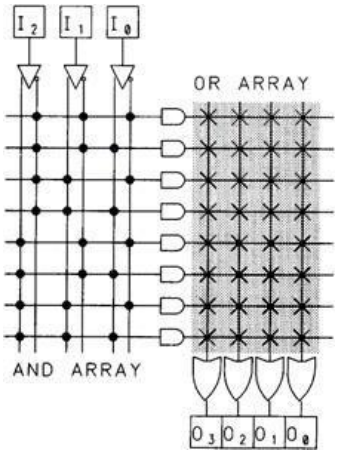
Universidad Nacional Autónoma de México
Facultad de Estudios Superiores Cuautitlán

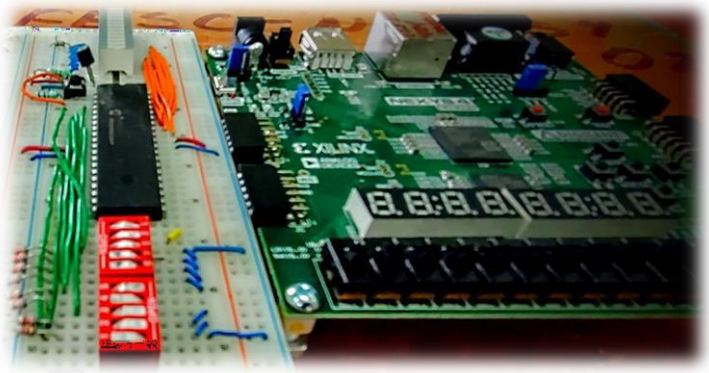
Departamento de Ingeniería
Sección Electrónica

Manual de prácticas del laboratorio de Dispositivos Lógicos Programables

SEMESTRE 2025 – 1

Asignatura: Dispositivos Lógicos Programables.
Clave de carrera: 130 **Clave de la asignatura: 1725**





Fecha de elaboración: enero de 2015.
Fecha de modificación: julio de 2024.

Autores: Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan.
Dr. Fernando Gudiño Peñaloza.

Este manual fue actualizado con el apoyo del proyecto
FESC - PIAPIME 1.31.26.23

ÍNDICE

CONTENIDO

OBJETIVO GENERAL DE LA ASIGNATURA

- Al finalizar el curso el estudiante conocerá y comprenderá la estructura y funcionamiento de los dispositivos lógicos programables (PLD) empleados en el diseño de sistemas digitales de última generación, así mismo podrá diseñar e implementar dispositivos con circuitos programables de muy alta escala de integración.

OBJETIVO DEL LABORATORIO

- Analizar y comprobar los fundamentos de los dispositivos lógicos programables (PLDs).
- Implementar circuitos electrónicos que empleen PLDs para realizar el control de sistemas.
- Conocer las diferentes arquitecturas empleadas en la construcción de PLDs y aplicarlas de acuerdo con el diseño del sistema.
- Conocer y utilizar las herramientas de hardware y software necesarias para la implementación de los circuitos que contienen PLDs.

INTRODUCCIÓN

Un PLD (*Programmable Logic Device*, Dispositivo Lógico Programable) es un dispositivo electrónico de gran escala de integración (*Large Scale Integration*, LSI) empleado para la implementación de diseños digitales. A diferencia de las compuertas lógicas básicas, un PLD es un arreglo genérico de compuertas y otros elementos digitales que no realiza una función definida.

Para implementar un sistema digital dentro de un dispositivo lógico programable, es necesario programarlo o configurarlo para establecer las conexiones que deberán realizarse internamente. Existe una gama muy amplia de PLDs pero en general están construidos con una matriz de compuertas inversoras, compuertas AND y compuertas OR, que se pueden configurar para conseguir funciones lógicas específicas.

Dependiendo de la complejidad del PLD, pueden contener además otro tipo de elementos digitales, tales como Flip-Flops, multiplexores, inversores de polaridad, memorias, registros y muchos elementos más formando las Macroceldas o Bloques Lógicos Configurables (CLBs).

Existen varios tipos de dispositivos que se clasifican como PLDs.

- PROM (*Programmable Read Only Memory*). Memoria programable de sólo lectura.
- PLA (*Programmable Logic Array*). Arreglos Lógicos Programables.

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

- PAL (*Programmable Array Logic*). Lógica de Arreglos Programables.
- GAL (*Generic Array Logic*). Arreglos Lógicos Genéricos.
- CPLD (*Complex Programmable Logic Device*) Dispositivos Lógicos Programables Complejos.
- FPGA (*Field Programmable Gates Array*) Arreglos de Compuertas Programables de Campo.

Básicamente, una matriz programable es una red de conductores distribuidos en filas y columnas con un fusible o conexión programable en cada punto de intersección (figura P.1) y que sirven para establecer las relaciones entre los elementos digitales que conforman al dispositivo.

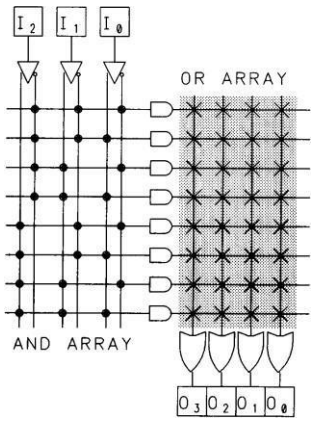


Figura 1

Las matrices de conexiones pueden ser fijas o programables por el usuario, también pueden ser programables una sola vez (*Once Time Programmable*, OTP) o reprogramables, utilizando programación volátil o programación permanente, con lo cual se obtiene un abanico muy amplio de dispositivos con diversas características y funcionalidades.

Los PLDs presentan muchas ventajas sobre los circuitos discretos implementados con compuertas básicas de baja escala de integración (*Small Scale Integration*, SSI) y de mediana escala de integración (*Medium Scale Integration*, MSI), estas características se pueden identificar como:

- Facilidad de diseño.
- Disponibilidad de herramientas.
- Prestaciones.
- Fiabilidad.
- Economía.
- Seguridad.
- Potencia consumida.
- Tamaño.

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

Además de los puntos mencionados, se puede añadir que los PLDs facilitan el ruteado de las placas de circuito impreso debido a la libertad de asignación de terminales de este tipo de dispositivos.

En este laboratorio se realizarán circuitos digitales que emplean diferentes dispositivos lógicos programables PLDs y también la utilización de varias de las herramientas de ambiente integrado *(Integrated Development Enviroment*, IDE) para implementar circuitos que contengan PLDs, tanto para su programación como para su simulación.

Es importante recalcar que el estudiante deberá comprender la necesidad de interacción entre las diferentes áreas de la ingeniería para llevar de los conceptos teóricos a la práctica un sistema digital que integre este tipo de dispositivos programables.

Las áreas que se encuentran relacionadas con esta asignatura son:

- Electrónica Analógica.
- Sistemas Digitales.
- Control Analógico.
- Control Digital.
- Computación.
- Diseño de Software.
- Electrónica de Potencia.
- Mecánica.
- Motores.

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

	UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO **FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN** **DEPARTAMENTO DE INGENIERÍA** **SECCIÓN ELECTRÓNICA**
	REGLAMENTO INTERNO DE LABORATORIOS

El presente reglamento de la sección electrónica tiene por objetivo establecer los lineamientos para el uso y seguridad de laboratorios, condiciones de operación y evaluación, que deberán de conocer y aplicar, estudiantes y profesores en sus cuatro áreas: comunicaciones, control, sistemas analógicos y sistemas digitales.

1. Queda estrictamente prohibido, al interior de los laboratorios

   a) Correr, jugar, gritar o hacer cualquier otra clase de desorden.
   b) Dejar basura en las mesas de trabajo y/o pisos.
   c) Fumar, consumir alimentos y/o bebidas.
   d) Realizar o responder llamadas telefónicas y/o el envío de cualquier tipo de mensajería.
   e) La presencia de personas ajenas en los horarios de laboratorio.
   f) Dejar los bancos en desorden y/o sobre las mesas.
   g) Mover equipos o quitar accesorios de una mesa de trabajo.
   h) Usar o manipular el equipo sin la autorización del profesor.
   i) Rayar y/o sentarse en las mesas del laboratorio.
   j) Energizar algún circuito sin antes verificar que las conexiones sean las correctas (polaridad de las fuentes de voltaje, multímetros, etc.).
   k) Hacer cambios en las conexiones o desconectar el equipo estando energizado.
   l) Hacer trabajos pesados (taladrar, martillar, etc.) en las mesas de trabajo.
   m) Instalar software y/o guardar información en los equipos de cómputo de los laboratorios.
   n) El uso de cualquier aparato o dispositivo electrónico ajeno al propósito para la realización de la práctica.
   o) Impartir clases teóricas, su uso es exclusivo para las sesiones de laboratorio.

2. Es responsabilidad del profesor y de los estudiantes revisar las condiciones del equipo e instalaciones del laboratorio al inicio de cada práctica (encendido, dañado, sin funcionar, maltratado, etc.). El profesor deberá generar el reporte de fallas de equipo o de cualquier anomalía y entregarlo al responsable de laboratorio o al jefe de sección.

3. Los profesores deberán de cumplir con las actividades y tiempos indicados en el “cronograma de actividades de laboratorio”.

4. Es requisito indispensable para la realización de las prácticas que el estudiante:

   a) Descargue el manual completo y actualizado al semestre en curso, el cual podrá obtener en (http://olimpia.cuautitlan2.unam.mx/pagina_ingenieria/)
   b) Presente su circuito armado en la tableta de conexiones para poder realizar la práctica (cuando aplique), de no ser así, tendrá una evaluación de cero en la sesión correspondiente.
   c) Realizar las actividades previas y entregarlas antes del inicio de la sesión de práctica, de no ser así, tendrá una evaluación de cero en la sesión correspondiente.

5. Estudiante que no asista a la sesión de práctica de laboratorio será evaluado con cero.

6. La evaluación de cada sesión debe realizarse con base en los criterios de evaluación incluidos en los manuales de prácticas de laboratorio y no podrán ser modificados. En caso contrario, el estudiante deberá reportarlo al jefe de sección.

7. La evaluación final del estudiante en los laboratorios será con base en lo siguiente:

   a) **(Aprobado) Cuando el promedio total de todas las prácticas de laboratorio sea mayor o igual a 6 siempre y cuando tengan el 90% de asistencia y el 80% de prácticas acreditadas con base en los criterios de evaluación.**
   b) **(No Aprobado) No cumplió con los requisitos mínimos establecidos en el punto anterior.**
   c) **(No Presentó) Cuando no asistió a ninguna sesión de laboratorio o que no haya entregado actividades previas o reporte alguno.**

8. Profesores que requieran hacer uso de las instalaciones de laboratorio para realizar trabajos o proyectos, es requisito indispensable que las soliciten por escrito al jefe de sección. Siempre y cuando no interfiera con los horarios de los laboratorios.

9. Estudiantes que requieran realizar trabajos o proyectos en las instalaciones de los laboratorios, es requisito indispensable que esté presente el profesor responsable del trabajo o proyecto. En caso contrario no podrán hacer uso de las instalaciones.

10. Correo electrónico del buzón para quejas y sugerencias para cualquier asunto relacionado con los laboratorios (seccion_electronica@cuautitlan.unam.mx).

11. El incumplimiento a estas disposiciones faculta al profesor para que instruya la salida del infractor y en caso de resistencia, la suspensión de la práctica.

12. A los usuarios que, por su negligencia o descuido inexcusable, cause daños al laboratorio, materiales o equipo deberá cubrir los gastos que se generen con motivo de la reparación o reposición, indicándose en el reporte de fallas correspondiente.

13. Los usuarios de laboratorio que sean sorprendidos haciendo uso indebido de equipos, materiales, instalaciones y demás implementos, serán sancionados conforme a la legislación universitaria que le corresponda, según la gravedad de la falta cometida.

14. Los casos no previstos en el presente reglamento serán resueltos por el Jefe de Sección, de acuerdo con los lineamientos generales para el uso de los laboratorios en la Universidad Nacional Autónoma de México.

SECCIÓN ELECTRÓNICA
“POR MI RAZA HABLARÁ EL ESPÍRITU”
Cuautitlán Izcalli, Estado de Méx. a 18 de junio de 2024

INSTRUCCIONES PARA LA ELABORACIÓN DE LOS REPORTES DE PRÁCTICAS

Los reportes deberán basarse en la metodología utilizada en los manuales de prácticas de laboratorio:

- Portada.
- Introducción.
- Procedimiento experimental.
- Esquemas / Diagramas.
- Imágenes de los circuitos armados y en funcionamiento.
- Tablas de datos.
- Mediciones.
- Gráficas.
- Comentarios.
- Observaciones.
- Cuestionario.
- Conclusiones.
- Bibliografía.

Será necesario incluir en cada actividad previa y reporte de práctica, una portada (obligatoria) que incluya como mínimo los datos mostrados a continuación, con un formato libre.

Universidad Nacional Autónoma de México
Facultad de Estudios Superiores Cuautitlán

Laboratorio de: _____ **Grupo:** _____

Profesor(a): _____

Estudiante: _____

Nombre de la práctica: _____ **No. de práctica:** _____

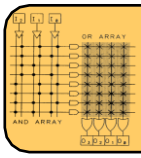
Fecha de realización: _____ **Fecha de entrega:** _____ **Semestre:** _____

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

CRITERIOS DE EVALUACIÓN

No. de criterio	Criterio de Evaluación para el laboratorio	Porcentaje
C1	Actividades previas	10%
C2	Escritura y compilación de los códigos con comentarios	30%
C3	Habilidad en el armado y funcionalidad de los sistemas	20%
C4	Reporte entregado con todos los puntos indicados	40%

Los criterios de evaluación indicados no podrán ser modificados.



Práctica 1. Decodificadores con EEPROM.

Objetivos

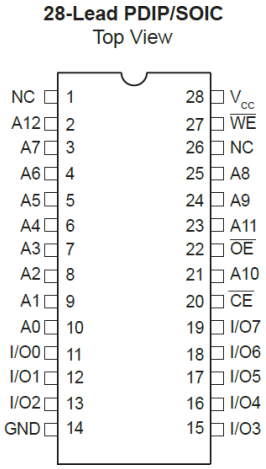
- Implementar un circuito que convierta un número decimal en el rango de 0 a 99 representado en binario en 7 bits, a su correspondiente código BCD5421 en 8 bits.
- Programar la memoria EEPROM con la tabla de verdad del convertidor de código.
- Comprobar el funcionamiento del convertidor de código.

Introducción

Entre los dispositivos lógicos programables básicos se encuentran las memorias de solo lectura (ROM's), en ellas se puede almacenar de forma directa las tablas de verdad que representan a los sistemas digitales, estas memorias tienen una capacidad de almacenamiento definido por el tamaño de su bus de direcciones (n líneas) y el tamaño de su bus de datos (m líneas), por lo tanto, pueden resolver sistemas que tengan "n" entradas por "m" salidas.

Una de las ventajas al integrar estas memorias es que no se requiere la reducción algebraica de los términos a través del "álgebra de Boole" o los "mapas de Karnaugh", ya que la tabla deberá ser insertada en su totalidad y cada dirección a la que se acceda proporcionará una de las combinaciones de la tabla.

El planteamiento de esta práctica describe el uso de una memoria AT28C64B (Fig. 1.1), teniendo una capacidad de 64K (8K x 8), por lo tanto, es capaz de almacenar tablas de 13 entradas (8,192 combinaciones de entrada) y puede proporcionar hasta 8 salidas independientes.



Pin Name	Function
A0 - A12	Addresses
CE	Chip Enable
OE	Output Enable
WE	Write Enable
I/O0 - I/O7	Data Inputs/Outputs
NC	No connect
DC	Don't Connect

Fig. 1.1 Terminales de la memoria AT28C64B

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

Sin embargo, puede emplearse alguna otra memoria EEPROM que cumpla con las especificaciones en cuanto a pines y funciones para llevar a cabo la práctica de laboratorio.

Planteamiento del problema:

Deberá realizarse el diseño de un circuito que sea capaz de convertir un número decimal que se encuentre en el rango de 0 a 99 (Tabla 1.1), que sea representado a través de un número binario de 7 bits y que produzca el valor equivalente en código decimal codificado en binario (BCD_{8421}) en 8 bit.

El rango de funcionamiento del circuito está restringido debido a que para números mayores a 99 y hasta 255 se requiere una representación BCD_{8421} de 12 bits, por lo tanto, la capacidad de la memoria se estaría excediendo, para estas condiciones entonces, el sistema tendrá una tabla de verdad de 7 entradas y 8 salidas, lo cual produce una tabla de 27 = 128 combinaciones de las cuales solo se emplearán las 100 primeras y las restantes deberán llenarse con ceros tal como se muestra en la Fig. 1.2.

Decimal	Binario	BCD8421
0	0000000	0000 0000
1	0000001	0000 0001
36	0100100	0011 0110
37	0100101	0011 0111
38	0100110	0011 1000
97	1100001	1001 0111
98	1100010	1001 1000
99	1100011	1001 1001

Tabla 1.1.

Actividades previas a la realización de la práctica

1. El alumno deberá realizar la lectura de la práctica.
2. El alumno definirá los valores que representan la conversión de los 100 números a convertir.
3. El alumno generará el archivo binario BIN_BCD.BIN empleando el software del programador SuperPro, para ello deberá insertar la tabla de 128 valores que representan a los códigos BCD8421 de salida.

Material

- 1 memoria EEPROM (se recomienda utilizar la AT28C16 o AT28C64B).
- 8 resistencias de 330 Ω.
- 1 barra de 8 LEDs o en su defecto 8 leds individuales.
- 1 tarjeta de conexiones.
- Alambres para conexiones.

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

Equipo

- 1 fuente de voltaje de CD.
- 1 multímetro.
- 1 programador universal.

Procedimiento Experimental

1. Generar la tabla de valores en el editor de buffer del software SuperPro como se muestra en la figura 1.3. Considere que los datos en el buffer son los valores de la tabla en código BCD8421 y solo deben llenarse 100 localidades, las restantes 28 deberán estar cargadas con el valor de 00.

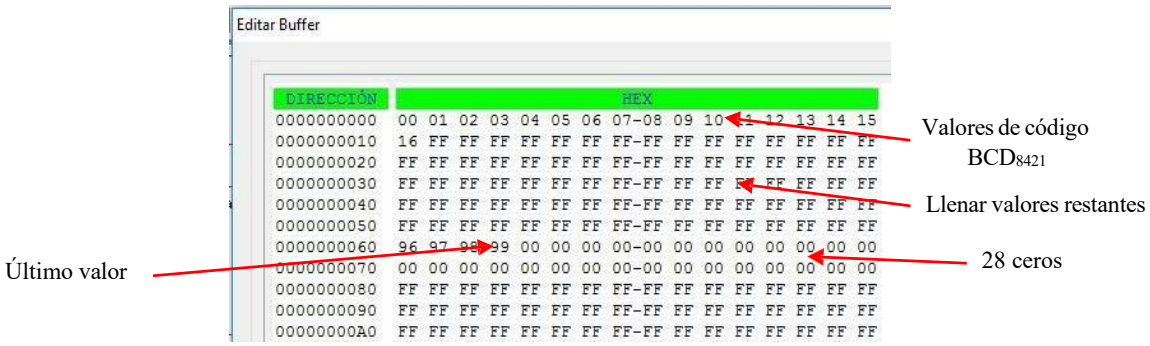


Fig. 1.3 Ejemplo incompleto de Buffer de Proteus

2. Salvar el buffer en formato binario seleccionando la pestaña de Archivo/Guardar y asignando el nombre BIN_BCD, designe la carpeta de almacenamiento que desee como se muestra en las figuras 1.4 y 1.5.

Fig. 1.4 y 1.5 Pestaña de selección de Archivo y Nombre de archivo

3. Programe la memoria EEPROM con el archivo BIN_BCD.BIN

4. Compruebe la función de conversión de código del circuito proponiendo una tabla de 16 valores diferentes.

5. Llene la tabla 1.3 con los 16 valores obtenidos del circuito y verifique que la operación se está realizando correctamente.

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

Decimal	Binario	BCD_{8421}

Tabla 1.3

6. El alumno diseñará un sistema que genere 4 códigos de salida a partir de un código de entrada de 4 bits:

   - Código gray (4 bits).
   - Código en complemento a 2 (4 bits).
   - Código XS3 (8 bits).
   - Código BCD7421 (8 bits).

7. El circuito debe incluir 2 señales de control para que se pueda seleccionar cuál de los 4 códigos se visualiza a la salida.

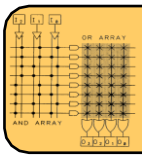
8. El alumno diseñará el circuito correspondiente, determinará las conexiones de la memoria EEPROM que se decidió emplear y comprobará las 4 tablas de verdad.

Cuestionario

1. Diseñe un circuito convertidor de código binario (4 bits) a BCD8421 (4 bits), utilizando herramientas computacionales de reducción para la obtención de las expresiones de álgebra de Boole y dibuje el circuito resultante con compuertas lógicas de cualquier número de entradas.

2. Describa las ventajas y desventajas del circuito resultante con respecto al circuito implementado en esta práctica a través de la memoria ROM, considerando que este diseño tiene 7 bits de entrada y 8 de salida.

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.



Práctica 2. Matriz de LEDs implementando dos máquinas secuenciales.

Objetivos

- Implementar 2 máquinas secuenciales individuales que controlen el desplegado de imágenes en un display matricial de LEDs de 8x8.
- Comprobar el funcionamiento de un registro de corrimiento implementado sobre una EEPROM.
- Implementar la sincronización del envío de datos y registro de corrimiento al display matricial de LEDs.

Introducción

En esta práctica realizaremos la activación de un display matricial de 64 leds, arreglados en 8 renglones y 8 columnas.

Figura 2.1 Display de 8 renglones y 8 columnas

Estos displays están construidos de tal forma que los LEDs de un renglón comparten la misma línea de conexión y asimismo los LEDs de una columna comparten una sola línea lo cual reduce el número de terminales para la activación de los 64 LEDs que de otra manera requerirían 128 terminales para su conexión individual.

Las configuraciones que se pueden encontrar en estos displays pueden ser:

- **Cátodo común en función de las columnas**. Para cátodo común, cada uno de los leds se enciende cuando se aplica un voltaje de 0V en la columna y 5V en el renglón.

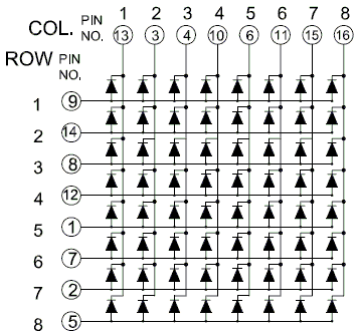


Figura 2.2

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

- **Ánodo común en función de las columnas**. Para ánodo común, cada uno de los leds se enciende cuando se aplica un voltaje de 5V en la columna y 0V en el renglón.

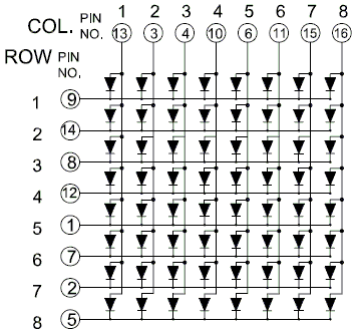


Figura 2.3

La referencia cátodo o ánodo comunes se considera en función de las columnas ya que estos displays son encendidos columna por columna en forma secuencial y los datos se insertarán vía los renglones.

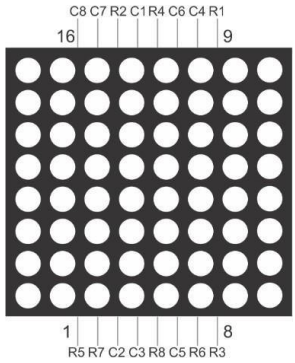


Figura 2.4. Asignación de terminales

El display que se utilizó en el desarrollo de la práctica es de ánodo común por lo que se deberá suministrar 5V en sus columnas y 0V en aquellos renglones que deseamos encender.

Debido a que el encendido del display se realiza columna por columna, se requiere construir un circuito de corrimiento de 8 bits que proporcione voltaje de 5V a cada una de las columnas en forma secuencial y que al mismo tiempo coincida con un cero en la posición del renglón del led que deseamos encender.

Para lograrlo el sistema deberá contar con 2 memorias:

- Una que sirva para generar el registro de corrimiento.
- Otra para almacenar los datos que vamos a desplegar en conjuntos de 8 datos de 8 bits.

Los datos que deberán almacenarse en la memoria de datos se muestran en la figura 2.5.

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

Estos caracteres se deben almacenar a partir de la localidad cero de la memoria de datos y cubriendo 8 localidades, cada letra se seleccionará a partir de 4 bits de control que definen una localidad de memoria a partir de la cual se despliega cada una de las máquinas de estado.

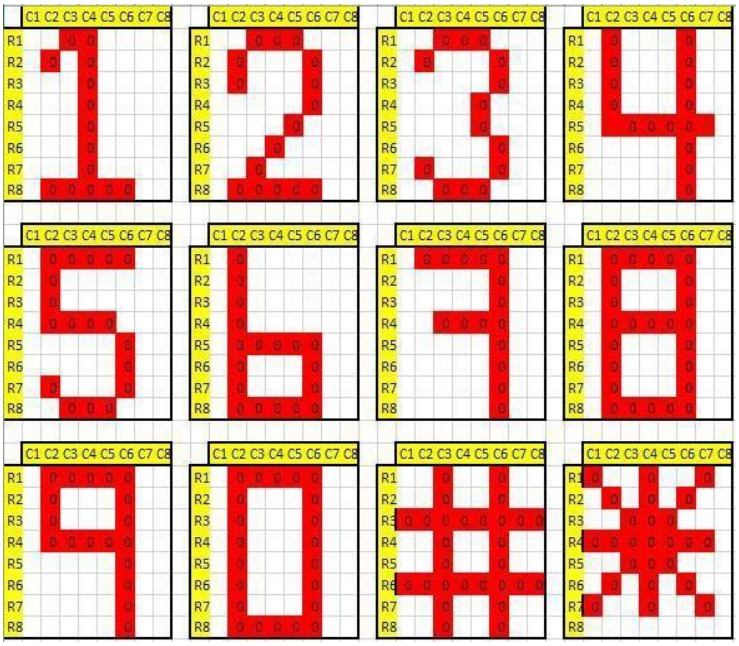


Figura 2.5 Caracteres a desplegar

La memoria 1 deberá contener un registro de corrimiento que encienda solo una de las columnas empezando por la columna 1 y terminando en la columna 8, en forma cíclica como se muestra en la figura 2.6 considerando que las terminales de columnas del display se deben asignar a datos de la memoria 1 de acuerdo con la tabla 2.1.

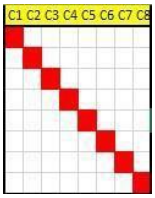


Figura 2.6 Registro de corrimiento para columnas

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Columna	C1	C2	C3	C4	C5	C6	C7	C8

Tabla 2.1 Asignación de terminales de datos de la memoria correspondientes a las columnas

La memoria 2 deberá contener los datos para cada una de las imágenes de los caracteres, los cuales deben definirse considerando la figura 2.5 en donde los bits en blanco deberán ponerse en 1 para apagar el led y los bits indicados con color rojo deberán ponerse en 0 para poder encender ese led.

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

Se debe considerar que las terminales de renglones del display se deben asignar a datos de la memoria 2 de acuerdo con la tabla 2.2.

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Renglón	R8	R7	R6	R5	R4	R3	R2	R1

Tabla 2.2. Asignación de terminales de datos de la memoria 2 correspondientes a los renglones.

Por ejemplo, para el número 2 se deberán cargar los datos indicados en la figura 2.7, considerando que al display se le aplican 5V en la columna que se desea encender y que los leds apagados (sin color en la figura) deberán tener un uno lógico en el renglón correspondiente y los leds que deseamos encender (rojos en la figura) deberán tener un 0 lógico en el renglón correspondiente.

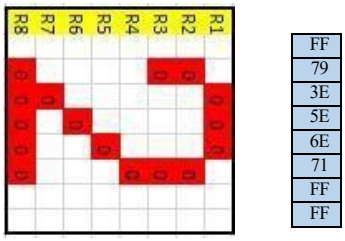


Figura 2.7 Datos hexadecimales para la representación del número 2

Actividades previas a la realización de la práctica

1. El alumno deberá realizar la lectura de la práctica.
2. El alumno realizará la programación de la primera memoria EEPROM con los datos del registro de corrimiento mostrados en la figura 2.6 y la segunda memoria se debe programar con la tabla de datos correspondiente a los caracteres mostrados en la figura 2.5. Ocho localidades para cada carácter y por lo tanto 128 localidades.
3. Como la tabla solo contiene 12 caracteres y se puede controlar la exhibición de 16, el alumno deberá generar 4 caracteres adicionales e incluirlos en la programación.
4. Simule el circuito de la figura 2.8. Considere emplear el circuito de memoria 2732 en sustitución del circuito AT28C16 de la práctica, considerando que tiene una terminal de direcciones extra. Esta sustitución es necesaria debido a que el simulador no tiene un modelo de simulación para la memoria AT28C16.
5. Incluya en el reporte los resultados de la simulación de forma digital.

Material

- 2 memorias EEPROM (se recomienda utilizar la AT28C16 o AT28C64B).
- 1 circuito CD4024.
- 1 display de matriz de leds de 8x8 de configuración ánodo o cátodo común.
- 1 tarjeta de conexiones.
- Alambres para conexiones.

Equipo

- 1 fuente de voltaje de CD.

- 1 generador de funciones.
- 1 programador universal.

Procedimiento Experimental

1. Implemente el circuito de la figura 2.8.

2. Programe la primera memoria EEPROM (U1 de acuerdo con el diagrama de la Figura 2.8) para que contenga el registro de corrimiento.

3. Programe la segunda memoria EEPROM (U2 de acuerdo con el diagrama de la Figura 2.8) para que contenga la tabla de datos de los caracteres que se desean desplegar.

4. Con una señal de reloj de 1Hz y niveles TTL, compruebe que se genera la secuencia adecuada del registro de corrimiento y se produce el encendido de cada una de las columnas del circuito para una entrada de selección en las terminales A3, A2, A1 y A0 con un valor de 0001 correspondiente a la imagen del número 2.

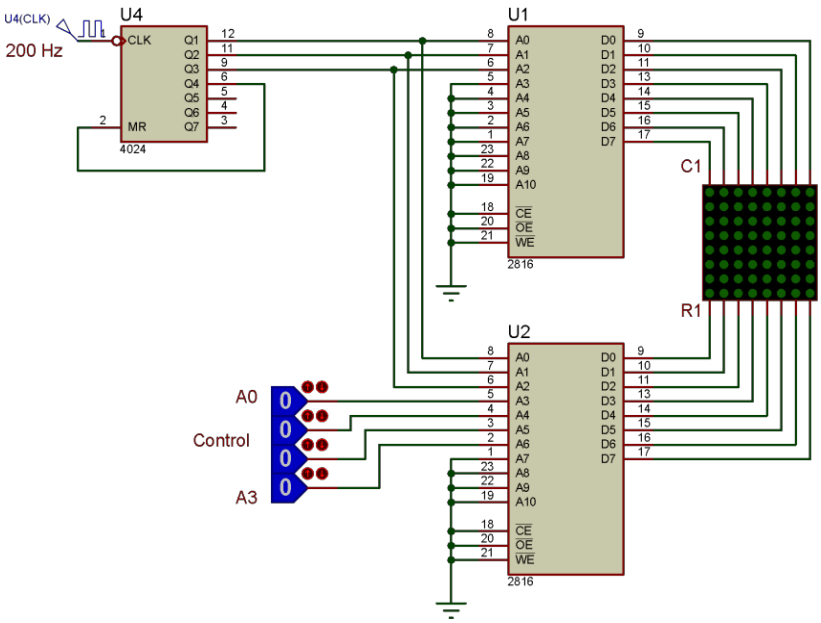


Figura 2.8

5. Incremente la frecuencia de reloj y verifique que la visualización es cada vez más rápida.

6. Aumente la frecuencia de reloj hasta que ya no pueda observarse el parpadeo de las columnas al apagarse, anote el valor de la frecuencia que produce este efecto.

7. Comprobar que se muestran todos los caracteres (incluyendo los 4 que se pidieron agregar).

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

Cuestionario

1. Indique cual es la frecuencia mínima para que se produzca el efecto de visualización fija y porque se puede ver el carácter completo como si el encendido fuera simultáneo.

2. Incluya las gráficas y valores para los 4 caracteres diseñados.

Práctica 3. Animación de figuras sobre matriz de LEDs.

Objetivos

- Implementar 2 máquinas secuenciales que permitan realizar una animación sobre un display de matriz de leds de 8 x 8.
- Comprobar el funcionamiento del sistema de animación.

Introducción

En esta práctica realizaremos una animación en un display matricial de 8 renglones y 8 columnas como el utilizado en la práctica anterior.

El sistema debe contar con 2 memorias, una que sirve para generar el registro de corrimiento y otra para almacenar las máquinas secuenciales que serán desplegadas en forma consecutiva para generar el proceso de animación las cuales serán seleccionadas a partir de un contador.

Cada máquina secuencial constará de 8 datos de 8 bits que representan una de las diferentes imágenes que se presentarán al usuario en forma consecutiva y cada cierto tiempo el sistema deberá acceder a la imagen siguiente para presentar la animación en forma continua.

Esta animación constará de 24 imágenes o cuadros que al ser presentadas una tras otra, darán la ilusión de tener movimiento como se hace en los sistemas de televisión o cine.

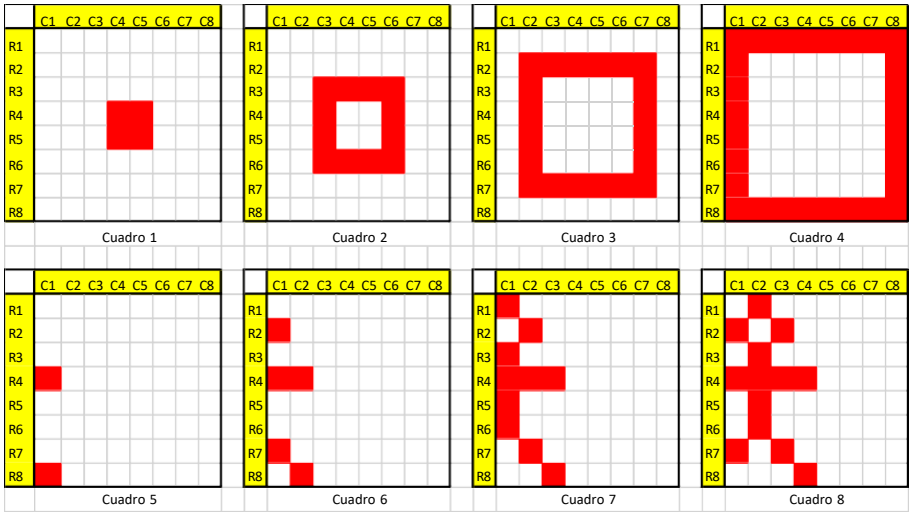


Figura 3.1a Cuadros de la animación.

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

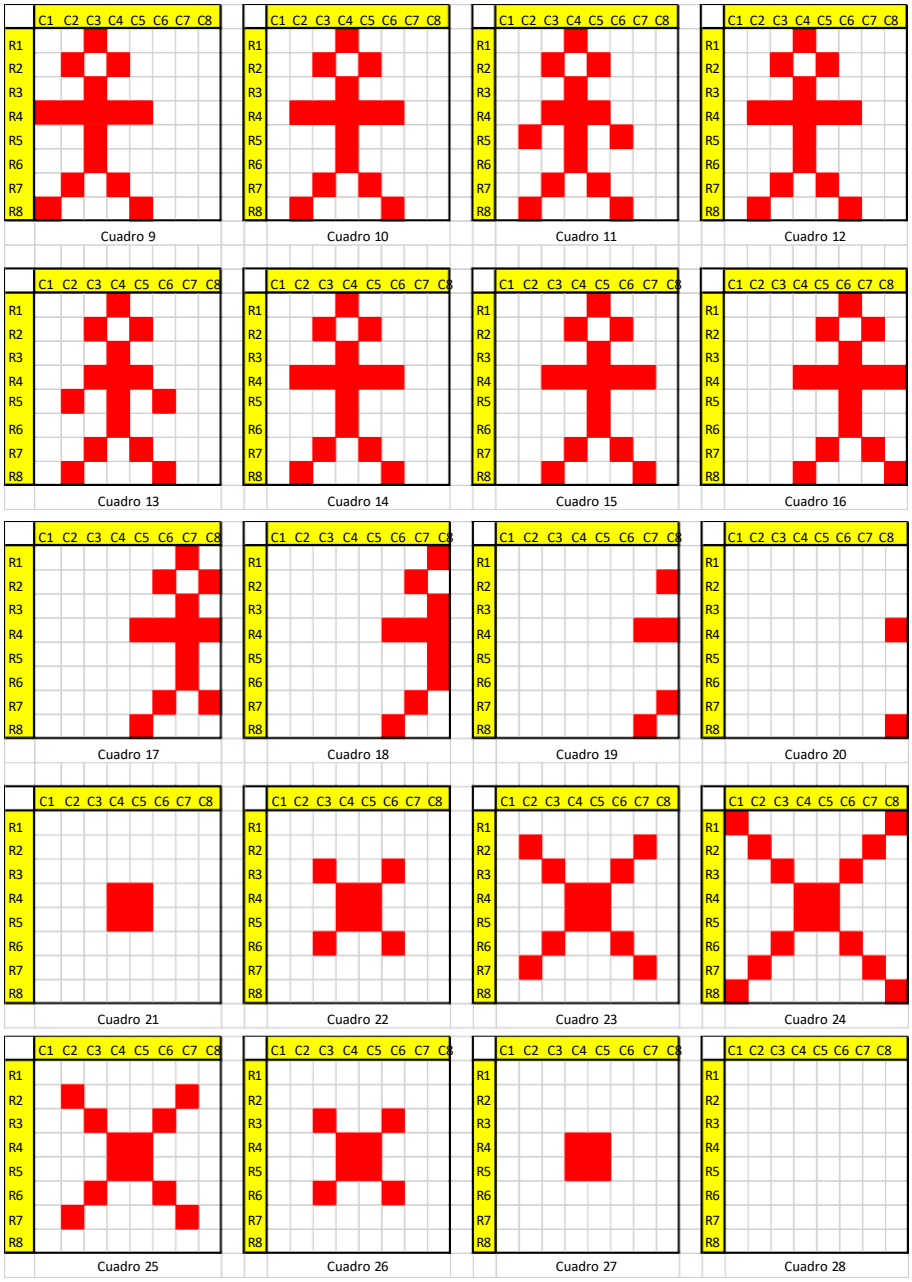


Figura 3.1b Cuadros de la animación.

Entre más imágenes contenga el sistema más se acercará a una ilusión de movimiento continuo, pero para ello se deberá seleccionar la correcta relación de los 2 relojes que alimentan al sistema, uno para definir la velocidad del registro de corrimiento de las columnas y otro reloj para definir la velocidad de cambio de cada imagen.

Las imágenes para desplegar se deben almacenar a partir de la localidad cero de la memoria de datos y cubriendo 8 localidades cada una, cada cuadro se seleccionará a partir de 5 bits de control que definen una localidad de memoria a partir de la cual se despliega cada una de las máquinas de estado.

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

Con 5 bits se pueden seleccionar 32 imágenes, pero debido a que este sistema solo tiene 28 cuadros, entonces se deberá restringir el valor máximo de conteo para la selección de cuadros utilizando una compuerta AND.

Actividades previas a la realización de la práctica

1. El alumno deberá realizar la lectura de la práctica.
2. El alumno realizará la programación de la primera memoria EEPROM (U8 en el diagrama de la Figura 3.2) con los datos del registro de corrimiento.
3. El alumno obtendrá los datos para cada una de las imágenes mostradas en la figura 3.1a y b, ocho localidades para cada imagen.
4. El alumno programará los datos de las imágenes en la segunda memoria EEPROM (U9 en el diagrama de la Figura 3.2).
5. El alumno diseñará un circuito de oscilación astable a partir de un circuito LM555 que pueda variar en un rango de 1 a 30 Hz el cuál se utilizará como reloj del contador de la memoria de datos.
6. Diseñar una animación que contenga 40 imágenes.

Material

- 2 memorias EEPROM (se recomienda utilizar la AT28C16 o AT28C64B).
- 2 circuitos CD4024.
- 1 circuito 7408
- 1 display de matriz de leds de 8x8 de configuración ánodo o cátodo común.
- 1 tarjeta de conexiones.
- Alambres para conexiones.

Equipo

- 1 fuente de voltaje de CD.
- 1 generador de funciones.
- 1 programador universal.

Procedimiento Experimental

1. Implemente el circuito de la figura 3.2.

2. Alimente un reloj de 200 Hz y niveles TTL para el contador del registro de corrimiento utilizando el generador de funciones.

3. Alimente con la señal generada por el oscilador astable del 555 el reloj del contador de la memoria de datos, utilizando una frecuencia aproximada de 10Hz, compruebe que se genera la animación de forma correcta en el display.

4. Si no se aprecia de forma clara el movimiento, se deberán ajustar sobre la práctica los valores de los relojes hasta lograr el funcionamiento correcto.

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

5. Comprobar la animación diseñada en el punto 6 de las actividades previas.

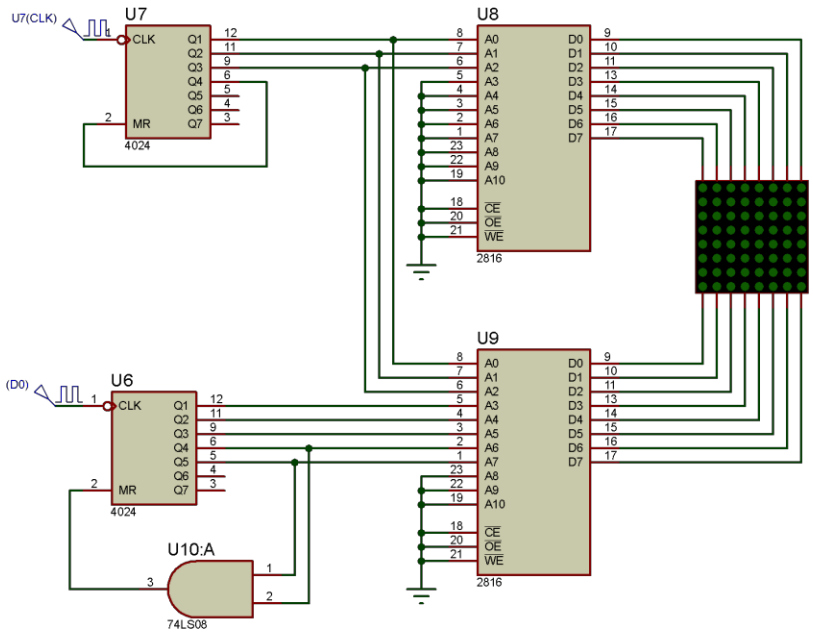


Figura 3.2.

Cuestionario

1. Investigue cuales son las características de frecuencia (cuadros por segundo), necesarias para la presentación correcta en los sistemas de TV. Y de cine.

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

Práctica 4. Compuertas lógicas con CPLD

Objetivos

- Implementar compuertas lógicas en un dispositivo CPLD.
- Conocer el entorno de programación Quartus Prime Lite Edition 18.1.
- Comenzar a utilizar lenguaje de descripción de hardware como VHDL.

Introducción

Además de los PLDs simples, existen dispositivos con mayor escala de integración, capaces de albergar en su memoria un mayor número de macroceldas, para esta práctica se utilizará la tarjeta de desarrollo CPL DIONE de la empresa mexicana NIBBLUX, esta tarjeta cuenta con un circuito integrado CPLD EPM240T100C5 de Altera el cual tiene una capacidad de 240 elementos lógicos que pueden ser programados en lenguaje VHDL. Para realizar la programación deberá utilizarse el entorno de desarrollo Quartus Prime Lite Edition 18.1 (**consulta aquí como instalar el software**).

Para poder adentrarnos y e ir reconociendo los elementos con los que cuenta la tarjeta CPLD DIONE, se realizará un ejercicio básico en donde se programen algunas compuertas lógicas, haciendo uso de los switches y los LEDs con los que cuenta la tarjeta (Figura 4.1).

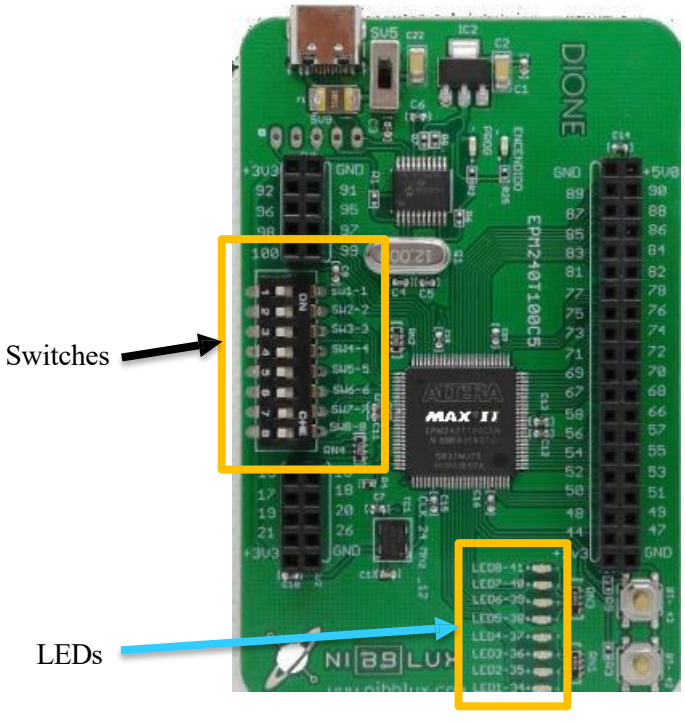


Figura 4.1 Tarjeta CPLD DIONE

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

Actividades previas a la realización de la práctica

1. El alumno deberá realizar la lectura de la práctica.
2. El alumno investigará que es una GAL y cuál es la diferencia que tiene con un CPLD.

Material

- No aplica

Equipo

- 1 tarjeta CPLD DIONE.
- 1 cable USB tipo C a USB tipo A.
- 1 computadora con el software Quartus Prime Lite Edition 18.1 instalado.

Procedimiento Experimental

1. Abra el programa Quartus Prime Lite Edition 18.1 y seleccione el botón: New Project Wizard (Figura 4.2).

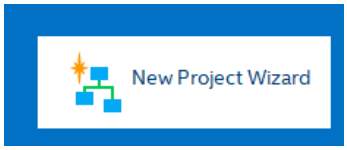


Figura 4.2

2. Realice el código que se muestra en la figura 4.3.

```
library ieee;
use ieee.std_logic_1164.all;

entity Compuertas is --Entity: Define la lista de un modelo

   port( A1, B1 : in std_logic;  --Se asignaron a A1 = SW1-1 y B1 = SW2-2
         A2, B2 : in std_logic;  --Se asignaron a A2 = SW5-5 y B2 = SW6-6
         A3 : in std_logic;      --Se asignó a A3 = SW8-8
         S1, S2, S3, S4, S5 : out std_logic); --Salidas que se asignarán a los LEDs

end Compuertas;

architecture comportamiento of Compuertas is --Architecture: Define una posible funcionalidad
begin

   --Declaración del funcionamiento del programa

   S1 <= A1 AND B1;         --LED8-41 Compuerta AND
   S2 <= A2 OR B2;          --LED7-40 Compuerta OR
   S3 <= NOT A3;            --LED6-39 Compuerta NOT
   S4 <= A1 NAND B1;     --LED5-37 Compuerta NAND
   S5 <= A2 NOR B2;      --LED4-36 Compuerta NOR


   --Tablas de verdad para las compuertas AND, OR NAND, NOR Y NOT

   -- | A1,A2 | B1,B2 |  S1  |  S2  |  S4  |  S5  |           |  A3  |  S3  |
   -- ---------------------------------------------           --------------
   -- |   0   |   0   |   0  |   0  |   1  |   1  |           |   0  |   1  |
   -- |   0   |   1   |   0  |   1  |   1  |   0  |           |   1  |   0  |
   -- |   1   |   0   |   0  |   1  |   1  |   0  |
   -- |   1   |   1   |   1  |   1  |   0  |   0  |

end comportamiento;
```

Figura 4.3 Código en VHDL con comentarios

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

3. Asigne los pines que se van a utilizar, para ello debe presionar el botón Pin Planner (figura 4.4) y se abrirá una ventana con la imagen del chip MAX II EMP240T100C5 (figura 4.5), en la cual vendrán los apartados del nombre del puerto (Node name) tal como se pusieron en el código, la dirección del puerto (Direction) para indicar si será de salida, entrada o entrada/salida, el pin que se utilizará (Location), para el A1 se utilizará el SW1-**1** (vea la nomenclatura impresa en la tarjeta), por lo tanto se deberá poner en la sección Location: PIN_**1**, para B1 se usará el SW2-**2**, por tanto, se deberá poner PIN_**2**, para la salida S1 se requiere utilizar el LED8-**41**, por tanto se debe escribir PIN_**41**.

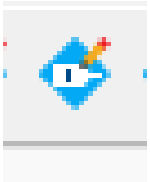


Figura 4.4 Botón “Pin planner”

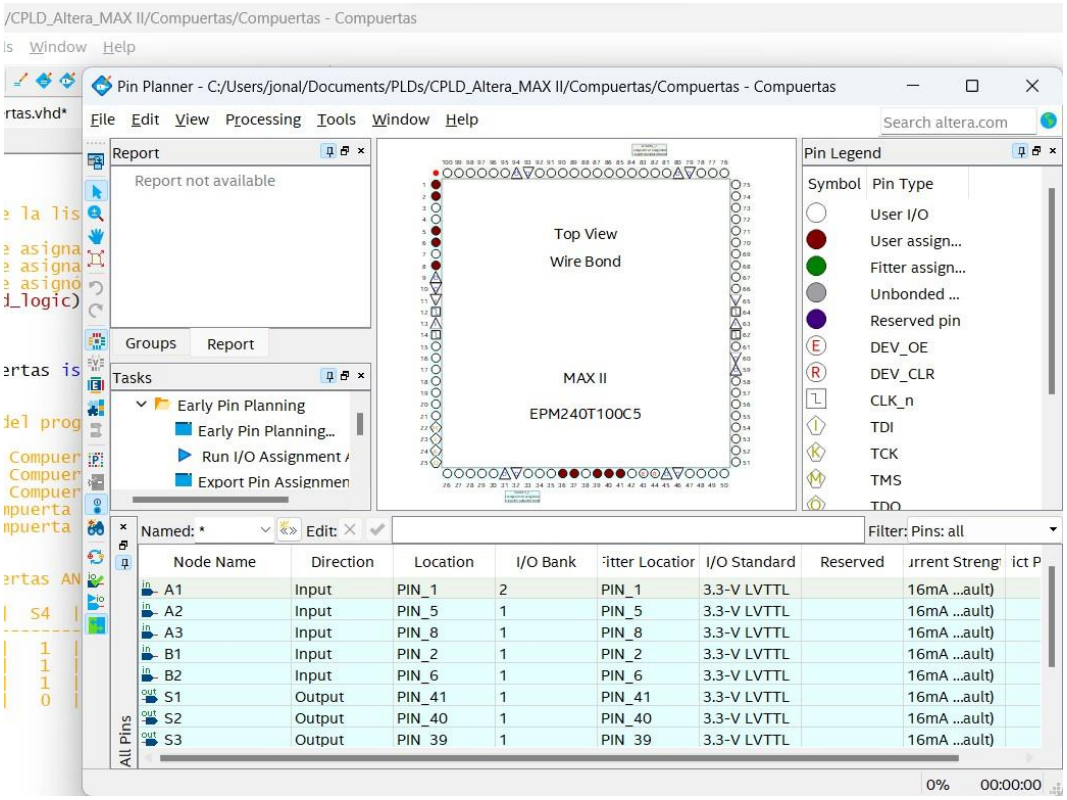


Figura 4.5 Asignación de pines en Quartus (aquí no se muestra la asignación para S4 y S5).

4. Una vez realizada la asignación de pines, se deberá cerrar la ventana de Pin Planer, guardar y sintetizar nuevamente el programa para que los cambios se registren y posteriormente se pueda proceder con la programación del dispositivo.

5. Conecte el cable USB tipo C a la tarjeta CPLD DIONE y el extremo con el puerto USB tipo A a la computadora, energice la tarjeta activando el switch y posteriormente presione el botón del programador (Figura 4.6) y se abrirá la ventana que se muestra en la figura 4.7.

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

Figura 4.6 Botón “Programmer”.

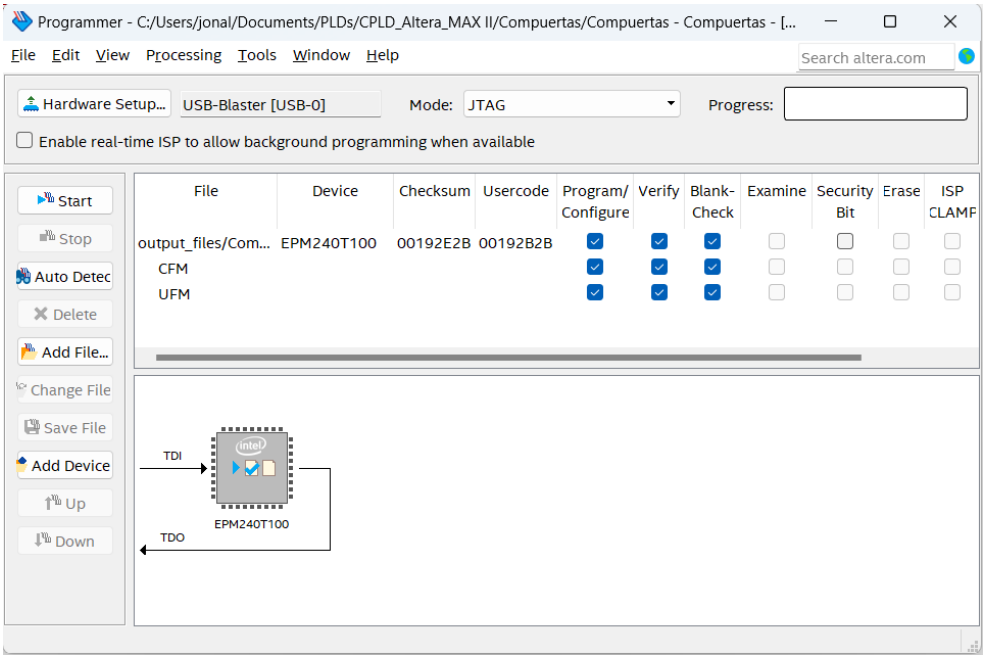


Figura 4.7 Vista del programador del dispositivo.

6. Presione el botón Start para comenzar a programar la tarjeta y pruebe que se cumpla con las tablas de verdad de las compuertas AND, OR, NOT, NAND y NOR (Ponga las evidencias del correcto funcionamiento).

Cuestionario

1. ¿Qué otro tipo de compuertas pueden implementarse en el CPLD?
2. ¿Indique cuantas fuentes de 3.3V tiene la tarjeta y cuantas de 5V?
3. ¿Cuál es la frecuencia del oscilador con el que cuenta esta tarjeta?

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

Práctica 5. Decodificador de teclado con CPLD

Objetivos

- Implementar un decodificador de teclado matricial telefónico de 12 teclas.
- Obtener la representación en algebra de Boole de la tabla de verdad del decodificador
- Implementar las ecuaciones de Boole dentro de un dispositivo CPLD.

Introducción

En esta práctica realizaremos la decodificación de un teclado matricial de 12 teclas utilizado para la marcación en los teléfonos como el mostrado en la figura 4.1. Este teclado presenta un conjunto de 12 teclas arregladas en un formato de matriz que activa un renglón y una columna en correspondencia con la tecla presionada.

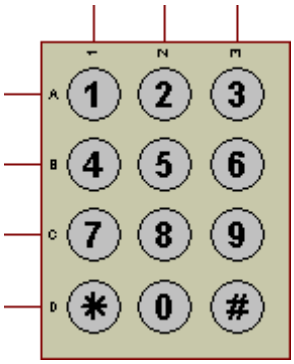


Figura 5.1

Existen varias formas de decodificarlo, una de ellas utiliza un registro de corrimiento que activa los renglones en forma consecutiva del renglón 1 al 4, repitiendo el proceso una y otra vez a una frecuencia elevada, analizando el valor que presentan las columnas y haciendo el análisis de que renglón y columna están activas en el instante de tiempo en que se presiona la tecla, se puede por lo tanto asignar el código correspondiente en binario u otro código, de cada una de las teclas.

En esta práctica emplearemos un método de decodificación que aprovecha una terminal del teclado que es común a todas las teclas y por lo tanto no requiere del registro de corrimiento externo.

Este proceso de decodificación es totalmente estático ya que se emplean 7 resistencias conectadas a la fuente de alimentación y donde la tecla presionada funciona como un switch conectado a tierra y genera un cero lógico tanto en el renglón como en la columna donde está posicionada la tecla.

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

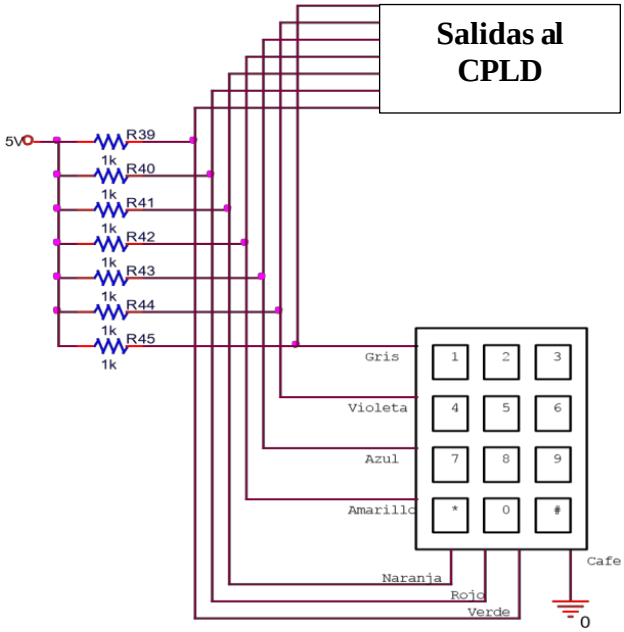


Figura 5.2 Decodificación estática del teclado

En esta práctica se requiere el planteamiento de las tablas de verdad para las 7 entradas; 3 columnas y 4 renglones, que se obtienen de las terminales de las resistencias y las 5 salidas de las cuales 4 proporcionan el código binario para identificar 12 teclas de las 16 posibles y además 1 salida adicional para indicar que una tecla se ha presionado y por lo tanto el código de salida es válido.

Esto es necesario porque no podría distinguirse entre el código de salida 0000 debido a que ninguna tecla está presionada y el código 0000 debido a la tecla 0 que es un código válido.

La asociación de códigos para cada tecla se muestra en la tabla 5.1, observe que el bit de código válido (V) está activo en 1 solo para las 12 combinaciones mostradas de las 128 posibles y por lo tanto los 116 restantes deben tener el bit V = 0.

Tecla	Columnas	Renglones	Código de tecla	Bit de código válido
	C1 C2 C3	R1 R2 R3 R4	S4 S3 S2 S1	Validador
1	0 1 1	0 1 1 1	0 0 0 1	1
2	1 0 1	0 1 1 1	0 0 1 0	1
3	1 1 0	0 1 1 1	0 0 1 1	1
4	0 1 1	1 0 1 1	0 1 0 0	1
5	1 0 1	1 0 1 1	0 1 0 1	1
6	1 1 0	1 0 1 1	0 1 1 0	1
7	0 1 1	1 1 0 1	0 1 1 1	1
8	1 0 1	1 1 0 1	1 0 0 0	1
9	1 1 0	1 1 0 1	1 0 0 1	1
*	0 1 1	1 1 1 0	1 0 1 0	1
0	1 0 1	1 1 1 0	0 0 0 0	1
#	1 1 0	1 1 1 0	1 0 1 1	1

Tabla 5.1

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

Actividades previas a la realización de la práctica

1. El alumno realizará la lectura de la práctica.
2. El alumno obtendrá las 5 ecuaciones de Boole que representan a cada una de las variables de salida de la tabla y comprobará que concuerdan con las ecuaciones del desarrollo. Hacer uso de herramientas computacionales para realizar las reducciones e incluir los pasos de solución en la actividad previa.

Material

- No aplica

Equipo

- 1 tarjeta CPLD DIONE.
- 1 cable USB tipo C a USB tipo A.
- 1 computadora con el software Quartus Prime Lite Edition 18.1 instalado.

Desarrollo experimental

1. Ejecute el software Quartus Prime Lite Edition 18.1 y genere un nuevo proyecto llamado “Teclado”.

2. Asigne las librerías, los puertos y genere una arquitectura (architecture) llamada “Decodificador” (Figura 5.3)

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  use ieee.std_logic_unsigned.all;
5
6  entity Teclado is
7
8     port (C1, C2, C3 : in std_logic;
9           R1, R2, R3, R4 : in std_logic;
10          S1, S2, S3, S4 : out std_logic;
11          Validador : out std_logic);
12
13    end;
14
15 architecture Decodificador of Teclado is
16
17 begin
18
```

Figura 5.3 Librerías, puertos y arquitectura en VHDL

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

3. Continue asignando las funciones de Boole como se muestra en la figura 5.4.

```
    S4 <= (C1 AND NOT C2 AND C3 AND R1 AND R2 AND NOT R3 AND R4) OR
          (C1 AND C2 AND NOT C3 AND R1 AND R2 AND NOT R3 AND R4) OR
          (NOT C1 AND C2 AND C3 AND R1 AND R2 AND R3 AND NOT R4) OR
          (C1 AND C2 AND NOT C3 AND R1 AND R2 AND R3 AND NOT R4);

    S3 <= (NOT C1 AND C2 AND C3 AND R1 AND NOT R2 AND R3 AND R4) OR
          (C1 AND NOT C2 AND C3 AND R1 AND NOT R2 AND R3 AND R4) OR
          (C1 AND C2 AND NOT C3 AND R1 AND NOT R2 AND R3 AND R4) OR
          (NOT C1 AND C2 AND C3 AND R1 AND R2 AND NOT R3 AND R4);

    S2 <= (C1 AND NOT C2 AND C3 AND NOT R1 AND R2 AND R3 AND R4) OR
          (C1 AND C2 AND NOT C3 AND NOT R1 AND R2 AND R3 AND R4) OR
          (C1 AND C2 AND NOT C3 AND R1 AND NOT R2 AND R3 AND R4) OR
          (NOT C1 AND C2 AND C3 AND R1 AND R2 AND NOT R3 AND R4) OR
          (NOT C1 AND C2 AND C3 AND R1 AND R2 AND R3 AND NOT R4) OR
          (C1 AND C2 AND NOT C3 AND R1 AND R2 AND R3 AND NOT R4);

    S1 <= (NOT C1 AND C2 AND C3 AND NOT R1 AND R2 AND R3 AND R4) OR
          (C1 AND C2 AND NOT C3 AND NOT R1 AND R2 AND R3 AND R4) OR
          (C1 AND NOT C2 AND C3 AND R1 AND NOT R2 AND R3 AND R4) OR
          (NOT C1 AND C2 AND C3 AND R1 AND R2 AND NOT R3 AND R4) OR
          (C1 AND C2 AND NOT C3 AND R1 AND R2 AND NOT R3 AND R4) OR
          (C1 AND C2 AND NOT C3 AND R1 AND R2 AND R3 AND NOT R4);

    Validador <= ((NOT C1 AND C2 AND C3) OR
                  (C1 AND NOT C2 AND C3) OR
                  (C1 AND C2 AND NOT C3)) AND
                  ((NOT R1 AND R2 AND R3 AND R4) OR
                  (R1 AND NOT R2 AND R3 AND R4) OR
                  (R1 AND R2 AND NOT R3 AND R4) OR
                  (R1 AND R2 AND R3 AND NOT R4));

end Decodificador;
```

Figura 5.4 Código a implementar en VHDL.

4. Realice la asignación de pines como se indica en la figura 5.5.

Named: * Edit: Filter: Pins: all

Node Name	Direction	Location	I/O Bank	itter Locatior	I/O Standard	Reserved	urrent Streng	ict Prese
A1	Input	PIN_1	2	PIN_1	3.3-V LVTTL		16mA ...ault)	
A2	Input	PIN_5	1	PIN_5	3.3-V LVTTL		16mA ...ault)	
A3	Input	PIN_8	1	PIN_8	3.3-V LVTTL		16mA ...ault)	
B1	Input	PIN_2	1	PIN_2	3.3-V LVTTL		16mA ...ault)	
B2	Input	PIN_6	1	PIN_6	3.3-V LVTTL		16mA ...ault)	
S1	Output	PIN_41	1	PIN_41	3.3-V LVTTL		16mA ...ault)	
S2	Output	PIN_40	1	PIN_40	3.3-V LVTTL		16mA ...ault)	
S3	Output	PIN_39	1	PIN_39	3.3-V LVTTL		16mA ...ault)	
S4	Output	PIN_37	1	PIN_37	3.3-V LVTTL		16mA ...ault)	
S5	Output	PIN_36	1	PIN_36	3.3-V LVTTL		16mA ...ault)	

Figura 5.5 Código a implementar en VHDL.

5. Sintetice, conecte la tarjeta CPLD y programe el dispositivo.

6. Compruebe el código de cada una de las teclas y el bit de validez de código.

Cuestionario

1. Indique cuales circuitos integrados tendrían que utilizarse para poder realizar las funciones que se implementaron en esta tarjeta.
2. ¿Cuál es la ventaja y desventaja de implementar este tipo de tarjetas en lugar de la circuitería convencional?
3. ¿Se podrían realizar las prácticas 1, 2 y 3 con el CPLD? (Justifique su respuesta)

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

Práctica 6. Control de entradas y salidas en una FPGA

Objetivos

- Programar una FPGA empleando lenguaje de descripción de hardware (VHDL).
- Conocer el entorno de desarrollo de Vivado 2022.1 y la tarjeta de desarrollo Nexys4 DDR o Nexys A7.

Introducción

En la actualidad existen diferentes Dispositivos Lógicos Programables (DLP o PLD por sus siglas en inglés) en donde cada uno de ellos cuenta con características muy específicas y dependerá de la situación que se presente el poder resolverla con algún determinado dispositivo.

Si el problema a resolver no requiere un gran procesamiento o uso de memoria, podemos elegir alguno de los PLD simples tales como PALs, PLAs o GALs, si la situación requiere un sistema un poco más complejo, se tendría que optar por un dispositivo lógico programable complejo CPLD y por último para diseños que involucren una mayor complejidad y necesidades de robustez sería conveniente utilizar un FPGA y esto va encaminado a la relación que debe haber entre el costo – beneficio.

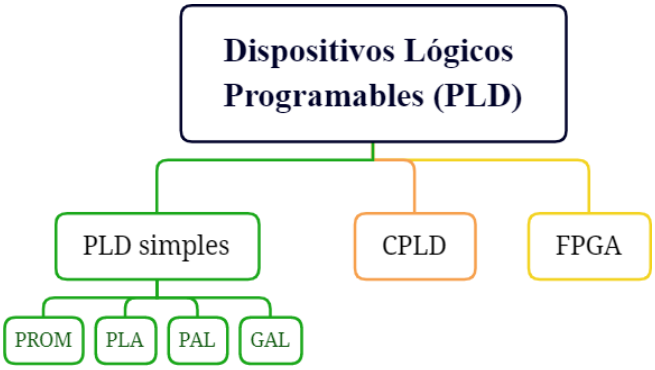


Figura 6.1 Clasificación de PLDs.

Actividades previas a la realización de la práctica

1. Realizar la lectura de la práctica de laboratorio.
2. Instalar en su PC personal el software de Vivado 2022.1 de acuerdo con las instrucciones descritas en: https://drive.google.com/file/d/1bvI1fx5o_JkwyzexfukJRKpw5afoftvz/view?usp=sharing
3. Instalar en su PC personal el software de la tarjeta DIGILENT en VIVADO de acuerdo con las instrucciones descritas en: https://drive.google.com/file/d/1_TbWXChnBu77Tx3szvvbI-5XkEdHW-A_/view?usp=sharing

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

4. Utilizando el software VIVADO desarrolle el proyecto mostrado en la figura 6.4 que se utilizará para programar la tarjeta de desarrollo Nexys4 DDR o Nexys A7 utilizando el procedimiento del primer proyecto con VIVADO descrito en la liga: https://drive.google.com/file/d/1kutnu5Q8sbqKGDTrhTlzubcjDwWnnmlK/view?usp=sharing
5. Descargar la carpeta generada por el proyecto en una USB para poder programar la tarjeta de desarrollo Nexys4 DDR o Nexys A7 utilizando la computadora del laboratorio.
6. Descargar la ficha técnica de la tarjeta Nexys4 DDR o Nexys A7 de Digilent.

Material

- No aplica.

Equipo

- 1 tarjeta Nexys4 DDR o Nexys A7.
- 1 computadora con Vivado 2022.1.

Procedimiento experimental

1. Siguiendo el proceso descrito en el manual de un nuevo proyecto en el software Vivado 2022.1, realice y cargue el código VHDL (figura 6.1) en la Nexys4 DDR o en la Nexys A7. https://drive.google.com/file/d/1kutnu5Q8sbqKGDTrhTlzubcjDwWnnmlK/view?usp=sharing.

```
--Con este programa se asignarán algunos Switches de la Nexys4 para poder
--controlar el encendido y apagado de los LEDs declarados como salidas

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity entradas_salidas is
    Port (--Asignación de entradas
            SW1: in std_logic;
            SW2: in std_logic;
            SW3: in std_logic;
            SW4: in std_logic;
          --Asignación de salidas
            LED1: out std_logic;
            LED2: out std_logic;
            LED3: out std_logic;
            LED4: out std_logic );

end entradas_salidas;

architecture Behavioral of entradas_salidas is

begin
--Asigna a LEDx 1 lógico cuando SWx sea igual a 1 lógico,
--de lo contrario, LEDx y SWx están en 0 lógico
    LED1 <= '1' when SW1 = '1' else '0';
    LED2 <= '1' when SW2 = '1' else '0';
    LED3 <= '1' when SW3 = '1' else '0';
    LED4 <= '1' when SW4 = '1' else '0';

end Behavioral;
```

Figura 6.1 Código a implementar

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

2. Anote el comportamiento de los LEDS en la tabla 6.1 y mencione que es lo que se está realizando.

ENTRADAS				SALIDAS			
SW4	SW3	SW2	SW1	LED4	LED3	LED2	LED1
0	0	0	0				
0	0	0	1				
0	0	1	0				
0	0	1	1				
0	1	0	0				
0	1	0	1				
0	1	1	0				
0	1	1	1				
1	0	0	0				
1	0	0	1				
1	0	1	0				
1	0	1	1				
1	1	0	0				
1	1	0	1				
1	1	1	0				
1	1	1	1				

Tabla 6.1 Accionamiento de Entradas, Salidas (casillas en blanco para anotar el comportamiento del LED).

Cuestionario

1. ¿Qué significan las siglas LVCMOS33 en la Nexys4 DDR o en la Nexys A7?

2. ¿Cuáles son las ventajas y desventajas (costos, tiempos de implementación, manejo de software, facilidad de armado, espacios, etc.) que se tienen al implementar este tipo de sistemas en una GAL, en un CPLD y cuales al utilizar un FPGA?

Práctica 7. Implementación de tablas de verdad.

Objetivos

- Implementar tablas de verdad de circuitos lógicos dentro de un dispositivo lógico programable considerando lenguaje VHDL.

Introducción

En esta práctica se realizarán 2 circuitos lógicos independientes empleando para ello la característica de concurrencia que tiene el lenguaje VHDL, el cual no ejecuta instrucciones de forma secuencial como lo hacen los circuitos microprocesadores y microcontroladores, sino de forma concurrente.

En este tipo de dispositivos el cambio en las variables de entrada se refleja inmediatamente en las variables de salida puesto que el sistema funciona como lo hace un circuito verdadero ya que este sistema no depende de la programación como lo hacen los microprocesadores o los microcontroladores.

En la figura 7.1 se muestra el esquema de conexión del conjunto de 8 displays de 7 segmentos que están integrados en la tarjeta de desarrollo NEXYS4, en donde cada uno de los ánodos está multiplexado en tiempo y las señales de cátodo son comunes a todos los dígitos, pero solo se pueden iluminar los segmentos del dígito cuya señal de ánodo correspondiente se decide utilizar.

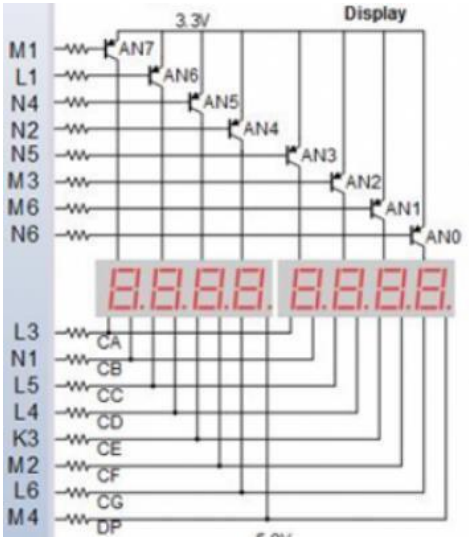


Figura 7.1 Conexión de los displays de 7 segmentos con la FPGA Artix-7

Para activar un segmento del display, el ánodo del display seleccionado debe estar en estado alto mientras que el cátodo del segmento correspondiente deberá estar en estado bajo.

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

Debido a que la Nexys4 DDR y la Nexys A7 usan transistores para conducir suficiente corriente al punto del ánodo común, las habilitaciones del ánodo se invierten a través de un transistor, por lo tanto, las señales AN7 – AN0 como CA – CG y DP se invierten cuando están activas.

El primer circuito desplegará los números decimales (0 a 9) en forma secuencial sobre cada uno de los displays de 7 segmentos comenzando del lado derecho y recorriéndolos hacia la izquierda, debido a que solo hay 8 displays los números 8 y 9 se desplegarán en sentido inverso. Debido a que el código BCD8421 tiene 6 códigos no válidos entonces se mostrará un guión en el display.

El segundo circuito mostrará los símbolos hexadecimales en 7 segmentos a partir de un número binario de 4 bits en el rango de 0 a F utilizando el mismo método de desplazamiento secuencial hacia la izquierda.

Actividades previas

1. El estudiante deberá leer la práctica de laboratorio.
2. Realizar la tabla de verdad para la conversión de código BCD8421 en 4 bits a código de 7 segmentos.
3. Realizar la tabla de verdad para la conversión de código binario de 4 bits a hexadecimal en 7 segmentos.
4. Generar el archivo binario_BCD en Vivado 2022.1.
5. Generar el archivo binario_hexadecimal en Vivado 2022.1.

Material

- No aplica

Equipo

- 1 tarjeta Nexys4 DDR o Nexys A7.
- 1 PC con Vivado 2022.1.

Procedimiento experimental

1. Utilizando el software de Vivado 2022.1, genere el proyecto en VHDL de la figura 7.2.

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity binario_BCD is
5      Port (binario: in std_logic_vector (3 downto 0);
6            BCD: out std_logic_vector (14 downto 0)
7            );
8  end binario_BCD;
9
10 architecture Behavioral of binario_BCD is
11
12 begin
13 -- Se ingresan datos en binario y se muestran en Hexadecimal
14 Entradas: process(binario) is
15     begin
16     case binario is
17         --    Bits             Display Segmentos    Dato a mostrar en el
18         --    8421             87654321ABCDEFG      display de 7 segmentos
19         when "0000" => BCD <= "111111100000001";    --0
20         when "0001" => BCD <= "111111011001111";    --1
21         when "0010" => BCD <= "111110110010010";    --2
22         when "0011" => BCD <= "111101110000110";    --3
23         when "0100" => BCD <= "111011111001100";    --4
24         when "0101" => BCD <= "110111110100100";    --5
25         when "0110" => BCD <= "101111110100000";    --6
26         when "0111" => BCD <= "011111110001111";    --7
27         when "1000" => BCD <= "101111110000000";    --8
28         when "1001" => BCD <= "110111110000100";    --9
29         when others => BCD <= "000000001111110";
30
31      end case;
32 end process Entradas;
33
34 end Behavioral;
```

Figura 7.2 Descripción en VHDL para BCD8421 a Display de 7 Segmentos.

2. Realice la asignación de puertos correspondiente.

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

3. Anote los resultados obtenidos en la tarjeta y comente acerca del comportamiento del sistema.

4. **Modifique el proyecto anterior para que la conversión se realice entre un número binario de 4 bits y un código hexadecimal** representado en un display de siete segmentos, realizando el corrimiento de derecha a izquierda de los números 0 al 7 y corrimiento de derecha a izquierda de los números 8 a F.

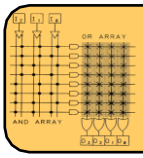
5. Anote los resultados obtenidos en la tarjeta y comente acerca del comportamiento del sistema.

Cuestionario

1. En los códigos descritos, ¿cómo se puede saber cuál segmento es el que se está activando y cuál es el que se está desactivando?

2. ¿Cuál es la diferencia entre usar std_logic_vector y std_logic?

3. Al utilizar std_logic_vector, ¿cuál es la diferencia entre utilizar downto y utilizar solamente to?



Práctica 8. Contadores Mod6 y Mod10.

Objetivos

- Implementar el contador Mod10 y Mod6 en una FPGA utilizando lenguaje VHDL.
- Comprobar el funcionamiento del contador.

Introducción

En la actualidad el diseño discreto de sistemas digitales muy grandes ya no es posible debido a la complejidad enorme de los sistemas actuales y es por eso que el lenguaje VHDL se emplea para describir circuitos en una forma más cercana al lenguaje humano y alejada de los múltiples detalles que hay que controlar en los sistemas de circuitos de SSI, MSI y VLSI.

VHDL es un lenguaje que tiene una sintaxis muy amplia y flexible y debido a ello es posible realizar diseños en forma estructural, en forma de flujo de datos y en forma comportamental.

Cada una de estas formas de diseño permite implementar a los sistemas digitales desde diferentes niveles de control y complejidad de los elementos que forman el sistema digital. Desde la forma más elemental de definición de terminales y compuertas hasta la forma más abstracta con elementos previamente diseñados y con el apoyo de lenguaje estructurado.

En esta práctica se utilizará el lenguaje VHDL para realizar la implementación de un contador en modulo 10 y otro en módulo 6 para producir un sistema que despliegue una numeración del 00 al 59 que se utilizarán para la realización de un reloj digital en la siguiente práctica.

El primer contador es un sistema de módulo 10 que se emplea para contar las unidades tanto de los segundos como las de los minutos (0 a 9).

El segundo contador es un sistema de módulo 6 que se emplea para contar las decenas tanto de los segundos como las de los minutos (0 a 5).

Actividades previas

1. El estudiante deberá leer la práctica de laboratorio.
2. Generar el archivo contador_Mod6 en Vivado 2022.1

Material

- No aplica.

Equipo

- 1 tarjeta Nexys4 DDR.
- 1 PC con Vivado 2022.1.

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

Procedimiento experimental

1. Abra el software de Vivado 2022.1 y genere un nuevo archivo llamado contador_Mod6.

2. Realice la descripción en VHDL tal como se muestra en la figura 8.1.

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_UNSIGNED.ALL;
4
5  entity contador_Mod6 is
6      Port (clk100mhz: in std_logic;
7            display: out STD_LOGIC_VECTOR(7 downto 0); -- PGFEDCBA (segmentos)
8            utilizar_display: out STD_LOGIC_VECTOR(7 downto 0) -- Display a utilizar (Selección)
9            );
10 end contador_Mod6;
11
12 architecture Behavioral of contador_Mod6 is
13
14     constant conteoMaximo: INTEGER := 50000000; -- 100,000,000 Hz/2 = 50000000 Hz
15     constant actMaxiConteo: INTEGER := 200000; -- 100 MHz/20,000 = 500 Hz
16
17     signal contador: INTEGER range 0 to conteoMaximo;
18     signal actualizar_conteo: INTEGER range 0 to actMaxiConteo;
19     signal estadoActualizado: STD_LOGIC_VECTOR(2 downto 0) := (others => '0');
20     signal clk_actual: STD_LOGIC := '0';
21     signal display_sel: STD_LOGIC_VECTOR(7 downto 0) := (others => '0');
22
23     shared variable segundos1: INTEGER range 0 to 10 := 0;
24     shared variable segundos2: INTEGER range 0 to 6 := 0;
25
26     function digito(numero:INTEGER) return STD_LOGIC_VECTOR is variable salida: STD_LOGIC_VECTOR(7 downto 0);
27
28 begin
29     case numero is
30         when 0 => salida := "11000000"; -- 0
31         when 1 => salida := "11111001"; -- 1
32         when 2 => salida := "10100100"; -- 2
33         when 3 => salida := "10110000"; -- 3
34         when 4 => salida := "10011001"; -- 4
35         when 5 => salida := "10010010"; -- 5
36         when 6 => salida := "10000010"; -- 6
37         when 7 => salida := "11111000"; -- 7
38         when 8 => salida := "10000000"; -- 8
39         when 9 => salida := "10010000"; -- 9
40         when others => salida := "11111111";
41     end case;
```

Figura 8.1 Descripción en VHDL para el Mod6 (Parte 1).

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

```
42      return(salida);
43    end digito;
44
45    begin
46      utilizar_display <= display_sel;
47
48      gen_clk: process(clk100mhz, clk_actual, contador)
49
50      begin
51      -- Proceso secuencial síncrono (flanco de subida)
52      if clk100mhz'event and clk100mhz='1' then
53          -- contador 1HZ
54          if contador < conteoMaximo then
55              contador <= contador + 1;
56          else
57              clk_actual <= not clk_actual;
58              contador <= 0;
59          end if;
60
61          -- contador 500Hz (para actualización del display)
62          if actualizar_conteo < actMaxiConteo then
63              actualizar_conteo <= actualizar_conteo + 1;
64          else
65              estadoActualizado <= estadoActualizado + 1;
66              actualizar_conteo <= 0;
67                  end if;
68          end if;
69      end process;
70
71    show_display: process(estadoActualizado)
72          begin -- selección del display
73              case estadoActualizado is
74                  when "000" => display_sel <= "01111111"; -- display 0
75                  when "001" => display_sel <= "10111111"; -- display 1
76                  when "010" => display_sel <= "11011111"; -- display 2
77                  when "011" => display_sel <= "11101111"; -- display 3
78                  when "100" => display_sel <= "11110111"; -- display 4
79                  when "101" => display_sel <= "11111011"; -- display 5
80                  when others => display_sel <= "11111111";
81              end case;
```

Figura 8.1 Descripción en VHDL para el Mod6 (Parte 2).

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

```
82
83              -- Formato del reloj: HH:MM:SS
84              case display_sel is
85                  when "11110111" => display <= digito(segundos2);     -- display 3
86                  when "11111011" => display <= digito(segundos1);     -- display 2
87                  when others => display <= "11111111";
88          end case;
89      end process;
90
91      persecond: process (clk_actual)
92      begin
93        if clk_actual'event and clk_actual='1' then
94
95          -- Si el digito de segundos2 es menor a 6 incrementa el segundo1
96          if segundos2 < 6 then
97              segundos1 := segundos1 + 1;
98          end if;
99
100         --Si segundos1 es igual a 10 incrementa el segundo2
101         if segundos1 = 10 then
102            segundos2 := segundos2 + 1;
103            segundos1 := 0;
104         end if;
105
106         -- Si segundos2 = 6 se reinicia el conteo
107         if segundos2 = 6 then
108             segundos1 := 0;
109             segundos2 := 0;
110         end if;
111       end if;
112     end process;
113 end Behavioral;
```

Figura 8.1 Descripción en VHDL para el Mod6 (Parte 3).

3. Realice la asignación de puertos como se muestra en la figura 8.2.

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

Cuestionario

1. ¿Qué función tiene la palabra reservada ‘event and?
2. ¿Cuál es la frecuencia del reloj interno con la que cuenta la Nexys4 DDR o la Nexys A7?
3. ¿Por qué se pide realizar la reducción para la frecuencia del reloj interno?

Práctica 9. Reloj Digital de 24 horas.

Objetivos

- Diseñar un reloj digital de 24 horas en VHDL y que se muestre en los displays de 7 segmentos de la NExys4 DDR o Nexys A7.
- Variar la frecuencia de reloj para poder ver los cambios en las horas, minutos y segundos.

Introducción

Los dispositivos lógicos programables permiten reducir el tamaño de la electrónica digital necesaria para la implementación de un sistema.

En la actualidad existen dispositivos PLDs de muy alta escala de integración y un número muy grande de terminales de entrada y salida que permiten integrar dentro de ellos sistemas digitales muy complejos, tales como los dispositivos FPGAs y los CPLDs que tienen miles de terminales.

En esta práctica se realizará la implementación de un reloj digital de 24 horas utilizando los contadores diseñados en la práctica 8. Para poder visualizar el reloj programaremos la tarjeta Nexys 4 DDR o Nexys A7, la cual cuenta con 8 displays de 7 segmentos de los cuales los primeros 2 displays permanecerán apagados, mientras que los otros 6 mostrarán horas, minutos y segundos en el formato HH:MM:SS (23:59:59).

El diseño deberá integrar un módulo que cuente los segundos en el rango de 0 a 59 con un reloj de entrada de 1Hz. y controlando el reloj del siguiente módulo, el cual deberá contar los minutos en el rango de 0 a 59 y controlando al tercer y último módulo que deberá contar las horas en un rango de 0 a 23.

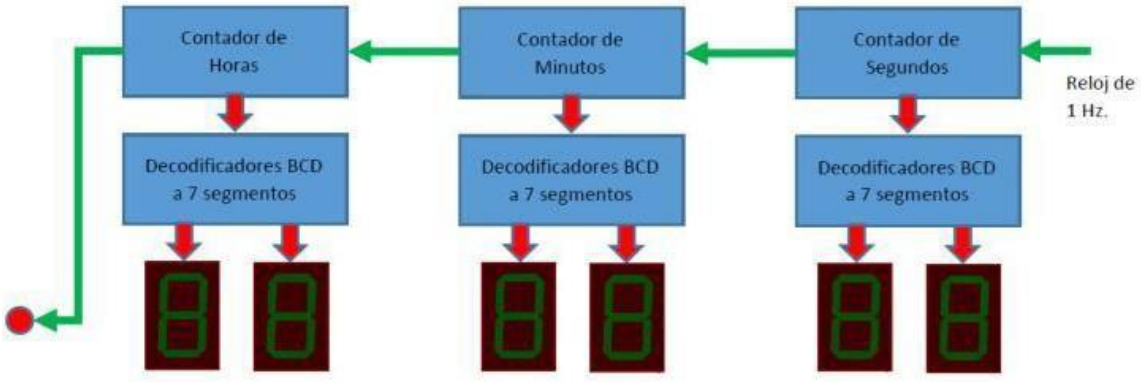


Figura 9.1

Actividades previas

1. El estudiante deberá leer la práctica de laboratorio.
2. Generar el archivo reloj_24h en Vivado 2022.1

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

Material

- No aplica.

Equipo

- 1 tarjeta Nexys4 DDR.
- 1 PC con Vivado 2022.1.

Procedimiento experimental

1. Abra el software de Vivado 2022.1 y genere un nuevo archivo llamado reloj_24h.

2. Copie el código de la figura 9.2.

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_UNSIGNED.ALL;--Declaramos la libreria para poder utilizar el operador + y -
4
5  entity reloj_24h is
6      Port (clk100mhz: in STD_LOGIC; --Utilizamos el Reloj de la Nexys 4 que trabaja a 100 MHz
7            display_Catodo: out STD_LOGIC_VECTOR(7 downto 0); -- "PGFEDCBA" segmentos del display
8            display_Anodo: out STD_LOGIC_VECTOR(7 downto 0) -- Display a utilizar
9            );
10 end reloj_24h;
11
12 architecture Behavioral of reloj_24h is
13
14     -- CONSTANTES
15     constant conteoMaximo: INTEGER := 50000000; -- 100,000,000 Hz/2 = 50000000 Hz
16     constant actMaxiConteo: INTEGER := 200000; -- 100 MHz/20,000 = 500 Hz
17
18     -- SEÑALES
19     signal contador: INTEGER range 0 to conteoMaximo;
20     signal actualizar_conteo: INTEGER range 0 to actMaxiConteo;
21     signal clk_actual: STD_LOGIC := '0';
22     signal estadoActualizado: STD_LOGIC_VECTOR(2 downto 0) := (others => '0');
23     signal selector_Anodo: STD_LOGIC_VECTOR(7 downto 0) := (others => '0');
24
25     -- VARIABLES COMPARTIDAS
26     shared variable hora1, hora2, minuto1, minuto2, segundo1: INTEGER range 0 to 10 := 0;
27     shared variable segundo2: INTEGER range 0 to 6 := 0; -- Conteo de 0 a 6
28
29     -- FUNCIÓN
30     function digito(valor_entero:INTEGER) return STD_LOGIC_VECTOR is variable salida: STD_LOGIC_VECTOR(7 downto 0);
31
32 begin
33     case valor_entero is
34        --       Orden de los segmentos del display
35        --                    PGFEDCBA
36         when 0 => salida := "11000000";       -- 0
37         when 1 => salida := "11111001";       -- 1
38         when 2 => salida := "10100100";       -- 2
39         when 3 => salida := "10110000";       -- 3
40         when 4 => salida := "10011001";       -- 4
41         when 5 => salida := "10010010";       -- 5
```

Figura 9.2 Descripción en VHDL para el reloj de 24 horas usando los displays de 7 segmentos (Parte 1).

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

```
42          when 6 => salida := "10000010";       -- 6
43          when 7 => salida := "11111000";       -- 7
44          when 8 => salida := "10000000";       -- 8
45          when 9 => salida := "10010000";       -- 9
46          when others => salida := "11111111";
47      end case;
48    return(salida);
49  end digito;
50
51    begin
52      display_Anodo <= selector_Anodo;
53
54      gen_clock: process(clk100mhz, clk_actual, contador)
55
56      begin
57      -- Proceso secuencial síncrono (flanco de subida)
58      if clk100mhz'event and clk100mhz='1' then
59          -- Funciones para el contador de 1HZ
60          if contador < conteoMaximo then
61              contador <= contador + 1;
62          else
63              clk_actual <= not clk_actual;
64              contador <= 0;
65          end if;
66
67          -- reloj de 500Hz para actualizar el display
68          if actualizar_conteo < actMaxiConteo then
69              actualizar_conteo <= actualizar_conteo + 1;
70          else
71              estadoActualizado <= estadoActualizado + 1;
72              actualizar_conteo <= 0;
73                  end if;
74          end if;
75      end process;
76
77    asignacion_Display: process(estadoActualizado)
78          begin -- Asignamos los display que se van a utilizar
79              case estadoActualizado is
80                  when "000" => selector_Anodo <= "01111111"; -- display 0
81                  when "001" => selector_Anodo <= "10111111"; -- display 1
82                  when "010" => selector_Anodo <= "11011111"; -- display 2
```

Figura 9.2 Descripción en VHDL para el reloj de 24 horas usando los displays de 7 segmentos (Parte 2).

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

```
83                  when "011" => selector_Anodo <= "11101111"; -- display 3
84                  when "100" => selector_Anodo <= "11110111"; -- display 4
85                  when "101" => selector_Anodo <= "11111011"; -- display 5
86                  when others => selector_Anodo <= "11111111";
87              end case;
88
89              -- Formato del reloj: HH:MM:SS
90              case selector_Anodo is
91                  when "01111111" => display_Catodo <= digito(hora2);      -- display 7
92                  when "10111111" => display_Catodo <= digito(hora1);      -- display 6
93                  when "11011111" => display_Catodo <= digito(minuto2);    -- display 5
94                  when "11101111" => display_Catodo <= digito(minuto1);    -- display 4
95                  when "11110111" => display_Catodo <= digito(segundo2);   -- display 3
96                  when "11111011" => display_Catodo <= digito(segundo1);   -- display 2
97                  when others => display_Catodo <= "11111111";
98          end case;
99      end process;
100
101     estadoReloj: process (clk_actual)
102     begin
103       if clk_actual'event and clk_actual='1' then
104
105           -- Si el digito de segundos2 es menor a 6 incrementa el segundo1
106           if segundo2 < 6 then
107               segundo1 := segundo1 + 1;
108           end if;
109
110           --Si segundos1 es igual a 10 incrementa el segundo2
111           if segundo1 = 10 then
112              segundo2 := segundo2 + 1;
113              segundo1 := 0;
114           end if;
115
116           -- primer digito hora
117           if segundo2 = 6 and minuto1 < 10 then
118               minuto1 := minuto1 + 1;
119               segundo1 := 0;
120               segundo2 := 0;
121           end if;
122
```

Figura 9.2 Descripción en VHDL para el reloj de 24 horas usando los displays de 7 segmentos (Parte 3).

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.

```
123          -- segundo digito minutero
124          if minuto1 = 10 then
125              minuto2 := minuto2 + 1;
126              minuto1 := 0;
127          end if;
128
129          -- primer digito hora
130          if minuto2 = 6 and hora1 < 10 then
131              hora1 := hora1 + 1;
132              minuto2 := 0;
133              minuto1 := 0;
134          end if;
135
136          -- segundo digito hora
137          if hora1 = 10 then
138              hora2 := hora2 + 1;
139              hora1 := 0;
140          end if;
141
142          if hora2=2 and hora1=4 then
143              hora2 := 0;
144              hora1 := 0;
145          end if;
146      end if;
147    end process;
148
149  end Behavioral;
```

Figura 9.2 Descripción en VHDL para el reloj de 24 horas usando los displays de 7 segmentos (Parte 4).

3. Realice la asignación de puertos correspondiente.

4. Cambie la frecuencia del reloj, de modo que se vea la diferencia al momento de que el sistema realice el conteo y mencione a que frecuencia elegida.

Cuestionario

1. ¿Qué diferencias en general se pueden apreciar al implementar estos sistemas en una FPGA?

2. ¿Qué es el diseño Top Down en VHDL?

3. ¿La descripción de este circuito de reloj de 24 h. podría mejorarse implementando el diseño Top Down?

Mtro. Jorge Buendía Gómez.
Ing. Jonathan Fuentes Euan
Dr. Fernando Gudiño Peñaloza.
2025-1.