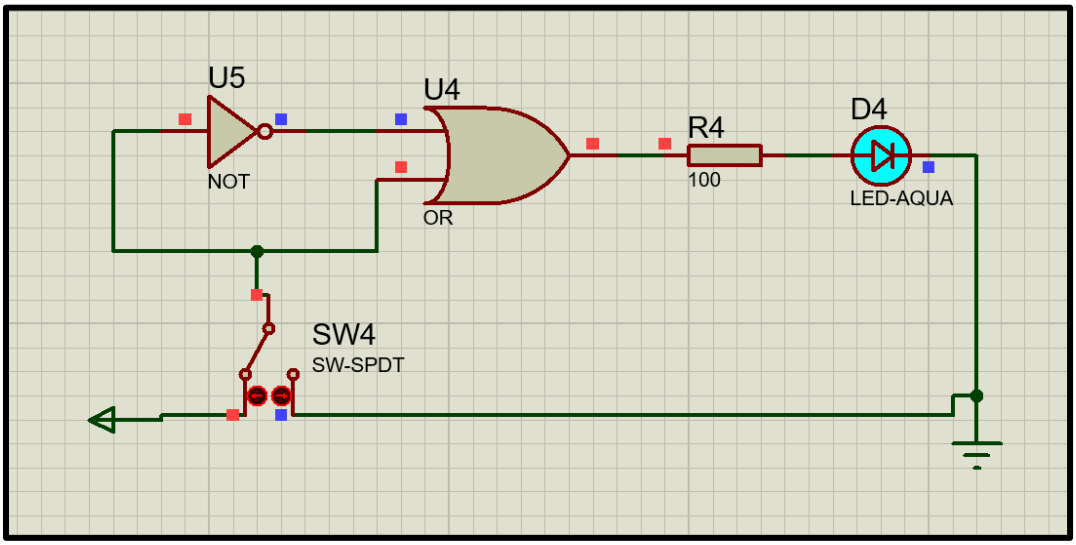
En **conclusión**, la conexión de las dos entradas de la compuerta OR a la misma señal a demuestra la ley booleana a + a = a: la salida del circuito seguirá el estado de la variable a sin alterarlo. El resistor y el LED simplemente sirven como indicador para comprobar si la salida se encuentra en alto o en bajo.

Cuarto Postulado:

El circuito incorpora una compuerta OR, que tiene la propiedad de que su salida es 1 (alta) si al menos una de sus entradas es 1. Esto se relaciona directamente con el postulado del álgebra booleana: $\boldsymbol{a} + \bar{\boldsymbol{a}} = \mathbf{1}$. En este caso, una de las entradas de la compuerta OR recibe directamente el valor de la variable lógica **a** , mientras que la otra entrada recibe su complemento, generado por la compuerta NOT.



Esto se refleja esencialmente en el LED conectado a la salida: **permanece encendido constantemente**, independientemente de la posición del interruptor, demostrando que la expresión lógica nunca produce un 0 en la salida.



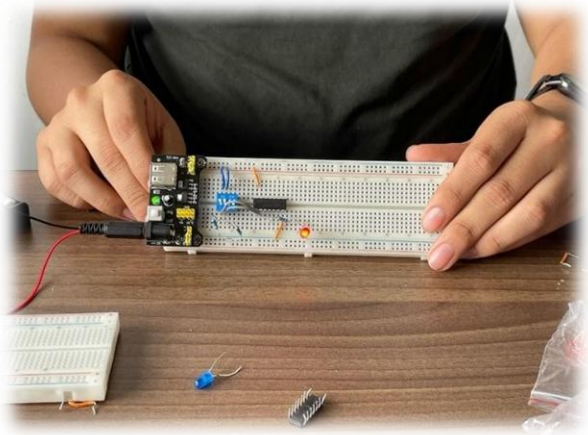
Al utilizar la propiedad $\boldsymbol{a + \overline{a} = 1}$, el diseño asegura que la salida no cambiará a 0 en ningún caso. Este tipo de configuración es útil en sistemas de control donde se desea una **respuesta predecible** y **constante**, como, por ejemplo, mantener encendido un indicador (LED) cuando se requiere garantizar una condición lógica activa sin importar los cambios en la variable de entrada.

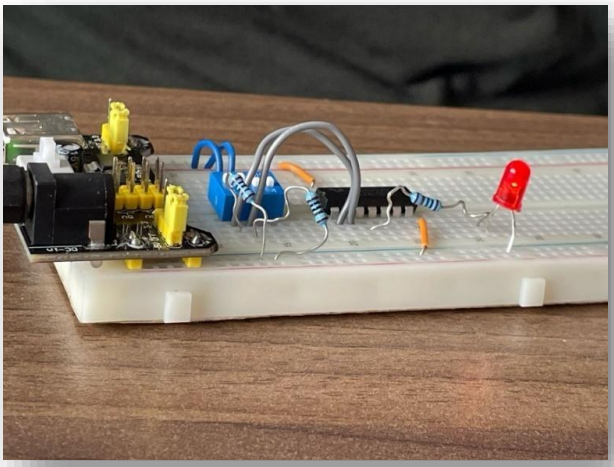
PRÁCTICA FÍSICA

A continuación, se presenta una redacción en la que se describen dos circuitos electrónicos reales, uno que ejemplifica el postulado a + 1 = 1 y otro que ejemplifica a + 0 = a, utilizando una implementación física.

1. Circuito Físico para, a + 1 = 1.

En este circuito se utiliza una tablilla de pruebas (protoboard) y una compuerta lógica OR con dos entradas. La configuración física se realiza de la siguiente manera:



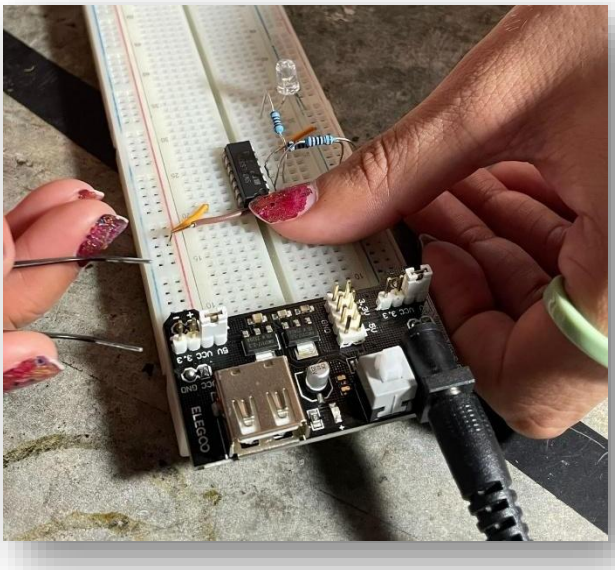


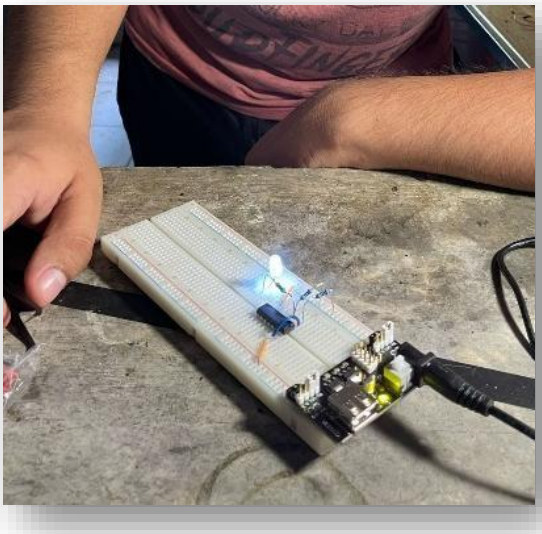
Funcionamiento físico:

Debido a que uno de los terminales de la OR está físicamente atado a +5 V (nivel 1), la compuerta genera una salida en estado alto sin importar el valor de la señal variable aa. Esto se evidencia con el encendido constante del LED, confirmando que en la práctica se cumple el postulado, a + 1 = 1.

2. Circuito Físico para, a + 0 = a.

Para este caso se monta otro circuito real sobre una protoboard utilizando nuevamente una compuerta lógica OR. La configuración es la siguiente:





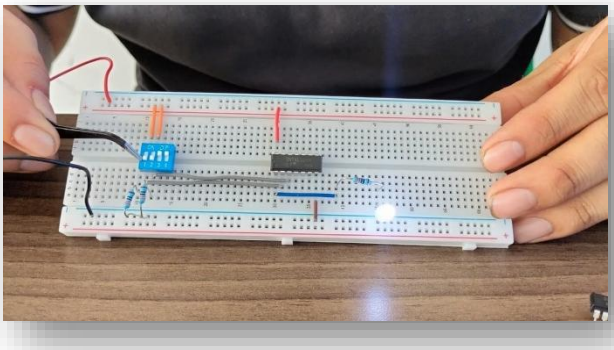
Funcionamiento físico:

En este montaje, la conexión física de una entrada a tierra obliga a que la salida dependa exclusivamente del valor de la señal variable aa. Así, si aa toma el valor 1 se observará que el LED se enciende, y si aa es 0 el LED permanecerá apagado.

Esto demuestra de forma tangible que la operación a+0a + 0 refleja el valor de la variable aa, en concordancia con el postulado a+0=aa + 0 = a.

3. Circuito Físico para, a + a = a.

En este circuito se utiliza una tablilla de pruebas (protoboard) y una compuerta lógica OR de dos entradas para demostrar el postulado a + a = a. La señal variable "a", que puede obtenerse a partir de un interruptor o un generador digital, se conecta simultáneamente a ambas entradas de la compuerta. De esta forma, se establece que la compuerta realiza la operación a OR a, lo cual, según el álgebra booleana, es equivalente a a.

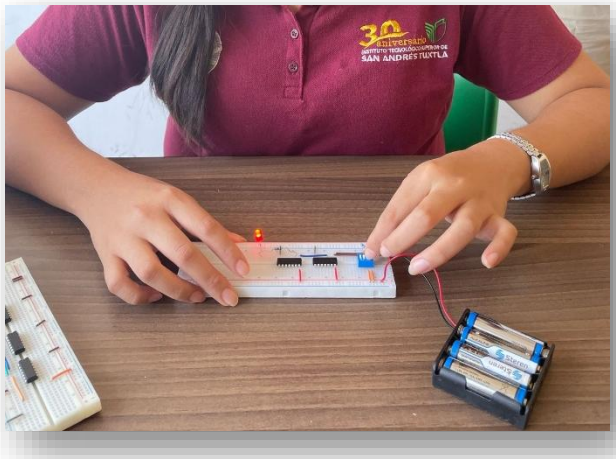


Funcionamiento físico:

El funcionamiento físico se fundamenta en el hecho de que, al ingresar la misma señal en ambos terminales de la OR, el resultado no sufre modificación, independientemente de que "a" sea 0 o 1. Así, si "a" es 1, ambas entradas son 1 y la salida es 1; si "a" es 0, ambas entradas son 0 y la salida permanece en 0. Esta salida se visualiza mediante un LED, conectado a la salida de la compuerta a través de un resistor que limita la corriente, lo que permite constatar de forma práctica que la salida sigue fielmente el valor de "a".

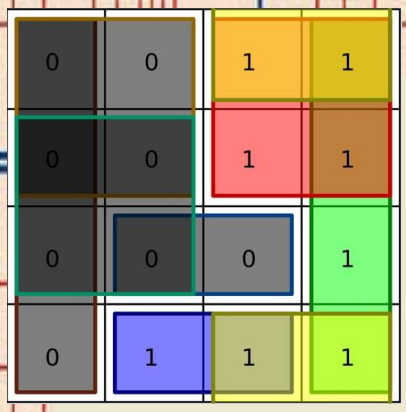
4. Circuito físico para, $a + \bar{a} = 1$

Contamos con un circuito físico compuesto por una compuerta OR (U4) y una compuerta NOT (U5). La señal de entrada "a" se conecta directamente a una entrada de la compuerta OR y también a la compuerta NOT, cuya salida (es decir, $\bar{a}$) se conecta a la segunda entrada de la compuerta OR. A la salida de la compuerta OR, se conecta una resistencia de 100 ohmios y un LED de color agua que indica el estado lógico de la salida.

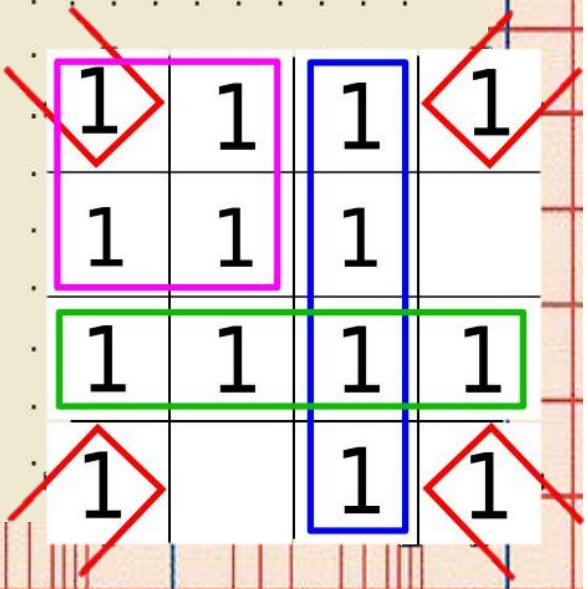


Funcionamiento físico:

Este circuito comprueba de forma tangible el postulado, $a + \bar{a} = 1$, ya que sin importar el estado de la señal "a" (controlada por un interruptor SPDT físico en la placa), la salida de la compuerta O **siempre se mantiene en estado alto (1 lógico)** . Es decir, el LED permanece encendido sin importar si "a" es 0 o 1, confirmando que la suma lógica de una variable y su inversa siempre es 1.

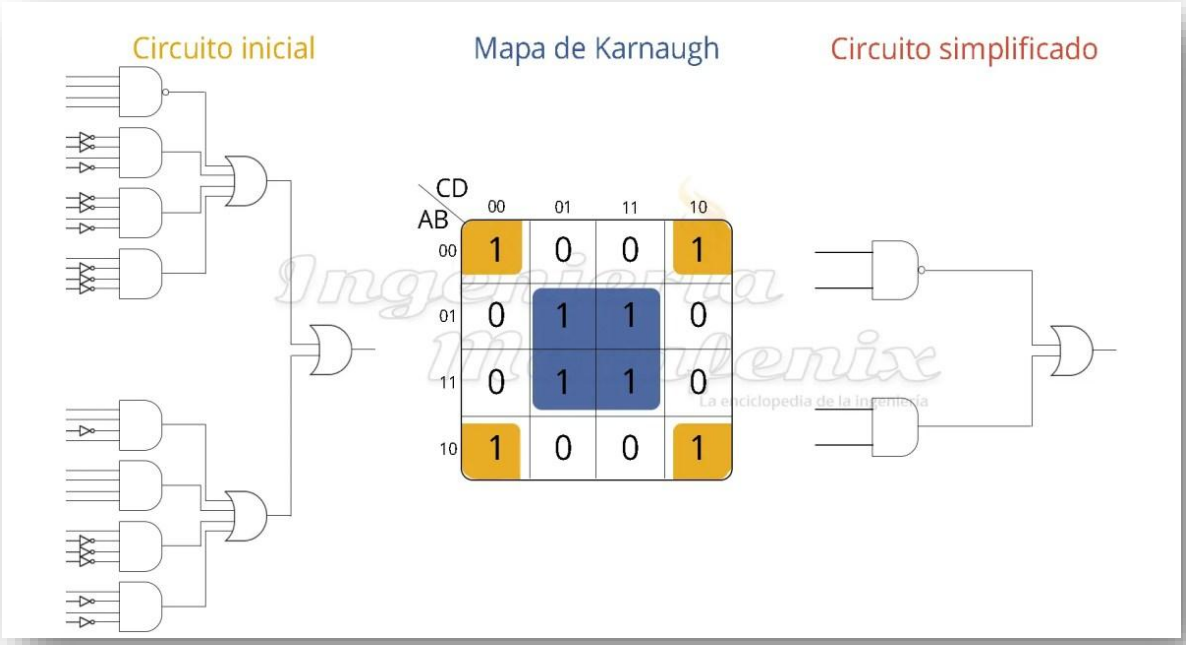


PRÁCTICA 4



MAPA DE KARNAUGH

El mapa de Karnaugh es una herramienta que se utiliza dentro de la electrónica digital (compuertas lógicas), para simplificar los circuitos de compuertas lógicas. Esta simplificación nos ayuda para reducir el tamaño del circuito, y para utilizar una menor cantidad de componentes, que al final se traduce en un menor consumo de corriente eléctrica y un menor gasto económico.



Problema: Cuarto de revelado

Angela ha acondicionado una habitación para revelar sus fotografías. Consiste en un circuito que tiene dos pulsadores para abrir la puerta: uno dentro de la habitación (P1) y otro fuera (P2). La puerta se abrirá cuando se pulse uno cualquiera de los dos pulsadores, o los dos a la vez. Además, para asegurarse de que nadie entre cuando está revelando las fotografías, ha montado un interruptor que acciona una luz roja fuera de la habitación. Por seguridad, cuando la luz roja esté encendida, la puerta no se podrá abrir ni desde fuera ni desde dentro.

Identificación de entradas y salidas.

- Entradas: pulsadores P1, P2 e interruptor I
- Salidas: Luz roja en el exterior de la habitación y motor que abre y cierra la puerta

Podemos observar que para que la puerta se abra (motor activado) tiene que ocurrir dos cosas:

- Que este la luz roja apagada
- Que se activen cualquiera de los pulsadores, o los dos a la vez

Tabla de verdad.

(P1)	(P2)	(I)	(Luz roja)	(Motor)
a	b	c	L	M
0	0	0	0	0
0	0	1	1	0
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	0	1
1	1	1	1	0

Análisis del Problema: Cuarto de Revelado

Descripción del Escenario

Angela ha diseñado un cuarto de revelado fotográfico con los siguientes controles:

- **P1:** Pulsador interno para abrir la puerta.
- **P2:** Pulsador externo para abrir la puerta.
- **I:** Interruptor que enciende una luz roja indicando que el cuarto está en uso.

Condiciones de Operación:

1. La **puerta** debe abrirse si se presiona cualquiera de los pulsadores (*P1* o *P2*), siempre que la luz roja esté apagada (*I = 0*).
2. La **luz roja** se enciende cuando el interruptor *I* está activado (*I = 1*), y en este estado, la puerta no debe abrirse por seguridad.

Desarrollo de la Tabla de Verdad

Para representar todas las combinaciones posibles de entradas y sus respectivas salidas, se construye la siguiente tabla:

P1	P2	I	Luz Roja (L)	Motor (M)
0	0	0	0	0
0	0	1	1	0
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	0	1
1	1	1	1	0

Observaciones:

- La **Luz Roja (L)** se activa únicamente cuando *I = 1*.
- El **Motor (M)** se activa cuando *I = 0* y al menos uno de los pulsadores (*P1* o *P2*) está presionado.

Obtención de las Expresiones Lógicas

Para la Luz Roja (L):

Dado que la luz roja se enciende directamente con el interruptor *I*, la expresión es:

$$L = I$$

Para el Motor (M):

El motor debe activarse cuando:

- La luz roja está apagada (*I = 0*), lo que implica *I'* (complemento de *I*).
- Al menos uno de los pulsadores está presionado (*P1* o *P2*).

La expresión lógica es:

$$M = (P1 + P2) \cdot I'$$

Simplificación mediante el Mapa de Karnaugh

Para simplificar la expresión del motor, utilizamos un mapa de Karnaugh de 3 variables (*P1*, *P2*, *I*).

Construcción del Mapa

P1\P2	00	01	11	10
I = 0	0	1	1	1
I = 1	0	0	0	0

Agrupación y Simplificación

Observamos que los valores '1' se agrupan en la fila donde *I = 0*, abarcando las combinaciones donde *P1* y *P2* tienen al menos un '1'. Esto confirma que la expresión simplificada es:

$$M = (P1 + P2) \cdot I'$$

CONCLUSIÓN

A través de esta práctica se logró comprobar de forma sencilla y clara algunos de los postulados básicos del álgebra de Boole, utilizando compuertas OR e implementando circuitos en una protoboard. Los resultados obtenidos en cada montaje coincidieron con lo que se esperaba teóricamente, lo que permitió confirmar el funcionamiento de las expresiones lógicas trabajadas.

La actividad ayudó a comprender mejor cómo se comportan las compuertas lógicas y cómo se aplican los conceptos del álgebra de Boole en la electrónica digital. Además, se reforzaron habilidades prácticas al manipular componentes electrónicos y observar cómo las señales de entrada afectan directamente la salida del circuito.

En general, la práctica fue útil para relacionar la teoría con la aplicación real, sirviendo como base para el estudio y diseño de sistemas digitales más avanzados.

LISTA DE COTEJO DE PRÁCTICAS

ELECTRÓNICA DIGITAL.

PRÁCTICA NÚMERO 3.

Nombre del estudiante: QUINO CAIXBA PERLA JOSELIN.

Tema: Circuitos Combinacionales.

Portada	5 %	5 %
Introducción	10 %	10 %
Desarrollo	20 %	20 %
Conclusiones	10 %	10 %
Referencias	5 %	3 %
Entrega en tiempo y forma	10 %	10 %
Total	60 %	58 %