Sistemas digitales y electrónica digital

Prácticas de laboratorio

PEARSON Prentice Hall ®

Juan Ángel Garza Garza

INCLUYE CD-ROM

Sistemas digitales y electrónica digital, prácticas de laboratorio

Sistemas digitales y electrónica digital, prácticas de laboratorio

Juan Ángel Garza Garza

Facultad de Ingeniería Mecánica y Eléctrica

Universidad Autónoma de Nuevo León

REVISIÓN TÉCNICA:

José Ignacio Vega Luna
Universidad Autónoma Metropolitana
Azcapotzalco.

Ricardo A. Gálvez Orozco
Instituto Tecnológico de Pachuca

Agustín Ramírez Agundis
Instituto Tecnológico de Celaya

PEARSON Educación ®

México • Argentina • Brasil • Colombia • Costa Rica • Chile • Ecuador
España • Guatemala • Panamá • Perú • Puerto Rico • Uruguay • Venezuela

Datos de catalogación bibliográfica

GARZA GARZA, JUAN ÁNGEL
Sistemas digitales y electrónica digital, prácticas de laboratorio. Primera edición

PEARSON EDUCACIÓN, México, 2006

ISBN: 970-26-0719-1
Área: Bachillerato

Formato: 18.5 × 23.5 cm Páginas: 352

Editor: Pablo Miguel Guerrero Rosas
e-mail: pablo.guerrero@pearsoned.com
Supervisor de desarrollo: Felipe Hernández Carrasco
Supervisor de producción: José D. Hernández Garduño

PRIMERA EDICIÓN, 2006

ISBN 970-26-0719-1

Impreso en México. *Printed in Mexico.*

1 2 3 4 5 6 7 8 9 0 - 09 08 07 06

Contenido

Agradecimientos

Un agradecimiento especial de parte del autor a las siguientes personas por sus valiosos comentarios para la publicación de esta obra:

Guadalupe Ignacio Cantú Garza.
Facultad de Ingeniería Mecánica y Eléctrica
Universidad Autónoma de Nuevo León.

Julián Eduardo Hernández Venegas.
Facultad de Ingeniería Mecánica y Eléctrica
Universidad Autónoma de Nuevo León.

Sergio Martínez Luna
Facultad de Ingeniería Mecánica y Eléctrica
Universidad Autónoma de Nuevo León..

Rogelio Guillermo Garza Rivera.
Director de la Facultad de Ingeniería Mecánica y Eléctrica.
Universidad Autónoma de Nuevo León.

José Antonio González Treviño.
Rector.
Universidad Autónoma de Nuevo León.

Soy esclavo de
mis propias ideas

Presentación

El objetivo principal de este libro de prácticas es brindar un recurso para adquirir conocimientos y desarrollar habilidades en el diseño de sistemas digitales. Está dirigido tanto a estudiantes como a profesionales de las carreras de ingeniería relacionadas con el área de electrónica digital y sistemas digitales.

En este libro se aplica el uso de nuevas tecnologías, como son los lenguajes de descripción de hardware (HDL), programas de captura esquemática y la implementación física mediante dispositivos lógicos programables (PLD).

Las prácticas están diseñadas de manera que permitan al estudiante reforzar el aprendizaje, extender los conocimientos conceptuales, desarrollar habilidades y obtener conocimientos, necesarios en su formación para el ejercicio de su profesión.

Se ha puesto mucho cuidado en asegurar que las prácticas sean útiles, pertinentes, realizables y estimulen el interés por el estudio de la materia.

El material usado en estas prácticas fue seleccionado para que esté al alcance de la economía del estudiante universitario y no sea necesario hacer una inversión significativa. El software utilizado es posible obtenerlo en forma gratuita en Internet o en un disco compacto distribuido por el fabricante con una licencia de uso por seis meses.

La idea de estas prácticas es que el alumno en forma individual o colaborativa fabrique, con las herramientas actuales de diseño, un circuito integrado a la medida (ASIC), lo cual le permitirá ahorrar tiempo, espacio y conexiones.

En el libro se propone una metodología de diseño combinacional que puede partir desde la descripción del problema, la tabla de verdad o las ecuaciones; asimismo, se propone la metodología para el diseño secuencial síncrono basada en cinco pasos: ya sea el modelo de Mealy o el de Moore. Para la solución de los sistemas secuenciales asíncronos se ofrece un método donde los resultados se implementan en un controlador lógico programable (PLC).

En el CD se encuentran: el software de diseño *ispLEVER starter* de la compañía *Lattice semiconductor*, crucigramas, presentaciones, manuales y dispositivos, entre otros recursos.

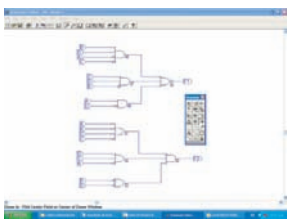
Introducción

En la actualidad el diseño de los sistemas digitales se simplifica gracias a los avances en las computadoras personales, las cuales son muy versátiles y poderosas. El software es muy amigable y está disponible en un ambiente de ventanas; además se cuenta con ayudas visuales en caso de algún error. Por su parte, los dispositivos electrónicos digitales son económicos y programables a la medida.

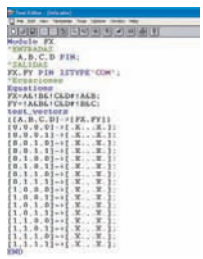
En la computadora personal para el diseño digital contamos con herramientas de captura esquemática (*schematic*) y el lenguaje de descripción de hardware (HDL) ABEL-HDL, la implementación física del diseño en un dispositivo lógico programable (PLD), en particular los dispositivos GAL16V8 y GAL22V10 que, por su versatilidad y capacidad, permiten implementar una gran variedad de diseños y aplicaciones de la lógica combinacional y secuencial, utilizando el mismo dispositivo en todas las prácticas (es reprogramable).

Para el desarrollo de las prácticas, es necesario contar con una computadora personal donde se instalará el programa compilador *Isp-Starter* de la compañía Lattice Semiconductor. Para este programa, se puede obtener una licencia gratuita en la página de Internet www.latticesemi.com.

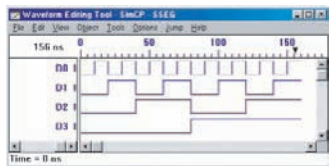
Dentro del programa *Isp-Starter* se incluyen los programas:



Schematic (captura esquemática), donde el diseño se representa usando símbolos.



Text Editor (editor de texto), para capturar el código en el lenguaje ABEL-HDL.



Waveforms (generador de diagramas de tiempo), para comprobar el funcionamiento del diseño, antes de implementarse físicamente.

Para la programación de los circuitos integrados es necesario un programador compatible con archivos en formato JEDEC, que soporte dispositivos lógicos programables como el GAL (arreglo lógico genérico).

Las prácticas fueron seleccionadas para sincronizar el laboratorio con los temas que se estudian en clase y así comprobar los conceptos propuestos en clase.

Material necesario para el desarrollo de las prácticas

Cantidad	Descripción
30	**Resistencias de 330 a 1/4 W**
14	**LED de 5 mm económicos de diferentes colores: ámbar, rojos y verdes.**
2	***Display* de 7 segmentos (ánodo o cátodo común)**
2	**Circuitos integrados decodificador de BCD a 7 segmentos SN 7447 (ánodo) o SN7448 (cátodo)**
1	**Tablilla de conexiones protoboard, 1 bloque 2 tiras**
1	**Metro de cable para alambrar calibre 24 o 26**
1	**DIP *switch* deslizable (8 interruptores deslizables)**
4	***Switch Push Micro NO* (interruptor de no retención normalmente abierto)**
1	**GAL16V8D o GAL22V10 (Lattice, Atmel o Cypress) o equivalente**
1	**Regulador 7805**
1	**Pila de 9V**
1	**Portapila para pila de 9V**
1	**Disco de 3.5 pulgadas de alta densidad formateado a 1.44 MB**
1	**Transistor 2N2222**

Para las prácticas 9 y 10 es necesaria una señal de sincronía de onda cuadrada (de preferencia de frecuencia variable) y un máximo de 5 volts de amplitud, que se puede obtener armando el circuito Timer, el cual se puede implementar con el material que se lista a continuación:

Cantidad	Descripción
4	**Resistencias de 1K a 1/4 W**
2	**Capacitores de 0.1 uF cerámico**
1	**Capacitor de 22 uF a 63V electrolítico**
1	**Potenciómetro de 100K tipo preset vertical**
1	**TIMER NE555V.**

En la Práctica 1 se explica el uso de la tablilla de conexiones y de la fuente de voltaje, así como la implementación de los circuitos de entrada y salida.

En la Práctica 2 se comprueban las tablas de verdad de los operadores *And, Or, Nand, Nor, Exor* de dos entradas, implementados con circuitos integrados de función fija TTL. Además, se comprueban algunas identidades del álgebra booleana. También se incluyen procedimientos y recomendaciones para el caso de falla. En esta práctica el alumno se familiariza con los circuitos integrados de función fija. En la Práctica 3 comprobarán las ventajas del uso de los DLP (dispositivos lógicos programables).

La Práctica 3 consiste en la implementación en un solo circuito integrado PLD con los operadores *And, Or, Nand, Nor, Exor* de tres entradas, usando el programa *Schematic* (captura esquemática); además de comprobar su tabla de verdad.

En la Práctica 4 se efectúa la simulación del circuito propuesto, obteniendo el diagrama de tiempos por medio de un archivo con formato ABEL Test_Vectors. Se propone un método analítico para obtener la tabla de verdad partiendo del diagrama esquemático (circuito).

La Práctica 5 consiste en la obtención de funciones booleanas a partir de la tabla de verdad y sus diferentes representaciones en el lenguaje ABEL-HDL.

Se proponen una metodología del diseño combinacional y 16 ejemplos a resolver siguiendo dicha metodología, para implementarlos en un dispositivo lógico programable usando el lenguaje de descripción de hardware ABEL-HDL, ya sea por ecuaciones, tabla de verdad o la descripción del problema. Además se incluye un resumen con un ejemplo completo del diseño e implementación de un multiplexor de 8 a una línea, resuelto de cuatro maneras diferentes.

En la Práctica 6 se propone la solución de sistemas combinacionales que no están totalmente especificados, y se incluye como ejemplo el diseño de un decodificador de BCD a siete segmentos, que parte de una metodología para la identificación de cada uno de los segmentos. Se expone la programación del dispositivo usando ABEL-HDL por tabla de verdad e incluyendo la instrucción *DC* (*Don't care*); además se incluyen ejemplos de convertidores de código binarios y decimales.

En la Práctica 8, se estudian la teoría básica de un Flip Flop partiendo de una estación de botones de paro y arranque, y la implementación de un circuito eliminador de rebotes, así como las tablas características de los Flip Flops, JK, RS, T y D.

En la Práctica 9, se propone una metodología del diseño secuencial síncrono en los modelos de Mealy y Moore, así como ejemplos a resolver por diferentes métodos usando ABEL-HDL, ecuaciones, tabla de estados, descripción del diagrama de transición o captura esquemática.

En la Práctica 10, se presentan problemas resueltos de diferentes tipos de contadores, como binarios, de décadas (BCD), ascendentes y/o descendentes, etcétera.

En la Práctica 11, se exponen los sistemas secuenciales asíncronos y se presenta un método en *n* pasos, que es el método que propone Charles H. Roth en el libro *Fundamentals of Logic Design*.

PRÁCTICA 1

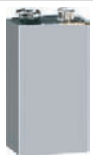
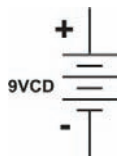
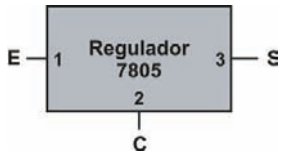
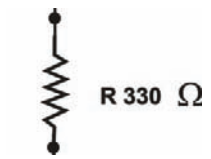
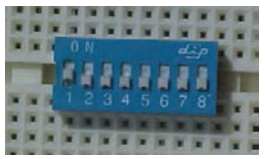
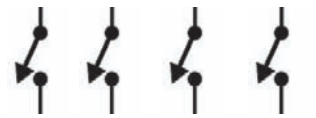
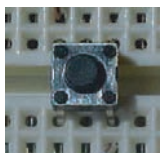
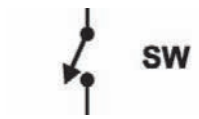
Introducción al laboratorio

Objetivos particulares

En esta práctica se explica el uso de la tablilla de conexiones (*protoboard*), y se describe una opción de fuente de alimentación usando una pila de 9 volts y el regulador 7805, así como la aplicación de los diodos emisores de luz (LED), que son esenciales para visualizar los valores de entrada y salida de los circuitos, incluyendo los interruptores y las resistencias necesarios para su adecuado funcionamiento.

El tiempo estimado para el estudio de esta práctica es de dos horas.

Material necesario para el desarrollo de esta práctica

Portapila		
Pila de 9 VCD		
Regulador de voltaje 7805		
Tres diodos emisores de luz. (Ver Apéndice A).		
Tres resistencias de 330 Ω 1/4 de W (naranja, naranja, café).		
Un *dip switch*.		
Tres *push buttons*.		

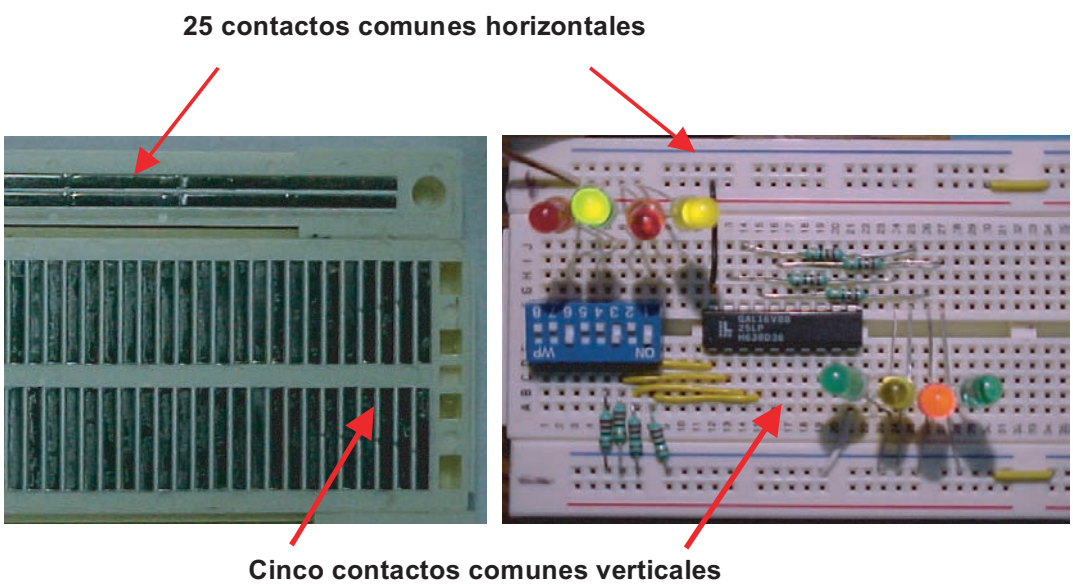
Fundamento teórico

1

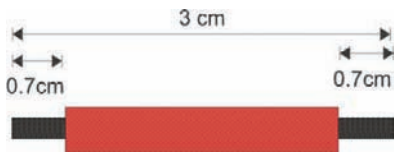
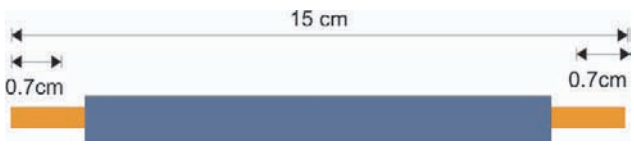
Distribución de la tablilla de conexiones (*protoboard*)

La tablilla de conexiones está construida por un bloque central y dos tiras en los extremos. El bloque central está formado por grupos de cinco contactos conectados en común, divididos por una canaleta central, de manera que cuando un componente o dispositivo se inserta en la tablilla, quedan cuatro contactos libres para interconexiones con las terminales del dispositivo.

En las tiras de los extremos hay ocho grupos de 25 contactos comunes, las cuales son convenientes para señales como VCD (voltaje de corriente directa o positivo), GND (tierra o negativo) o cualquier señal que requiera más de cinco contactos comunes. Es recomendable usar terminales o alambre de calibre 24 o 26 para la interconexión, ya que usar alambre de calibre más grueso muy probablemente dañaría los contactos de la tablilla de terminales.



Para la interconexión de los elementos del circuito dentro de la tablilla de conexiones, se recomienda preparar alambres descubriendo la parte metálica de los extremos con las siguientes medidas:

10 alambres de 3 cm.	
15 alambres de 5 cm.	
10 alambres de 10 cm.	
10 alambres de 15 cm.	

En los cuales los extremos deberán estar descubiertos por lo menos 0.7 cm con el propósito de establecer una buen contacto en la tablilla de conexiones, ya que el plástico que lo cubre es aislante.

Para descubrir los extremos se recomienda utilizar un par de pinzas: la primera pinza para sujetar firmemente el alambre; y la segunda, para cortar sólo el plástico y estirar.

Fuente de alimentación de 5 VCD

Para su funcionamiento la mayoría de los circuitos utilizados en las prácticas de este libro requieren de una alimentación de 5 volts de corriente directa; además, se recomienda utilizar un regulador de voltaje 7805 para asegurar un voltaje máximo de 5 volts, con el propósito de no exceder el límite definido por los fabricantes y evitar dañar el dispositivo.

Un regulador de voltaje mantiene el voltaje de salida de una fuente de CD en un valor constante (idealmente), aunque varía la carga. Si ésta excede el límite permitido como en el caso de un corto circuito, el regulador ofrece protección interrumpiendo el suministro de voltaje.

ELECTRICAL CHARACTERISTICS FOR L7805 (refer to the test circuits, T_j = -55 to 150 °C, V_i = 10V, I_o = 500 mA, C_i = 0.33 µF, C_o = 0.1 µF unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_o	Output Voltage	T_J = 25 °C	4.8	5	5.2	V
V_o	Output Voltage	I_o = 5 mA to 1 A $P_o \leq$ 15 W V_i = 8 to 20 V	4.65	5	5.35	V
ΔV_o*	Line Regulation	V_i = 7 to 25 V T_j = 25 °C V_i = 8 to 12 V T_j = 25 °C		3 1	50 25	mV mV
ΔV_o*	Load Regulation	I_o = 5 to 1500 mA T_j = 25 °C I_o = 250 to 750 mA T_j = 25 °C			100 25	mV mV
I_d	Quiescent Current	T_j = 25 °C			6	mA
ΔI_d	Quiescent Current Change	I_o = 5 to 1000 mA			0.5	mA
ΔI_d	Quiescent Current Change	V_i = 8 to 25 V			0.8	mA
$\frac{\Delta V_o}{\Delta T}$	Output Voltage Drift	I_o = 5 mA		0.6		mV/°C
eN	Output Noise Voltage	B = 10Hz to 100KHz T_j = 25 °C			40	µV/V_O
SVR	Supply Voltage Rejection	V_i = 8 to 18 V f = 120 Hz	68			dB
V_d	Dropout Voltage	I_o = 1 A T_j = 25 °C		2	2.5	V
R_o	Output Resistance	f = 1 KHz		17		mΩ
I_{sc}	Short Circuit Current	V_i = 35 V T_j = 25 °C		0.75	1.2	A
I_{scp}	Short Circuit Peak Current	T_j = 25 °C	1.3	2.2	3.3	A

Trabajo solicitado

1. Armar en la tablilla de conexiones los siguientes circuitos:

a) Fuente de alimentación 5 volts de corriente directa.

b) Circuito para visualizar los valores de entrada.

c) Circuito para visualizar los valores de salida.

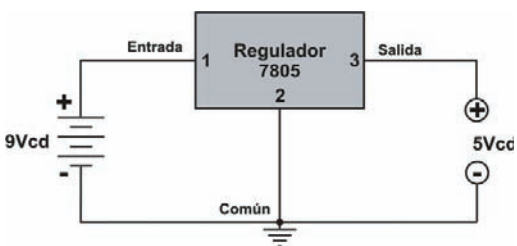
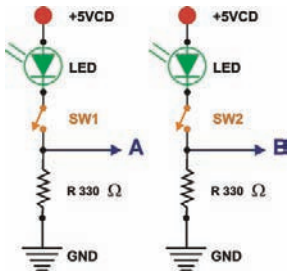
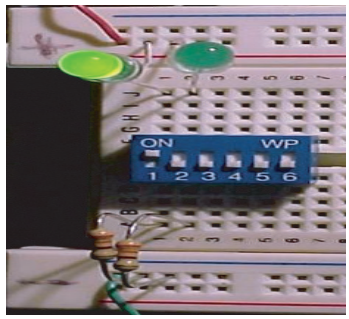


Diagrama eléctrico de la fuente de alimentación.

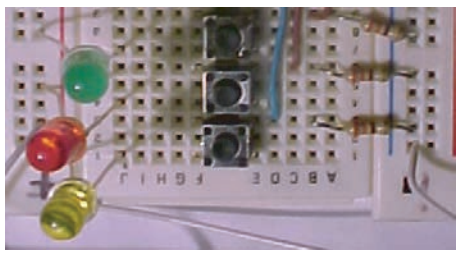
Diagrama eléctrico del circuito de entrada.



Fotografía del circuito de entrada usando interruptores deslizables de dos hileras de terminales DIP switch (*deslizable 6*)

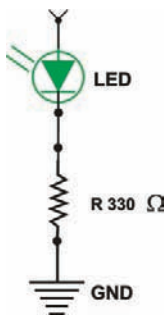


Fotografía del circuito de entrada usando interruptores de no retención normalmente abiertos push micro NO

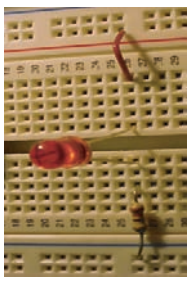


Circuito de salida:

Diagrama eléctrico del circuito de salida.



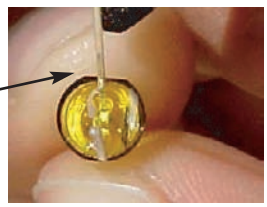
Fotografía del circuito de salida.



Procedimiento

Circuito de entrada usando el DIP *switch*

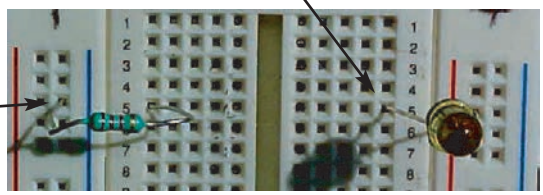
1. Identifique las terminales del diodo emisor de luz ánodo y cátodo.



Si observamos el contorno inferior del encapsulado del LED notaremos una parte plana, como lo indica la figura. La terminal del lado plano es el cátodo y la otra terminal es el ánodo.

2. Conecte el ánodo del LED en una de las tiras de la tablilla de conexiones que corresponde al positivo de la fuente (+5VCD); y el cátodo del LED, en una de las hileras del bloque central.

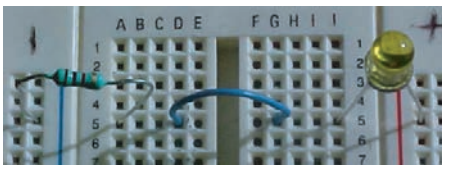
3. A continuación conectamos la resistencia de 330 K (franjas naranja, naranja y café) en la tablilla de conexiones, donde una de sus terminales se coloca en una tira del extremo que corresponderá a la tierra (GND); y la otra terminal, en la misma fila central donde colocamos el LED (paso anterior). Cabe mencionar que en las hileras centrales, los comunes están representados en forma horizontal y la función de la resistencia es limitar la corriente que pasará a través del LED, ya que si éste lo conectáramos directamente a la fuente, posiblemente el LED se quemaría por no tener un límite de corriente.



4. Probamos tres opciones para efectuar la conexión y cerrar el circuito entre el LED y la resistencia.

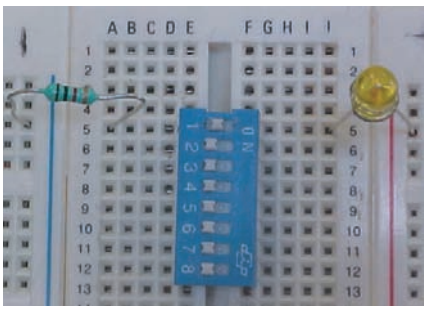
a) *Cable de conexión:*

La interconexión de un LED y una resistencia se realiza fácilmente usando un alambre, como lo representa la siguiente figura.



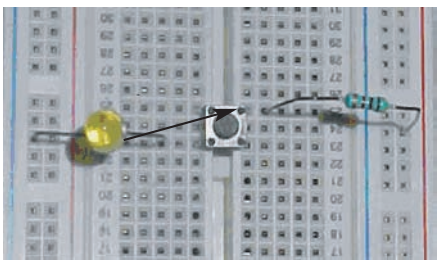
b) DIP switch:

Si ahora deseamos conectar una mayor cantidad de LEDS y resistencias, utilizamos un DIP *switch*, el cual consta de 8 switchs, que se pueden utilizar de uno en uno, o bien, todos a la vez. En la siguiente figura se representa en su conexión más simple.



c) Push button

Otra manera de instaurar el circuito anterior es mediante el uso de un *push button*. Al oprimir el botón se cierra el circuito, permitiendo el paso de la corriente. Note que la conexión de este dispositivo es en forma diagonal.



Cuestionario

1. ¿Qué es un LED?

2. Dibuje el circuito para la conexión de un LED.

3. ¿Cuál es la ecuación para determinar la corriente de un LED?

4. ¿Qué pasaría con un LED si se conecta directo a la fuente sin resistencia?

5. ¿De qué depende la intensidad luminosa de un LED?

6. ¿Cuál es el voltaje en terminales de la resistencia de 330 Ω del circuito de entrada?

7. ¿A qué rango de voltaje se le considera 1 lógico?

8. ¿A qué rango de voltaje se le considera 0 lógico?

Recomendaciones

1. Considere que el plástico del cable no es conductor y que sólo la parte metálica del extremo es la que se debe introducir para hacer contacto con la tablilla de conexiones.
2. En caso de que algún LED no encienda, confirme que el LED esté en la posición correcta y la resistencia sea de 330Ω (naranja, naranja, café).
3. Si lo anterior está correcto y aún no enciende el LED, usando un multímetro verifique la polaridad de la fuente y el voltaje alimentado. (Quizá la pila esté descargada o haya un corto circuito).
4. En algunos tipos de tablilla de conexiones los extremos están divididos por secciones sin tener necesariamente continuidad entre sus líneas.

Reporte

Elabore el reporte correspondiente a cada práctica con las siguientes especificaciones:

1.1 Portada
   a) Nombre de la práctica
   b) Fecha de realización
   c) Nombre y número de matrícula
   d) Nombre del instructor

1.2 Introducción (explicar el objetivo de la práctica)

1.3 Procedimiento y metodología[1]

[1] Tanto el procedimiento como la metodología deben explicarse.

1.4 Representación de la función mediante diagrama de alambrado, diagrama esquemático, circuito, ecuación o tabla de verdad[2]

1.5 Resultados, conclusiones y recomendaciones[3]

1.6 Cuestionario resuelto que aparece al final de la práctica, en su caso

1.7 Referencias bibliográficas

NOTAS:

[2] Un reporte con diagramas y sin explicaciones ni comentarios carece de valor.

[3] Los resultados deben de analizarse y comentarse.

PRÁCTICA 2

Operadores lógicos con circuitos TTL

Objetivos particulares

Durante el desarrollo de esta práctica se conocerá el funcionamiento de los distintos operadores lógicos **And**, **Or**, **Not**, **Nand**, **Nor**, **Exor** y **Exnor,** analizando su símbolo, tabla de verdad y ecuación.

Para lograr el objetivo de esta práctica, el alumno deberá:

- Conocer el símbolo, la expresión matemática y la tabla de verdad de los operadores lógicos **And**, **Or**, **Not**, **Nand**, **Nor**, **Exor** y **Exnor**.
- Identificar las terminales de los circuitos utilizados.
- Aprender a interconectar y armar circuitos usando la tablilla de conexiones (*protoboard*).
- Obtener las tablas de verdad de cada uno de los operadores.

El tiempo de estudio estimado para el desarrollo de esta práctica es de tres horas (una hora para la explicación y dos horas adicionales como mínimo en trabajo de laboratorio).

Material necesario para el desarrollo de esta práctica

- Una fuente de voltaje de 5VCD
- Una tablilla de conexiones (*protoboard*)
- Circuitos integrados SN7400, SN7402, SN7408, SN7432 y SN7486
- Un DIP deslizable de 8 o 4 *switch push micro* NO
- Ocho LEDS (sin importar el color)
- Ocho resistencias de 330 Ω
- Alambre para conexiones

Fundamento teórico

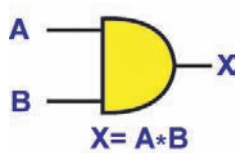
Operaciones booleanas

Operador And (y) o condición[1]

La operación **And** esta relacionada con el término condición y es exactamente igual a la multiplicación ordinaria de **unos** y **ceros**. Una salida igual a **1** ocurre sólo en el único caso en que todas las entradas son **1**. La salida es **cero** cuando una o más de las entradas son iguales a **0**.

El símbolo de la compuerta **And** se muestra en la figura. La expresión matemática de esta operación puede representarse por:

X = A B o, también, **X = A*B** y **X = A ∩ B**.



En otras palabras, la compuerta **And** es un circuito que opera de forma tal que su salida es **ALTA** sólo cuando todas sus entradas son **ALTAS**; o bien, su salida es **BAJA** cuando cualesquiera de sus entradas son **BAJAS**.

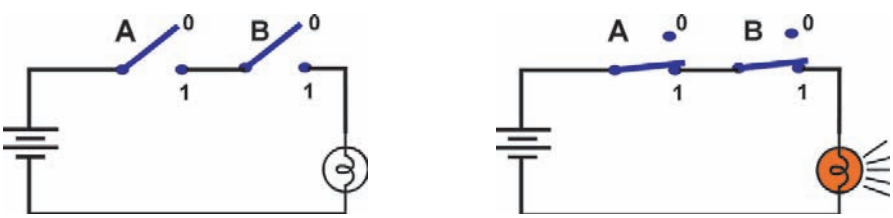
La tabla de verdad para la compuerta **And** de dos entradas, **A** y **B,** y la salida **X** se muestra a continuación:[2]

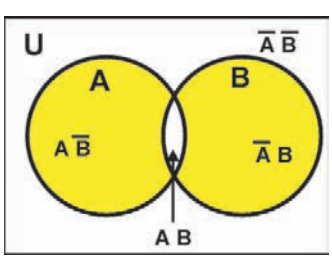
[1] Condición es la cláusula obligatoria de la que depende la validez de un acto.

[2] La letra *m* se refiere al número de combinación de la tabla de verdad.

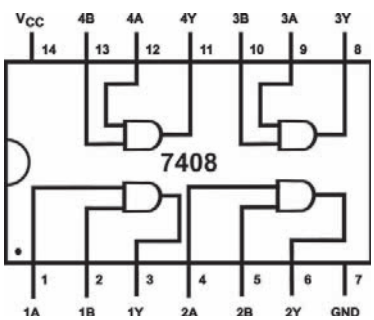
	Tabla de Verdad		
m	A	B	X = AB
0	0	0	0
1	0	1	0
2	1	0	0
3	1	1	1

A continuación vemos el circuito eléctrico para un operador **And** donde el foco enciende sólo cuando los interruptores **A** y **B** están en posición **1** (cerrados).



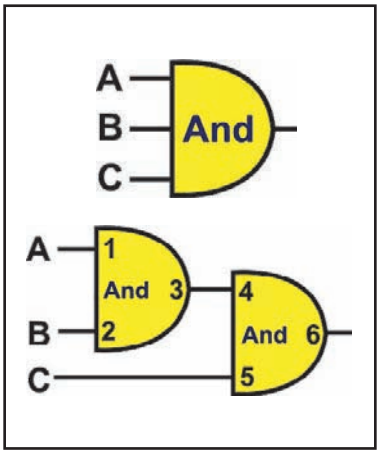


La operación **And** en un diagrama de la teoría de conjuntos se representa con la intersección **A ∩ B.**



Un circuito integrado TTL[3] con cuatro operadores y **And** de dos entradas.

[3]TTL significa tecnología *Transistor-Transistor Logic*.



Esta figura representa la operación **And** de tres entradas implementada con dos **And** de dos entradas.

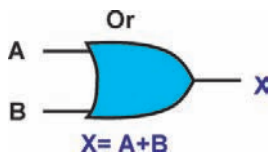
m	A B C	And
0	0 0 0	0
1	0 0 1	0
2	0 1 0	0
3	0 1 1	0
4	1 0 0	0
5	1 0 1	0
6	1 1 0	0
7	1 1 1	1

Aquí tenemos la tabla de verdad para una operación **And** de tres entradas.

Operador Or (o) o alternativa[4]

La operación **Or** está relacionada con el término *alternativa* y produce un resultado **1** cuando cualquiera de las variables de entrada es **1**. La operación **Or** genera un resultado de **0** sólo cuando todas las variables de entrada son **0**.

El símbolo de la compuerta **Or** se muestra en esta figura. La expresión matemática de la operación **Or** es: **X = A + B** o también **X = A ∪ B**.

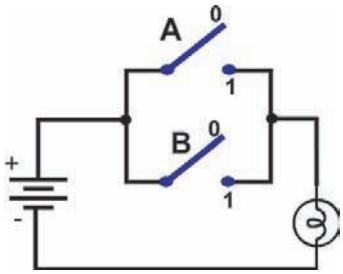


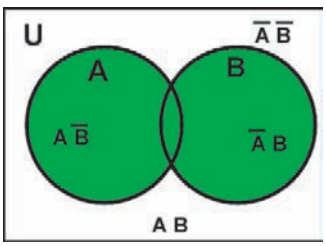
La tabla de verdad para la compuerta **Or** de dos entradas **A** y **B,** y la salida **X** se presenta a continuación:

	Tabla de verdad		
m	A	B	X=A+B
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	1

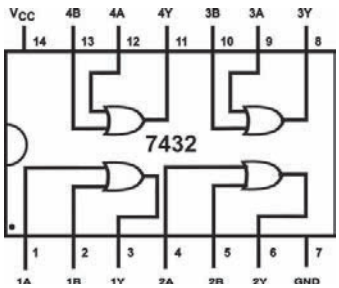
[4]Alternativa es una opción entre dos cosas, ya sea una, la otra, o ambas.

Circuito eléctrico para un operador **Or** donde el foco enciende cuando cualquiera de los interruptores **A** o **B** están en posición **1,** o ambos (es decir, cerrados).

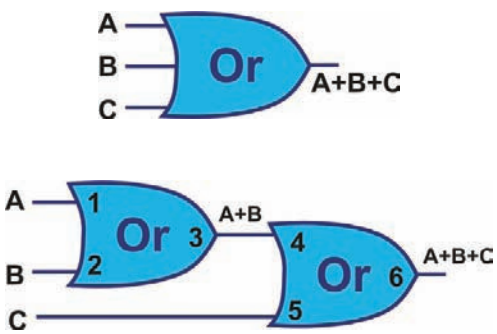




La operación **Or** en un diagrama de la teoría de conjuntos se representa con la unión **A ∪ B**.



Circuito integrado TTL con cuatro operadores **Or** de dos entradas.



Operación **Or** de tres entradas implementada con dos **Or** de dos entradas.

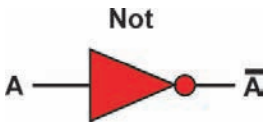
M	A B C	Or
0	0 0 0	0
1	0 0 1	1
2	0 1 0	1
3	0 1 1	1
4	1 0 0	1
5	1 0 1	1
6	1 1 0	1
7	1 1 1	1

Tabla de verdad para una operación **Or** de tres entradas.

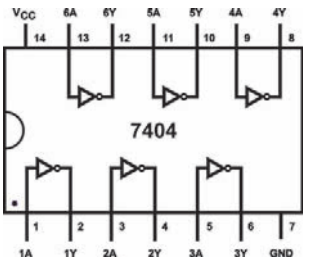
Operador Not (negación)

La operación **Not** está definida para una sola variable y es muy simple, ya que sólo tiene dos posibilidades: si la entrada es **0** la salida es igual a **1**, y viceversa.

Símbolo	Tabla de verdad	Circuito integrado TTL con seis operadores **Not**.



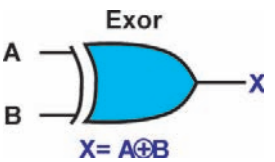
m	A	$\overline{A}$
0	0	1
1	1	0



Operador Exor (Or exclusiva)[5]

La operación **Exor** produce un resultado de **1** cuando un número impar de variables de entrada vale **1**.

El símbolo de la compuerta **Exor** se muestra en esta figura, en tanto que la expresión matemática para una compuerta **Exor** de dos entradas es:
X = A ⊕ B.

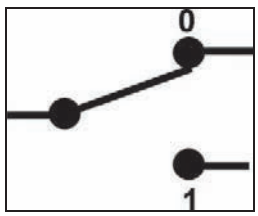


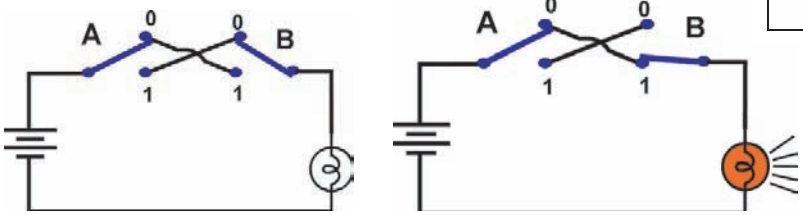
La tabla de verdad para la compuerta **Exor** de dos entradas, **A** y **B,** y la salida **X** se presentan a continuación:

	Tabla de verdad		
m	A	B	X= A+B
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	0

[5]Alternativa exclusiva es una opción entre dos cosas, una u otra *pero no ambas*.

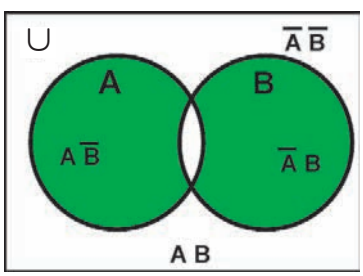
El interruptor usado en el circuito eléctrico para la demostracion del **Exor** es diferente a los utilizados en los circuitos de la **And** y **Or**; este interruptor se conoce como un tiro y dos polos, como se observa en la figura.



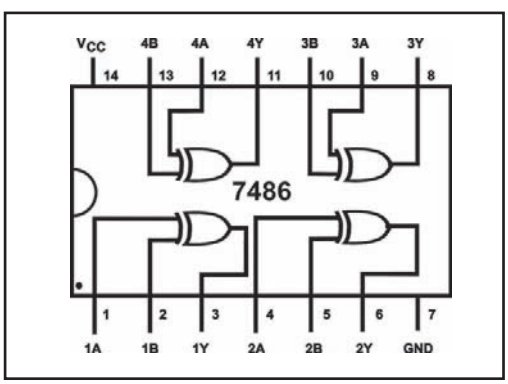


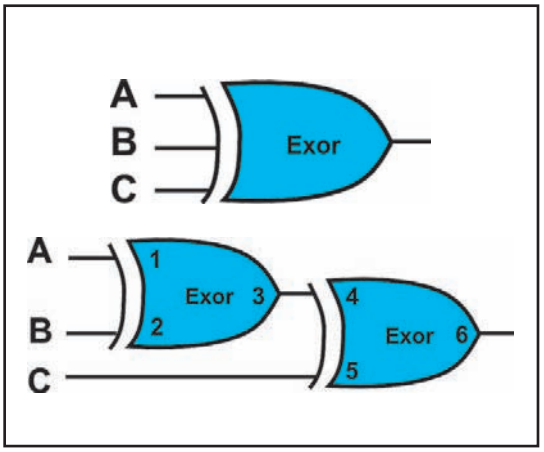
Ahora vemos un circuito eléctrico para un operador **Exor**, donde el foco enciende cuando cualquiera de los interruptores **A** o **B** están en posición **1** (cerrados), pero no ambos.

En un diagrama de la teoría de conjuntos, la operación **Exor** se representa con el área iluminada.



Circuito integrado TTL con cuatro operadores **Exor** de dos entradas.





Operación **Exor** de tres entradas implementada con dos **Exor** de dos entradas.

m	A B C	Exor
0	0 0 0	0
1	0 0 1	1
2	0 1 0	1
3	0 1 1	0
4	1 0 0	1
5	1 0 1	0
6	1 1 0	0
7	1 1 1	1

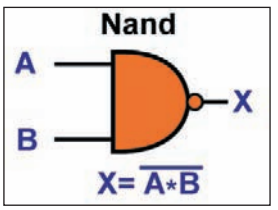
Tabla de verdad para una operación **Exor** de tres entradas.

Operador Nand (And negado)

La operación **Nand** es la negación de la salida de la operación **And**.

El símbolo de la compuerta **Nand** se muestra en la siguiente figura. La expresión matemática de la compuerta **Nand** se describe como:

$\overline{X}$ **= A B, (A B)'** o, también, **X = A ↑ B**.

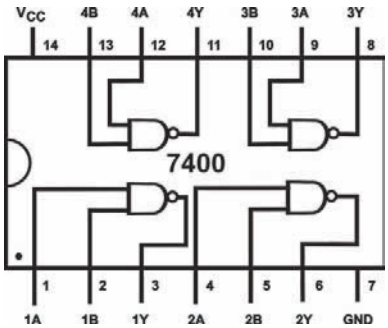


En otras palabras, la compuerta **Nand** es un circuito que opera de tal forma que su salida es **BAJA** sólo cuando todas sus entradas son **ALTAS**. O, también, su salida es **ALTA** cuando cualquiera de sus entradas es **BAJA**.

La tabla de verdad para la compuerta **Nand** de dos entradas **A** y **B,** y la salida **X** se muestran a continuación.

m	A	B	X= $\overline{AB}$
0	0	0	1
1	0	1	1
2	1	0	1
3	1	1	0

Tabla de verdad



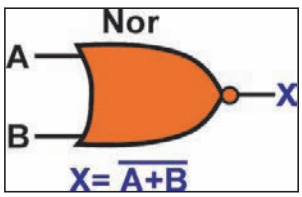
Circuito integrado TTL con cuatro operadores **Nand** de dos entradas.

Operador Nor (Or negado)

La operación **Nor** es la negación de la salida de la operación **Or**.

El símbolo de la compuerta **Nor** se muestra en la siguiente figura. La expresión matemática de la compuerta **Nor** es:

$$\mathbf{X = \overline{A+B}},\ \mathbf{(A+B)'}\ \text{o, también,}\ \mathbf{X = A \downarrow B}.$$

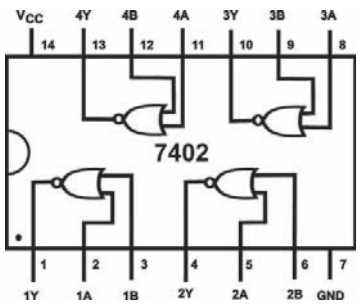


En otras palabras, la compuerta **Nor** es un circuito que opera para que su salida sea **BAJA** cuando cualquiera de sus entradas es **ALTA**. O, también, su salida es **ALTA** sólo cuando todas sus entradas son **BAJAS**.

La tabla de verdad para la compuerta **Nor** de dos entradas **A** y **B,** y la salida **X** se muestran a continuación.

m	A	B	X= $\overline{A+B}$
0	0	0	1
1	0	1	0
2	1	0	0
3	1	1	0

Tabla de verdad



Circuito integrado TTL con cuatro operadores **Nor** de dos entradas.

Operador Exnor (Exor negado)

Su símbolo y tabla de verdad para dos entradas son los siguientes.

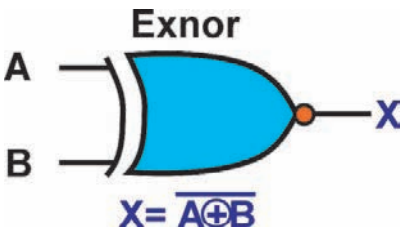
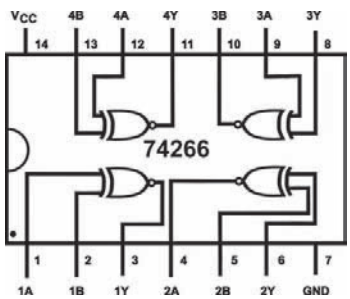


	Tabla de verdad		
m	A	B	$X=\overline{A \oplus B}$
0	0	0	1
1	0	1	0
2	1	0	0
3	1	1	1

Circuito integrado TTL con cuatro operadores **Exnor** de dos entradas.

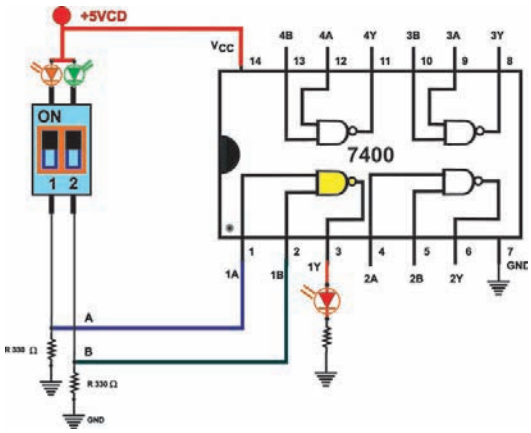


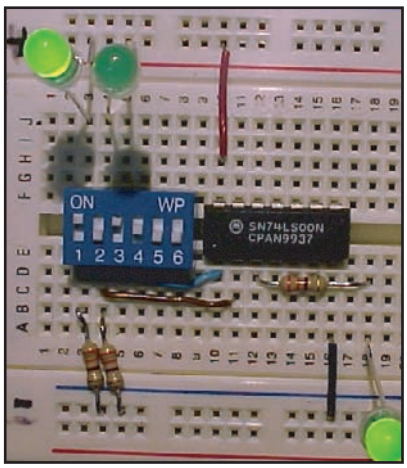
Trabajo solicitado

En la tablilla de conexiones (*protoboard*) armar el circuito que se muestra abajo para comprobar las tablas de verdad de cada uno de los operadores **And**, **Or**, **Exor**, **Nand** y **Nor**, de dos entradas llamadas **A**, **B**, alimentadas eléctricamente mediante un DIP SW. Obtener la salida en un LED que indique encendido cuando la salida sea 1; y apagado, cuando la salida tenga el valor de **0**.

Procedimiento

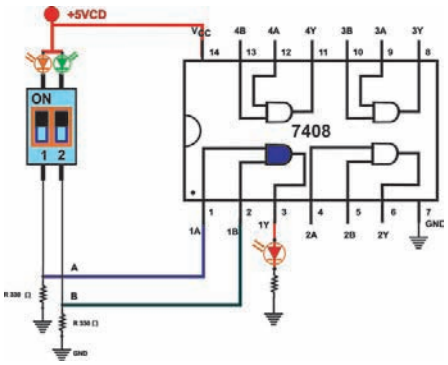
1. Efectúe las conexiones para obtener el circuito mostrado en la figura. Obtenga los valores de salida para las combinaciones de entrada **00**, **01**, **10** y **11** (tabla de verdad) de la operación **Nand** con su circuito integrado 7400.





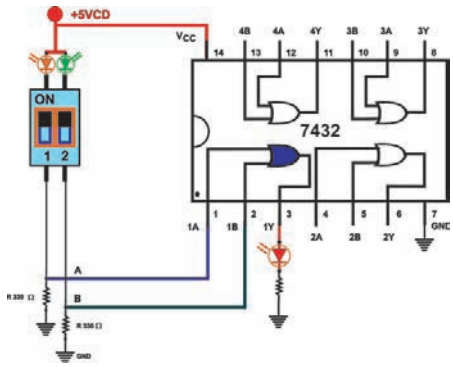
NOTA: Asegúrese de que la terminal positiva de 5 VCD se conecte a la terminal 14 del circuito, y la negativa GND a la terminal 7, pues un error al conectar podría dañar el circuito integrado.

2. Haga las conexiones del circuito integrado SN7408, señalado en la figura, para obtener los valores de salida para las combinaciones de entrada **00**, **01**, **10** y **11** (tabla de verdad) de la operación **And** con su circuito.



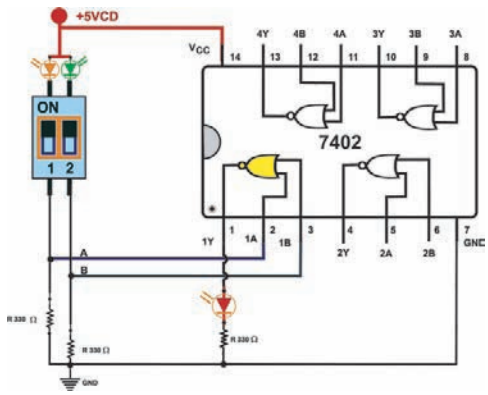
3. Realice las conexiones del circuito integrado SN7432, indicado en la figura, para obtener los valores de salida para las combinaciones de entrada **00**, **01**, **10** y **11** (tabla de verdad) de la operación **Or**.

Si se dejara una terminal de entrada (1 o 2) sin conectar, ¿qué valor tomaría?

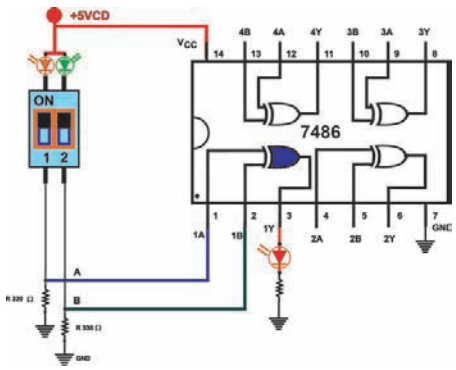


4. Efectúe las conexiones del circuito integrado SN7402 que se muestra en la figura, para obtener los valores de salida para las combinaciones de entrada **00, 01, 10** y **11** (tabla de verdad) de la operación **Nor**.

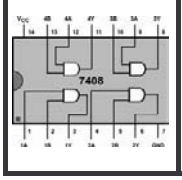
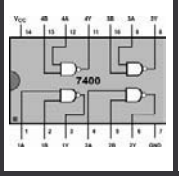
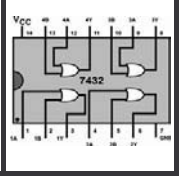
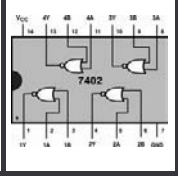
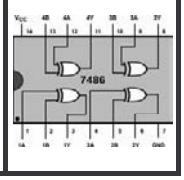
Observe que la distribución de terminales es diferente de los circuitos anteriores.



5. Haga las conexiones del circuito integrado SN7486 indicado en la figura, para obtener los valores de salida para las combinaciones de entrada **00**, **01**, **10** y **11** (tabla de verdad) de la operación **Exor**.



6. Coloque los valores obtenidos en la tabla de verdad para cada operador, indicando con **1** encendido, y con **0** apagado.

m	A B	Nand	And	Or	Nor	Exor
0	0 0					
1	0 1					
2	1 0					
3	1 1					
						

Cuestionario

1. ¿Quién desarrolló el álgebra booleana?

_____

_____

_____

2. ¿Cómo formaría una operación **And** de tres entradas usando compuertas **And** de sólo dos entradas? Dibuje el circuito.

3. ¿Qué valor lógico se considera cuando una entrada no está conectada? (Pruebe con el circuito Or 7432.)

4. ¿Cuál es el significado de TTL?

5. ¿Cuál es el significado de VCC?

6. ¿Cuál es el máximo valor de voltaje de alimentación para un circuito típico TTL?

7. ¿Cuál es el significado de GND?

Reporte

Elabore el reporte correspondiente a cada práctica con las siguientes especificaciones:

2.1 Portada

a) Nombre de la práctica

b) Fecha de realización

c) Nombre y número de matrícula

d) Nombre del instructor

2.2 Introducción (explicar el objetivo de la práctica)

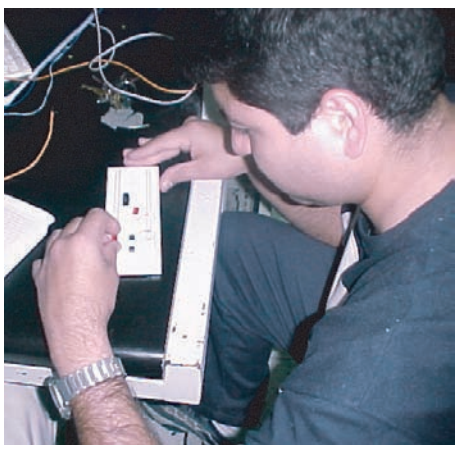
2.3 Procedimiento y metodología[6]

2.4 Representación de la función mediante diagrama de alambrado, diagrama esquemático, circuito, ecuación o tabla de verdad[7]

2.5 Resultados, conclusiones y recomendaciones[8]

2.6 Cuestionario resuelto que aparece al final de la práctica, en su caso

2.7 Referencias bibliográficas



[6] Tanto el procedimiento como la metodología deben explicarse.

[7] Un reporte con diagramas y sin explicaciones ni comentarios carece de valor.

[8] Los resultados deben de analizarse y comentarse.

PRÁCTICA 3

Captura esquemática

Objetivos particulares

Durante el desarrollo de esta práctica se implementarán los operadores lógicos **And**, **Or**, **Nand**, **Nor** y **Exor** de tres entradas en un dispositivo lógico programable (PLD), utilizando un programa de aplicación de captura esquemática; asimismo se comprobarán sus tablas de verdad.

Para lograr el objetivo de esta práctica, el alumno deberá:

- Conocer el símbolo, la expresión matemática y la tabla de verdad de los operadores lógicos **And**, **Or**, **Nand**, **Nor** y **Exor**.
- Familiarizarse con el programa de captura esquemática (*Schematic*).
- Conocer las características básicas del GAL16V8.
- Aplicar el proceso de compilación (*ISP Starter*).
- Programar el GAL16V8.
- Saber identificar las terminales de un circuito integrado a partir del archivo reporte (*pin out*).

- Polarizar el circuito integrado GAL16V8 a una fuente de 5V (5V la terminal 20, y GND la terminal 10).
- Conectar las terminales de entrada a interruptores.
- Conectar las terminales de salida a LEDS.
- Comprobar de forma práctica las tablas de verdad de cada operador lógico, alimentando las combinaciones del 0 al 7 binario (**000** al **111**), y obtener los valores de salida para cada combinación.

El tiempo estimado para el estudio de esta práctica es de dos horas.

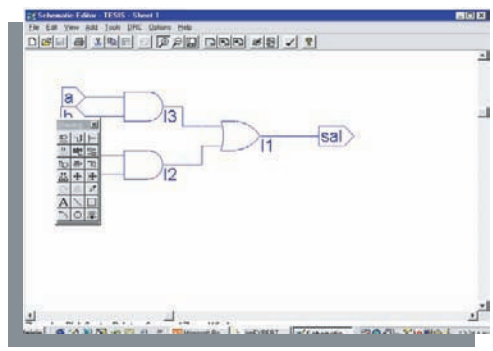
Material necesario para el desarrollo de esta práctica

- Una fuente de voltaje de 5VCD.
- Una tablilla de conexiones (*protoboard*).
- Un circuito integrado GAL16V8 (*lattice* semiconductor) o equivalente.
- Un DIP deslizable de 8 *switch* o 3 *switch push micro* NO.
- Ocho LEDS sin importar el color.
- Ocho resistencias de 330 ohms.
- Alambre para conexiones.
- Un disco de 3.5 pulgadas de alta densidad formateado a 1.44 MB.

Fundamento teórico

Captura esquemática

Usando la captura esquemática es posible fabricar un circuito integrado a la medida, con diagramas que representen los diferentes componentes del circuito y efectuando solamente interconexiones entre ellos.

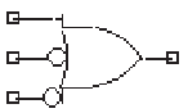


La gran ventaja de usar esta herramienta radica en la posibilidad de realizar los diseños por computadora, donde los errores se detectan y se corrigen fácilmente. Todo lo anterior agiliza el procedimiento, ya que se evita la fabricación de varios circuitos integrados (chip) para verificar su funcionamiento, reduciendo así tanto el ciclo de diseño como el tiempo de obtención de un producto.

La desventaja surge en los diseños grandes, los cuales son difíciles de comprender a causa de que hay demasiados componentes e interconexiones.

Los cuatro componentes básicos de la captura esquemática son los **símbolos**, los **conectores**, las **etiquetas** y los **puertos** de entrada y/o salida.

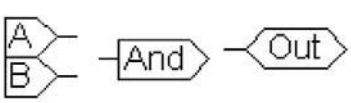
Los **símbolos** son una representación gráfica de los componentes.



Los **conectores** (alambre) sirven para la interconexión entre las terminales de los símbolos o dispositivos de entrada/salida.

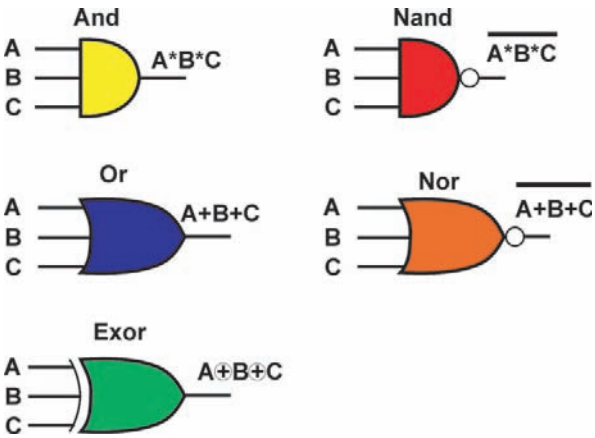
Las **etiquetas** (variables) son los nombres para la identificación de las entradas o salidas.

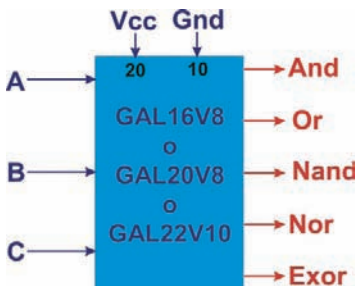
Los **puertos de entrada/salida** definen un puerto de entrada, salida o bidireccional.



Trabajo solicitado

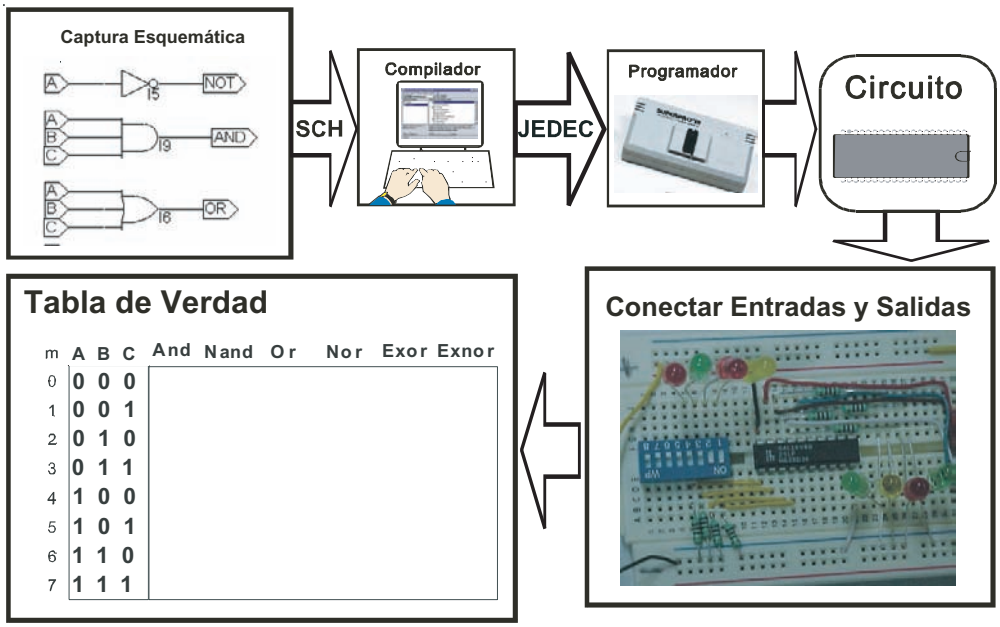
Diseñar un circuito que incluya las compuertas básicas **And**, **Or**, **Exor**, **Nand**, y **Nor** de tres entradas llamadas **A**, **B** y **C**, implementadas en un dispositivo programable GAL (*Generic Logic Array*), usando el programa de captura esquemática y el compilador *Isp Expert System Starter Software*. Además hay que obtener la tabla de verdad de cada una de las compuertas.





Procedimiento

Los pasos para obtener el circuito integrado a la medida por medio de captura esquemática se muestran a continuación:



A. Inicio.

1A. Abrir el programa *Isp System Starter*.

2A. *Crear un nuevo proyecto* (*File, New Project*).

3A. Dar nombre del proyecto.

4A. Seleccionar el dispositivo GAL16V8ZD.

5A. Seleccionar el nuevo archivo fuente.

B. Captura esquemática.

1B. Dar nombre del archivo.

2B. Seleccionar componentes en la caja de herramientas.

3B. Seleccionar la biblioteca de símbolos GATES:LIB

4B. Conectores.

5B. Etiquetas.

6B. Puertos a cada entrada o salida.

7B. Guardar el archivo SCH.

C. Enlazar.

1C. *Update All Schematic Files* (actualizar los archivos de captura esquemática).

2C. *Link Design* (enlazar el diseño).

3C. *Fit Design* (tamaño del diseño).

4C. *Create Fuse Map* (obtener los archivos del mapa de fusibles y reporte).

D. Programar el dispositivo.

1D. Ejecutar el programa del programador.

2D. Seleccionar del dispositivo en el menú Select.

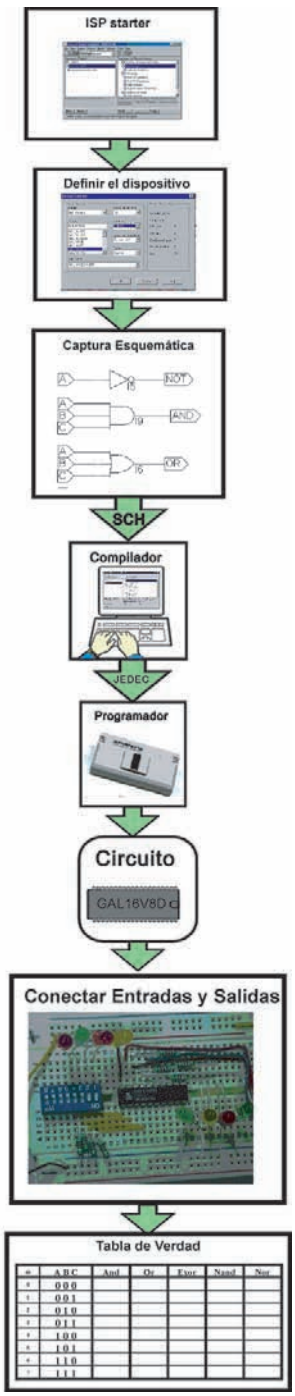
3D. Cargar del archivo JEDEC (F3) .

4D. Colocar el dispositivo en el socket.

5D. Programar (F5).

6D. Borrar el dispositivo (*Erase*).

7D. Programar el dispositivo (*Program*).

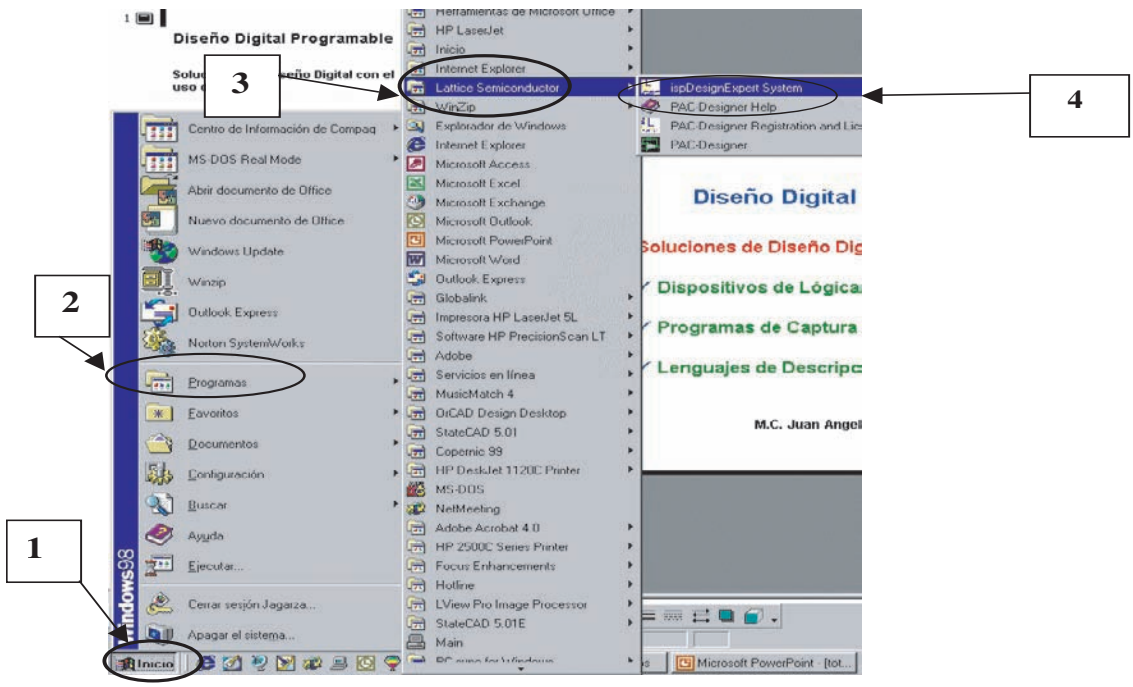


Captura esquemática: compilación y programación paso a paso

A. Inicio

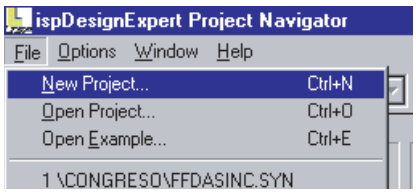
Una vez encendida la computadora, hay que buscar en la parte inferior izquierda del escritorio (pantalla) el botón *Inicio*.

1A. Abra el programa *Isp System Starter* (siga los pasos señalados en la figura).

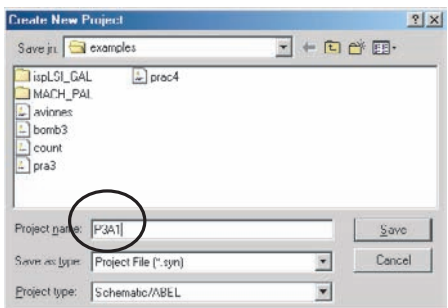


O bien, dé doble clic con el apuntador del mouse en el icono que se muestra en el escritorio de la pantalla de Windows.

2A. Cree un nuevo proyecto (*File, New Project*).

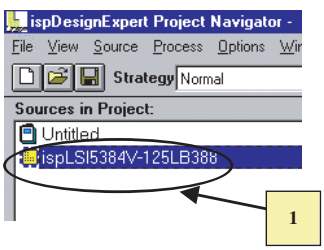


3A. Dé nombre del proyecto (*Project name*).

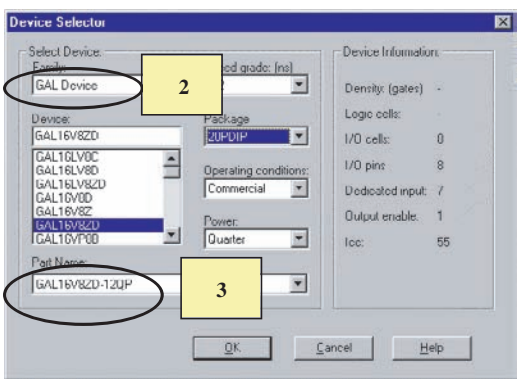


El nombre que se le dé al proyecto no debe exceder de ocho caracteres. Se sugiere usar nombres como P3A1, donde P3 se refiera al de práctica (en este caso la 3), y A1 el equipo de trabajo que lo realiza. La extensión que identifica al proyecto es .syn. Por ejemplo: P3A1.SYN.

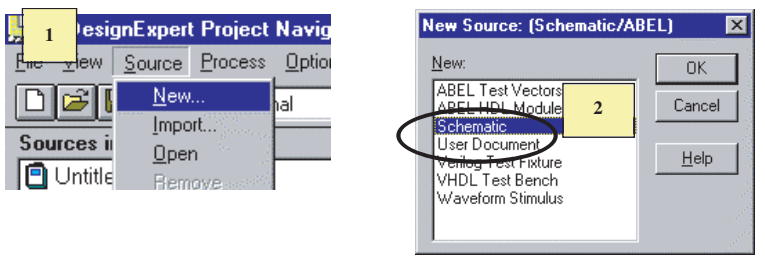
4A. Seleccione el dispositivo GAL16V8ZD. (Los pasos se muestran en la figura.)



Doble clic con el mouse.



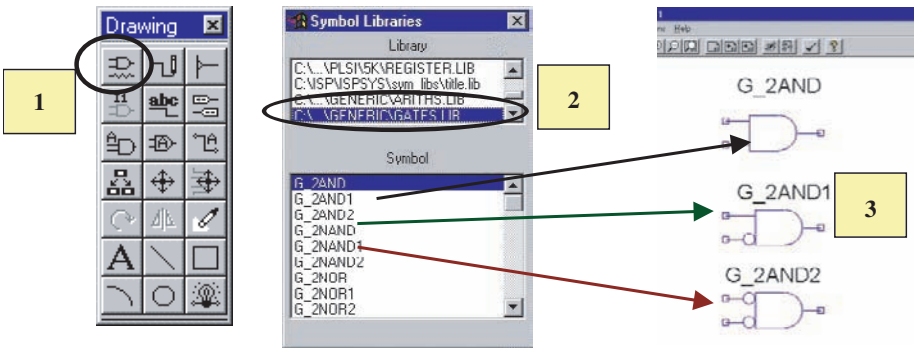
5A. Seleccione el nuevo archivo fuente. (Los pasos se indican en la figura).



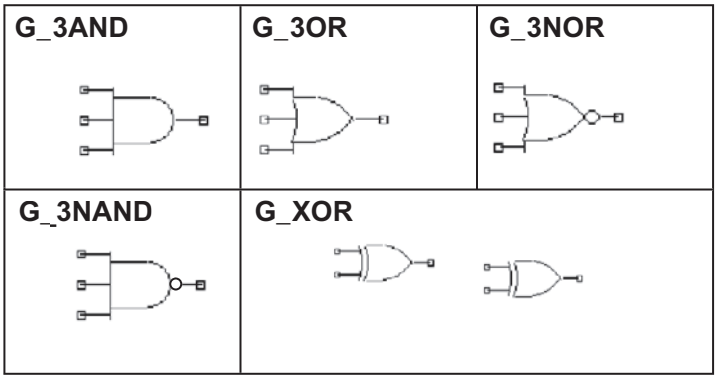
B. Captura esquemática

1B. Nombre del archivo. El nombre que se asigne para identificar este archivo no debe exceder de ocho caracteres. Se recomienda emplear el mismo nombre del proyecto P3A1, ya que la extensión que identifica al archivo de captura esquemática es sch. Por ejemplo: P3A1.SCH.

2B. Seleccione los componentes en la *Caja de herramientas* dentro de la *Biblioteca de símbolos GATES.LIB* y colóquelos en la hoja de trabajo. (Siga los pasos que se muestran en la figura).



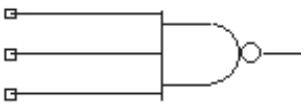
3B. Los componentes solicitados están en la *Biblioteca de símbolos GATES:LIB* y son:



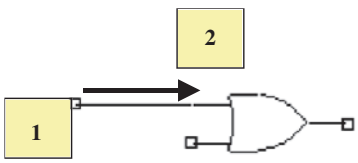
Observe que para el operador **Exor (G_XOR)** no se encuentran disponibles símbolos de tres entradas, por lo cual se usarán dos símbolos de dos entradas.

4B. Conectores. Tanto las entradas y salidas deberán de llevar un conector, de lo contrario el programa lo tomará como entrada o salida invalidada.

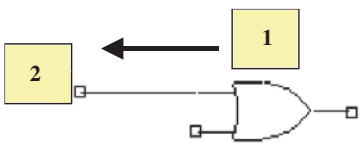
Para obtener un conector seleccione el icono señalado en la figura de la caja de herramientas *Drawing*.



Para trazar un conector en línea recta desde un punto hacia la terminal de un componente, haga un clic del mouse para iniciar (1) y otro para terminar el conector (2).



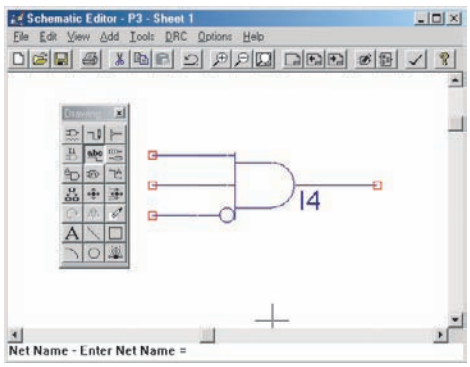
Para trazar un conector en línea recta desde la terminal hacia un punto dé un clic del mouse para iniciar (1) sobre el extremo de la terminal y doble clic para terminar el conector (2).

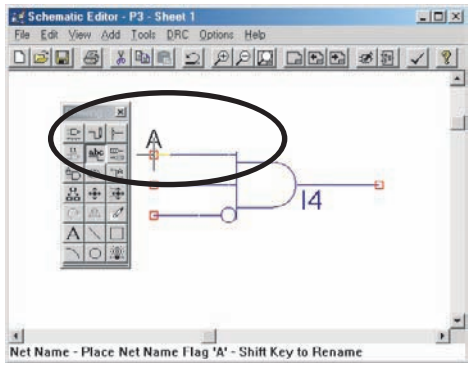


Para trazar un conector desde la terminal de salida de un componente hacia una entrada de otro componente, coloque el puntero del mouse en cualquiera de las dos terminales a conectar, dé un clic para iniciar el trazo, desplace el puntero del mouse sobre la otra terminal y dé doble clic. Los componentes quedarán interconectados.

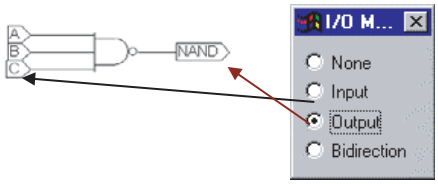
5B. Etiquetas (variables). Para obtener las etiquetas, en la caja de herramientas seleccione *Drawing* y el icono con **abc**. En la parte inferior de la pantalla aparecerán *Net Name - Enter Net Name =*

Teclee el nombre de la variable y posteriormente oprima la tecla *Enter*. Con el cursor posicione la variable al final del conector deseado y de nuevo un *Enter*.

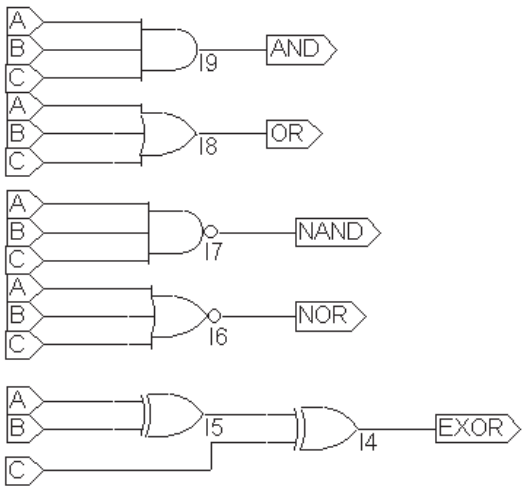




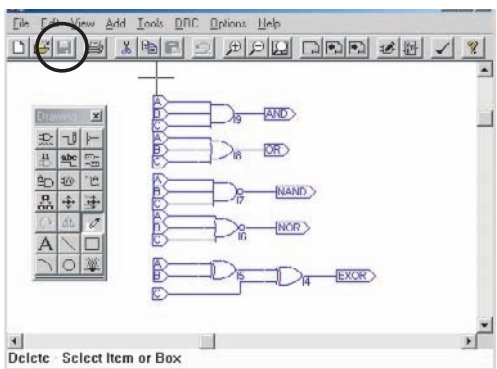
6B. Puertos de entrada o salida. Seleccione de la caja de herramientas *Drawing* el icono mostrado en la figura. Aparecerá un menú de opciones titulado *I/O M*. Aquí debe elegir el tipo de puerto a usarse (*None, Input, Output* y *Bidirection*).



El circuito terminado quedará de la siguiente forma:

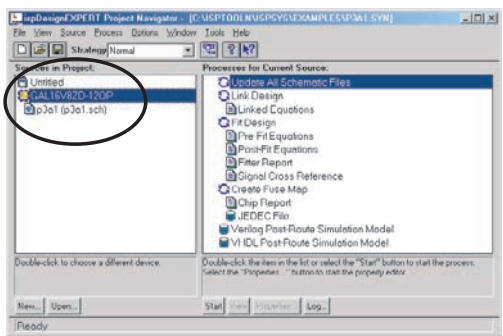


7B. Una vez terminada la captura esquemática, guarde el archivo utilizando el icono del disco que se muestra en la figura.



C. Enlazar (*Link*)

Regrese a la ventana de *Isp System Starter* y en el recuadro izquierdo (*Sources in Project*) asegure la presencia del dispositivo definido (GAL16V8/ZD); en el mismo recuadro asegure la presencia del archivo con extensión *.sch* (*P3A1.SCH*). Como lo indica la figura, es posible iniciar el proceso de compilación ejecutando las rutinas que aparecen en el recuadro derecho (*Processes for Current Source*).



1C. *Update All Schematic Files* (actualizar todos los archivos de captura esquemática).

En esta parte del proceso actualice los archivos que se tomarán en cuenta para la compilación.

2C. *Link Design* (enlazar el diseño).

Verifique si el o los archivos contienen un código válido. En caso de que no se acepte aparecerá un mensaje que incluye una explicación y un código de error.

3C. *Fit Design* (tamaño del diseño).

En algunas ocasiones, los requerimientos del diseño sobrepasan la capacidad del dispositivo seleccionado. Esta rutina verifica si el diseño cabe en el dispositivo. En caso de que sea demasiado grande, se sugiere elegir un dispositivo de mayor capacidad como GAL20V8 o GAL 22V10, etcétera.

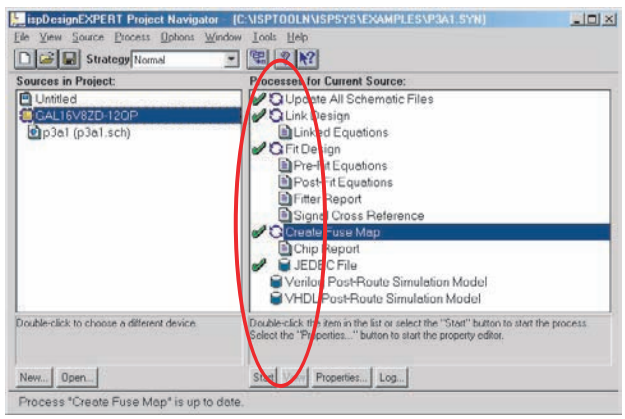
4C. *Create Fuse Map* (obtener el archivo del mapa de fusibles).

En este proceso se obtienen dos archivos:

El archivo reporte con extensión *.rep* contiene la información de las ecuaciones, la distribución de terminales *pin out*, el porcentaje de utilización del dispositivo, etcétera.

El archivo JEDEC con extensión *.jed* tiene el mapa de fusibles, el cual será utilizado para programar el dispositivo.

Para efectuar todos los pasos de este proceso, dé doble clic con el apuntador del mouse sobre los iconos que están en la ventana de *Processes for Current Source*. Al realizar la operación correctamente aparecerá una señal de aprobación en cada uno de ellos, como lo muestra la siguiente figura.



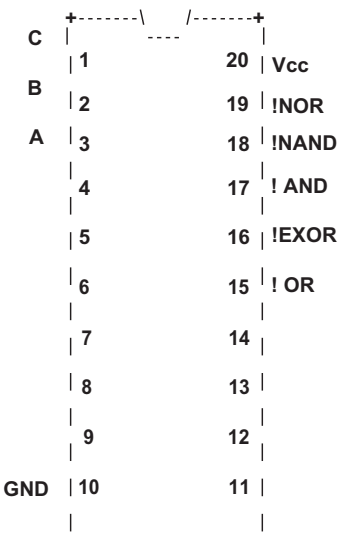
Archivo Reporte P3A1.rep

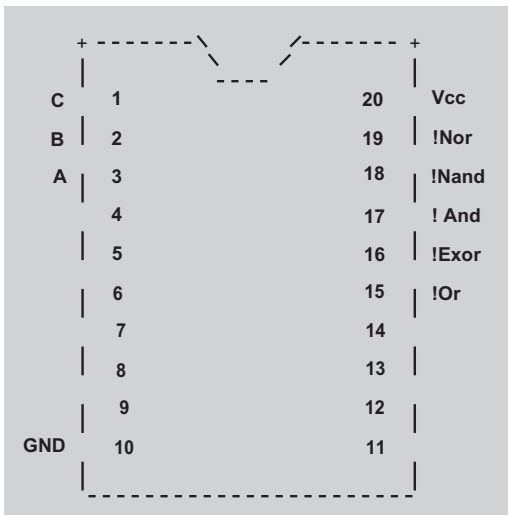
Este archivo se genera como resultado de la compilación.

Ecuaciones:

AND = (C & B & A);
NAND = !(C & B & A);
NOR = (!C & !B & !A);
OR = !(!C & !B & !A);
EXOR = !(!C & B & A # C & !B & A # C & B & !A # !C & !B & !A);

Chip Diagram:





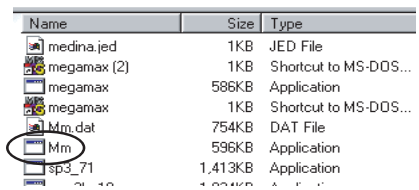
NOTA: Si la distribución de terminales (*pin out*) descrita en el archivo *Chip Report* aparece sin asignación (en blanco), es probable que el archivo de captura esquemática esté grabado en un directorio diferente del esperado (C:/ISPTOOLS/ISPSYS/BIN).

D. Programar el dispositivo

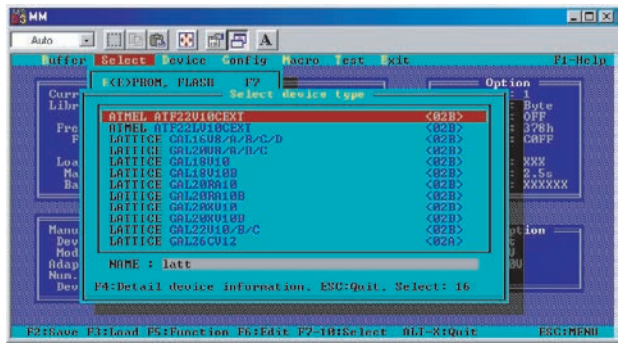
Es necesario tener un programador universal que soporte la programación de dispositivos lógicos programables, como el Mega Max–4G, que incluye tanto unos conectores adicionales que se seleccionan, dependiendo el dispositivo a programar, como un programa que se ejecuta en ambiente DOS llamado Mm.exe.

Procedimiento para el uso del programador Mega Max–4G.

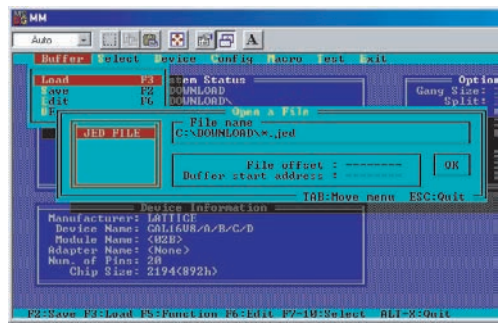
1D. Ejecute el programa Mm.exe. Este programa normalmente está en un directorio llamado **MM**. Antes de ejecutarlo es conveniente asegurarse de que el programador esté encendido y conectado al puerto paralelo de la computadora.



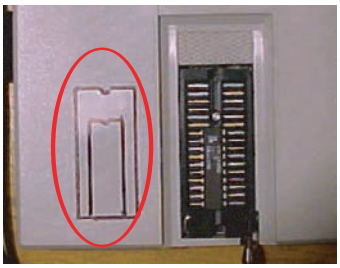
2D. Seleccione el dispositivo en el menú Select (Alt + S). Aquí aparecen varias opciones de dispositivos. Elija **PLD** y posteriormente LATTICE **GAL16V8/A/B/C/D.** En la parte derecha del dispositivo se indica la tarjeta que se debe insertar en el programador: **<02B>**.



3D. Cargue el archivo JEDEC. En el menú *Buffer* (F3), en la opción *Load*, seleccione el archivo P3A1.jed que se generó al enlazar en el programa *ISP Starter*.



4D. Coloque el dispositivo en el *socket* y baje la palanca, asegurándose de que la colocación del dispositivo es igual a la forma que se indica en el programador.

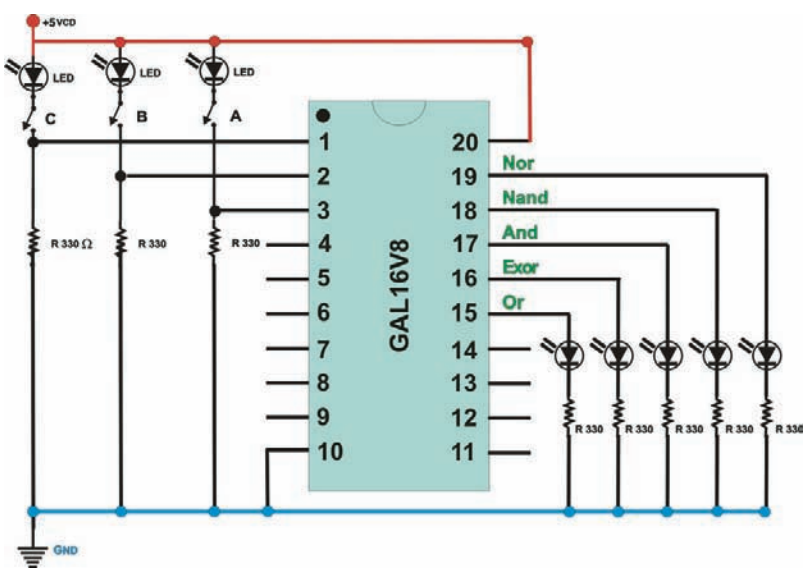


5D. Programar (F5 *Function*). Una vez definido el dispositivo y cargado el archivo JEDEC, oprima la tecla **F5**, y aparecerá el menú que se muestra en la figura.

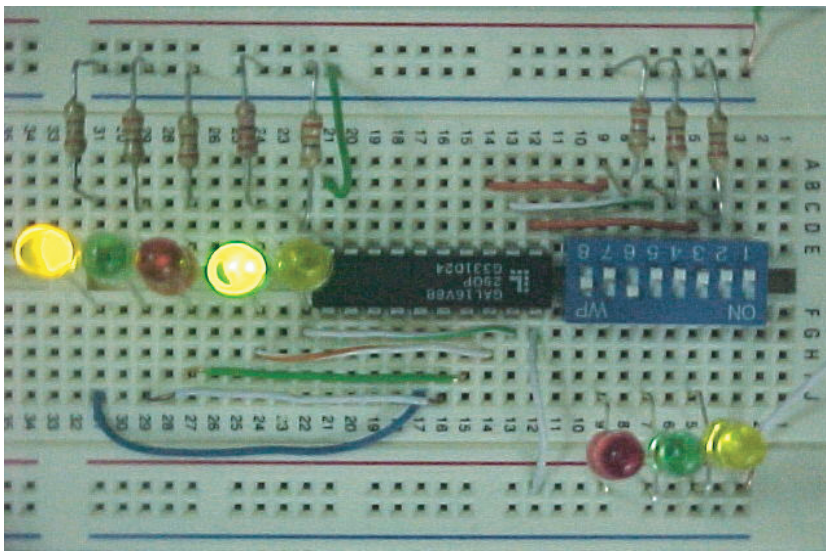
6D. Ejecute el comando *Erase,* una vez terminado.

7D. Ejecute *Program*. Si aparece el comentario *success,* entonces el dispositivo está listo para probarse.

Ahora implemente el circuito en la tablilla de conexiones siguiendo el diagrama obtenido en el archivo. Reporte como lo indica la siguiente figura y realice la tabla de verdad.



La distribución de terminales se asigna en forma aleatoria por el programa, de manera que quizá su resultado sea diferente de la distribución que se presenta en esta imagen.



Obtenga para la tabla de verdad los valores para cada una de las salidas.

M	A B C	And	Or	Exor	Nand	Nor
0	0 0 0					
1	0 0 1					
2	0 1 0					
3	0 1 1					
4	1 0 0					
5	1 0 1					
6	1 1 0					
7	1 1 1					

Cuestionario

1. ¿Cuál es el significado de las siglas GAL?

2. ¿Cuántas entradas como máximo puede tener el GAL16V8?

3. ¿Cuántas salidas como máximo puede tener el GAL16V8?

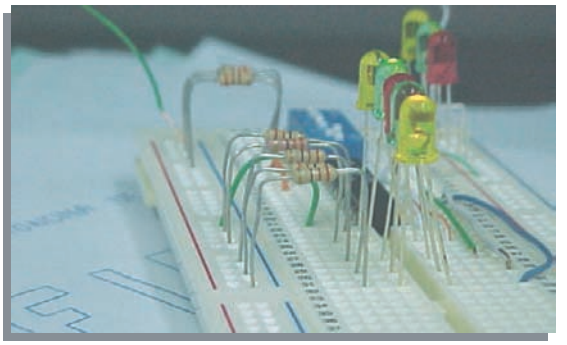
4. ¿Qué significa JEDEC?

5. Calcule el número de circuitos integrados TTL que se requieren para realizar esta práctica.

Recomendaciones

1. Tenga cuidado al insertar correctamente el circuito integrado en el programador, pues colocarlo en forma diferente de lo especificado podría dañar su dispositivo.

2. Al programar se recomienda que primero seleccione el circuito, borre su contenido y, posteriormente, cargue el archivo JEDEC y, por último, programe el dispositivo.



Reporte

Elabore el reporte correspondiente a cada práctica con las siguientes especificaciones:

3.1 Portada

a) Nombre de la práctica

b) Fecha de realización

c) Nombre y número de matrícula

d) Nombre del instructor

3.2 Introducción (explicar el objetivo de la práctica)

3.3 Procedimiento y metodología[1]

3.4 Representación de la función mediante diagrama de alambrado, diagrama esquemático, circuito, ecuación o tabla de verdad[2]

3.5 Resultados, conclusiones y recomendaciones[3]

3.6 Cuestionario resuelto que aparece al final de la práctica, en su caso

3.7 Referencias bibliográficas

[1] Tanto el procedimiento como la metodología deben explicarse.

[2] Un reporte con diagramas y sin explicaciones ni comentarios carece de valor.

[3] Los resultados deben de analizarse y comentarse.

PRÁCTICA 4

Simulación

Objetivos particulares

Durante el desarrollo de esta práctica se obtendrá el circuito a partir de la ecuación; y la tabla de verdad, a partir de la implementación del circuito. También se conseguirá el diagrama de tiempos usando el archivo TEST_VECTORS. Además, se conocerán las características básicas del GAL16V8D.

El tiempo estimado de estudio para esta práctica es de dos horas.

Material necesario para el desarrollo de esta práctica

- Un fuente de voltaje de 5VCD.
- Una tablilla de conexiones (*protoboard*).
- Un GAL16V8D (*Lattice semiconductor*) o equivalente.

- Un DIP de ocho entradas o cuatro switch *push micro NO*.
- Seis LEDS (diodos emisores de luz) sin importar el color.
- Seis resistencias de 330 ohms.
- Alambre preparado para conexiones.
- Un disco de 3.5 pulgadas formateado a 1.44 MB.

El tiempo estimado para el estudio de esta práctica son dos horas.

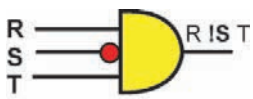
Fundamento teórico

Los elementos más usuales para describir una función booleana son:

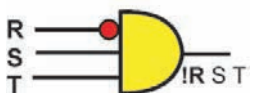
Circuito o diagrama esquemático	Representación gráfica de una expresión booleana mediante la interconexión de símbolos que corresponden a los operadores lógicos.
Ecuación	Representación matemática de una función booleana.
Tabla de verdad	Representación tabular del valor de salida para cada una de las posibles combinaciones de entrada.
Diagrama de tiempos	Representación gráfica de los valores de salida para las combinaciones de entrada en un tiempo dado.

Obtención de la ecuación a partir del circuito

A partir de una ecuación booleana es posible obtener su circuito o diagrama esquemático por el orden de sus operaciones. Por ejemplo:



En la ecuación Fx (R, S, T) = R **!S** T + **!R** S T, donde se realizan como primera operación la multiplicación **And** de los dos términos R **!S** T al mismo nivel **!R** S T, como lo indica la figura de la derecha. El resultado se suma por medio de una **Or**, como lo muestra la figura inferior.



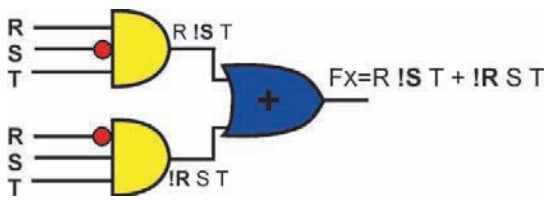
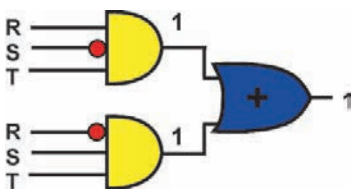


Tabla de verdad

Para realizar la tabla de verdad de un circuito se pueden probar, una por una, todas las combinaciones de entrada posibles y obtener el valor de salida de cada una de ellas, lo cual, no obstante, sería un método muy largo.

Otro método consiste en suponer un valor de salida y verificar qué combinaciones de entrada cumplen con el valor propuesto.



Por ejemplo: Fx $_{(R, S, T)}$ = R **S'** T + **R'** S T (forma SOP suma de productos). En este circuito se supone un valor de **1** a la salida de la **Or**, lo cual genera una alternativa, ya que cualquier entrada igual a 1 en la operación **Or** produce una salida ¡ (una, otra, o ambas).

Una vez analizada la salida de la **And** de arriba, la salida es 1 solo; cuando todas sus entradas son 1, entonces: **R = 1, S = 0 y T = 1**, lo cual se presenta en la combinación 5 de la tabla de verdad (m = 5).

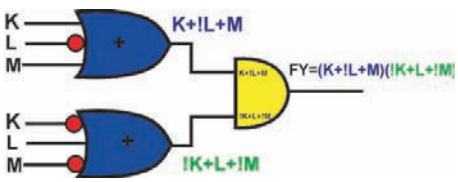
En la **And** de abajo, la salida es 1 cuando todas sus entradas son 1. Entonces **R = 0, S = 1 y T = 1** se presenta en la combinación 3 de la tabla de verdad (*m* = 3). Todas las demás combinaciones serán iguales a 0.

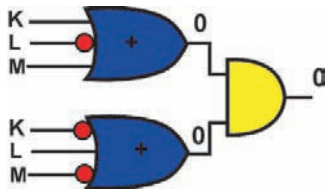
Tabla de verdad de la función Fx

M	R S T	Fx
0	0 0 0	0
1	0 0 1	0
2	0 1 0	0
3	0 1 1	1
4	1 0 0	0
5	1 0 1	1
6	1 1 0	0
7	1 1 1	0

Para el caso de la función FY (K, L, M) = (K **+ !L +** M) **(!K + L + !M)** (forma POS productos de suma), primero se efectúan las operaciones **Or**, sumadas antes que la **And** producto.

Aquí la operación OR (K **+ !L +** M) se realiza al mismo nivel que la operación Or (**!K +** L**+ !M**). Posteriormente, con la salida de estas dos se efectúa la operación **And**, como lo muestra la figura.





Para obtener la tabla de verdad de este circuito se supone un valor de **0** a la salida de **And**; esto genera una alternativa, ya que cualquier entrada 0 en la operación **And** produce una salida 0 (una, otra o ambas).

Una vez analizada la salida de la **Or** de arriba, la salida es 0 sólo cuando todas sus entradas son 0; entonces: **K = 0, L = 1 y M = 0**, lo cual se presenta en la combinación 2 de la tabla de verdad (m = 2).

En la **Or** de abajo la salida es cero cuando **K = 1, L = 0 y M = 1**. Esto sucede en la combinación 5 de la tabla de verdad (m = 5). Todas las salidas para las demás combinaciones serán iguales a 1.

Tabla de verdad para la función F_Y

m	K L M	F_Y
0	0 0 0	1
1	0 0 1	1
2	0 1 0	0
3	0 1 1	1
4	1 0 0	1
5	1 0 1	0
6	1 1 0	1
7	1 1 1	1

Trabajo solicitado

Para F1 y F2 obtenga:

a) La tabla de verdad en forma analítica.

b) La tabla de verdad de la implementación del circuito.

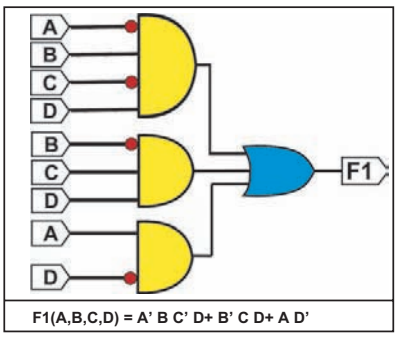
c) El diagrama de tiempos usando el archivo TEST_VECTORS.

Funciones: **F1** (A, B, C, D) = **A'** B **C'** D + **B'** C D + A **D'** (SOP)

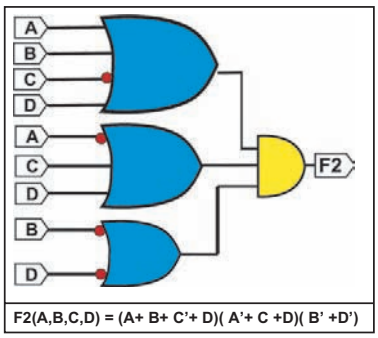
F2 (A, B, C, D) = (A+ B+ **C'**+ D)(**A'**+ C +D)(**B'** +**D'**) (POS)

Procedimiento

1. Dibuje el circuito de F1 y F2.



F1(A,B,C,D) = A' B C' D+ B' C D+ A D'



F2(A,B,C,D) = (A+ B+ C'+ D)(A'+ C +D)(B' +D')

2. Obtenga la tabla de verdad mediante el análisis de F1 y F2.

M	A B C D	F1	
0	0 0 0 0	0	
1	0 0 0 1	0	
2	0 0 1 0	0	
3	0 0 1 1	1	**B'** C D
4	0 1 0 0	0	
5	0 1 0 1	1	**A'** B **C'** D
6	0 1 1 0	0	
7	0 1 1 1	0	
8	1 0 0 0	1	A **D'**
9	1 0 0 1	0	
10	1 0 1 0	1	A **D'**
11	1 0 1 1	1	**B'** C D
12	1 1 0 0	1	A **D'**
13	1 0 0 1	0	
14	1 1 1 0	1	A **D'**
15	1 1 1 1	0	

F2	
1	
1	
0	A + B + **C'**+ D
1	
1	
0	**B' + D'**
1	
0	**B' + D'**
0	A' **+ C + D**
1	
1	
1	
0	A' **+ C + D**
0	**B' + D'**
1	
0	**B'** + **D'**

Tabla de verdad obtenida en forma analítica

M	A B C D	F1	F2
0	0 0 0 0	0	1
1	0 0 0 1	0	1
2	0 0 1 0	0	0
3	0 0 1 1	1	1
4	0 1 0 0	0	1
5	0 1 0 1	1	0
6	0 1 1 0	0	1
7	0 1 1 1	0	0
8	1 0 0 0	1	0
9	1 0 0 1	0	1
10	1 0 1 0	1	1
11	1 0 1 1	1	1
12	1 1 0 0	1	0
13	1 1 0 1	0	0
14	1 1 1 0	1	1
15	1 1 1 1	0	0

3. Programe las funciones F1 y F2 en el circuito integrado GAL16V8D mediante captura esquemática.

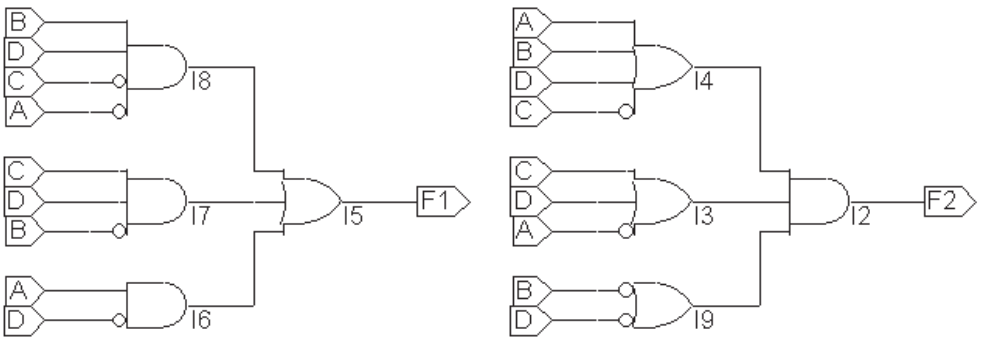
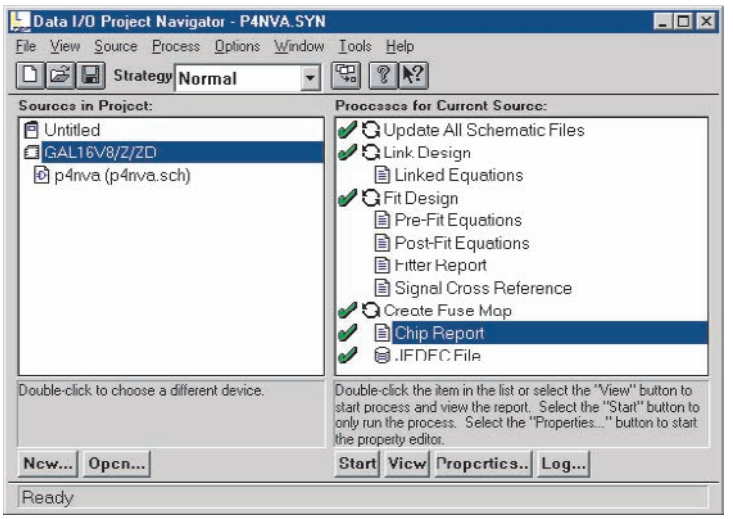
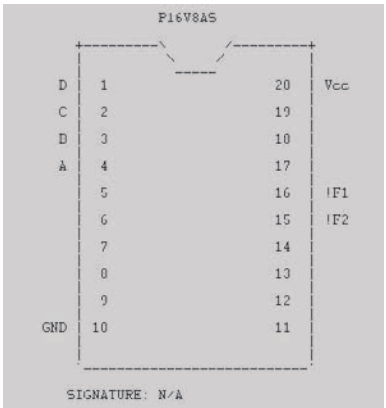


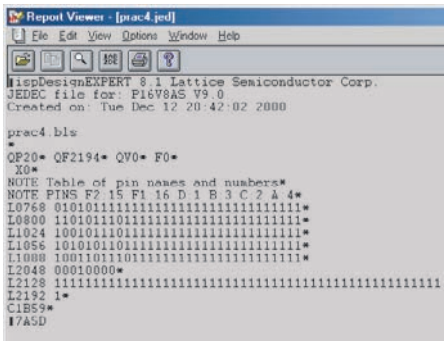
Figura de captura esquemática

4. Efectúe el proceso de compilación.





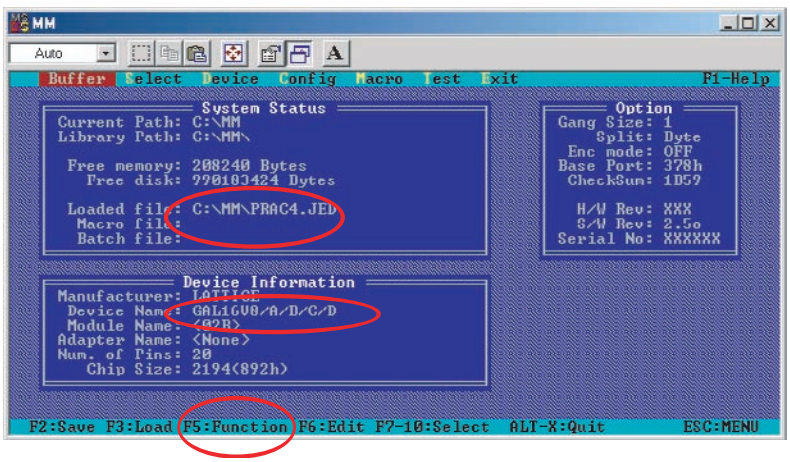
Archivo reporte que indica la distribución de terminales del circuito integrado (*pin out*).



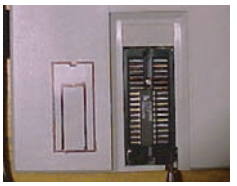
Archivo JEDEC necesario para programar el GAL16V8.

4

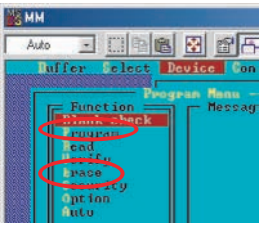
5. Programe el GAL16V8, y asegúrese de cargar el archivo JEDEC y de definir el dispositivo que se va a programar.



Presione la tecla de función F5 para programar.

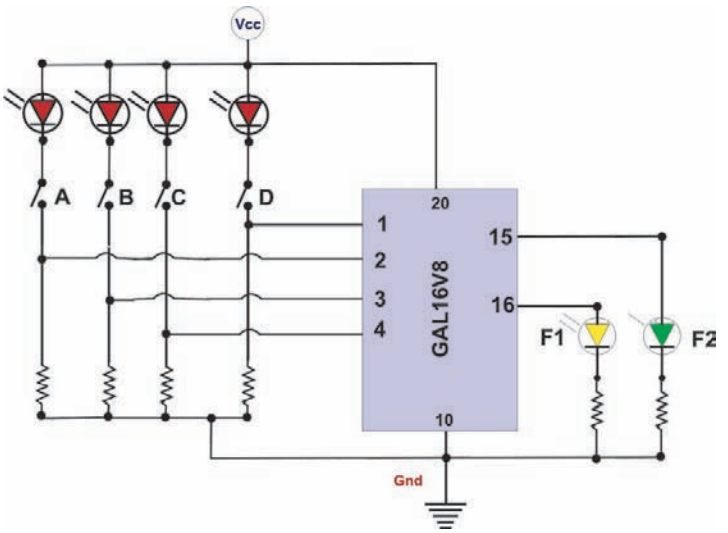


Inserte el GAL, asegure su correcta colocación en el socket y baje la palanca.

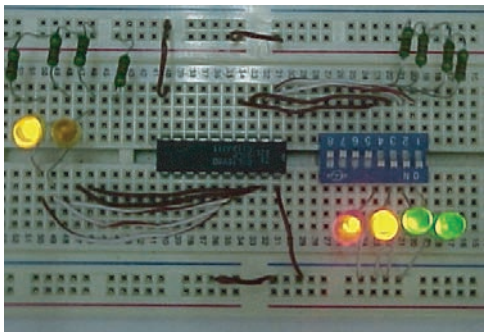


Utilice el comando *Erase* para borrar el dispositivo y, posteriormente, use *Program*.

6. Elabore un diagrama de alambrado con base en el Archivo reporte para obtener las tablas de verdad e implementarlo en la tablilla de conexiones. Para facilitar la prueba, coloque los interruptores en el orden A, B, C, D.



7. Implemente el circuito en la tablilla de conexiones y obtenga los valores de F1 y F2 para cada combinación m (de m = 0 a m = 15) de la tabla de verdad y compárela con la tabla obtenida en forma analítica.

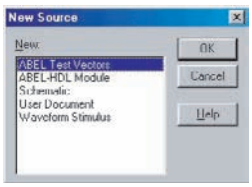


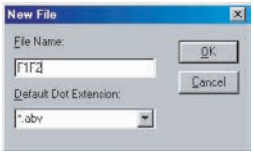
m	A B C D	F1	F2
0	0 0 0 0		
1	0 0 0 1		
2	0 0 1 0		
3	0 0 1 1		
4	0 1 0 0		
5	0 1 0 1		
6	0 1 1 0		
7	0 1 1 1		
8	1 0 0 0		
9	1 0 0 1		
10	1 0 1 0		
11	1 0 1 1		
12	1 1 0 0		
13	1 1 0 1		
14	1 1 1 0		
14	1 1 1 1		

8. En la simulación capture el programa *ABEL Test Vectors*.

a) En el menú de *Source* seleccione *New* y, después, *ABEL Test Vectors*.

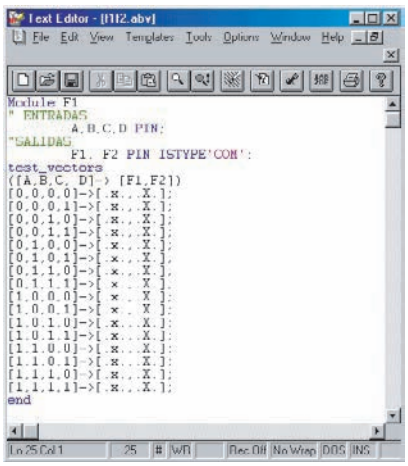
b) Teclee el nombre del archivo.





c) En la ventana del *Text Editor* teclee el siguiente archivo:

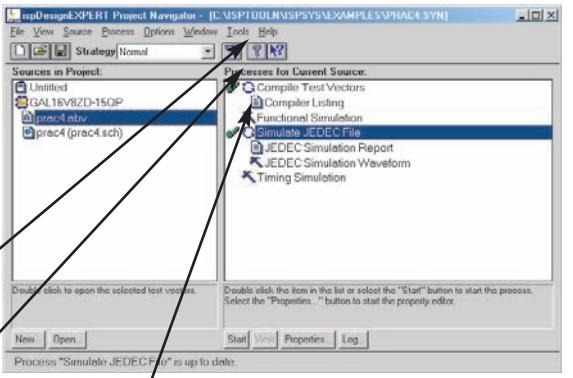
```
Module F1
" ENTRADAS
  A,B,C,D PIN;
  "SALIDAS
F1, F2 PIN ISTYPE'COM';
  TEST_VECTORS
([A,B,C, D]-> [F1,F2])
  [0,0,0,0]->[.x.,.x.];
  [0,0,0,1]->[.x.,.x.];
  [0,0,1,0]->[.x.,.x.];
  [0,0,1,1]->[.x.,.x.];
  [0,1,0,0]->[.x.,.x.];
  [0,1,0,1]->[.x.,.x.];
  [0,1,1,0]->[.x.,.x.];
  [0,1,1,1]->[.x.,.x.];
  [1,0,0,0]->[.x.,.x.];
  [1,0,0,1]->[.x.,.x.];
  [1,0,1,0]->[.x.,.x.];
  [1,0,1,1]->[.x.,.x.];
  [1,1,0,0]->[.x.,.x.];
  [1,1,0,1]->[.x.,.x.];
  [1,1,1,0]->[.x.,.x.];
  [1,1,1,1]->[.x.,.x.];
    End
```

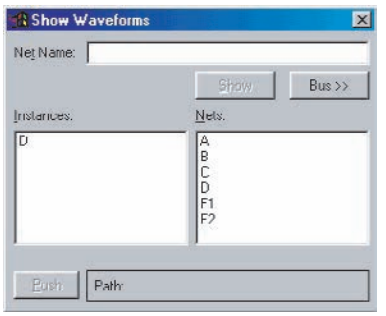


4

d) Compile el archivo *Test_Vectors*.

- Regrese al programa *Project Navigator* donde aparecerá incluido el archivo F1F2.ABV.
- Efectúe la compilación *Compile Test Vectors*.
- Ejecute *Simulation JEDEC File*.
- Ejecute *JEDEC Simulation Waveform*.
- En el programa *Waveform Viewer* aparecerá una nueva pantalla; seleccione *Edit* y posteriormente *Show*.
- Seleccione las variables A y oprima *Show*; posteriormente elija la variable B y de nuevo *Show*. Haga lo mismo con las demás variables C, D, F1 y F2.
- Cierre la ventana *Show Waveform* y compare la gráfica con la tabla de verdad.





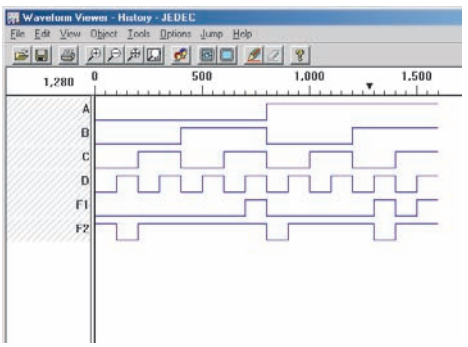


Diagrama de tiempos de las funciones F1 y F2.

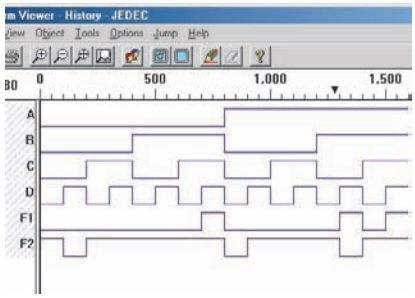


Diagrama de tiempos

m	A B C D	F1	F2
0	0 0 0 0		
1	0 0 0 1		
2	0 0 1 0		
3	0 0 1 1		
4	0 1 0 0		
5	0 1 0 1		
6	0 1 1 0		
7	0 1 1 1		
8	1 0 0 0		
9	1 0 0 1		
10	1 0 1 0		
11	1 0 1 1		
12	1 1 0 0		
13	1 1 0 1		
14	1 1 1 0		
15	1 1 1 1		

Tabla de verdad

Notas

1. En el proceso de enlace se obtiene la función mínima y en algunas funciones la reducción puede incluir algunas variables. Cuando esto sucede, el programa lo indica con el símbolo de admiración ¡ (warning).
2. Es recomendable que, una vez insertado correctamente el dispositivo en el programador, primero seleccione el circuito, borre su contenido, cargue el archivo JEDEC y, por último, programe el dispositivo.

Trabajo solicitado

Para cada uno de los ejercicios obtenga lo siguiente:

a) La tabla de verdad en forma analítica.

b) La tabla de verdad de la implementación del circuito.

c) El diagrama de tiempos usando el archivo TEST_VECTORS.

(Solicite a su instructor que le asigne un ejercicio).

1. **F1** (**A,B,C,D**) = **A' B' C' D'** + **A' B' D'** + A B **C' D'**

   F2 (**A,B,C,D**) = (A + B + **C'** + D) (A + B + **D'**) (**A'** + B + **C'** + **D'**)

2. **F1** (**A,B,C,D**) = **A' B' C' D'** + **A' B' D'** + **C' D'**
   F2 (**A,B,C,D**) = (A + B + **C'** + D) (A + B + **D'**) (**A'** + **C'**+ **D'**)
3. **F1** (**A,B,C,D**) = **A' B' C' D'** + A B **D'** C + **C' D'**
   F2 (**A,B,C,D**) = (**A'** + B + **C'** + **D'**) (A + B + **D'**) (**A'** + **C'** + **D'**)
4. **F1** (**A,B,C,D**) = **A' B' C'** D + **A'** C **D'** + **C'** D
   F2 (**A,B,C,D**) = (**A'** + B + **C'** + D)(A + C + D)(**A'** + B + **C'** + **D'**)
5. **F1** (**A,B,C,D**) = **A'** B **C'** D + **A'** C **D'** + **C'** D
   F2 (**A,B,C,D**) = (**A'** + **B'** + **C'** + D)(A + C + D)(**A'** + B + **C'** + **D'**)
6. **F1** (**A,B,C,D**) = **A'** B **C'** D + **A'** C **D'** + **A'** D
   F2 (**A,B,C,D**) = (**A'** + **B'** + **C'** + D)(A + **C'** + D)(**A'** + B + **C'** + **D'**)
7. **F1** (**A,B,C,D**) = **A'** B **C'** D + **A'** C **D'**+ **A'** B
   F2 (**A,B,C,D**) = (**A'** + **B'**+ **C'**+ D)(A + B + D)(**A'** + B + **C'** + **D'**)
8. **F1** (**A,B,C,D**) = **A'** B **C'** D + **A' D'** + **A'** B D
   F2 (**A,B,C,D**) = (A + **B'** + **C'** + D)(**A'** + B + D)(**A'** + B + **C'** + **D'**)
9. **F1**(**X, Y, Z, W**) = **X' Z' W'** + **X'** Y **W'** + **X'** Y
   F2(**X, Y, Z, W**) = (X + **Y'**+ **Z'** + W)(**X'** + Y + W)(**Y'** + **Z'** + **W'**)
10. **F1**(**X, Y, Z, W**) = **X' Z'** Y **W'** + **X'** Y **W'**+ **X'** Y
    F2(**X, Y, Z, W**) = (X + **Z'** + W)(**X'** + Y + W)(**Y'** + **Z'** + **W'**)
11. **F1**(**X, Y, Z, W**) = **X' Z'** Y **W'** + **X' W'** + X Y
    F2(**X, Y, Z, W**) = (**X'** + **Z'** + W)(**X'** + Y + W)(Y + **Z'** + **W'**)
12. **F1**(**X, Y, Z, W**) = **X'** **Z'** Y W + **X'** W + **X'** Y
    F2(**X, Y, Z, W**) = (**X'** + **Z'** + Y + W)(**X'** + Y + W)(Y + **W'**)
13. **F1** (**X, Y, Z, W**) = **X'** Z + **Z' W'** + **X'** Y W
    F2 (**X, Y, Z, W**) = (X + Z + **W'**)(**X'** + **Z'** + W)(Y + **W'**)
14. **F1** (**X, Y, Z, W**) = **X' Y' W'** + **X'** Y W + **X' Z' W'**
    F2 (**X, Y, Z, W**) = X (Y + Z + **W'**) (**Y'** + W)
15. **F1** (**X, Y, Z, W**) = Y Z **W'** + **X'** Z **W'** + **X'** Y
    F2 (**X, Y, Z, W**) = (X + W) (X + **Z'**) (X + **Y'**) (**Y'** + W) (**Y'** + **Z'**)
16. **F1** (**X, Y, Z, W**) = **Y' W'** + **X' Z' W'** + **X' Y'**
    F2 (**X, Y, Z, W**) = (X + Y) (X + W) (Y + Z) (Y + W)

Cuestionario

1. ¿Cuál es el significado de OLMC?

2. ¿Cuál es el significado de E^2CMOS?

3. Además del GAL16V8, ¿qué otros tipos de GAL existen?

4. ¿Qué significado tiene la expresión .X. en el archivo *Test_Vectors*?

Reporte

Elabore el reporte correspondiente a cada práctica con las siguientes especificaciones:

4.1 Portada

a) Nombre de la práctica

b) Fecha de realización

c) Nombre y número de matrícula

d) Nombre del instructor

4.2 Introducción (explicar el objetivo de la práctica)

4.3 Procedimiento y metodología[1]

4.4 Representación de la función mediante diagrama de alambrado, diagrama esquemático, circuito, ecuación o tabla de verdad[2]

4.5 Archivo reporte que indique las terminales del circuito.

4.6 La gráfica obtenida en la simulación *Test_Vectors*.

4.7 Las ecuaciones mínimas en las formas SOP y POS de F1 y F2 del ejercicio que se le asignó. Para obtener los resultados puede utilizar manipulación algebraica o mapas de Karnaugh.

4.8 Resultados, conclusiones y recomendaciones[3]

4.9 Cuestionario resuelto que aparece al final de la práctica, en su caso

4.10 Referencias bibliográficas

[1] Tanto el procedimiento como la metodología deben explicarse.

[2] Un reporte con diagramas y sin explicaciones ni comentarios carece de valor.

[3] Los resultados deben de analizarse y comentarse.

PRÁCTICA 5

Ecuaciones booleanas y el uso del lenguaje de descripción de hardware ABEL-HDL

Objetivos particulares

Para lograr el objetivo de esta práctica el alumno obtendrá:

- La ecuación partiendo de una tabla de verdad y utilizando la selección de minitérminos y/o maxitérminos.
- La tabla de verdad partiendo de la descripción del problema y, posteriormente, las ecuaciones y el circuito.

- El archivo en lenguaje de descripción de hardware en formato ABEL-HDL y, con las ecuaciones obtenidas anteriormente, programar el GAL16V8D.
- El diagrama de tiempos usando el archivo TEST_VECTORS.
- La implementación del circuito.

El tiempo estimado para el estudio de esta práctica es de dos horas.

Material necesario para el desarrollo de esta práctica

- Una fuente de voltaje de 5VCD.
- Una tablilla de conexiones (*protoboard*).
- Un GAL16V8D (*Lattice semiconductor*) o equivalente.
- Un DIP de ocho o cuatro entradas.
- Seis LED sin importar el color.
- Seis resistencias de 330 OHMS.
- Alambre para conexiones.
- Un disco de 3.5 pulgadas de alta densidad formateado a 1.44 MB.

Fundamento teórico

Minitérmino	Término producto **AND** que contiene todas las variables de la función, ya sea en su forma normal o complementada, cuyo valor de salida es 1 únicamente en una combinación de variables.
Maxitérmino	Término suma **OR** que contiene todas las variables de la función, ya sea en su forma normal o complementada y su valor de salida es 0 únicamente en una combinación de variables.

Por medio de los minitérminos y/o maxitérminos se pueden obtener las ecuaciones, partiendo de una tabla de verdad. Por ejemplo:

m	A B C	F3	F4
0	0 0 0	1	0
1	0 0 1	0	1
2	0 1 0	0	1
3	0 1 1	1	1
4	1 0 0	1	0
5	1 0 1	0	1
6	1 1 0	0	1
7	1 1 1	0	0

En **F3,** la salida es 1 solamente *si se da* la combinación **000** (m_0), la combinación **011** (m_3) o la combinación **100** (m_4).

Para que la combinación **000** (m_0) sea 1 en la salida, se requiere de una **AND**, donde se nieguen sus tres entradas. Se puede escribir: **A' B' C'**. Esta expresión es un minitérmino, ya que es un **AND** que contiene todas las variables de la función, y su valor de salida es 1 únicamente para la combinación especificada.

Asimismo, para los otros dos casos, las expresiones serían las siguientes: para la combinación **011** (m_3), **A' B C;** y para la combinación **100** (m_4), **A B' C'. F3** se puede expresar como:

F3(A, B, C) = **A' B' C' + A'** B C **+** A **B' C'** forma **SOP** sumatoria de productos.

F3(A, B, C) = Σm (0, 3, 4) forma canónica o expansión de minitérminos, donde se indican sólo las combinaciones de la tabla cuyo valor de salida es 1.

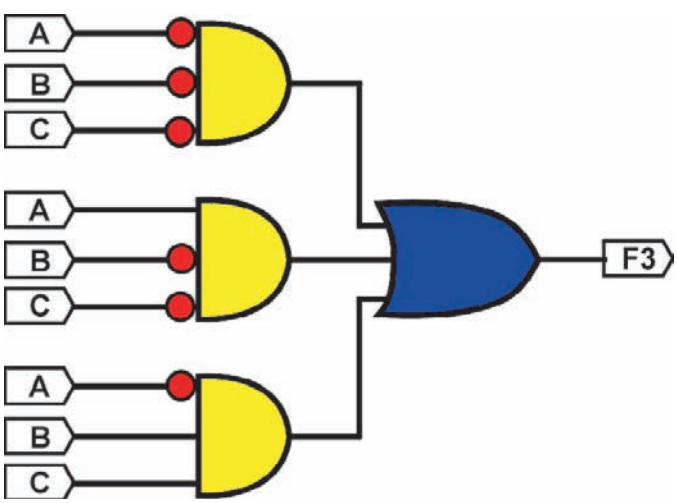


Diagrama esquemático de la función **F3**

En **F4**, la salida es uno solamente si *no se dan* las combinaciones **000** (m_0), **100** (m_4) y **111** (m_7).

m	A B C	F3	F4
0	0 0 0	1	0
1	0 0 1	0	1
2	0 1 0	0	1
3	0 1 1	1	1
4	1 0 0	1	0
5	1 0 1	0	1
6	1 1 0	0	1
7	1 1 1	0	0

Para que la salida de la combinación **000** (m_0) ***no sea 1***, es decir, igual a cero, se requiere un operador **OR,** donde sus tres entradas sean afirmadas **A + B + C**. Esta expresión es un maxitérmino, ya que es un **OR** que contiene todas las variables y su valor de salida es 0 solamente en la combinación especificada.

Asimismo, para los otros dos casos, las expresiones serían **A'**+ B + C para la combinación 100 (m_4), y **A'+ B' + C'** para la combinación 111 (m_7).

Se puede expresar **F4** como:

F4(A, B, C) = (A + B + C)(**A'** + B + C)(**A' + B' + C'**) forma **POS** producto de sumas.

F4(A, B, C) = Πm (0, 4, 7) forma canónica, donde se indican sólo las combinaciones de la tabla cuya salida vale 0.

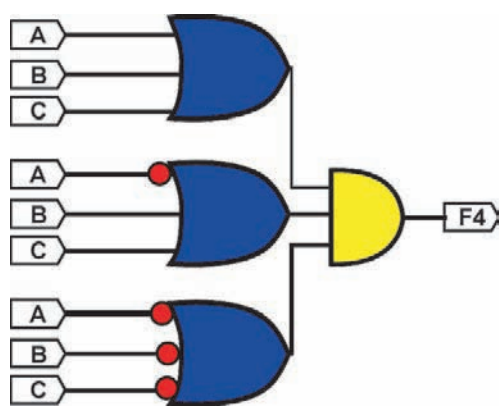


Diagrama esquemático de la función F4

Lenguaje de descripción de hardware ABEL-HDL

A continuación se presenta una descripción de algunas características y sintaxis del lenguaje ABEL-HDL.

ABEL(*Advanced Boolean Expression Language*, lenguaje avanzado de expresiones booleanas) fue desarrollado por Data I/O Corporation para la implementación de funciones booleanas en dispositivos lógicos programables (PLD).

ABEL se utiliza para describir el comportamiento de un sistema digital partiendo de:

- Ecuaciones booleanas.

- La descripción del comportamiento usando instrucciones WHEN-THEN.
- Tablas de verdad.
- Tablas de estado.
- Diagramas de transición.

ABEL es un **archivo de texto** que contiene los siguientes elementos:

1.	Documentación, incluyendo nombre del programa y comentarios.
2.	Declaraciones que identifican las entradas y salidas de las funciones lógicas que serán efectuadas.
3.	Instrucciones que especifican las funciones lógicas que se realizarán.
4.	Declaración del tipo de dispositivo en que las funciones lógicas especificadas se implementarán.
5.	Vectores de prueba que especifican las salidas esperadas de las funciones lógicas para ciertas entradas.

ABEL necesita un procesador de lenguaje llamado compilador, cuyo trabajo consiste en traducir el archivo de texto de ABEL a un mapa de fusibles (JEDEC) del dispositivo físico seleccionado, pasando por un proceso de validación de las instrucciones, así como de minimización de las funciones para ajustar, si es posible, la capacidad del dispositivo elegido.

Sintaxis básica de ABEL-HDL

Identificadores

Los identificadores se emplean para definir variables, cuyas reglas de uso son:

1.	Los identificadores no pueden ser mayores de 31 caracteres. Por ejemplo: Este_es_un_identificador_largo Esteesunidentificadorlargo
2.	Deben de iniciar con un carácter alfabético o con un guión bajo. Por ejemplo: HELLO Hello _K5input P_h
3.	Los identificadores sí son sensibles a mayúsculas o minúsculas. Por ejemplo: el identificador *output* es un identificador diferente de *Output* o de *OUTPUT*.
4.	Los identificadores pueden separarse por comas: A, B, C.
5.	En las expresiones, los identificadores o números pueden separarse por operadores (o donde los paréntesis ofrecen la separación).

Identificadores no válidos:

7_ $4 Pues deben de comenzar con letra o guión bajo.

Hel.lo Pues no se deben usar puntos.

B6 kj Pues no se debe utilizar espacio, y se interpreta como dos identificadores B6 y kj.

Palabras clave (*keywords*)

Las palabras clave son identificadores reservados que se pueden escribir con minúsculas o mayúsculas, o una combinación de ambas. A continuación se listan las palabras clave más comunes:

Declarations	device	else	End	equations
Goto	If	istype	Macro	module
Pin	State	state_diagram	state_register	test_vectors
Then	Title	truth_table	When	With

Las palabras clave deben ir separadas, al menos, por un espacio. En tanto que las líneas escritas en un archivo ABEL deben cumplir con los siguientes requisitos:

1.	Una línea no puede exceder de 150 caracteres.
2.	Empezar los comentarios con comillas (**"**).
3.	Las líneas o instrucciones terminan con punto y coma (**;**).

Los caracteres ASCII soportados son:

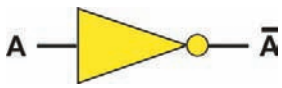
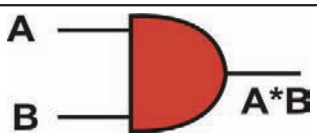
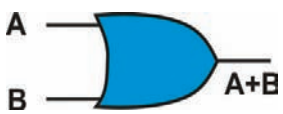
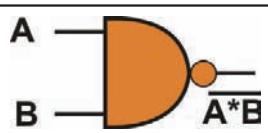
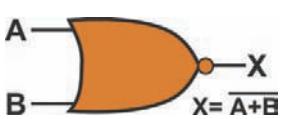
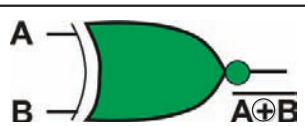
a - z (alfabeto minúsculo).

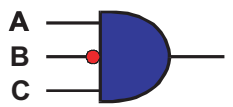
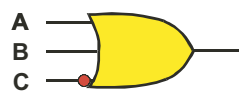
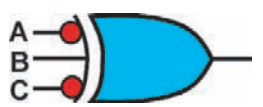
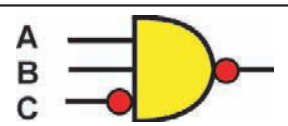
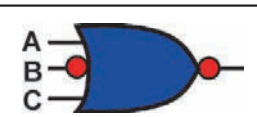
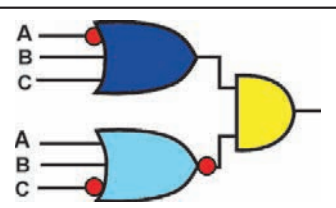
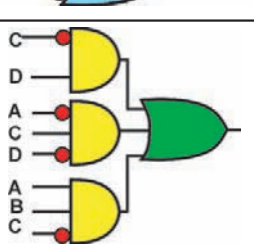
A - Z (alfabeto mayúsculo).

0 - 9 (dígitos).

<space> <tab>

! @ # $? + & * ( ) -_. = + [ ] { } ; : ' " ` \ | , < > . / ^ %.

Operadores lógicos			
Operador	**Descripción**	**Ecuación**	**Símbolo**
!	NOT	!A	
&	AND	A&B	
#	OR	A#B	
$	EXOR	A$B	
!&	NAND	!(A&B)	
!#	NOR	!(A#B)	
!$	EXNOR	!(A$B)	

Ejemplos de ecuaciones con ABEL-HDL	
Circuito	**Ecuación**
	**A & !B & C**
	**A # B # !C**
	**!A $ B $!C**
	**!(A & B & !C)**
	**!(A # !B # C)**
	**(!A # B # C)& !(A # B # !C)**
	**!C&D # !A&C&!D # A&B&!C**

Números (*numbers*)

Los números se utilizan en cuatro diferentes bases: binario, octal, decimal y hexadecimal.

Si no se especifica una base ABEL-HDL, se tomará como base decimal. Para indicar una base diferente del decimal es necesario utilizar el símbolo **^** y la inicial de la base.

Nombre	Base	Símbolo
Binario	**2**	**^b**
Octal	**8**	**^o**
Decimal	**10**	**^d** *(default)*
Hexadecimal	**16**	**^h**

Ejemplos:

Base	Especificación en ABEL	Valor decimal
Decimal	**35**	**35**
Hexadecimal	**^h35**	**53**
Binario	**^b101**	**5**
Octal	**^o22**	**18**

Declaraciones

Es una colección de señales o constantes usadas como referencia de un grupo de expresiones simplificadas en un solo nombre.

Ejemplos:

Y = [D0, D1, D2, D4, D5]:

X= [A, B, C, D];

aset = [a2,a1,a0]; bset = [b2,b1,b0];

COUNT = [Q9, Q8, Q7, Q6, Q5, Q4, Q3, Q2, Q1, Q0];

En ABEL-HDL es posible cambiar las expresiones **.X.** simplemente por **X** o **.C.** por C usando una igualdad como se indica a continuación:

X = .X.; **C = .C.** o las dos a la vez **C, X = .c.,.x.;**

Set

Es una lista de constantes o variables que están separadas por comas o por dos puntos seguidos (**..**) que indican el rango del operador. En esta opción se requieren los paréntesis rectangulares.

Ejemplos:

[D0..D6]	Rango [D0,D1,D2,D4,D5,D6]
[b6..b0]	" Decremento el rango
[D7..D15]	Rango parcial
[b1,b2,a0..a3]	" Combinación de variables y rango
[!S7..!S0]	" Decremento el rango con nivel activo bajo

NOTA: Dentro del rango no se permite usar diferentes nombres de variables [**X0..D5**}.

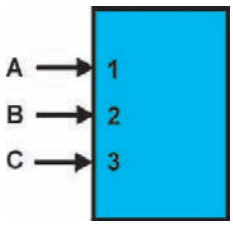
Partes de un programa en ABEL-HDL

1.	*Module*. Inicio del programa.
2.	*TITLE.* Líneas de título y comentarios (opcional).
3.	*Declaration.* Asignación de las terminales de entrada y salida del dispositivo.
4.	*Descripción lógica*. Ecuaciones, tablas de verdad, etcétera.
5.	*TEST_VECTORS*. Vectores de prueba (opcional).
6.	*End*. Final del programa.

Construcción del archivo en ABEL-HDL

1. Al inicio todo programa debe contener la instrucción *Module* y, al final, *End* que indican el principio y el final del programa.
2. Los cometarios y las líneas de título son opcionales, pero es conveniente utilizarlos para describir el funcionamiento y las partes del programa. Éstos deberán empezar con comillas (**"**). Por ejemplo: "Entradas.
3. *Declarations.* Usando este comando es posible declarar las entradas y salidas del sistema.

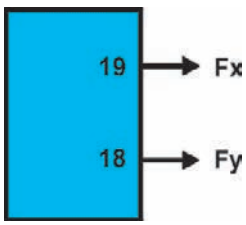
La declaración de las variables de entrada y la asignación de terminales dentro de un GAL, como se muestra en la figura, se describen en la siguiente línea: **A, B, C PIN 1,2,3;**



Observe que las variables de entrada están separadas por comas (**,**) seguidas del comando **PIN** y la terminal correspondiente a cada variable; además se cierra la instrucción con punto y coma (**;**).

Es necesario que la terminal asignada como entrada cumpla con esa función en el GAL. En su caso **GAL16V8** tiene ocho terminales definidas como entradas exclusivas 2,3,4,5,6,7,8,9; además, la terminal 1 sirve como entrada o señal de reloj (*Ck*), y las otras ocho pueden ser entradas o salidas: 11,12,13,14,15,16,17,18 y 19.

Al definir tanto las variables de salida como su asignación de terminales, es necesario incluir el comando *ISTYPE 'COM'* para indicar que son salidas y combinaciones. Por ejemplo:



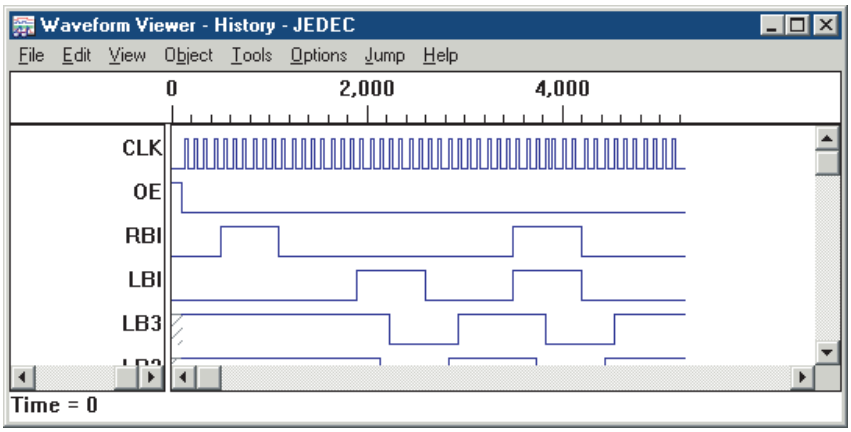
FX,FY PIN 19,18 *ISTYPE* 'COM';

Las terminales designadas como salidas en un **GAL16V8** son 11,12,13,14,15,16,17,18 y 19.

4. *Descripciones Lógicas.* En esta sección se usan los comandos:

EQUATIONS	Permite expresar las ecuaciones.
TRUTH_TABLE	Permite declarar una tabla de verdad o tabla de estados.
WHEN y **THEN**	Permite referir el comportamiento en algunos casos.
STATE_TABLE	Permite describir el comportamiento del diagrama de transición.

5. *Vectores de prueba* (*Test_Vectors*). Esta parte es opcional y es posible efectuar la comprobación o simulación del diseño sin necesidad de implementarlo.



Ejemplo de simulación de vectores de prueba (*Test_Vectors*).

Estructura del archivo en lenguaje ABEL-HDL

Encabezado	**MODULE EQ**
	Declarations
	“Entradas
Declaraciones	**A,B,C PIN 1,2,3;**
	“Salidas
	FX,FY PIN 19,18 ISTYPE ‘COM’;
	EQUATIONS
Descripciones lógicas	**FX= A & ¡B & C # ¡B & C;** **FY = (A # !B # C) & (A # !C);**
Vectores de prueba	**TEST_VECTORS**
	([A,B,C]->[FX,FY])
	[0,0,0]->[.X.,.X.];
	[0,0,1]->[.X.,.X.];
	[0,1,0]->[.X. ,.X.];
	[0,1,1]->[.X. ,.X.];
	[1,0,0]->[.X. ,.X.];
	[1,0,1]->[.X. ,.X.];
	[1,1,0]->[.X. ,.X.];
	[1,1,1]->[.X. ,.X.];
Final	**END**

Ejemplo 5.1

Construya la tabla de verdad con base en el análisis del problema.

En un auditorio se tienen grupos de cuatro sillas llamadas **A**, **B**, **C** y **D,** distribuidas como se indica en la siguiente figura:

Cada una de ellas contiene un sensor, de manera que detecta cuando está ocupada, indicando con un **1,** y cuando está vacía con un **0**.

Obtenga la función booleana $\mathbf{F}_{(A, B, C, D)}$, la cual será 1 cuando dos sillas adyacentes se encuentren vacías.

Para obtener la $F_{(A, B, C, D)}$ solicitada, se construye una tabla de verdad donde para las cuatro variables de entrada se tienen 16 posibles combinaciones, que se muestran a continuación:

M	A	B	C	D	F
0	0	0	0	0	
1	0	0	0	1	
2	0	0	1	0	
3	0	0	1	1	
4	0	1	0	0	
5	0	1	0	1	
6	0	1	1	0	
7	0	1	1	1	
8	1	0	0	0	
9	1	0	0	1	
10	1	0	1	0	
11	1	0	1	1	
12	1	1	0	0	
13	1	1	0	1	
14	1	1	1	0	
15	1	1	1	1	

Usando un valor de 1 en la salida **F** se indica para cuáles combinaciones cumple, cuando dos sillas adyacentes se encuentren vacías, como se muestra en la tabla siguiente:

Tabla de verdad de la función F

M	A	B	C	D	F
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0

Como la cantidad de unos es igual a la cantidad de ceros en la función $F_{(A, B, C, D),}$ la función se expresa en minitérminos o maxitérminos.

$F_{(A, B, C, D)} = \Sigma$ (0, 1, 2, 3, 4, 8, 9, 12)

$F_{(A, B, C, D)}$ = **A' B' C' D'** + **A' B' C'** D + **A' B'** C **D'** + **A' B'** C D

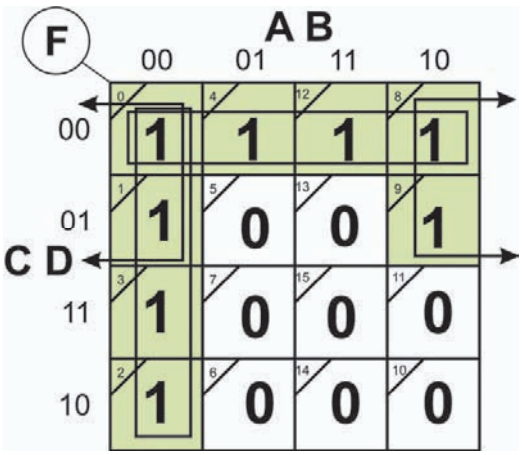
+**A'** B **C' D'** + A **B' C' D'**+ A **B' C'** D + A B **C' D'**

$F_{(A, B, C, D)} = \Pi$ (5, 6, 7, 10, 11, 13, 14, 15)

$F_{(A, B, C, D)}$ = (A **+ B' +** C **+ D´**)(A **+ B' + C' +** D)(A **+ B' + C' + D´**)(**A' +** B **+ C' +** D) **(A' +** B **+ C' + D')** (**A' +** B' + C **+ D**') (**A**' **+ B' + C' +** D) **(A**' **+ B' + C' + D')**

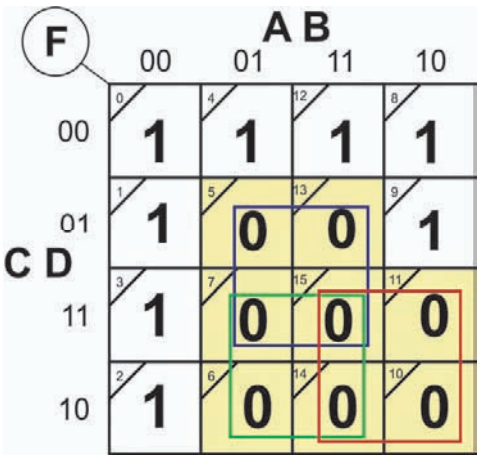
Estos resultados quizá no sean la mínima expresión de la función. Podemos comprobarlo a través de manipulación algebraica, mapas de Karnaugh o algún software para reducir funciones booleanas.

Con los mapas de Karnaugh es posible obtener la expresión mínima de la función.



Donde el resultado de la expresión mínima en forma SOP es:

$F_{(A, B, C, D)}$ = **A' B' + B' C' + C' D'**

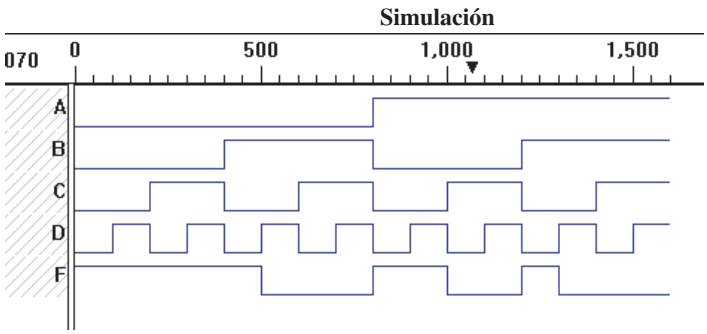


Donde el resultado de la expresión mínima en forma POS es:

$$F_{(A, B, C, D)} = (\ \mathbf{B'} + \mathbf{D'}\ )(\ \mathbf{A'} + \mathbf{C'}\ )(\ \mathbf{B'} + \mathbf{C'}\ )$$

Archivo en formato ABEL-ABL para la expresión mínima de SOP

```
MODULE sillas
“entradas
A,B,C,D Pin 1..4;
“Salida
F pin 19 istype ‘com’;
“miniterminos F(A,B,C,D)= (0,1,2,3,4,8,9,12)
Equations
F=!A &!B  + !B &!C + !C & !D;
Test_vectors
([A,B,C,D]->F)
[0,0,0,0]->.x.;
[0,0,0,1]->.x.;
[0,0,1,0]->.x.;
[0,0,1,1]->.x.;
[0,1,0,0]->.x.;
[0,1,0,1]->.x.;
[0,1,1,0]->.x.;
[0,1,1,1]->.x.;
[1,0,0,0]->.x.;
[1,0,0,1]->.x.;
[1,0,1,0]->.x.;
[1,0,1,1]->.x.;
[1,1,0,0]->.x.;
[1,1,0,1]->.x.;
[1,1,1,0]->.x.;
[1,1,1,1]->.x.;
END
```



Trabajo solicitado

Obtenga las ecuaciones para **F3** en minitérminos y **F4** en maxitérminos, capture el archivo en el editor de texto de ABEL-HDL, compile y programe en un circuito GAL16V8, compruebe su tabla de verdad e incluya la simulación.

Procedimiento

1. A partir de la tabla de verdad, obtenga las ecuaciones correspondientes a las funciones **F3** en la forma de *suma de productos SOP* (minitérminos), y **F4** en la forma *productos de suma POS* (maxitérminos).

Tabla de verdad propuesta

M	A B C D	F3	F4
0	0 0 0 0	0	1
1	0 0 0 1	0	1
2	0 0 1 0	1	1
3	0 0 1 1	0	1
4	0 1 0 0	0	1
5	0 1 0 1	0	0
6	0 1 1 0	1	1
7	0 1 1 1	1	1
8	1 0 0 0	0	1
9	1 0 0 1	0	1
10	1 0 1 0	1	1
11	1 0 1 1	0	1
12	1 1 0 0	0	0
13	1 1 0 1	0	0
14	1 1 1 0	1	0
15	1 1 1 1	0	1

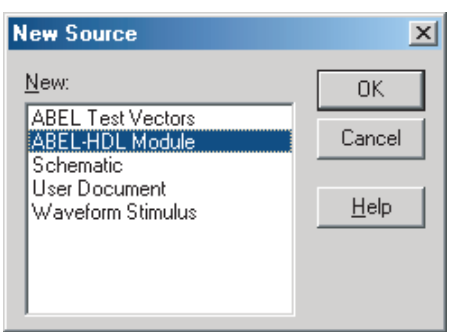
2. Obtenga las ecuaciones para:

F3 (A, B, C, D) =		**(SOP)**
F3 (A, B, C, D) =		**Canónica**
F4 (A, B, C, D) =		**(POS)**
F4 (A, B, C, D) =		**Canónica**

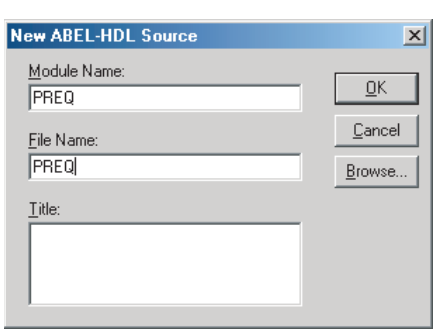
3. Cree un nuevo proyecto en *IspEXPERT System* (asigne un nombre).

4. Seleccione GAL16V8D/ZD.

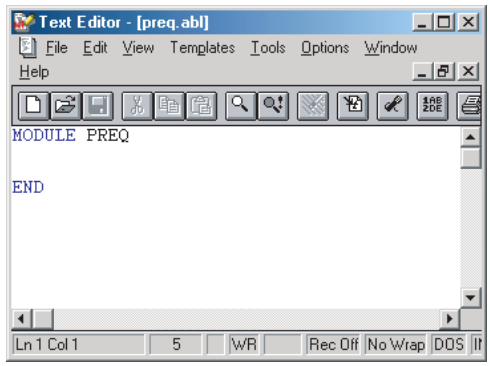
5. Elija una nueva fuente (*Source*) en *ABEL-HDL Module*.

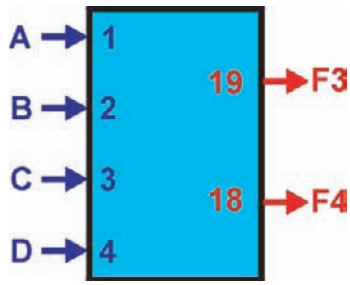


6. Defina el nombre del módulo y el archivo ABEL.

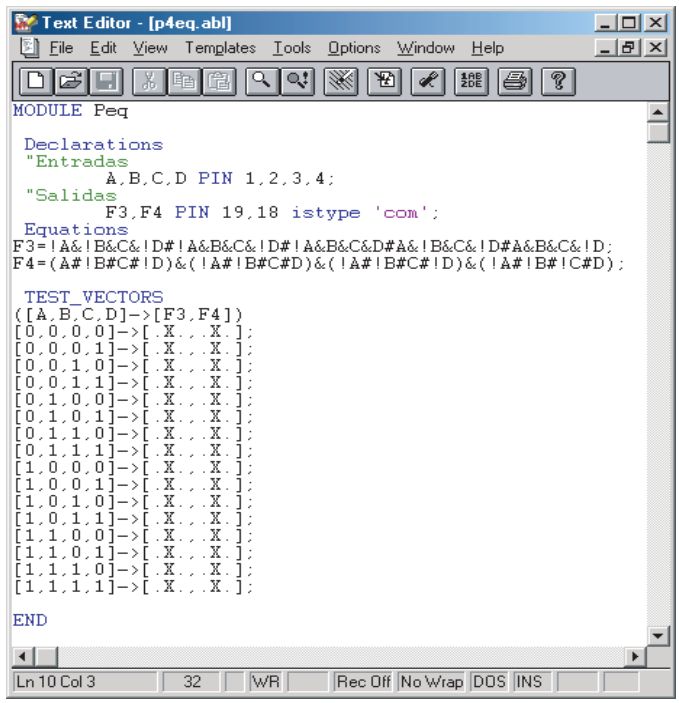


7. Capture el archivo ABEL-HDL para las funciones **F3** y **F4** en el editor de textos de ABEL y asigne las terminales 1, 2, 3 y 4 a A, B, C y D, respectivamente. Para las salidas F3, F4 asigne las terminales 19, 18.



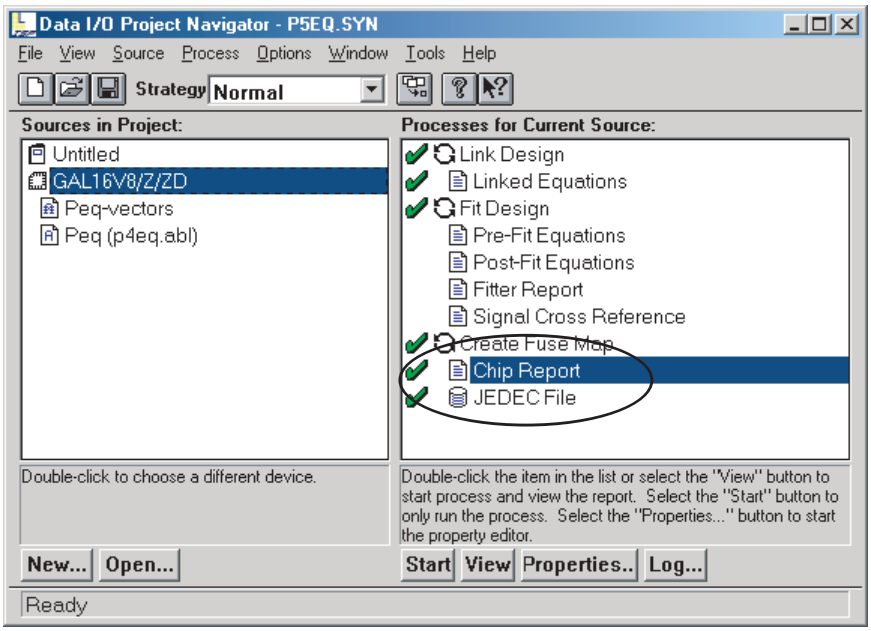


8. Guarde el archivo completo una vez que cumpla con la estructura.
9. Compile el archivo.

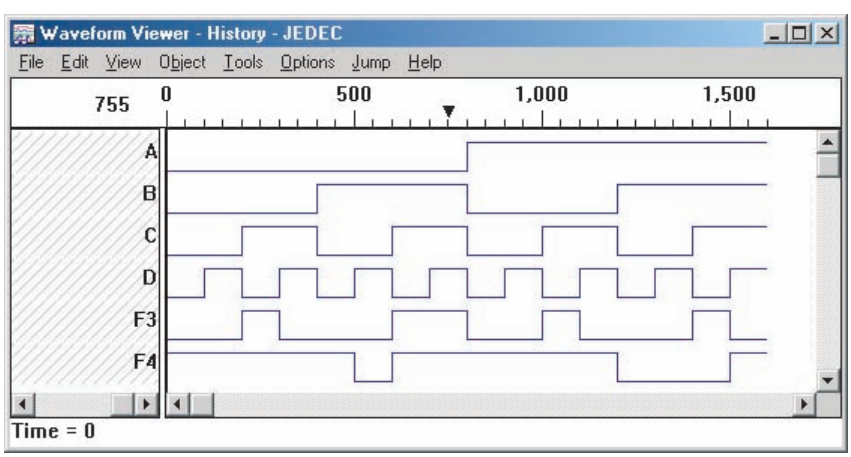


Archivo ABEL-HDL incluyendo *TEST_VECTORS*.

10. Obtenga los archivos Reporte y JEDEC.

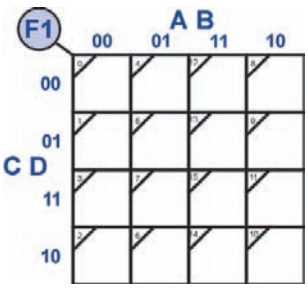


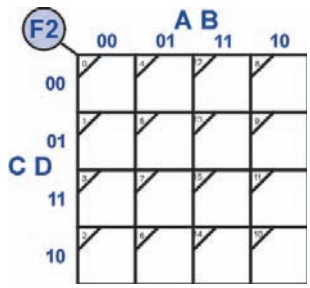
12. Obtenga el diagrama de tiempos.

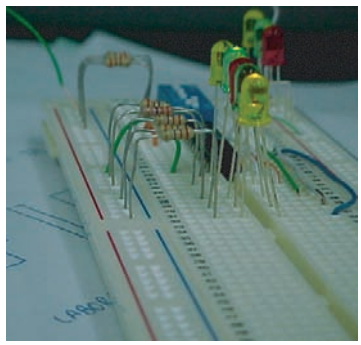


13. Grabe el GAL16V8D y compare su tabla de verdad con el diagrama de tiempos.

14. Usando mapas de Karnaugh simplifique las funciones F3 y F4, y compare los resultados con los que aparecen en el archivo reporte.







Manual de ABEL en http://www.cs.bilkent.edu.tr/~baray/cs223/abel_primer.html

PRÁCTICA 6

Diseño combinacional

Objetivos particulares

Durante el desarrollo de esta práctica se aplicará la metodología del diseño combinacional. Asimismo se obtendrá la implementación del circuito a partir del archivo ABEL-HDL, empleando los comandos *Equations*, TRUTH_TABLE o WHEN THEN para el uso de ecuaciones, tablas de verdad o descripción del problema para programar en un GAL16V8D. También se calculará el diagrama de tiempos usando el archivo TEST_VECTORS.

El tiempo estimado para el estudio de esta práctica es de dos horas.

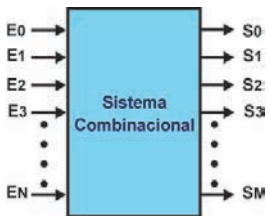
Material necesario para el desarrollo de esta práctica

- Una fuente de voltaje de 5VCD.
- Una tablilla de conexiones (*protoboard*).

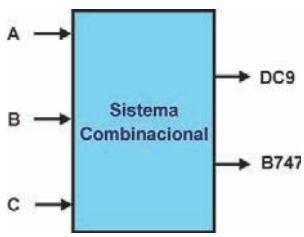
- Un GAL16V8D (*Lattice semiconductor*).
- Seis resistencias de 330 ohms.
- Un DIP de ocho entradas.
- Seis LED sin importar el color.
- Seis resistencias de 330 ohms.
- Alambre para conexiones.
- Un disco de 3.5 pulgadas de alta densidad formateado a 1.44 MB.

Fundamento teórico

Un *sistema combinacional* es aquel donde los valores de salida dependen únicamente de las combinaciones de entrada. En este sistema el número de entradas puede ser mayor, menor o igual al número de salidas.



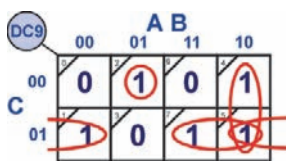
Metodología del diseño combinacional

1. Especificar el sistema.	
2. Determinar entradas y salidas.	

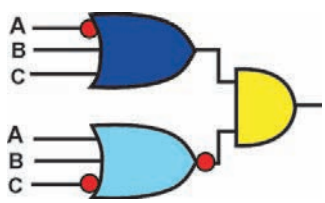
3. Trasladar el comportamiento del sistema a una tabla de verdad.

m	A B C	DC9	B747
0	0 0 0	0	0
1	0 0 1	1	0
2	0 1 0	1	0
3	0 1 1	0	1
4	1 0 0	1	0
5	1 0 1	1	0
6	1 1 0	0	1
7	1 1 1	1	1

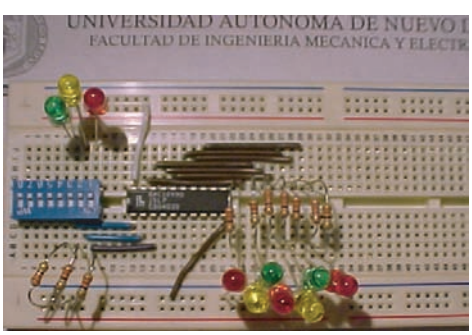
4. Minimizar.



5. Elaborar diagrama esquemático.



6. Implementar.



Ejemplo 6.1

Diseñe un sistema combinacional capaz de cubrir las necesidades de control de aterrizaje de un pequeño aeropuerto, el cual consta de tres pistas llamadas ***A, B*** y ***C***. En ese aeropuerto aterrizan dos tipos de aviones: un **DC9** que requiere de una sola pista para aterrizar y un **B747** que necesita de dos pistas para hacerlo. El avión **B747** tiene prioridad de aterrizar respecto del **DC9**.

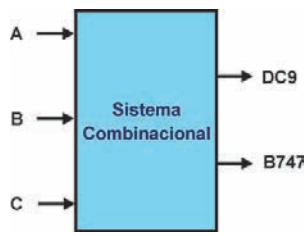
Diseñe un sistema combinacional que determine qué tipo de avión podría aterrizar en función de las pistas disponibles.

1. *Especificar el sistema*. Las variables que intervienen son:

   PISTAS A, B y C { Disponible = **1**; No disponible = **0**

   Aviones DC9 y B747 { Permiso para aterrizar = **1**; No permiso para aterrizar = **0**

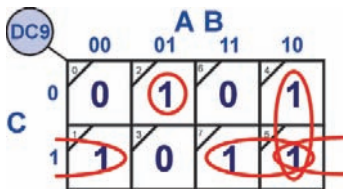
2. *Determinar entradas y salidas*. Donde las pistas ***A***, ***B***, ***C*** son las entradas del sistema; mientras que el permiso para aterrizar para el **DC9** o el **B747** son las salidas que a continuación se representan en un diagrama de bloques.



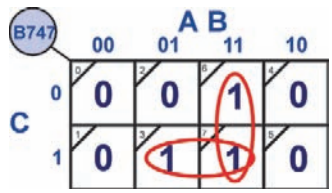
3. *Trasladar el comportamiento del sistema a una tabla de verdad*. Hay que decidir el valor de las salidas (**0** o **1**) para cada una de las combinaciones de entrada:

m	A B C	DC9	B747
0	0 0 0	0	0
1	0 0 1	1	0
2	0 1 0	1	0
3	0 1 1	0	1
4	1 0 0	1	0
5	1 0 1	1	0
6	1 1 0	0	1
7	1 1 1	1	1

4. *Minimizar*. Para hacerlo se utilizan los mapas de Karnaugh para simplificar las funciones **DC9** y **B747**.

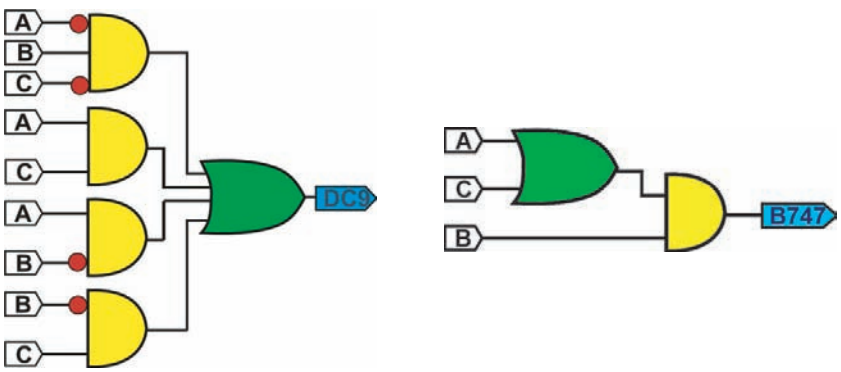


FDC9 (A, B, C) = **A**'B**C**' + A**B**' + AC + **B**'C



FB747 (A, B, C) = AB + BC = B(A + C)

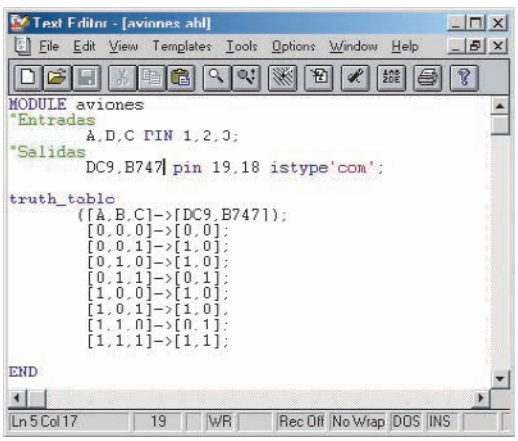
5. *Diagrama esquemático*.



6. *Implementación*. En la implementación, usando el GAL16V8 y ABEL-HDL, es posible eliminar los pasos de *diagrama esquemático* y de *minimizar,* a partir de la tabla de verdad, usando el comando *TRUTH_TABLE*, donde al enlazar se obtienen las ecuaciones minimizadas. Esto último facilita el procedimiento de diseño y optimiza el uso del circuito integrado.

Ejemplo del problema anterior implementado en un dispositivo lógico programable usando el comando TRUTH_TABLE en un archivo formato ABEL-HDL:

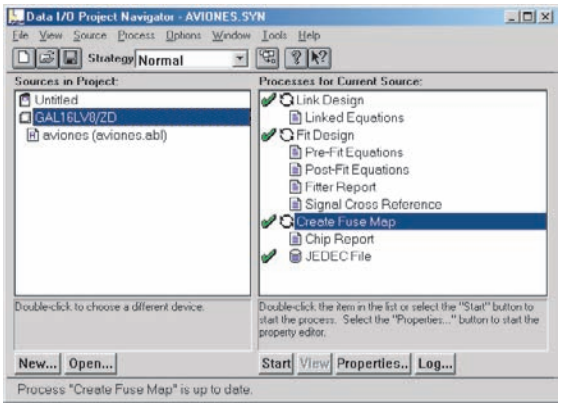
```
MODULE aviones
"Entradas
    A,B,C PIN 1,2,3;
"Salidas
    DC9,B747 pin 19,18 istype'com';
TRUTH_Table
([A,B,C]->[DC9,B747])
[0,0,0]->[0,0];
[0,0,1]->[1,0];
[0,1,0]->[1,0];
[0,1,1]->[0,1];
[1,0,0]->[1,0];
[1,0,1]->[1,0];
[1,1,0]->[0,1];
[1,1,1]->[1,1];
END
```



En el proceso de **compilación** (*link*) se efectúa una minimización partiendo de la tabla de verdad. Las ecuaciones obtenidas se presentan en el Archivo reporte y son las mismas que se obtuvieron al simplificar por medio del mapa de Karnaugh.

DC9 = (!A & B & !C # A & !B # !B & C # A & C);

B747 = (A & B # B & C);



Estructura del archivo ABEL-HDL usando el comando *TRUTH_TABLE*

Encabezado	**MODULE TV**
Declaraciones	Declarations
	"Entradas
	A,B,C PIN 1,2,3;
	"Salidas
	FZ,FW PIN 19,18 ISTYPE 'COM';
Descripciones lógicas	**TRUTH_TABLE**
	([A,B,C]->[FZ,FW]) [0,0,0]->[0,1]; [0,0,1]->[1.0]; [0,1,0]->[1,1]; [0,1,1]->[1,0]; [1,0,0]->[1,1]; [1,0,1]->[0,0]; [1,1,1]->[1,0]; Final **END**

NOTA: La combinación **1, 1, 0** no aparece en la tabla de verdad. Las salidas correspondientes a la combinación serán tomadas con un valor de cero:

[1,1,0]->[0,0];.

Trabajo solicitado

Obtenga la tabla de verdad del problema que le asigne su instructor, elabore el archivo en formato ABEL-HDL correspondiente al ejemplo por ecuaciones y/o tabla de verdad. Incluya vectores de prueba para su simulación e impleméntelo en un PLD.

Problema 1

Diseñe un sistema combinacional donde sea posible comparar dos números binarios de dos bits cada número.

Problema 2

Diseñe un sistema combinacional operable para multiplicar dos números binarios de dos bits cada número.

Problema 3

Diseñe un sistema combinacional con la posibilidad de sumar cuatro números binarios de un solo bit cada número.

Problema 4

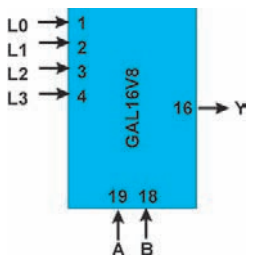
Diseñe un sistema combinacional donde sea posible sumar dos números binarios de dos bits cada número.

Problema 5

Diseñe un sistema combinacional operable para restar dos números binarios de dos bits cada número, tomando en cuenta una salida ***S*** adicional al resultado, que indique con 0 cuando la salida sea positiva o nula (S = 0), y con 1 cuando la diferencia sea negativa (S = 1).

Problema 6

Diseñe un sistema multiplexor (selector datos) de 4 a 1 líneas. Como entradas de datos: **L0**, **L1**, **L2**, **L3.** Como entradas de control: **A** y **B.** Y como salida: **Y.** Debe cumplir con la siguiente tabla.

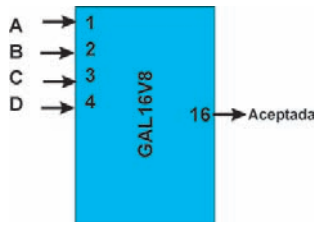


m	A B	Y
0	0 0	L0
1	0 1	L1
2	1 0	L2
3	1 1	L3

(Véase el ejemplo de multiplexor de ocho a una líneas, al final de la práctica).

Problema 7

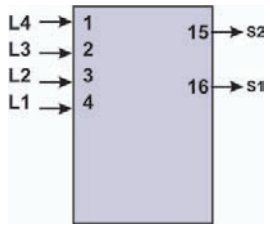
Diseñe un sistema combinacional capaz de indicar la decisión de aceptada o rechazada de una propuesta, cuando el porcentaje de las acciones a favor sea mayor o igual al 60 por ciento.



Tomando en cuenta que la empresa tiene cuatro accionistas y donde el accionista ***A*** tiene el **40%**, el ***B*** el **30%**, el ***C*** el **20%** y el ***D*** el **10%**, en función de la votación de cada uno de los accionistas a favor = **1** o en contra = **0**, el sistema deberá indicar con un uno (Y = 1) en la salida **Y** si cumple con el 60% o más, de lo contrario notificará con un cero (Y = 0).

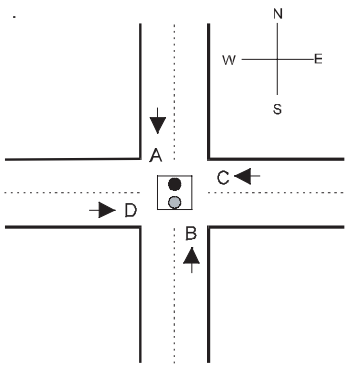
Problema 8

Un sistema de alarma contra incendios se conectará a cuatro conmutadores **X1**, **X2**, **X3** y **X4.** Si se activa uno de estos conmutadores deberá encenderse una sirena **S1.** Si se activan dos o más conmutadores en forma simultánea deberán dar aviso la sirena **S1** y una segunda sirena **S2**.



Problema 9

La figura muestra la intersección de una autopista principal con un camino de acceso secundario. Se colocan detectores de vehículos a lo largo de los carriles **C** y **D** (camino principal) y en los carriles **A** y **B** (camino de acceso).



Las lecturas o salidas de los detectores son **BAJAS** "**0**" cuando no pasa ningún vehículo, y **ALTAS** "**1**" cuando pasa algún transporte. El semáforo del crucero se controlará de acuerdo con la siguiente lógica:

a) El semáforo **E-W** indicará luz verde siempre que los carriles **C** y **D** estén ocupados.

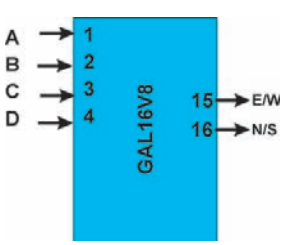
b) El semáforo **E-W** indicará luz verde siempre que **C** o **D** estén ocupados, siempre y cuando **A** y **B** no estén ocupados.

c) El semáforo **N-S** indicará luz verde siempre que los carriles **A** y **B** estén ocupados, siempre y cuando **C y D** no estén ocupados.

d) El semáforo **N-S** también indicará luz verde siempre que los carriles **A** o **B** estén ocupados, en tanto **C** y **D** estén vacantes.

e) El semáforo **E-W** indicará luz verde cuando no haya vehículos transitando.

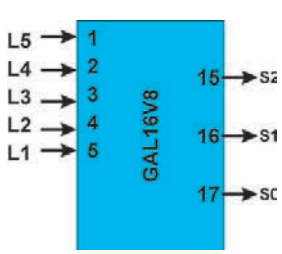
Utilizando las salidas de los sensores **A**, **B**, **C** y **D** como entradas de un sistema combinacional, diseñe un circuito lógico para controlar el semáforo.



Debe tener dos salidas **N/S** y **E/W** que sean ALTO "1" cuando corresponda la luz verde, y bajo "0" cuando sea el turno de la luz roja.

Problema 10 (*encoder*)

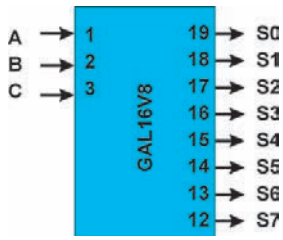
Diseñe un sistema combinacional que contenga cinco entradas llamadas **L5**, **L4**, **L3**, **L2** y **L1** capaz de indicar (mediante un código binario **S2**, **S1**, **S0**) cuál entrada tiene valor 1. En caso de que se presenten dos o más unos (1) en la entrada, la salida tomará el valor de la línea de mayor peso; por ejemplo:



Si se presenta la combinación de entrada L5 = **0**, L4 = **1**, L3 = **0**, L2 = **1**, L1 = **1**, la salida será igual a S2 = 1, S1 = 0, S0 = 0 que corresponde a la línea 4 ($100_{(2)}$). En el caso de que todas las líneas de entrada sean iguales a 0, entonces la salida será igual a S2 = 0, S1 = 0, S0 = 0 ($000_{(2)}$).

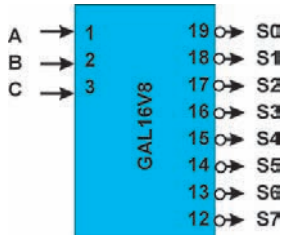
Problema 11 (*decoder*)

Diseñe un sistema combinacional de tres entradas llamadas **A**, **B** y **C** con ocho salidas llamadas **S7**, **S6**, **S5**, **S4, S3**, **S2**, **S1**, **S0,** de manera que cuando la entrada sea igual a A = 0, B = 0 y C = 0, sólo la salida S0 sea igual a 1. Si la entrada es igual a A = 0, B = 0 y C = 1, sólo la salida S1 sea igual a 1 y así sucesivamente, hasta que la entrada sea A = 1, B = 1 y C = 1; entonces sólo la salida S7 será igual a 1.



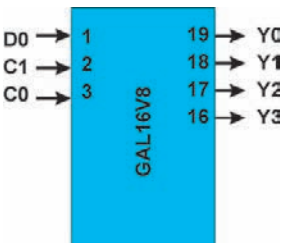
Problema 12 (*decoder*)

Diseñe un sistema combinacional de tres entradas llamadas **A**, **B** y **C** con ocho salidas llamadas **S7**, **S6**, **S5**, **S4**, **S3**, **S2**, **S1**, **S0,** de manera que cuando la entrada sea igual a A = 0, B = 0 y C = 0, sólo la salida S0 = 0 sea igual a 0. Si la entrada es igual a A = 0, B = 0 y C = 1, sólo la salida S1 = 0 sea igual a 0, y así sucesivamente, hasta que la entrada sea A = 1, B = 1 y C = 1; entonces sólo la salida será S7 = 0.



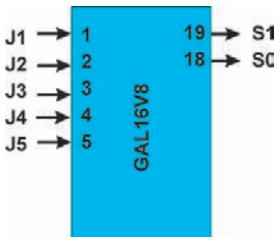
Problema 13 (*demultiplexer*)

Diseñe un sistema combinacional que contenga una entrada de dato **D0**, dos entradas de control **C1**, **C0** y cuatro señales de salida llamadas **Y0**, **Y1**, **Y2** y **Y3**, de manera que si la entrada de control es **C1 = 0**, **C0 = 0**, la salida **Y0** tomará el valor del **D0** y las demás salidas tomarán el valor de 0. Si la entrada de control es **C1 = 0**, **C0 = 1,** la salida **Y1** tendrá el valor del **D0** y las demás salidas el valor de 0. Si la entrada de control es **C1 = 1**, **C0 = 0**, la salida **Y2** tomará el valor del **D0** y las demás salidas el valor de 0. Si la entrada de control es **C1=1**, **C0=1**, la salida **Y3** tendrá el valor del **D0** y las demás salidas el valor de 0.



Problema 14

Hay cinco personas que actúan como jueces en una competencia. El voto de cada uno de ellos se indica en una línea de señal con un **1** cuando el participante pasa la prueba, o con un **0** cuando fracasa. Las cinco líneas **J1**, **J2**, **J3 J4** y **J5** son la entrada de un sistema combinacional.

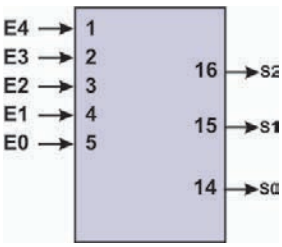


Las reglas de la competencia sólo permiten la diferencia de un voto. Si la votación es 2-3 o 3-2, la competencia debe continuar.

El sistema deberá tener dos salidas **S1** y **S0**. Si el voto es 4-1 o 5-0 a favor, entonces la salida será igual a S1, S0 = 1, 1. Si el voto es 4-1 o 5-0 en contra, la salida será igual a S1, S0 = 0, 0. Si el voto es 3-2 o 2-3 la salida será igual a S1, S0 = 1, 0.

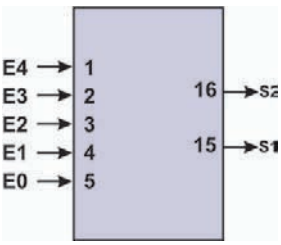
Problema 15

Diseñe un sistema combinacional de cinco entradas (***E4***, ***E3***, ***E2***, ***E1***, y ***E0***) que contenga tres salidas, de manera que la primera (***S2***) señale con **1** las combinaciones de entrada que sean números primos, la segunda (***S1***) indique de la misma forma las combinaciones pares, y la última salida (***S0***), las combinaciones impares.



Problema 16

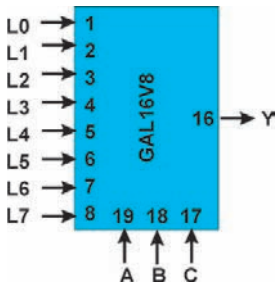
Diseñe un sistema combinacional de cuatro entradas (***E3***, ***E2***, ***E1*** y ***E0***) que contenga dos salidas, de tal forma que la primera (***S2***) señale con **1** las combinaciones de entrada que sean divisibles entre tres, y la segunda (***S1***) indique de la misma forma las combinaciones que sean divisibles entre dos.



Resumen

A continuación se muestran las diferentes formas de programación en ABEL-HDL para el diseño de un **multiplexor de ocho a una líneas.**

Para este sistema combinacional se tienen ocho líneas, desde **L0** hasta **L7,** como entradas de datos; las entradas **A**, **B** y **C,** como entradas control; y una sola salida **Y**, donde si las entradas de control son A = 0, B = 0 y C = 0, la salida **Y** deberá ser igual a la línea L0. Así, si A = 1, B = 1 y C = 1, la salida **Y** deberá ser igual a la línea L7, como lo indica la siguiente tabla.



m	A B C	Y
0	0 0 0	L0
1	0 0 1	L1
2	0 1 0	L2
3	0 1 1	L3
4	1 0 0	L4
5	1 0 1	L5
6	1 1 0	L6
7	1 1 1	L7

1. Archivo en formato ABEL-HDL por ecuaciones booleanas.

```
MODULE muxeq
“Entradas de datos
        L0..L7 pin 1..8;
“Entradas de control
        A,B,C pin 19,18,17;
“Salida
        Y pin 16 istype ‘com’;
Equations
Y = !A&!B&!C&L0 # !A&!B&C&L1 #!A&B&!C&L2# !A&B&C&L3#
A&!B&!C&L4# A&!B&C&L5 # A&B&!C&L6 # A&B&C&L7;
END
```

2. Archivo ABEL-HDL por tabla de verdad, donde al incluir el comando **.X.** (“Don’t Care”, no importa) en las entradas, se simplifican de 2,048 combinaciones posibles a sólo 16.

```
MODULE mux
"DECLARACIONES
" X = Don't Care
X = .x.;
" Entradas de datos
        L0..L7 pin 1..8;
"Entradas de control
        A,B,C pin 19,18,17;
"Salida
        Y pin 16 istype 'com';
truth_table
([A,B,C,L7,L6,L5,L4,L3,L2,L1,L0]->[Y])
  [0,0,0,X,X,X,X,X,X,X,0]->[0];
  [0,0,0,X,X,X,X,X,X,X,1]->[1];
  [0,0,1,X,X,X,X,X,X,0,X]->[0];
  [0,0,1,X,X,X,X,X,X,1,X]->[1];
  [0,1,0,X,X,X,X,X,0,X,X]->[0];
  [0,1,0,X,X,X,X,X,1,X,X]->[1];
  [0,1,1,X,X,X,X,0,X,X,X]->[0];
  [0,1,1,X,X,X,X,1,X,X,X]->[1];
  [1,0,0,X,X,X,0,X,X,X,X]->[0];
  [1,0,0,X,X,X,1,X,X,X,X]->[1];
  [1,0,1,X,X,0,X,X,X,X,X]->[0];
  [1,0,1,X,X,1,X,X,X,X,X]->[1];
  [1,1,0,X,0,X,X,X,X,X,X]->[0];
  [1,1,0,X,1,X,X,X,X,X,X]->[1];
  [1,1,1,0,X,X,X,X,X,X,X]->[0];
  [1,1,1,1,X,X,X,X,X,X,X]->[1];
  END
```

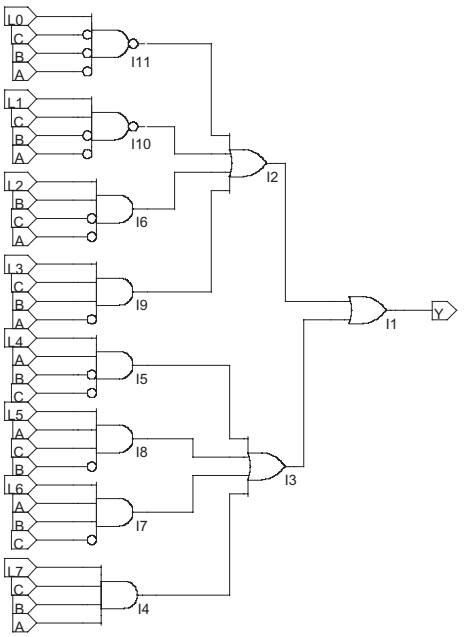
3. Además de las ecuaciones y la tabla de verdad en el archivo ABEL-HDL, se puede usar el comando WHEN, THEN, ELSE como una manera adicional para describir cada uno de los casos, facilitando así su programación.

El formato para el uso se define:

WHEN *descripción lógica*
THEN *ecuación verdadera*
ELSE *ecuación falsa*

```
MODULE muxeq
" Entradas de datos
        L0..L7 pin 1..8;
"Entradas de control
        A,B,C pin 19,18,17;
"Salida
        Y pin 16 istype 'com';
Equations
        WHEN !A&!B&!C THEN Y=L0;
        WHEN !A&!B&C THEN Y=L1;
        WHEN !A&B&!C THEN Y=L2;
        WHEN !A&B&C THEN Y=L3;
        WHEN A&!B&!C THEN Y=L4;
        WHEN A&!B&C THEN Y=L5;
        WHEN A&B&!C THEN Y=L6;
        WHEN A&B&C THEN Y=L7;
END
```

4. Una cuarta opción sería por captura esquemática, utilizada en la práctica 3, como lo muestra la figura.



PRÁCTICA 7

Sistemas combinacionales que no están totalmente especificados

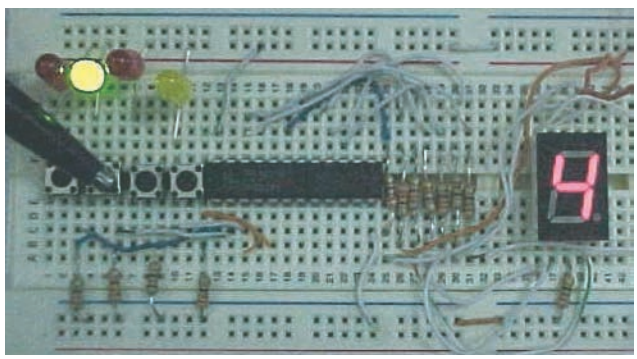
Objetivos particulares

Durante el desarrollo de esta práctica se diseñarán sistemas combinacionales que no están completamente definidos, como el decodificador de un código BCD a siete segmentos o convertidores de código; además analizaremos la conveniencia y el uso de las instrucciones **SET** y **Don't care** para simplificar el archivo en formato ABEL.

El tiempo estimado para el estudio para esta práctica es de dos horas.

Material necesario para el desarrollo de esta práctica

- Una fuente de voltaje de 5VCD.
- Una tablilla de conexiones (*protoboard*).
- Un GAL16V8D (*Lattice semiconductor*) o equivalente.
- Un DIP de ocho entradas.
- Ocho LED.
- Once resistencias de 330 ohms.
- Un *display* de siete segmentos.
- Alambre para conexiones.
- Un disco de 3.5 pulgadas de alta densidad formateado a 1.44 MB.



Fundamento teórico

Un sistema combinacional puede considerase como no completamente especificado, por dos razones:

Can't happen No puede suceder	Ya que una o varias combinaciones de entrada, debido a las características del sistema, se pueden presentar.
Don't care No Importa.	Se trata de un valor de salida al que no importa el valor que se le asigne (no se afecta el sistema no).

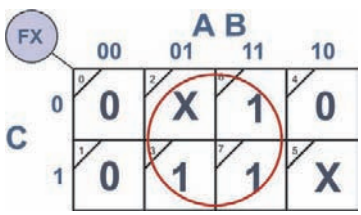
En ambos casos se aprovecha que la entrada no se presente o que en la salida no importe el valor, asignándole un valor de X a la salida en la tabla de verdad. Aquí ese valor de X individualmente se toma como 0 o 1, según convenga a una mejor minimización. Por ejemplo, en el siguiente mapa tenemos:

FX(A, B, C) = ∑m (3, 6, 7), d (2, 5)

m	A B C	FX
0	0 0 0	0
1	0 0 1	0
2	0 1 0	X
3	0 1 1	1
4	1 0 0	0
5	1 0 1	X
6	1 1 0	1
7	1 1 1	1

FX(A, B, C) = B

FX = B



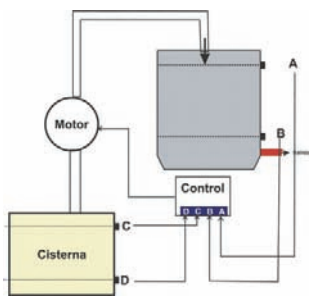
Donde la **X** de la combinación 2 se toma como 1 para contribuir a formar un grupo de cuatro unos y tener una mayor simplificación; en tanto que la **X** de la combinación 5 se toma como 0 para no incluir un grupo más.

Ejemplo 7.1

Sistema de llenado de un tinaco

Diseñe un sistema combinacional capaz de controlar el llenado de un tinaco mediante un sistema hidráulico que está compuesto de una cisterna, un motor y un tinaco.

El sistema cuenta con cuatro sensores[1] de nivel, dos llamados **A** y **B**, correspondientes al nivel alto y bajo, respectivamente, del tinaco. Otros dos llamados **C** y **D** corresponden a los niveles alto y bajo, respectivamente, de la cisterna, como lo indica la siguiente figura.



Los sensores son iguales a **cero** cuando no hay líquido presente y son iguales a **uno** cuando sí lo hay.

[1] Los sensores tienen un rango de operación para evitar oscilaciones del sistema.

Se requiere de una salida **M** que activa el motor de una bomba, con el cual se llevará el agua de la cisterna al tinaco.

El sistema deberá de encender el motor (M = 1) solamente cuando la cisterna no esté vacía y el tinaco no esté lleno.

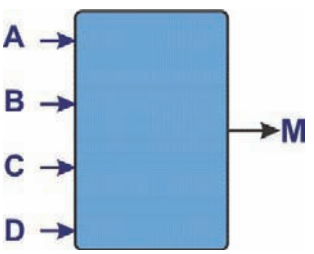
Procedimiento

1. Especifique el sistema. Las variables que intervienen son:

Sensores A, B, C y D { agua = **1**; sin agua = **0**

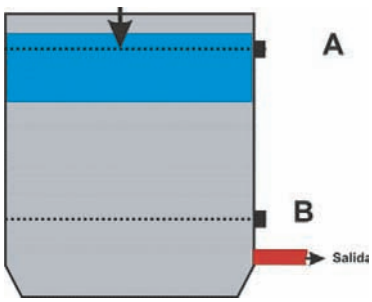
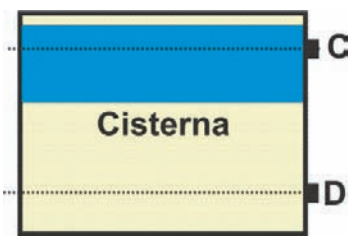
Motor M { encendido = **1**; apagado = **0**

2. Determine entradas y salidas; se puede decir que los sensores **A**, **B**, **C**, **D** son las entradas del sistema, mientras que el motor **M** es la salida que representamos a continuación en un diagrama de bloques.



3. Para trasladar el comportamiento a una tabla de verdad, decida la salida para cada una de las 16 posibles combinaciones de entrada, desde m = 0 hasta m = 15, con la siguiente lógica: el sistema deberá de encender el motor (M = 1) sólo cuando la cisterna no esté vacía y el tinaco no esté lleno.

Observe que no es posible que se presenten las combinaciones descritas en la tabla de abajo, ya que no resulta viable que sólo se detecte agua en el nivel superior sin tener agua en la parte de abajo, como se indica en la figura.

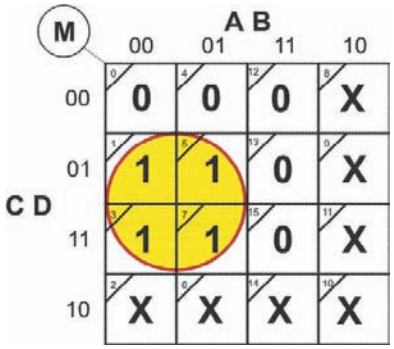
Para todos los casos donde **A = 1 y B = 0,** **1 0 X X** **combinaciones** **8, 9, 10 y 11 de la tabla de verdad.**	
También para los casos donde se presente **C = 1 y D = 0,** **X X 10** **combinaciones** **2, 6, 10 y 14.**	

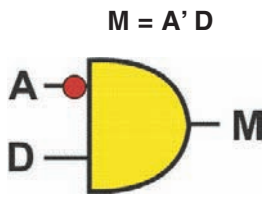
Si no es posible que se presente, es conveniente que en la tabla de verdad se le asigne el valor de X.

Tabla de verdad

m	A B C D	M	comentario
0	0 0 0 0	0	La cisterna está vacía.
1	0 0 0 1	1	Tinaco vacío, cisterna no vacía.
2	0 0 1 0	X	No es posible que se presente.
3	0 0 1 1	1	Tinaco vacío, cisterna llena.
4	0 1 0 0	0	La cisterna está vacía.
5	0 1 0 1	1	Tinaco no lleno, cisterna no vacía.
6	0 1 1 0	X	No es posible que se presente.
7	0 1 1 1	1	Tinaco no lleno, cisterna llena.
8	1 0 0 0	X	No es posible que se presente.
9	1 0 0 1	X	No es posible que se presente.
10	1 0 1 0	X	No es posible que se presente.
11	1 0 1 1	X	No es posible que se presente.
12	1 1 0 0	0	La cisterna está vacía.
13	1 1 0 1	0	Tinaco lleno.
14	1 1 1 0	X	No es posible que se presente.
15	1 1 1 1	0	Tinaco lleno.

4. Obtenga la ecuación simplificada usando un mapa de Karnaugh y el diagrama esquemático.





A continuación se presentan tres formas de archivo en formato ABEL-HDL:

a) Tabla de verdad, incluyendo la instrucción **DC don't care**.

b) Tabla de verdad con la instrucción **@DCSET**.

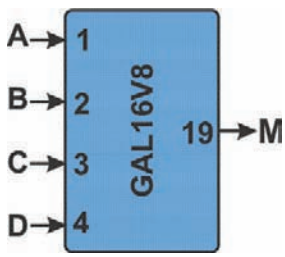
c) Mediante la ecuación obtenida en el mapa de Karnaugh.

Archivo en formato ABEL-HDL listando la tabla de verdad e incluyendo la instrucción DC (don't care)

Para obtener ventajas de las combinaciones que no se presentan (**Can't happen**) o las salidas donde no importa el valor (**Don't care**), en el lenguaje ABEL-HDL es necesario incluir la instrucción **DC** (**Don't care**) en la línea de las declaraciones de salida: **M PIN 19 istype 'dc,com';** si alguna combinación de la tabla de verdad no se incluye, ésta se tomará como **X;** y si se listara, es necesario sustituir el valor de salida por **.X.**

En el archivo en formato ABEL-HDL para la solución en un GAL16V8 utilizando la tabla de verdad, no es necesario minimizar la función, ya que el programa incluye una rutina de simplificación.

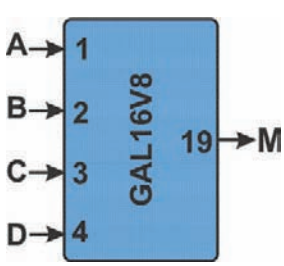
```
MODULE tinaco
“Entradas
        A,B,C,D PIN 1,2,3,4;
“Salida
        M pin 19 istype ‘dc-
,com’;
Truth_Table
([A,B,C,D]->[M])
[0,0,0,0]->[0];
[0,0,0,1]->[1];
[0,0,1,0]->[.x.];
[0,0,1,1]->[1];
[0,1,0,0]->[0];
[0,1,0,1]->[1];
[0,1,1,0]->[.x.];
[0,1,1,1]->[1];
[1,0,0,0]->[.x.];
[1,0,0,1]->[.x.];
[1,0,1,0]->[.x.];
[1,0,1,1]->[.x.];
[1,1,0,0]->[0];
[1,1,0,1]->[0];
[1,1,1,0]->[.x.];
[1,1,1,1]->[0];
End
```



Archivo en formato ABEL-HDL listando la tabla de verdad e incluyendo la instrucción @DCSET

Es posible usar la instrucción **@DCSET** en vez de **DC** y obtener el mismo resultado, como lo muestra el siguiente archivo:

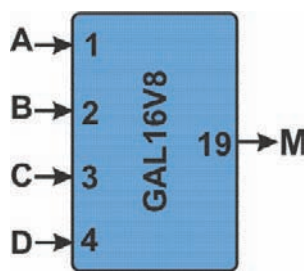
```
MODULE tinaco
 @DCSET
“Entradas
        A,B,C,D PIN
1,2,3,4;
“Salida
        M pin 19 istype ‘com’;
```



```
Truth_Table
([A,B,C,D]->[M])
[0,0,0,0]->[0];
[0,0,0,1]->[1];
[0,0,1,0]->[.x.];
[0,0,1,1]->[1];
[0,1,0,0]->[0];
[0,1,0,1]->[1];
[0,1,1,0]->[.x.];
[0,1,1,1]->[1];
[1,0,0,0]->[.x.];
[1,0,0,1]->[.x.];
[1,0,1,0]->[.x.];
[1,0,1,1]->[.x.];
[1,1,0,0]->[0];
[1,1,0,1]->[0];
[1,1,1,0]->[.x.];
[1,1,1,1]->[0];
End
```

Archivo en formato ABEL-HDL mediante la ecuación obtenida

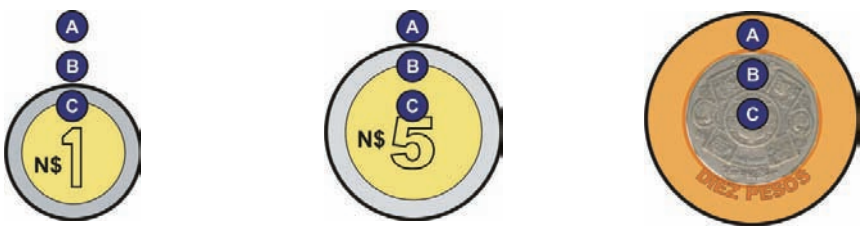
```
MODULE tinaco
“Entradas
        A,B,C,D PIN 1,2,3,4;
“Salida
        M pin 19 istype ‘dc,com’;
Equations
M=!A&D;
End
```



Ejemplo 7.2

Detectores de monedas

Se desea detectar qué tipos de monedas se insertan en una máquina expendedora. Las monedas que se aceptan son de $1 (UP), $5 (CP) y $10 (DP). Para ello se colocan tres fotoceldas a distancia conveniente, de manera que la moneda de $1 sólo cubra la fotocelda **C;** la moneda $5 sólo las fotoceldas **B** y **C**; y la moneda de $10 sólo las tres fotoceldas **A**, **B** y **C**. Observe la siguiente figura:



El sistema consta de tres entradas **A**, **B** y **C**, donde toman el valor de 1 cuando hay moneda presente y de 0 cuando no hay moneda en esa fotocelda.

Es conveniente incluir una cuarta salida **M** que tome el valor de 1 cuando ocurra una combinación de entrada no prevista. Cuando la moneda es la indicada, la salida tomará un valor de 1.

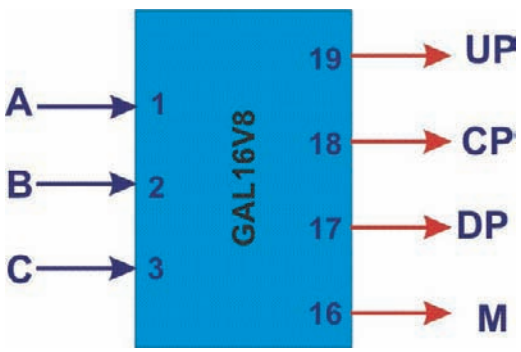


Diagrama de bloques

Tabla de verdad

m	A B C	UP	CP	DP	M
0	0 0 0	0	0	0	0
1	0 0 1	1	0	0	0
2	0 1 0	X	X	X	1
3	0 1 1	0	1	0	0
4	1 0 0	X	X	X	1
5	1 0 1	X	X	X	1
6	1 1 0	X	X	X	1
7	1 1 1	0	0	1	0

En condiciones normales de funcionamiento sólo es posible que se presenten cuatro combinaciones:

- La combinación m = 0 (0, 0, 0) donde no hay moneda presente.
- La combinación m = 1 (0, 0, 1) donde sólo se tapa la fotocelda **C** que corresponde a la moneda de **$1** (UP).
- La combinación m = 3 (0, 1, 1) donde está la moneda de **$5** (CP) y se cubren solamente las fotoceldas **B** y **C**.
- La combinación m = 7 correspondiente a una moneda de **$10** (DP) y se cubren las fotoceldas **A**, **B** y **C**.

No es posible que las combinaciones 2, 4, 5 y 6 se presenten en condiciones normales; aunque si esto ocurriera sólo sería en condiciones de mantenimiento **M**. Se aprovecha el hecho de que no se presenten (**Can't happen**) para asignarle un valor de **X** a la salida donde cada valor de **X** puede tomar el valor de 0 o 1, según convenga, como se expuso anteriormente.

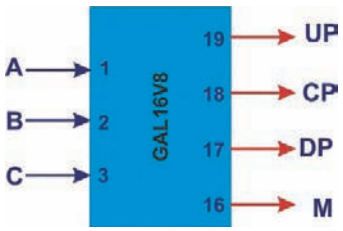
En el lenguaje ABEL-HDL para obtener ventaja de las combinaciones que no se presentan (**Can't happen**) o las salidas en que no importa el valor (**Don't care**), es necesario incluir el comando **DC** (**Don't care**) en la línea de las declaraciones de salida *UP,CP,DP,M PIN 19..16 istype **'dc,com'**;* si alguna combinación de la tabla de verdad no se incluye, ésta se tomará como X, y si se lista, es necesario sustituir el valor de salida por .x.

También es posible simplificar la entrada de tabla definiendo un **SET.** Por ejemplo:

E=[A, B, C];

Archivo en formato ABEL-HDL que incluye la simulación

```
MODULE monedas
“Detector de monedas
“Simplificación de variables
x=.x.;
“Entradas
      A,B,C pin 1,2,3;
“Salidas
      UP,CP,DP,M pin 19..16 istype ‘dc,com’;
“SET
E=[A,B,C];
truth_table
(E->[UP,CP,DP,M])
 0->[0, 0, 0,0];
 2->[x,x,x,1];
 1->[1, 0, 0,0];
 3->[0, 1, 0,0];
 4->[x,x,x,1];
 5->[x,x,x,1];
 6->[x,x,x,1];
 7->[0, 0, 1,0];

TEST_VECTORS
(E->[UP,CP,DP])
 0->[x, x, x];
 1->[x, x, x];
 2->[x, x, x];
 3->[x, x, x];
 4->[x, x, x];
 5->[x, x, x];
 6->[x, x, x];
 7->[x, x, x];
END
```



Ecuaciones obtenidas:

UP = (!B & C);

Es una moneda de un peso, siempre que se tape la fotocelda **C**,
pero no se tape **B**.

CP = (!A & B);

Es una moneda de cinco pesos cuando se tape la fotocelda B,
pero no se tape A; se asume que C se tiene que tapar.

DP = (A);

Siempre que se tape la fotocelda **A** es una moneda de diez pesos, se asume que **B** y **C** se tienen que tapar

M = (B & !C # A & !B);

Ejemplo 7.3

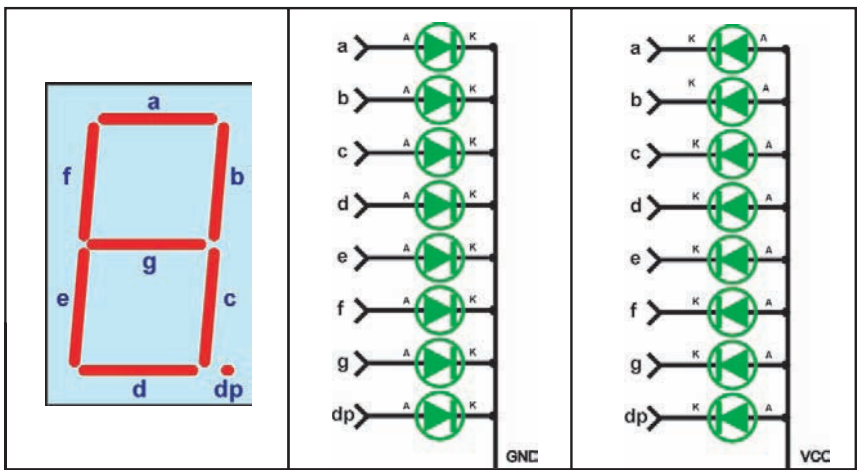
Decodificador de BCD a siete segmentos

Definiciones

***Decodificador*:** Proceso que permite pasar de un lenguaje codificado a otro legible directamente.

BCD: Código decimal expresado en binario, cada dígito del decimal se representa por cuatro bits. Ejemplo: $4678_{(10)}$ 0100 0110 0111 1000 $_{(BCD).}$

7 Segmentos: Se refiere a un *display* (dispositivo para mostrar resultados) compuesto por LED (diodos emisores de luz) distribuidos de tal suerte que se puedan mostrar los dígitos del 0 al 9.



Display	Cátodo común	Ánodo común

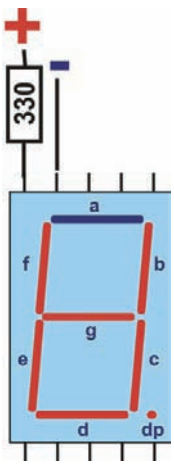
Procedimiento

1. Identifique las terminales del *display*.
2. Elabore la tabla de verdad para el decodificador.
3. Desarrolle el archivo en ABEL-HDL usando el comando TRUTH_TABLE.
4. Compile y programe el GAL16V8D.
5. Arme el circuito y compruebe su funcionamiento.

Identificación de la distribución de terminales mediante una fuente de VCD y una resistencia de 330Ω

1. Identificación del punto común

En una de las terminales del *display* conecte el positivo de la fuente a través de una resistencia de 330Ω. Con el negativo de la fuente pruebe cada una de las terminales hasta que encienda algún segmento o punto decimal. La terminal negativa donde encendió indica el *punto común*; se trata de un *display* de cátodo común.

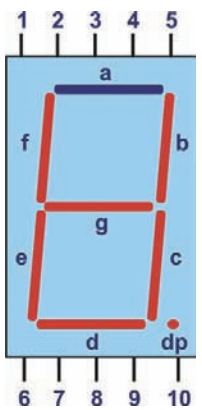


En caso de que no encienda ningún segmento o punto decimal, invierta la polaridad de la fuente, pruebe de nuevo cada una de las terminales hasta que encienda algún segmento o punto decimal. La terminal positiva en donde encendió indica el *punto común,* de esta manera sabremos que se trata de un *display* de ánodo común.

2. Identificación de los segmentos

Una vez identificado el punto común, conecte éste a la fuente (negativo para cátodo común, positivo para ánodo común) a través de la resistencia de 330Ω, y con la otra terminal identifique cada segmento.

Terminal	Segmento
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	



3. Tabla de verdad para su *display*

	BCD	7 segmentos						
m	A B C D	a	b	c	d	e	f	g
0	0 0 0 0							
1	0 0 0 1							
2	0 0 1 0							
3	0 0 1 1							
4	0 1 0 0							
5	0 1 0 1							
6	0 1 1 0							
7	0 1 1 1							
8	1 0 0 0							
9	1 0 0 1							
10	1 0 1 0	X	X	X	X	X	X	X
11	1 0 1 1	X	X	X	X	X	X	X
12	1 1 0 0	X	X	X	X	X	X	X
13	1 1 0 1	X	X	X	X	X	X	X
14	1 1 1 0	X	X	X	X	X	X	X
15	1 1 1 1	X	X	X	X	X	X	X

NOTA: Las combinaciones del 10 al 15 no pertenecen al código **BCD** y se les asigna el valor de **X**, de manera que toman el valor que más convenga para la minimización. Al definir las variables de salida, se debe incluir el comando **DC** para que las combinaciones que no aparezcan en la tabla las tome como **Don't care**. Por ejemplo:

"SALIDAS

a,b,c,d,e,f,g pin 19..13 istype `com,**dc**`;

Observe que las terminales de salida se indican como 19..13 mediante dos puntos seguidos que indican 19,18,17,16,15,14,13.

Para simplificar las entradas **A**, **B**, **C**, **D** de la tabla de verdad y asignarlas en decimal, en lugar de binario, se define una variable **X** igual a **A**, **B**, **C**, **D**.

X = [A, B, C, D]; Ejemplo en vez de teclear [0,1,1,0]; sólo se indicaría el valor decimal que es [6].

Archivo ABEL-HDL

```
MODULE BCD7
"ENTRADAS
    A,B,C,D PIN 1,2,3,4;
"SALIDAS
    a,b,c,d,e,f,g pin 19..13 istype `com,dc`;
X= [A,B,C,D];

"Tabla de verdad para cátodo común.
truth_table
([X]->[a,b,c,d,e,f,g])
[0]-> [1,1,1,1,1,1,0];
[1]-> [0,0,1,1,0,0,0];
[2]-> [                ];
[3]-> [                ];
[4]-> [                ];
[5]-> [                ];
[6]-> [                ];
[7]-> [                ];
[8]-> [1,1,1,1,1,1,1];
[9]-> [                ];
END
```

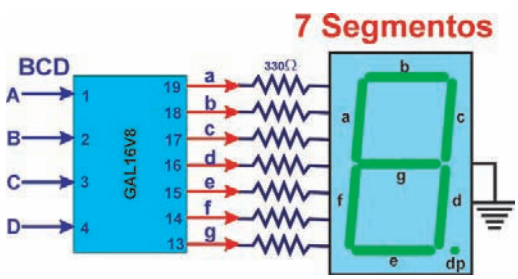
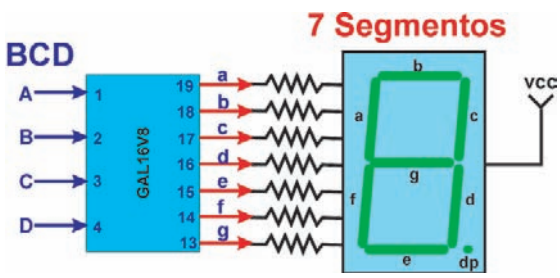


Tabla de verdad para ánodo común

```
truth_table
([X]->[a,b,c,d,e,f,g])
[0]-> [0,0,0,0,0,0,1];
[1]-> [1,1,0,0,1,1,1];
[2]-> [                ];
[3]-> [                ];
[4]-> [                ];
[5]-> [                ];
[6]-> [                ];
[7]-> [                ];
[8]-> [0,0,0,0,0,0,0];
[9]-> [                ];
```



4. Armar el circuito y comprobar su funcionamiento

Reporte

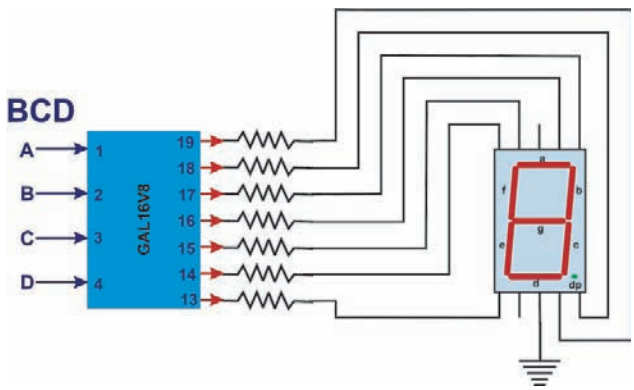
Elabore el archivo en formato ABEL-HDL para que el decodificador de **BCD** a siete segmentos indique lo siguiente en las combinaciones del 10 al 15:

a) Una E de error.

b) Permanezca apagado.

c) Las letras **a**, **b**, **c**, **d**, **e** y **f** correspondientes a las combinaciones 10, 11, 12, 13, 14 y 15, respectivamente.

d) Asigne las terminales del GAL16V8 a las salidas **a**, **b**, **c**, **d**, **e** y **f**, de manera que no existan cruces en el alambrado hacia el *display*, como se indica en la siguiente figura.



Fundamento teórico

Convertidores de códigos

La palabra *código* se define de varias maneras; por ejemplo:

- Sistema de signos y reglas que permiten formular y comprender un mensaje.
- Cifra o signo para comunicar algo de forma secreta.

Los códigos binarios numéricos facilitan la comunicación entre dispositivos. Los más usados se dividen en decimales y binarios, como se muestra en la siguiente tabla:

	Códigos decimales expresados en binario			Códigos binarios	
	BCD	Exceso 3	2421	Binario $N_{(2)}$	Gray
	A B C D	**E F G H**	**I J K L**	**R S T V**	**X Y Z W**
0	**0 0 0 0**	**0 0 1 1**	**0 0 0 0**	**0 0 0 0**	**0 0 0 0**
1	**0 0 0 1**	**0 1 0 0**	**0 0 0 1**	**0 0 0 1**	**0 0 0 1**
2	**0 0 1 0**	**0 1 0 1**	**0 0 1 0**	**0 0 1 0**	**0 0 1 1**
3	**0 0 1 1**	**0 1 1 0**	**0 0 1 1**	**0 0 1 1**	**0 0 1 0**
4	**0 1 0 0**	**0 1 1 1**	**0 1 0 0**	**0 1 0 0**	**0 1 1 0**
5	**0 1 0 1**	**1 0 0 0**	**1 0 1 1**	**0 1 0 1**	**0 1 1 1**
6	**0 1 1 0**	**1 0 0 1**	**1 1 0 0**	**0 1 1 0**	**0 1 0 1**
7	**0 1 1 1**	**1 0 1 0**	**1 1 0 1**	**0 1 1 1**	**0 1 0 0**
8	**1 0 0 0**	**1 0 1 1**	**1 1 1 0**	**1 0 0 0**	**1 1 0 0**
9	**1 0 0 1**	**1 1 0 0**	**1 1 1 1**	**1 0 0 1**	**1 1 0 1**
10				**1 0 1 0**	**1 1 1 1**
11				**1 0 1 1**	**1 1 1 0**
12				**1 1 0 0**	**1 0 1 0**
13				**1 1 0 1**	**1 0 1 1**
14				**1 1 1 0**	**1 0 0 1**
15				**1 1 1 1**	**1 0 0 0**

Por ejemplo, el número $382_{(10)}$ podría expresarse en los diferentes códigos.

BCD	**0011 1000 0010**	En donde cada dígito del decimal es expresado por un grupo de cuatro bits.
Exceso 3	**0110 1011 0101**	
2421	**0011 1110 0010**	
Binario $N_{(2)}$	**$101111110_{(2)}$**	El binario se obtiene de la conversión partiendo del decimal. Para el **Gray** se parte del binario.
Gray	**$111000001_{(Gray)}$**	

Objetivo particular

Durante el desarrollo de esta práctica se diseñará un sistema combinacional capaz de convertir de un código a:

a) EX3 a un código BCD.	*b*) EX3 a un código 2421.
c) 2421a un código EX3.	*d*) 2421 a un código BCD.
e) BCD a un código EX3.	*f*) BCD a un código 2421.
g) Binario a un código Gray.	*h*) Gray a un código binario.

Solicite a su instructor que le asigne uno de los problemas.

Ejemplo 7.4

Convierta de un código EX3 a un código 2421 T5.

	Código de entrada	Código de salida			
m	E F G H	I	J	K	L
0	0 0 0 0	X	X	X	X
1	0 0 0 1	X	X	X	X
2	0 0 1 0	X	X	X	X
3	0 0 1 1	0	0	0	0
4	0 1 0 0	0	0	0	1
5	0 1 0 1	0	0	1	0
6	0 1 1 0	0	0	1	1
7	0 1 1 1	0	1	0	0
8	1 0 0 0	1	0	1	1
9	1 0 0 1	1	1	0	0
10	1 0 1 0	1	1	0	1
11	1 0 1 1	1	1	1	0
12	1 1 0 0	1	1	1	1
13	1 1 0 1	X	X	X	X
14	1 1 1 0	X	X	X	X
15	1 1 1 1	X	X	X	X

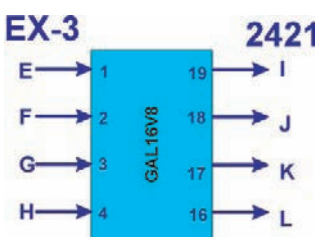
Procedimiento

1. Seleccione uno de los convertidores de código arriba mencionados.
2. Elabore la tabla de verdad.
3. Desarrolle el archivo en ABEL-HDL utilizando el comando TRUTH_TABLE, y use una variable para representar entrada y salida en su valor decimal.
4. Compile y programe el GAL16V8D.

Para simplificar el archivo en formato ABEL-HDL, se utilizará la instrucción **DC** (**Don't care**) para tomar como **X** aquellas combinaciones de entrada que no se listen en el archivo. Además, se simplificará la tabla de verdad al usar los **SET x = [E,F,G,H]; y = [I,J,K,L];**

Archivo en formato ABEL-HDL

```
MODULE EXtres
E,F,G,H pin 1,2,3,4;
I,J,K,L pin 19,18,17,16 istype 'dc, com';
x= [E,F,G,H];
y= [I,J,K,L];
truth_table
     ([x] ->[y])
        [3] -> [0];
        [4] -> [1];
        [5] -> [2];
        [6] -> [3];
        [7] -> [4];
        [8] -> [11];
        [9] -> [12];
        [10]-> [13];
        [11]-> [14];
        [12] -> [15];
END
```



Su instructor le asignara uno de los ejercicios de los incisos descritos anteriormente. Elabore la tabla de verdad:

	Código de entrada	Código de salida			
M					
0	**0 0 0 0**				
1	**0 0 0 1**				
2	**0 0 1 0**				
3	**0 0 1 1**				
4	**0 1 0 0**				
5	**0 1 0 1**				
6	**0 1 1 0**				
7	**0 1 1 1**				
8	**1 0 0 0**				
9	**1 0 0 1**				
10	**1 0 1 0**				
11	**1 0 1 1**				
12	**1 1 0 0**				
13	**1 1 0 1**				
14	**1 1 1 0**				
15	**1 1 1 1**				

PRÁCTICA 8

Flip Flops

Objetivos particulares

Durante el desarrollo de esta práctica se analizará el comportamiento del **FF "RS"** (*Reset-Set*), partiendo de un circuito con relevadores y contactos de arranque y paro, incluyendo su implementación en un dispositivo lógico programable GAL16V8 usando ecuaciones en un archivo de formato ABEL-HDL. Además se realizará el análisis y la implementación del **FF "SC"** (*Set-Clear*) en su aplicación como eliminador de rebotes, así como las tablas de los Flip Flops **T** y **D.**

El tiempo estimado para el estudio de esta práctica es de dos horas.

Material necesario para el desarrollo de esta práctica

- Una fuente de voltaje de 5VCD.
- Una tablilla de conexiones (*protoboard*).
- Un GAL16V8D (*Lattice semiconductor*).
- Un DIP de ocho entradas o dos *push button*.
- Cuatro LED sin importar el color.
- Cuatro resistencias de 330 OHMS.
- Alambre para conexiones.
- Un disco de 3.5 pulgadas de alta densidad formateado a 1.44 MB.
- Una resistencia de 4.7 KΩ.
- Un capacitor de 0.1 μF.
- Un capacitor de 10 μF.

Fundamento teórico

Flip Flop es un elemento de memoria con capacidad para almacenar un solo bit. Los tipos estándares de Flip Flops son:

RS *Reset-Set*

JK No está definido el nombre. Se originó en la Huges Aircraft Company durante la década de 1950.

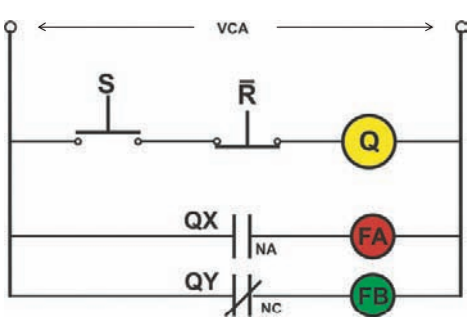
T *Toggle* o cambio de estado.

D *Delay* (retardo); *Data* (dato).

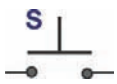
SC *Set-Clear*.

Circuito, arranque y paro de Flip Flop RS (*Reset-Set*)

El circuito descrito cuenta con dos botones llamados **S** y **R**, además de un relevador que contiene una bobina **Q** y dos contactos llamados **QX** y **QY**, con los cuales se encienden dos focos llamados **FA** y **FB**.



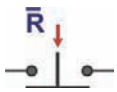
El botón **S** es de no retención y se llama normalmente abierto (NO). Cuando está en condiciones normales se encuentra abierto.



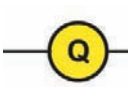
Al oprimirlo se cierra y al soltarlo se vuelve a abrir.

El botón **R'** de no retención se llama normalmente cerrado (**R'**), es decir, en condiciones normales está cerrado.

Si se oprime el botón **R'** se abren sus contactos, y al soltarlo se vuelven a cerrar. Se considera negado por tener una acción contraria al botón **S**.

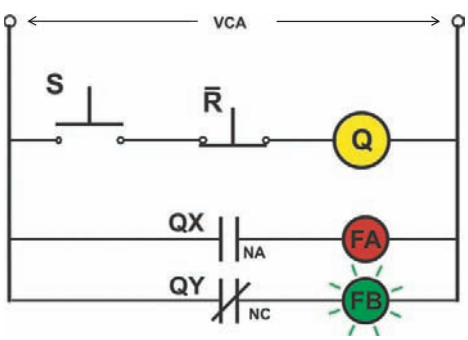


El relevador contiene una bobina **Q**, que al recibir voltaje en sus terminales hace que sus contactos cambien su posición de abierto a cerrado, o viceversa.

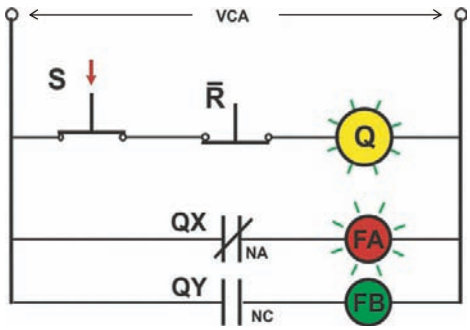


Dos contactos: uno abierto **QX** (NA) y otro cerrado **QY** (NC).

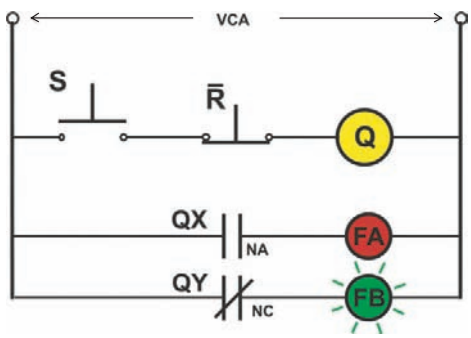
En los contactos **QX** y **QY** se conectan en serie dos focos llamados **FA** y **FB**.



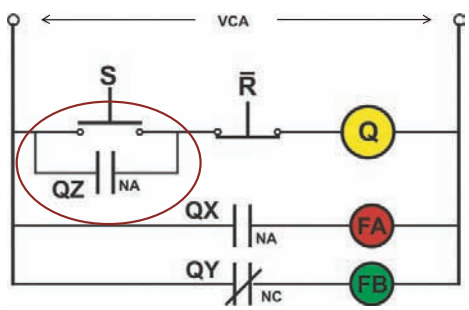
Si se oprime el botón **S** se energetiza la bobina **Q** del relevador, se cierra el contacto **QX**, enciende el foco **FA** y se abre el contacto **QY** apagando el foco **FB**.



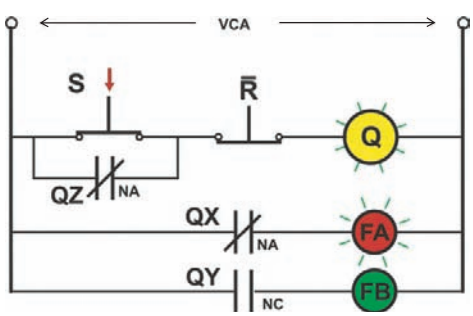
Al soltar el botón **S** se desenergetiza la bobina **Q**, y los contactos regresan a la condición inicial (apagando el foco A y manteniendo encendido el foco B).



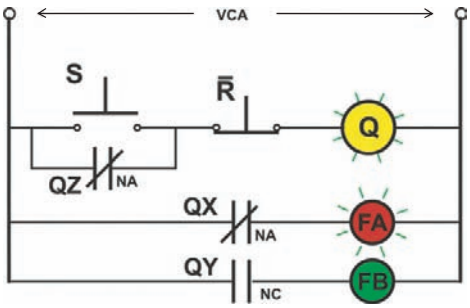
Si se agrega un contacto **QZ**, normalmente abierto en paralelo con el botón **S**, como lo indica la figura, se obtiene una condición de memoria.



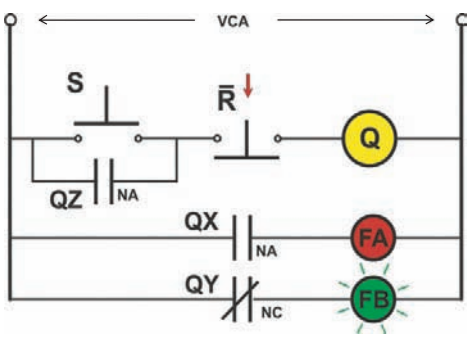
De modo que si se oprime de nuevo el botón **S** se energetiza la bobina **Q** del relevador, se cierra el contacto **QX** encendiendo **FA**, se abre el contacto **QY** apagando **FB** y el contacto **QZ** se cierra puenteando el botón **S**.



De modo que al soltar el botón **S** se queda energetizada la bobina **Q** del relevador a través del contacto **QZ**. Se puede considerar una memoria desde que se oprimió el botón **S** (arranque).



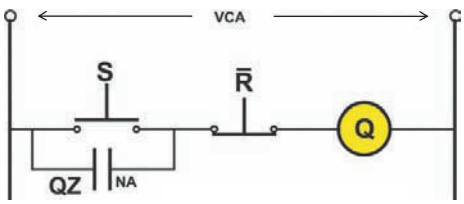
Al oprimir el botón **R** se desenergetiza la bobina del relevador y el sistema vuelve a sus condiciones iniciales o paro.



Si se oprimen por error los dos botones al mismo tiempo, el sistema no arrancará.

Después de oprimir erróneamente los dos botones, al soltarlos no es posible asegurar el estado que resultará, ya sea de encendido o apagado.

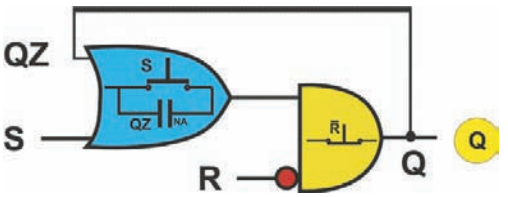
De manera que tenemos un circuito con memoria de arranque usando el botón **S,** y de paro con el botón **R**.



Para obtener el circuito equivalente con compuertas, se considera que el botón **S** y el contacto **QZ** están en paralelo, por lo que se tiene la operación **OR (S + QZ)** y, a la vez, están en serie con el botón **R'**, lo cual implica una operación **AND**. A continuación se obtiene la ecuación y el circuito equivalente con compuertas lógicas.

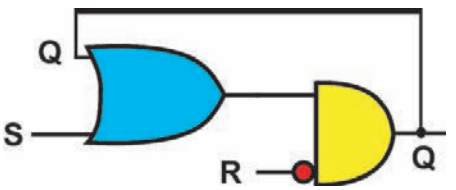
La compuerta **OR** es el paralelo del botón **S** y el contacto **QZ**. Éstos a la vez están en serie con el botón **R'** (operación **AND**).

Q = R' (S + QZ).

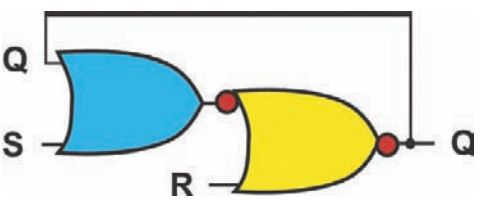


Si consideramos que **QZ** es un contacto de **Q** (**QZ = Q**), el circuito y la ecuación resultante serían:

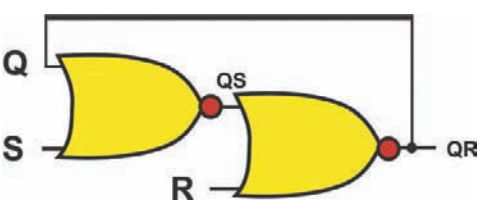
Q = R' (S + Q).



Si reemplazamos la compuerta **And** por **Nor** con las **entradas negadas** (teorema de De Morgan), el circuito queda de la siguiente forma:



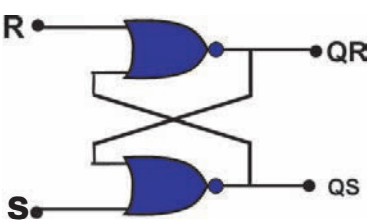
Podemos obtener dos compuertas **NOR**, cuyas salidas se llamarán **QS** y **QR**, que corresponden a cada entrada, como lo muestra la figura.



Cambiando la distribución del circuito se obtiene el circuito equivalente de un Flip Flop RS:

QR = (R + QS)'

QS = (S + QR)'

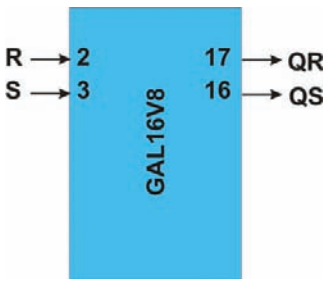


Trabajo solicitado

Implemente con ABEL-HDL el Flip Flop RS.

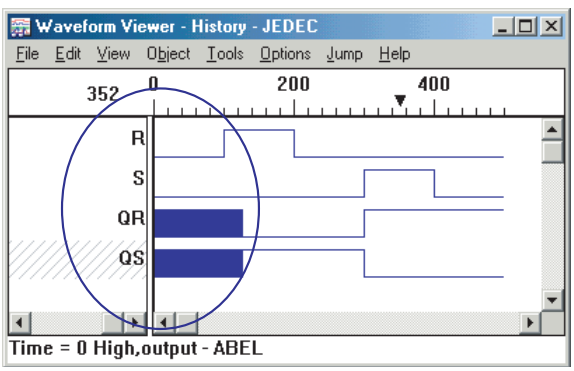
Ecuaciones en ABEL-HDL. QR = ¡(R # QS) QS = ¡(S # QR)

```
MODULE ffrs
“Entradas
   R,S pin 2,3;
“Salidas
   QR,QS pin 17,16 istype’com’;
equations
   QR = !( R # QS);
   QS = !( S # QR);
test_vectors
([R,S]->[QR,QS])
[ 0,0]->[.x.,.x.];
[ 1,0]->[.x.,.x.];
[ 0,0]->[.x.,.x.];
[ 0,1]->[.x.,.x.];
[ 0,0]->[.x.,.x.];
END
```



Archivo en formato ABEL_HDL que incluye la simulación (test_vectors)	Diagrama de bloques

Al efectuar la simulación se observa que cuando se inicia con los valores de **R** = 0 y **S** = 0, si no se conocen los valores de **QR** y **QS**, la salida es incierta; es decir, podrían tomar el valor de 0 o 1, como se indica en la figura.



Implemente el circuito en la tablilla de conexiones y obtenga la tabla de estados, siguiendo el orden de la secuencia de valores descrita en la parte inferior izquierda.

Secuencia de valores.

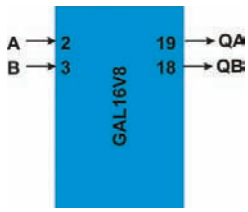
sec	R	S
1	1	0
2	0	0
3	0	1
4	0	0

Tabla característica del FF **RS**.

R S	QR	QS
0 0		
0 1		
1 0		
1 1		

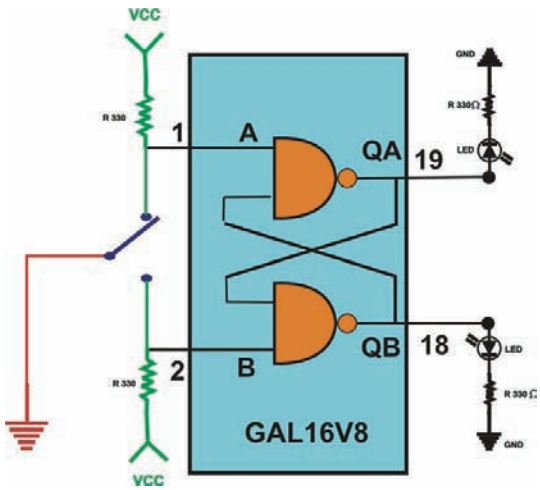
Procedimiento

Eliminador de rebotes FF "SC" (Set-Clear)



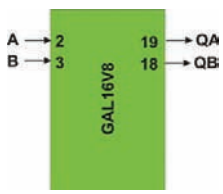
1. Genere un archivo en formato ABEL-HDL para las ecuaciones de **QA** y **QB**, asignando las terminales 2 y 3 a las entradas A y B, respectivamente, y las terminales 19 y 18 a las salidas **QA** y **QB**.

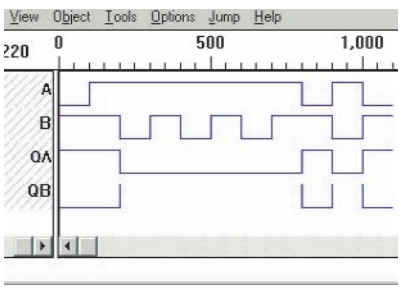
2. Compile el archivo, implemente el circuito agregando los elementos descritos en la figura abajo mostrada, y obtenga los valores de entrada y salida para cada uno de los 11 tiempos marcados en el diagrama de tiempos de la página siguiente.

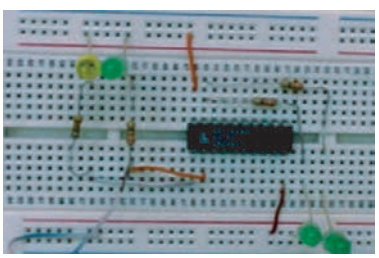


Archivo en formato ABEL_HDL.

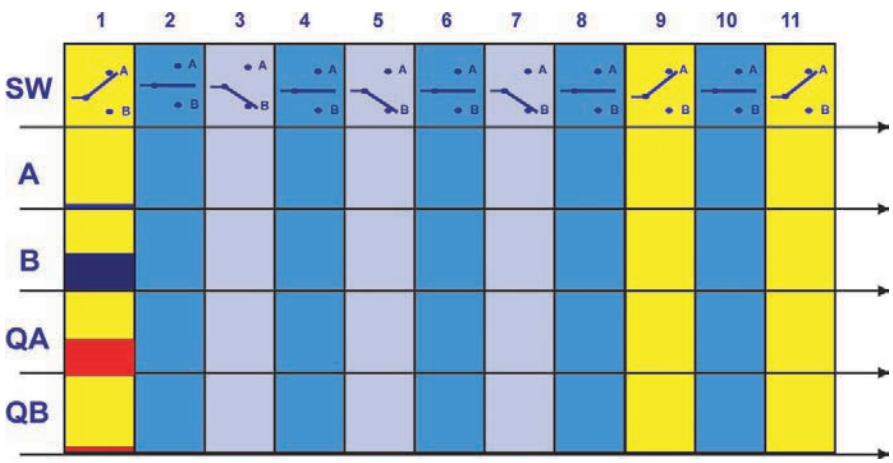
```
MODULE ffsc
“Entradas
      A,B pin 2,3;
 “Salidas
     QA,QB pin 19,18 istype’com’;
 equations
    QA = !( A & QB);
    QB = !( B & QA);
test_vectors
([A,B]->[QA,QB])
[ 0,1]->[.x.,.x.];
[ 1,1]->[.x.,.x.];
[ 1,0]->[.x.,.x.];
[ 1,1]->[.x.,.x.];
[ 1,0]->[.x.,.x.];
[ 1,1]->[.x.,.x.];
[ 1,0]->[.x.,.x.];
[ 1,1]->[.x.,.x.];
[ 0,1]->[.x.,.x.];
[ 1,1]->[.x.,.x.];
[ 0,1]->[.x.,.x.];
END
```







3. Implemente el circuito en la tablilla de conexiones y obtenga los valores de la gráfica inferior, siguiendo la posición del **SW** en los tiempos del 1 al 11.



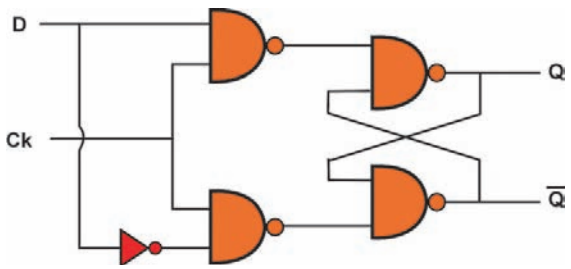
4. Obtenga la tabla característica del **Flip Flop "SC"**. Una vez implementado el circuito, calcule los valores de **QA** y **QB** para cada una de las combinaciones de las entradas **A** y **B**.

sec	A	B
1	1	0
2	1	1
3	0	1
4	1	1

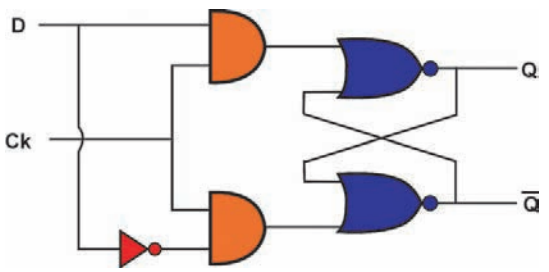
A B	QA	QB
0 0		
0 1		
1 0		
1 1		

Flip Flop D

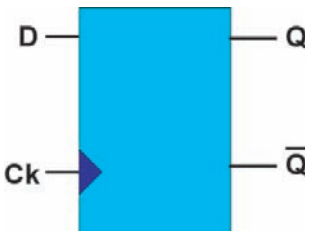
Flip Flop D circuito a partir de compuertas NAND'S



Flip Flop D circuito a partir de compuertas **NOR** y **AND**

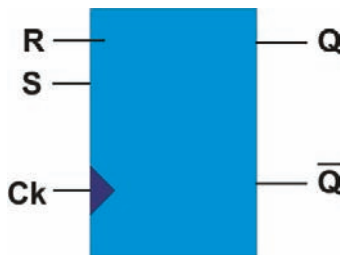


Símbolo y tabla característica del Flip Flop D con **Ck** (señal de sincronía)



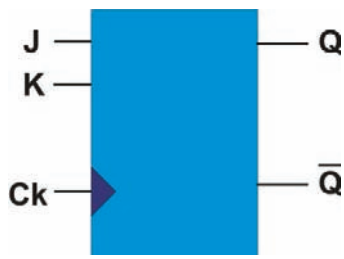
Ck	D	Qn + 1
↑	L	L
↑	H	H
L	X	Qn

Símbolo y tabla característica del Flip Flop RS con **Ck**



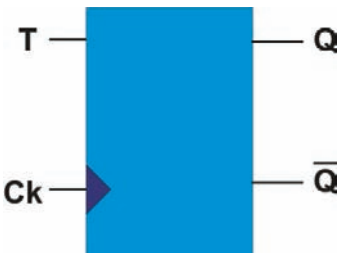
Ck	R	S	Qn+1
↑	L	L	Qn
↑	L	H	H
↑	H	L	L
↑	H	H	**H
L	X	X	Qn

Símbolo y tabla característica del Flip Flop JK con **Ck**



Ck	J	K	Qn+1
↑	L	L	Qn
↑	L	H	L
↑	H	L	H
↑	H	H	(Qn)'
L	X	X	Qn

Símbolo y tabla característica del Flip Flop T con **Ck**



Ck	T	Qn + 1
↑	L	Qn
↑	H	(Qn)'
L	X	Qn

Otras entradas de control a los Flip Flops Clear y Preset

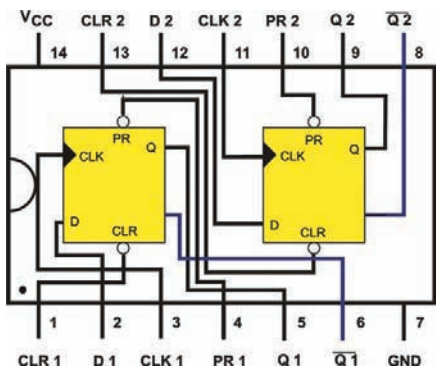
Las funciones *Clear* (Clr) y *Preset* (Pr) son entradas asíncronas, es decir, no requieren del pulso de reloj **Ck** para ser activadas.

La función *Clear* tiene el propósito de hacer que la salida **Q** tenga el valor de cero (Q = 0), mientras que *Preset* establece que la salida **Q** adquiera el valor de uno (Q = 1).

A continuación se muestran el símbolo y la tabla característica del circuito integrado SN7474 que corresponde a dos Flip Flop D con reloj de transición positiva ↑ e incluyen las entradas *Clear* y *Preset*.

Entradas				Salidas	
PR	**CLR**	**Ck**	**D**	**Qn**	**(Qn)'**
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	***E**	***E**
H	H	↑	H	H	L
H	H	↑	L	L	L
H	H	L	H	Qn	(Qn)'

* **E =** Combinación no estable.

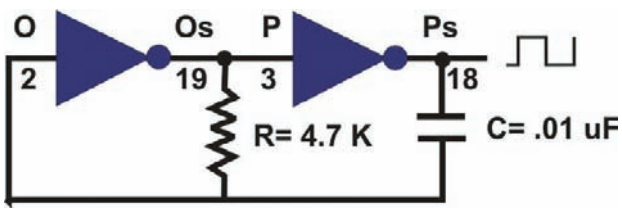


8

Generación de pulso de reloj Ck usando un dispositivo lógico programable

Es posible fabricar en un DLP y un oscilador de onda cuadrada con una frecuencia aproximada de 400 Hz, retroalimentando dos compuertas **Not** a través de un circuito **RC**.

Diagrama del oscilador



Archivo en formato ABEL-HDL para la implementación del oscilador

```
Module osci
“Entradas
O,P pin 2,3;
“Salidas
Os, Ps pin 19,18 istype ‘com’;
EQUATIONS
        Os=!O;
        Ps=!P;
  END
```

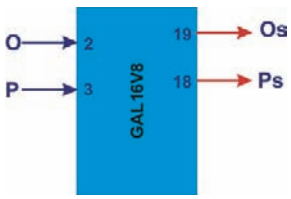
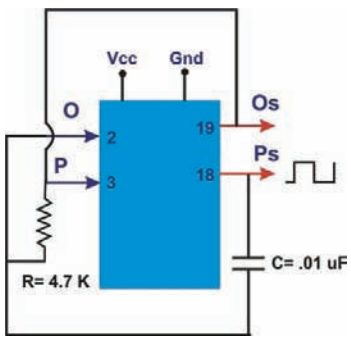


Diagrama completo del oscilador (no olvide conectar el Vcc y Gnd del dispositivo)



Si requiere de una frecuencia menor, cambie el capacitor por uno de mayor capacidad (por ejemplo 10μF, 100μF, etcétera).

PRÁCTICA 9

Diseño secuencial

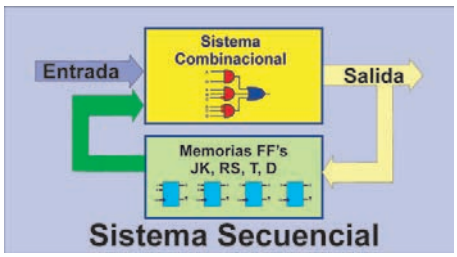
Objetivos particulares

Durante el desarrollo de esta práctica se aplicará la metodología para diseñar un sistema secuencial síncrono y se implementará por medio de captura esquemática o un lenguaje de descripción de hardware en un dispositivo lógico programable.

El tiempo estimado para el estudio de esta práctica es de dos horas.

Fundamento teórico

En el sistema secuencial síncrono los valores de salida no dependen únicamente de las combinaciones de entrada, sino también de la salida misma. Los cambios de estado están sujetos a una señal de sincronía de los Flip Flops llamada reloj o Clk.



Modelos secuenciales y sus representaciones

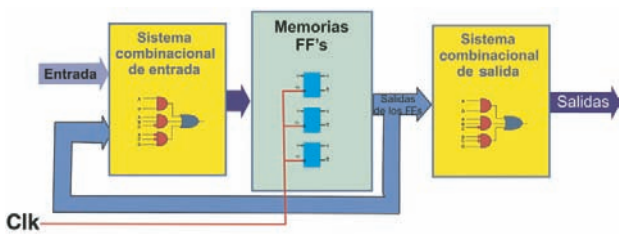
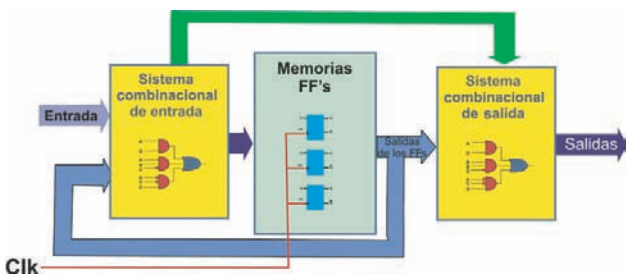
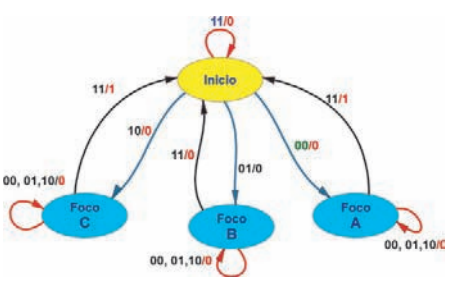
Máquina de **Moore:** En ésta la salida sólo depende del estado presente.	
Máquina de **Mealy:** Aquí la salida está sujeta tanto a la entrada como al estado presente.	

Diagrama de transición

Una forma muy explícita de especificar los eventos en un sistema secuencial es usando un diagrama de transición.



Un diagrama de transición se compone de los siguientes elementos:

Estados:

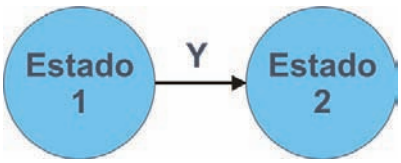
Una condición o situación en la vida de un objeto, durante la cual satisface una condición, realiza una actividad o está esperando un evento.

Transición en el mismo estado:

Una entrada X cuyo estado próximo es el mismo que el anterior.

Transición entre dos estados:

Una relación entre estados que indica que un objeto, que está en el primer estado, realizará una acción especificada, y entrará en el segundo estado cuando un evento **Y** ocurra y se satisfagan ciertas condiciones especificadas.



Entradas:

Combinaciones que establecen un cambio de evento.

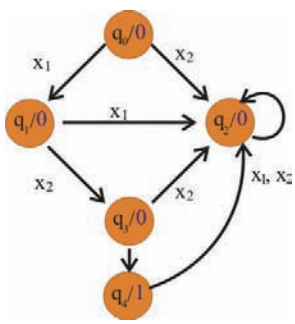
Y

Salidas:

Valores combinacionales que determinan un evento.

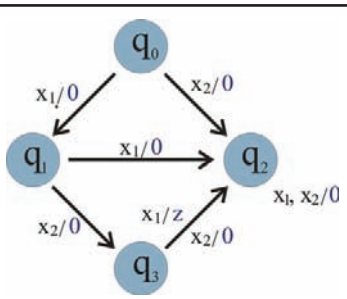
Máquina de Moore

En esta representación secuencial la salida está asociada directamente al estado.

Diagrama de transición de la máquina de Moore Entrada de pulso Salida de nivel La salida se indica en cada estado. (La salida depende sólo del valor del estado).	

Máquina de Mealy

En esta representación secuencial, la salida está relacionada con el estado y el valor de entrada, además de la transición.

Diagrama de transición de la máquina de Mealy. Entrada de pulso Salida de pulso (La salida depende del valor del estado y la entrada).	

Metodología del diseño secuencial, concluyendo con la implementación mediante captura esquemática

1. Especificar el sistema (diagrama de transición).
2. Determinar la cantidad de Flip Flops.
3. Asignar los valores a los estados.
4. Determinar las entradas y salidas.
   a) Entrada de sincronía reloj
   b) Entradas combinacionales
   c) Salidas combinacionales
   d) Salidas registradas (FF)
5. Construir una tabla de estados.
6. Minimizar.
7. Obtener diagrama esquemático.
8. Realizar la implementación.

Procedimiento

1. Especificar el sistema.

   Para especificar el comportamiento del sistema se puede emplear el diagrama de transición, donde se indican entradas, salidas y estados.

2. Determinar la cantidad de Flip Flops.

   La cantidad de Flip Flops depende del número de estados utilizados en el diagrama de transición, como lo indica la siguiente tabla:

Estados	Cantidad de Flip Flops
2	1
3 o 4	2
5 a 8	3
9 a 16	4
17 a 32	5
33 a 64	6
65 a 128	7
129 a 256	8
257 a 512	9

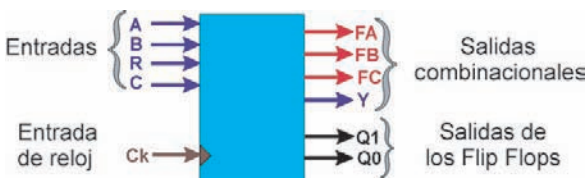
3. Asignar los valores a los estados.

La asignación de valores a los estados puede ser al azar y corresponde a las combinaciones posibles que generan las salidas **Q** de los Flip Flops.

Estados	Salidas FF
	Q1 Q0
CI	**0 0**
Foco A	**0 1**
Foco B	**1 0**
Foco C	**1 1**

4. Determinar las entradas y salidas.

En esta parte se recomienda identificar las entradas y las salidas del sistema secuencial usando un diagrama de bloques, como se indica en la siguiente figura.



5. Construir una tabla de estados.

a) Tabla de estados es usada para describir el comportamiento secuencial.

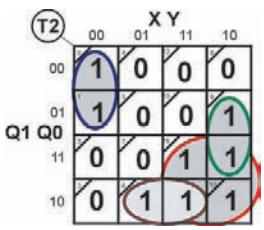
Estados presentes	**Estados próximos**		**Salida**	
	X=0	X=1	X=0	X=1

b) Tabla de estados empleando entradas de control para diseñar con un Flip Flop específico.

Entradas	**Estados presentes**	**Estados próximos**	**Entradas de control** Para Flip Flop T			**Salida**
A B	$Q_2\ Q_1\ Q_0$	$Q_{2+1}\ Q_{1+1}\ Q_{0+1}$	T_2	T_1	T_0	Y

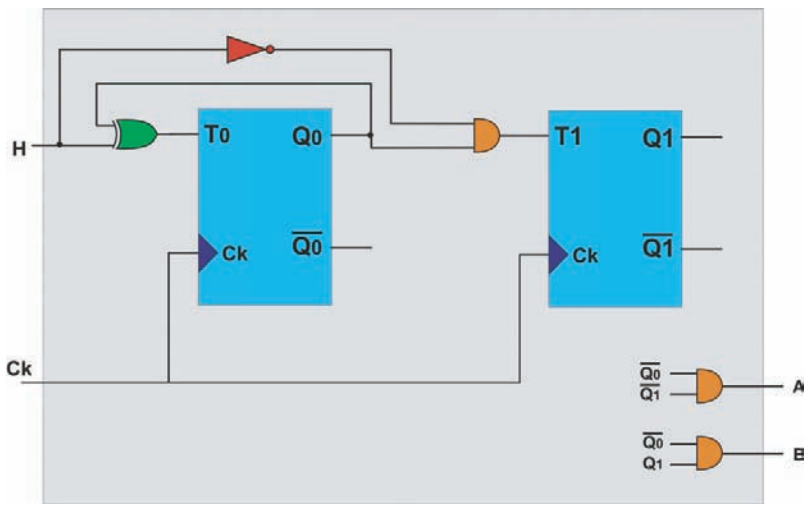
6. Minimizar.

Para obtener las ecuaciones simplificadas se utiliza la manipulación algebraica, los mapas de Karnaugh o algún programa de minimización de funciones booleanas.

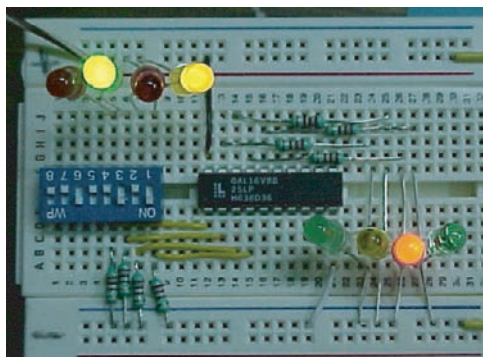



```
T2= !X&!Y&!Q1#X&!Y&Q0#Y&Q1&!Q0#X&Q1;
```

7. Obtener diagrama esquemático.



8. Realizar la implementación.



Ejemplo 9.1

Máquina de Moore

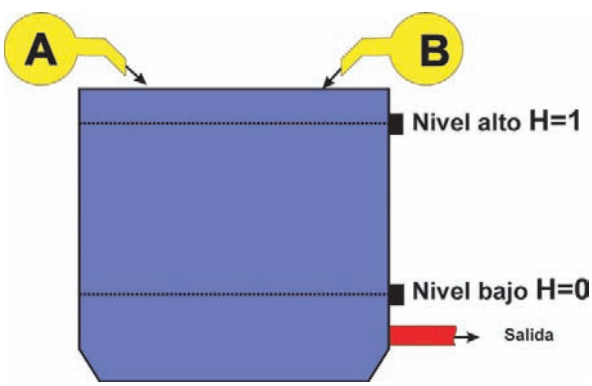
Diseñe un sistema secuencial que controle el llenado de un tanque con las siguientes características:

a) El sistema consta de dos bombas llamadas **"A"** y **"B"**.

b) Un sensor de nivel **"H"** que indica con H = 1 tanque lleno y H = 0 tanque vacío.

c) Partiendo de que el tanque se encuentra vacío (H = 0), el llenado deberá iniciarse encendiendo la bomba "A" hasta llenar el tanque (H = 1), para posteriormente apagarse.

d) Si de nuevo se vacía el tanque (H = 0), el llenado deberá hacerse encendiendo ahora la bomba "B", hasta llenar el tanque (H = 1) para que finalmente se apague. Si nuevamente se vacía el tanque, el llenado deberá hacerse con la bomba "A", y así sucesivamente, de tal forma que las bombas alternen en su funcionamiento.

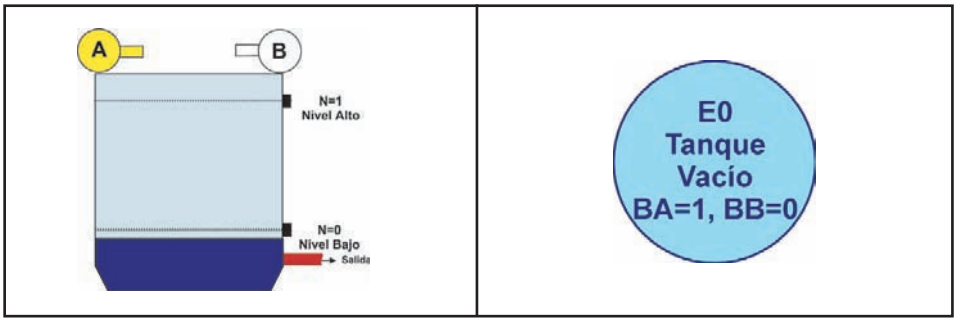


Procedimiento

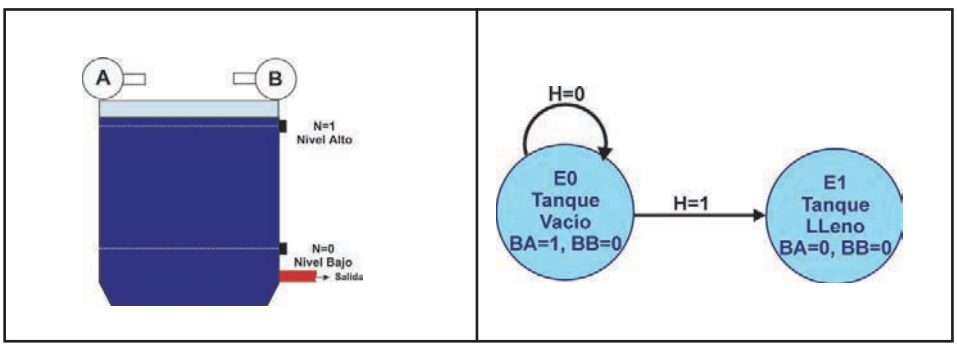
1. Especifique el sistema.

Diagrama de transición

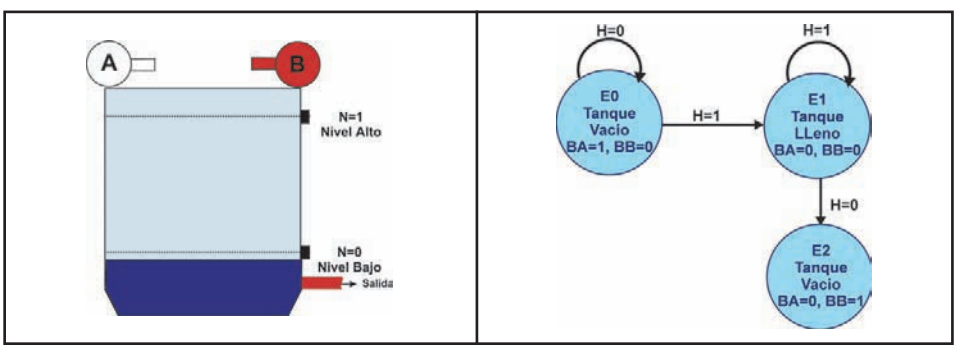
El estado **E0** tiene como finalidad definir que el tanque está vacío y en proceso de llenado; además de especificar que la bomba A está trabajando y la B está apagada.



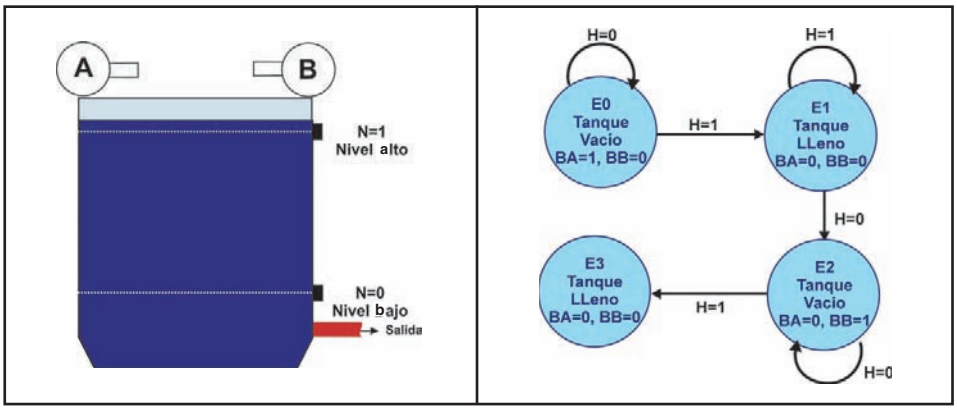
Aún en el estado **E0** quizás esté vacío (H = 0), por lo que el siguiente estado debe ser el mismo **E0**. O puede suceder que se llene (H = 1); entonces el sistema deberá ir a otro estado, **E1,** que indique tanque lleno cuya finalidad será apagar la bomba A.



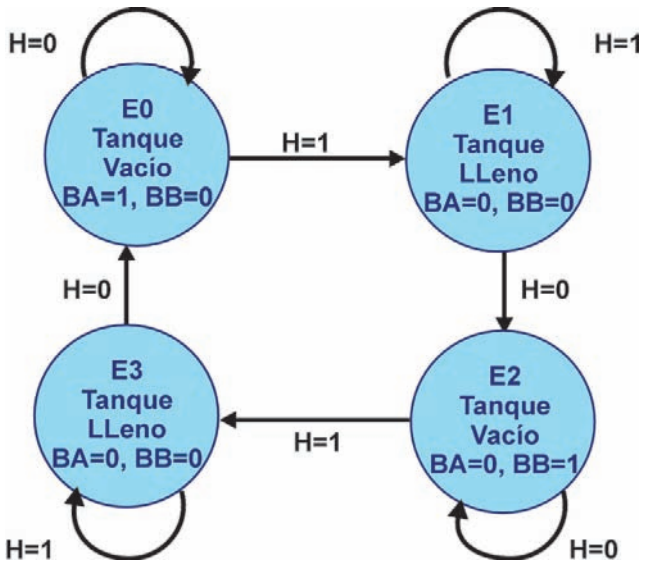
O bien, en el estado **E1** se puede presentar que aún siga lleno (H = 1); por consiguiente, el estado siguiente será el mismo **E1**. O se podría vaciar (H = 0) y debería ir a otro estado **E2** que indique que el tanque está vacío o en proceso de llenado y que está trabajando la bomba B y apagada la bomba A.



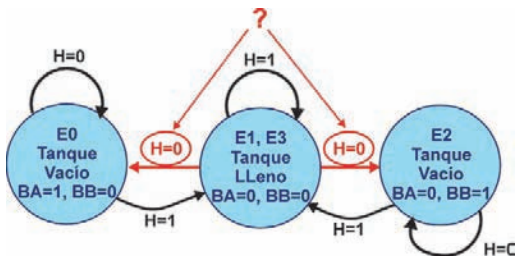
También podría ocurrir que en el estado **E2** aún esté vacío (H = 0). Entonces el siguiente estado será **E2**. O bien, que se llene (H = 1) y, por consiguiente, el sistema deberá ir a otro estado **E3** que indique tanque lleno y tendría el objetivo de apagar la bomba B.



En el estado **E3** podría ocurrir que aún siga lleno (H = 1) y el estado siguiente será el mismo **E3.** O se puede vaciar (H = 0), entonces deberá ir a otro estado que podría ser **E0** y que cerraría el ciclo del funcionamiento.



Los estados **E1** y **E3** tienen el mismo propósito, aunque la diferencia es que el estado próximo de **E1** para H = 0 es **E2**, en tanto que el estado próximo de **E3** para H = 0 es **E0**, de manera que no pueden ser equivalentes. Si **E1** y **E3** se sustituyeran por un sólo estado cuando la entrada fuera H = 0, no estaría definido claramente si el estado siguiente sería **E0** o **E2**, como lo indica la figura.

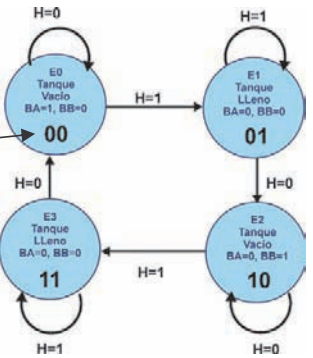


2. Determine la cantidad de Flip Flops.

Con cuatro estados es necesario utilizar dos Flip Flops.

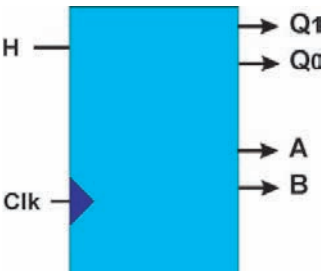
3. Asigne valores a los estados.

Estados		Salidas FF
		Q1 Q0
E0	Tanque Vacío	0 0
E1	Tanque Lleno	0 1
E2	Tanque Vacío	1 0
E3	Tanque Lleno	1 1



4. Determine las entradas y salidas.

H	Entrada del nivel.
Clk	Entrada de sincronía.
Q1, Q2	Salidas de los Flip Flops.
A y B	Salidas de las bombas.



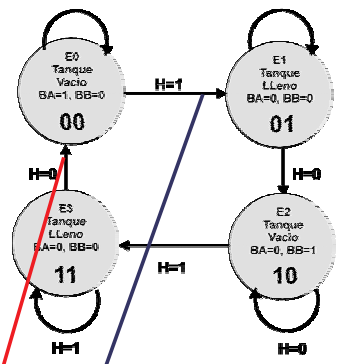
5. Construya una tabla de estados.

Estados presentes	Estados próximos	
	H=0	H=1
E0	**E0**	**E1**
E1	**E2**	**E1**
E2	**E2**	**E3**
E3	**E0**	**E3**

Tabla de estados con asignación de valores para un Flip Flop T

	Entrada	Estados presentes			Estados próximos		Entradas de control	
M	H		Q_1	Q_0	Q_{1+1}	Q_{0+1}	T1	T0
0	0	E0	0	0				
1	0	E1	0	1				
2	0	E2	1	0				
3	0	E3	1	1				
4	1	E0	0	0				
5	1	E1	0	1				
6	1	E2	1	0				
7	1	E3	1	1				

Los valores de los estados próximos (**Q1 + 1, Q0 + 1**) se obtienen a partir del diagrama de transición, donde cada hilera corresponde a una transición. Por ejemplo, en el estado **E0**, donde la asignación de valores es: **Q1 = 0** y **Q0 = 0**, se tienen dos opciones: una para **H = 0**, donde el estado próximo será el mismo **E0** y los valores **Q1 + 1 = 0** y **Q0 + 1 = 0**; y otra con **H = 1**, cuyos valores del estado próximo serán **Q1 + 1 = 0 y Q0 + 1 = 1**.



m	Entrada	Estados Presentes		Estados Próximos		Entradas de Control	
	H	Q_1	Q_0	Q_{1+1}	Q_{0+1}	T1	T0
0	0	0	0	0	0		
1	0	0	1				
2	0	1	0				
3	0	1	1				
4	1	0	0	0	1		
5	1	0	1				
6	1	1	0				
7	1	1	1				

A continuación se presenta la tabla de estados con los valores de los estados próximos:

	Entrada	Estados presentes		Estados próximos		Entradas de control	
m	H	Q_1	Q_0	Q_{1+1}	Q_{0+1}	T1	T0
0	0	0	0	0	0		
1	0	0	1	1	0		
2	0	1	0	1	0		
3	0	1	1	0	0		
4	1	0	0	0	1		
5	1	0	1	0	1		
6	1	1	0	1	1		
7	1	1	1	1	1		

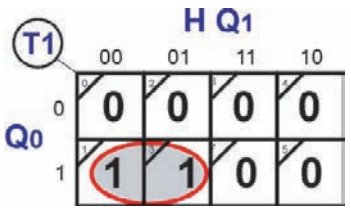
Los valores de las entradas de control (**T1, T0**) se obtienen de las tablas de excitación.

Tablas de excitación de los Flip Flops

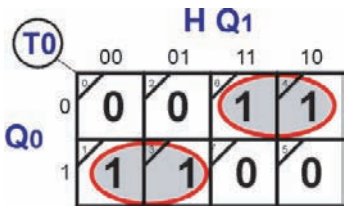
Estado presente	Estado próximo	Entradas de control					
Q_n	Q_{n+1}	R_n	S_n	J_n	K_n	T_n	D_n
0	0	X	0	0	X	0	0
0	1	0	1	1	X	1	1
1	0	1	0	X	1	1	0
1	1	0	X	X	0	0	1

	Entrada	Estados presentes		Estados próximos		Entradas de control	
m	H	Q_1	Q_0	Q_{1+1}	Q_{0+1}	T1	T0
0	0	0	0	0	0	0	0
1	0	0	1	1	0	1	1
2	0	1	0	1	0	0	0
3	0	1	1	0	0	1	1
4	1	0	0	0	1	0	1
5	1	0	1	0	1	0	0
6	1	1	0	1	1	0	1
7	1	1	1	1	1	0	0

6. Minimice.

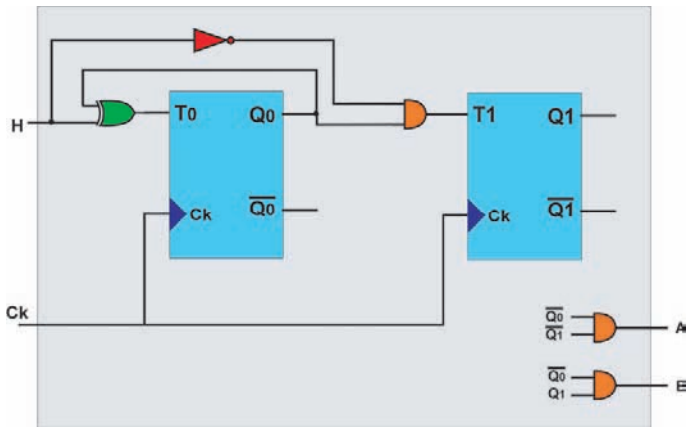


$T1 = \overline{H}\, Q0$



$T0 = H\,\overline{Q0} + \overline{H}\, Q0 = H \oplus Q0$

7. Elabore el diagrama esquemático.



Si se cambiara al Flip Flop D, la tabla de estados quedaría de esta forma:

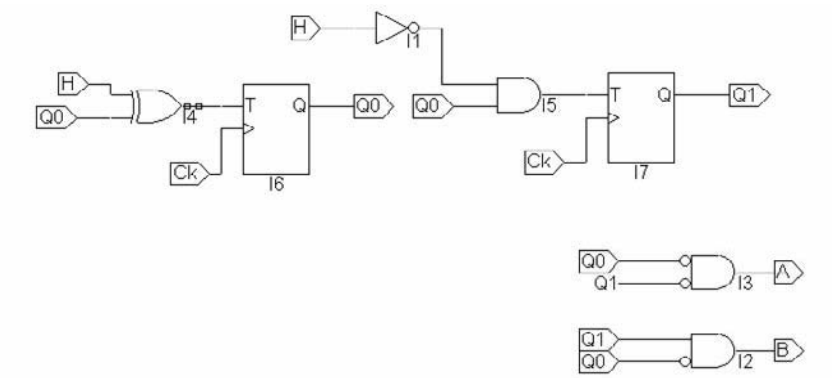
	Entrada	Estados presentes		Estados próximos		Entradas de control	
m	H	Q_1	Q_0	Q_{1+1}	Q_{0+1}	D1	D0
0	0	0	0	0	0	0	0
1	0	0	1	1	0	1	0
2	0	1	0	1	0	1	0
3	0	1	1	0	0	0	0
4	1	0	0	0	1	0	1
5	1	0	1	0	1	0	1
6	1	1	0	1	1	1	1
7	1	1	1	1	1	1	1

8. Realice la implementación.

Para implementar en un dispositivo lógico programable se proponen las siguientes opciones:

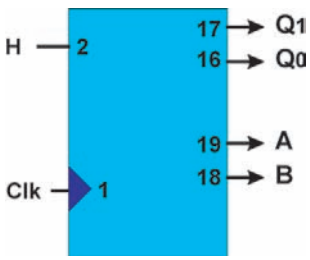
a) Captura esquemática.
b) Ecuaciones en ABEL-HDL.
c) Tabla de estados en ABEL-HDL.
d) Descripción del diagrama de transición.

a) Captura esquemática.



b) Ecuaciones en ABEL-HDL (*equations*).

```
MODULE b2eq
“Entradas
        Clk, H Pin 1,2;
“ Salidas Combinacionales
        A,B Pin 19,18 istype’com’;
“ Salidas Registradas
        Q1,Q0 pin 17,16 istype’reg’;
DECLARATIONS
        sreg=[Q0,Q1];
EQUATIONS
“ Conectar el Clk a los dos Flip Flops
        sreg.clk=Clk;
Equations
        Q1:= H&!Q0;
        Q0:= H&!Q0# !H& Q0;
        A= !Q1&!Q0;
        B= Q1&!Q0;
        END
```



c) Uso del comando TRUTH_TABLE.

Este mismo comando, usado en sistemas combinacionales, podría utilizarse para describir las tablas de estados cambiando **->** por **:>** en la tabla.

([Entrada, estados presentes]:>[estados próximos])

Es necesario sincronizar los Flip Flops usados en el diseño conectando a un mismo punto las entradas de Clk para que todos los Flip Flops cambien al mismo tiempo (sistema secuencial síncrono). En ABEL-HDL se realiza lo siguiente:

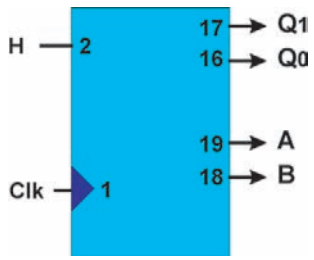
sreg=[Q0,Q1];

Hay que definir en una sola variable las Q de los Flip Flops, y posteriormente hacer que esa variable dependa de la misma señal de sincronía Clk.

EQUATIONS
sreg.clk=Clk;

Archivo en ABEL-HDL.

```
MODULE b2te
“Entradas
Clk, H Pin 1,2;
“ Salidas Combinacionales
A,B Pin 19,18 istype’com’;
“ Salidas Registradas
Q1,Q0 pin 17,16 istype’reg’;
sreg=[Q0,Q1];
EQUATIONS
“ Conectar el Clk a los dos Flip Flops
sreg.clk=Clk;
TRUTH_TABLE
([ H, Q1,Q0]:>[Q1,Q0])
 [ 0,0,0]:>[0,0];
 [ 0,0,1]:>[1,0];
            [ 0,1,0]:>[1,0];
 [ 0,1,1]:>[0,0];
 [ 1,0,0]:>[0,1];
 [ 1,0,1]:>[0,1];
            [ 1,1,0]:>[1,1];
 [ 1,1,1]:>[1,1];
END
```



Entrada	Estados Presentes		Estados Próximos	
H	Q1	Q0	Q1+1	Q0+1
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	0
1	0	0	0	1
1	0	1	0	1
1	1	0	1	1
1	1	1	1	1

Incluir tabla de verdad para A y B

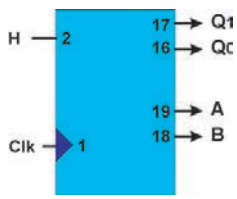
d) Sentencia STATE_DIAGRAM, STATE y los comandos IF, THEN y ELSE.

Las máquinas de estados finitos también se diseñan utilizando la sentencia **STATE_DIAGRAM**, donde se pueden incluir los comandos **IF**, **THEN** y **ELSE** para explicar el comportamiento del sistema secuencial, con sólo seguir el diagrama de transición, con reducción del tiempo de diseño al obtener la tabla de estados, las ecuaciones o el diagrama esquemático.

Descripción del diagrama de transición mediante la sentencia STATE_DIAGRAM y los comandos IF, THEN y ELSE.

Los elementos necesarios para solucionar el sistema secuencial mediante la sentencia **STATE_DIAGRAM** son:

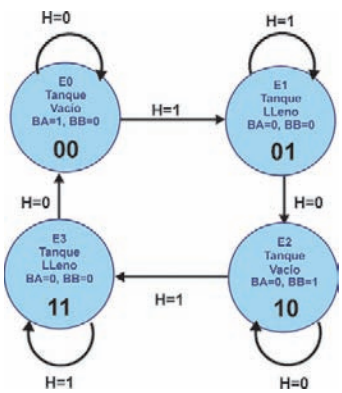
El bloque en donde se indican entradas y salidas.



La asignación de valores a los estados.

Estados		Salidas FF's
		Q_1 Q_0
E0	**Tanque Vacío**	**0 0**
E1	**Tanque Lleno**	**0 1**
E2	**Tanque Vacío**	**1 0**
E3	**Tanque Lleno**	**1 1**

El diagrama de transición.

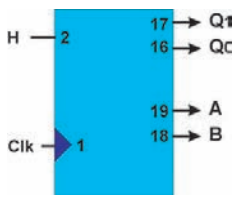


El archivo en formato ABEL-HDL para la solución del problema se divide en cuatro partes:

1. La declaración de entradas y salidas.
2. Sincronización de los FF.
3. Asignación de valores a los estados.
4. Descripción del diagrama de transición.

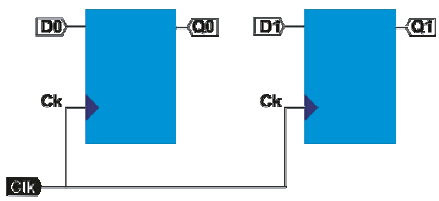
1.

```
MODULE bdt
“Entradas
    Clk, H Pin 1,2;
“ Salidas Combinacionales
    A,B Pin 19,18 istype’com’;
“ Salidas Registradas
    Q1,Q0 pin 17,16 istype’reg’;
```



2.

```
“ Conectar el Clk a los dos Flip Flops
sincronizar
    DECLARATIONS
        sreg=[Q0,Q1];
EQUATIONS
        sreg.clk=Clk;
```



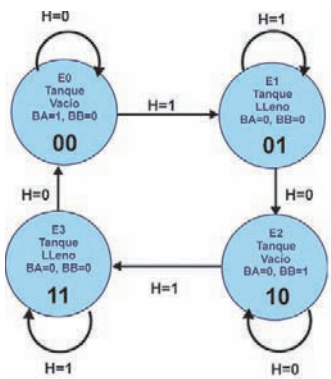
3.

```
“ Asignar Valores a los estados
DECLARATIONS
    E0=[0, 0];
    E1=[0, 1];
    E2=[1, 0];
    E3=[1, 1];
```

Estados		Salidas FF's
		Q1 Q0
E0	Tanque Vacío	0 0
E1	Tanque Lleno	0 1
E2	Tanque Vacío	1 0
E3	Tanque Lleno	1 1

4.

```
state_diagram sreg;
state E0:
    A=1;
    B=0;
    if H then E1 else E0;
state E1:
    A=0;
    B=0;
    if H then E1 else E2;
state E2:
    A=0;
    B=1;
    if H then E3 else E2;
state E3:
A=0; B=0;
    if H then E3 else E0;
END
```



Simulación

Al igual que en los sistemas combinacionales, para efectuar la simulación se podría utilizar el comando **TEST_VECTORS**; además, para generar los pulsos de reloj se recomienda usar **.C.** Para simplificar la captura de .c. y teclear solamente C, se declara una constante, como se indica a continuación:


```
“Constantes
      C,x = .c.,.x.;
```

Esto es, al principio del archivo ABEL-HDL y después de la instrucción MODULE.

Por ejemplo:

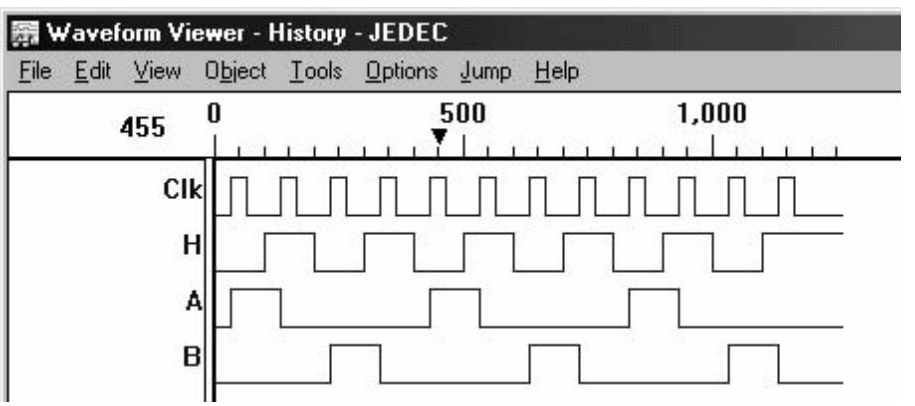
```
MODULE bddt
“Constantes
      C,x = .c.,.x.;
“Entradas
    Clk, H Pin 1,2;

“Salidas Combinacionales
    A,B Pin 19,18 istype’com’;

“Salidas Registradas
    Q1,Q0 pin 17,16 istype’reg’;

DECLARATIONS
    sreg=[Q0,Q1];

EQUATIONS
“Conectar el Clk a los dos Flip Flops
sincronizar
    sreg.clk=Clk;

“Asignar Valores a los estados
DECLARATIONS
    E0=[0, 0];
    E1=[0, 1];
    E2=[1, 0];
    E3=[1, 1];
```

```
state_diagram sreg;
state E0:
    A=1, B=0;
    if H then E1 else E0;
state E1:
    A=0, B=0;
    if H then E1 else E2;
state E2:
    A=0, B=1;
    if H then E3 else E2;
state E3:
    A=0 , B=0;
    if H then E3 else E0;
“Simulación
Test_Vectors ([Clk ,H] -> [A,B])
    [ C ,0 ] -> [x,x];
    [ C ,1 ] -> [x,x];
    [ C ,0 ] -> [x,x];
[ C ,1 ] -> [x,x];
    [ C ,0 ] -> [x,x];
    [ C ,1 ] -> [x,x];
    [ C ,0 ] -> [x,x];
[ C ,1 ] -> [x,x];
    [ C ,0 ] -> [x,x];
    [ C ,1 ] -> [x,x];
    [ C ,0 ] -> [x,x];
    [ C ,1 ] -> [x,x];
end
```

En la simulación secuencial es necesario incluir la señal de sincronía Clk.

```
Test_Vectors
([Clk ,H] -> [A,B])
    [ C,0 ] -> [x,x];
    [ C,1 ] -> [x,x];
    [ C,0 ] -> [x,x];
    [ C,1 ] -> [x,x];
    [ C,0 ] -> [x,x];
    [ C,1 ] -> [x,x];
    [ C,0 ] -> [x,x];
    [ C,1 ] -> [x,x];
    [ C,0 ] -> [x,x];
    [ C,1 ] -> [x,x];
    [ C,0 ] -> [x,x];
    [ C,1 ] -> [x,x];
end
```

La instrucción C (.c.) equivale a un pulso de reloj Clk, la entrada H corresponde a la entrada de nivel, mientras que A y B son las bombas.

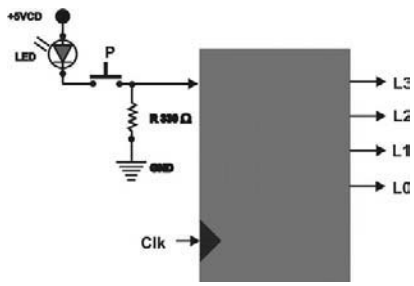


Ejemplo 9.2

Máquina de Moore, no completamente especificada

Diseñe un sistema secuencial síncrono con las siguientes características:

El sistema cuenta con un botón de entrada **P** y cuatro lámparas de salida **L3**, **L2**, **L1** y **L0**.



El sistema tiene una entrada de sincronía para los Flip Flops, llamada Clk, la cual establece el cambio de evento.

La operación de la secuencia deseada se describe a continuación:

a) En condiciones iniciales, las cuatro lámparas deberán estar apagadas.

b) Si se oprime por primera vez **P**, encenderá sólo la lámpara **L3**.

c) Si se oprime de nuevo **P**, prenderán las lámparas **L3** y **L2.**

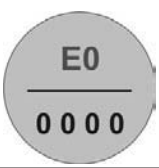
d) Si se oprime **P** por tercera vez, encenderán las lámparas **L3**, **L2** y **L1**.

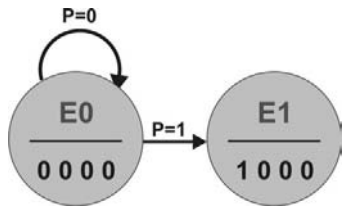
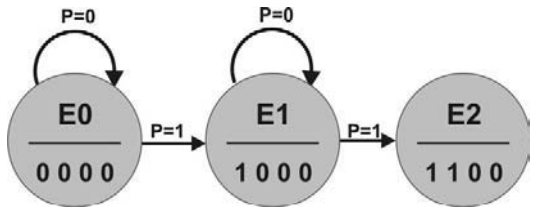
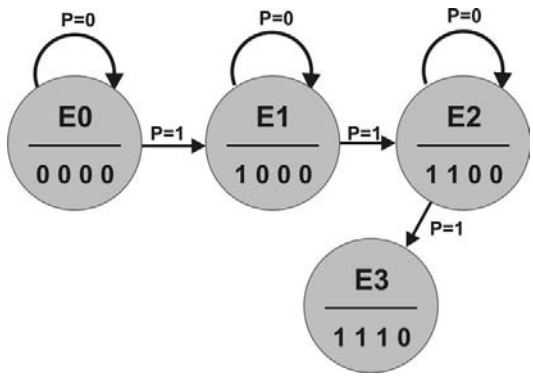
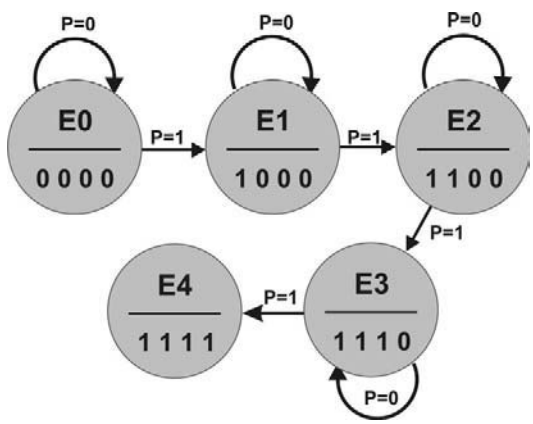
e) Si se oprime **P** por cuarta vez, prenderán todas las lámparas **L3**, **L2**, **L1** y **L0**.

f) Si se oprime **P** por quinta vez, regresará a las condiciones iniciales, en donde las cuatro lámparas deberán estar apagadas para repetir el ciclo nuevamente.

Solución:

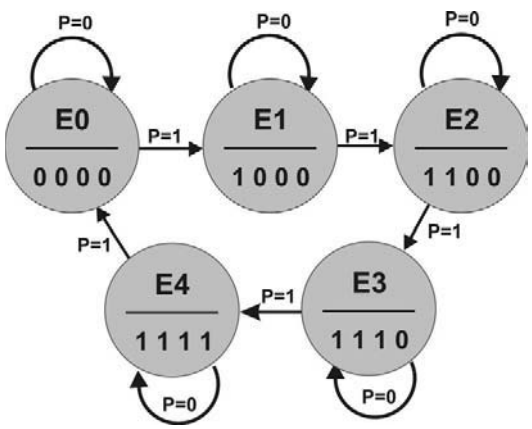
1. Diagrama de transición.

Los estados estarán definidos por EX, donde X es un valor decimal (E0, E1, E2, etc.), que lo diferencia de los demás estados. Los valores de salida se indican con L3, L2, L1 y L0.	
Se inicia con un estado E0 que representa las condiciones iniciales, y donde las lámparas L3, L2, L1 y L0 están apagadas.	
En el estado E0, si no se oprime el botón P (P = 0), el estado próximo es el mismo E0, como lo indica la figura.	

En el estado E0, si se presiona el botón P (P = 1), el estado próximo será E1 (véase figura). El estado E1 tiene como salida la L3 encendida (L3 = 1); y las demás están apagadas.	
En el estado E1, si P = 0, continuará en el mismo estado E1. Si se oprime el botón P (P = 1), el estado próximo será E2. El estado E2 tiene como salidas la L3 y L2 encendidas; y las demás estarán apagadas.	
Colocándose en el estado E2, si P = 0, seguirá en el mismo estado E2. Si se presiona el botón P (P = 1), el estado próximo será E3. El estado E3 tiene como salidas la L3, L2 y L1 encendidas; y la L0 apagada.	
Situándose en el estado E3, si P = 0, permanecerá en el mismo estado E3. Si se oprime el botón P (P = 1) el estado próximo será E4. El estado E4 tiene todas salidas L3, L2, L1 y L0 encendidas.	

En el estado E4, si P = 0, continuará en el mismo estado E4.

Si se presiona el botón P (P = 1), el estado próximo será E0.



2. Cantidad de Flip Flops.

Una vez que se obtiene el diagrama de transición con el número de estados (cinco), se determina la cantidad de Flip Flops.

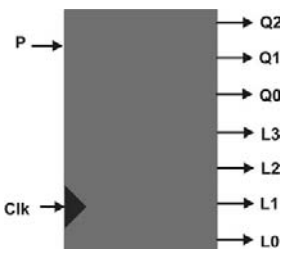
Para tener cinco estados se requieren mínimo tres Flip Flops que se llamarán Q2, Q1 y Q0. Con tres Flip Flops es posible tener hasta ocho estados diferentes, aunque sólo se utlilizarán cinco.

3. Asignar valores a los estados.

Estado	Valor			Salidas			
	Q2	Q1	Q0	L3	L2	L1	L0
E0	0	0	0	0	0	0	0
E1	0	0	1	1	0	0	0
E2	0	1	0	1	1	0	0
E3	0	1	1	1	1	1	0
E4	1	0	0	1	1	1	1

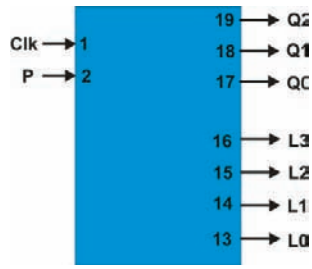
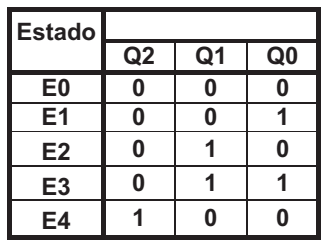
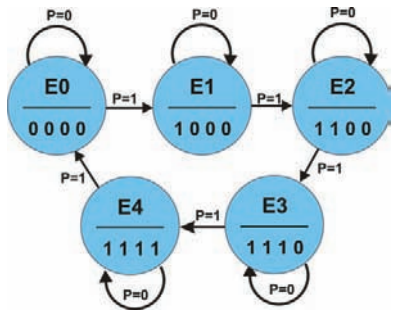
4. Determinar las entradas y las salidas.

P	Entrada botón.
Clk	Entrada de sincronía de los Flip Flops.
Q2, Q1 y Q0	Salidas de los Flip Flops.
L3, L2, L1 y L0	Salidas combinacionales.



Para la solución del problema 2 por medio de la instrucción STATE_DIAGRAM, en el archivo en formato ABEL-HDL se requiere de:

1. Bloque de entradas y salidas.
2. La asignación de valores a los estados.
3. El diagrama de transición.

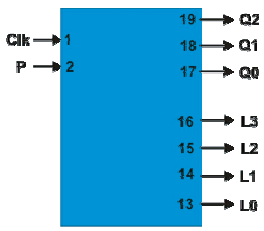
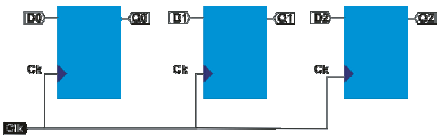
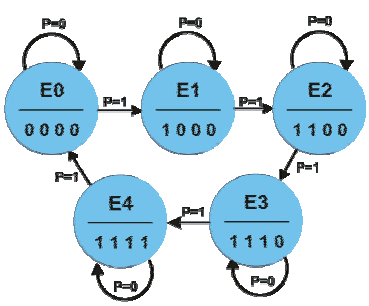
1.	El bloque en donde se indican entradas y salidas, así como la asignación de terminales.	
2.	La definición de valores a los estados.	
3.	El diagrama de transición.	

Estado	Q2	Q1	Q0
E0	0	0	0
E1	0	0	1
E2	0	1	0
E3	0	1	1
E4	1	0	0

El archivo en formato ABEL-HDL para la solución del problema se divide en cuatro partes:

1. La declaración de entradas y salidas.
2. Sincronización de los FF.

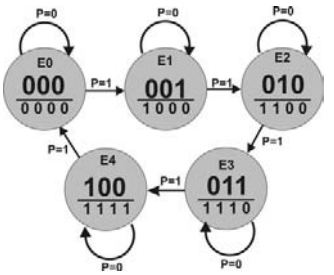
3. Asignación de valores a los estados.
4. Descripción del diagrama de transición.

<table>
<tr>
<td>1.</td>
<td>MODULE botón
“Entradas
Clk, P Pin 1,2;
“Salidas Combinacionales (luces)
L3,L2,L1,L0 Pin 16,15,14,13 istype-’com’;
“Salidas Registradas (Flip Flops)
Q2,Q1,Q0 pin 19,18,17 istype’reg’;</td>
<td> 
</td>
</tr>
<tr>
<td>2.</td>
<td>“Conectar el Clk a los tres Flip Flops (sincronizar)
DECLARATIONS
sreg=[Q2,Q1,Q0];
EQUATIONS
sreg.clk=Clk;</td>
<td> 
</td>
</tr>
<tr>
<td>3.</td>
<td>“Asignar Valores a los estados
DECLARATIONS
E0=[0, 0,0];
E1=[0,0,1];
E2=[0,1,0];
E3=[0,1,1];
E4=[1,0,0];</td>
<td>
<table>
<tr><th rowspan="2">Estado</th><th colspan="3"></th></tr>
<tr><th>Q2</th><th>Q1</th><th>Q0</th></tr>
<tr><td>E0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>E1</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>E2</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>E3</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>E4</td><td>1</td><td>0</td><td>0</td></tr>
</table>
</td>
</tr>
<tr>
<td>4.</td>
<td>state_diagram sreg;
state E0:
L3=0;L2=0;L1=0;L0=0;
if P then E1 else E0;
state E1:
L3=1;L2=0;L1=0;L0=0;
if P then E2 else E1;
state E2:
L3=1;L2=1;L1=0;L0=0;
if P then E3 else E2;
state E3:
L3=1;L2=1;L1=1;L0=0;
if P then E4 else E3;
state E4:
L3=1;L2=1;L1=1;L0=1;
if P then E0 else E4;
END</td>
<td> 
</td>
</tr>
</table>

Una segunda forma de solucionar el problema es mediante la tabla de estados.

Hay que considerar que los tres Flip Flops pueden generar hasta ocho estados y para este ejemplo sólo utilizamos cinco de ellos. Los tres restantes son considerados como un valor **X**.

Los valores de la tabla de estados se obtienen a partir del diagrama de transición, donde los estados toman el valor que se les asignó previamente, por ejemplo, el estado E0 = 000.



En las primeras ocho combinaciones, P = 0 (de m = 0 hasta m = 7), el estado próximo es igual al estado presente; las segundas ocho son para P = 1 (de m = 8 hasta m =15), donde el estado próximo es diferente del estado presente.

m	Entrada P	Estados presentes			Estados próximos			Salidas			
		Q2	Q1	Q0	Q2+1	Q1+1	Q0+1	L3	L2	L1	L0
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	1	1	0	0	0
2	0	0	1	0	0	1	0	1	1	0	0
3	0	0	1	1	0	1	1	1	1	1	0
4	0	1	0	0	1	0	0	1	1	1	1
5	0	1	0	1	X	X	X	X	X	X	X
6	0	1	1	0	X	X	X	X	X	X	X
7	0	1	1	1	X	X	X	X	X	X	X
8	1	0	0	0	0	0	1	0	0	0	0
9	1	0	0	1	0	1	0	1	0	0	0
10	1	0	1	0	0	1	1	1	1	0	0
11	1	0	1	1	1	0	0	1	1	1	0
12	1	1	0	0	0	0	0	1	1	1	1
13	1	1	0	1	X	X	X	X	X	X	X
14	1	1	1	0	X	X	X	X	X	X	X
15	1	1	1	1	X	X	X	X	X	X	X

Archivo en formato ABEL-HDL usando el comando TRUTH_TABLE

```
MODULE botonte
“Entradas
Clk, P Pin 1,2;
“ Salidas Combinacionales (luces)
L3,L2,L1,L0 Pin 16,15,14,13 istype’dc,com’;
“Salidas Registradas
Q2,Q1,Q0 pin 19,18,17 istype’dc,reg’;
```

```
“Conectar el Clk a los tres Flip Flops (sincronizar)
DECLARATIONS
        sreg=[ Q2,Q1,Q0];
EQUATIONS
        sreg.clk=Clk;
```

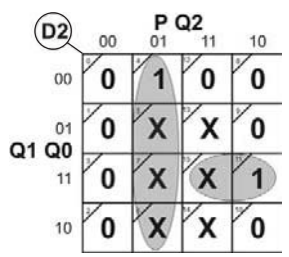
```
“Tabla de estados
TRUTH_TABLE
        ([P,Q2,Q1,Q0]:>[Q2,Q1,Q0])
        [0,0,0,0]:>[0,0,0];
        [0,0,0,1]:>[0,0,1];
        [0,0,1,0]:>[0,1,0];
        [0,0,1,1]:>[1,0,0];
        [0,1,0,0]:>[1,0,0];
        [1,0,0,0]:>[0,0,1];
        [1,0,0,1]:>[0,1,0];
        [1,0,1,0]:>[0,1,1];
        [1,0,1,1]:>[0,0,0];
        [1,1,0,0]:>[0,0,0];
```

```
“Decodificador de salida
TRUTH_TABLE
        ([Q2,Q1,Q0]->[L3,L2,L1,L0])
        [0,0,0]->[0,0,0,0];
        [0,0,1]->[1,0,0,0];
        [0,1,0]->[1,1,0,0];
        [0,1,1]->[1,1,1,0];
        [1,0,0]->[1,1,1,1];
END
```

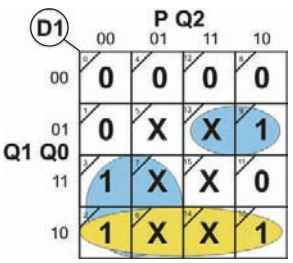
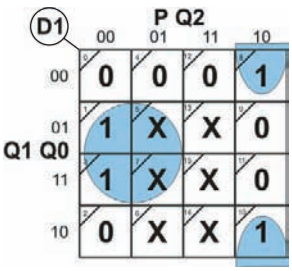
Una tercera forma de solucionar el sistema secuencial del botón es por medio de las ecuaciones obtenidas partiendo de la tabla de estados. Si como Flip Flop se señala el FF D, las entradas de control serán los mismos valores que los asignados a los estados próximos.

	Entrada	Estados presentes			Estados próximos			Entradas de control			Salidas			
m	P	Q2	Q1	Q0	Q2+1	Q1+1	Q0+1	D2	D1	D0	L3	L2	L1	L0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	1	0	0	1	1	0	0	0
2	0	0	1	0	0	1	0	0	1	0	1	1	0	0
3	0	0	1	1	0	1	1	0	1	1	1	1	1	0
4	0	1	0	0	1	0	0	1	0	0	1	1	1	1
5	0	1	0	1	X	X	X	X	X	X	X	X	X	X
6	0	1	1	0	X	X	X	X	X	X	X	X	X	X
7	0	1	1	1	X	X	X	X	X	X	X	X	X	X
8	1	0	0	0	0	0	1	0	0	1	0	0	0	0
9	1	0	0	1	0	1	0	0	1	0	1	0	0	0
10	1	0	1	0	0	1	1	0	1	1	1	1	0	0
11	1	0	1	1	1	0	0	1	0	0	1	1	1	0
12	1	1	0	0	0	0	0	0	0	0	1	1	1	1
13	1	1	0	1	X	X	X	X	X	X	X	X	X	X
14	1	1	1	0	X	X	X	X	X	X	X	X	X	X
15	1	1	1	1	X	X	X	X	X	X	X	X	X	X

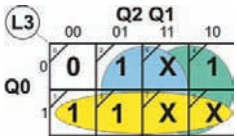
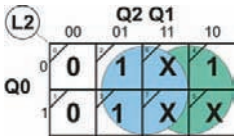
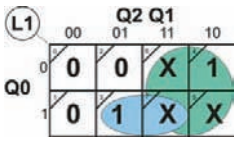
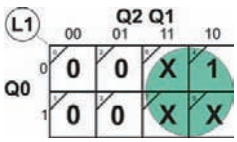
Se obtendrán las ecuaciones para D2, D1 y D0, además de las salidas L3, L2, L1 y L0 y se hará uso de mapas de Karnaugh para calcular la ecuación simplificada.



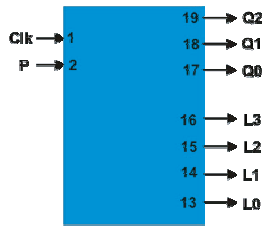
D2 = P' Q2 + P Q1 Q0

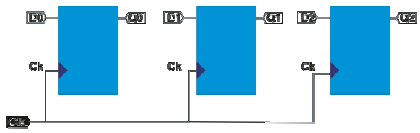
	D1 = P' Q1 + Q1 Q0' + P Q1' Q0
	D0 = P' Q0 + P Q2' Q0

Para las salidas L3, L2, L1 y L0 no es necesario incluir la variable P, ya que sólo dependen de los valores de Q2, Q1 y Q0 (máquina de Mealy).

	L3 = Q2 + Q1 + Q0
	L2 = Q2 + Q1
	L1 = Q2 + Q1 Q0
	L0 = Q2

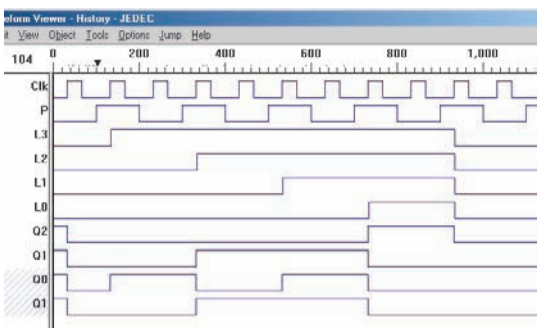
Archivo en formato ABEL-HDL usando el comando *equations*

MODULE botonec “Entrada Clk, P Pin 1,2; “Salidas Combinacionales (luces) L3,L2,L1,L0 Pin 16,15,14,13 istype’dc,com’; “Salidas Registradas Q2,Q1,Q0 pin 19,18,17 istype’dc,reg’;	

“conectar el Clk a los tres Flip Flops (sincronizar)
DECLARATIONS
sreg=[Q2,Q1,Q0];
 	
EQUATIONS
sreg.clk=Clk;
	

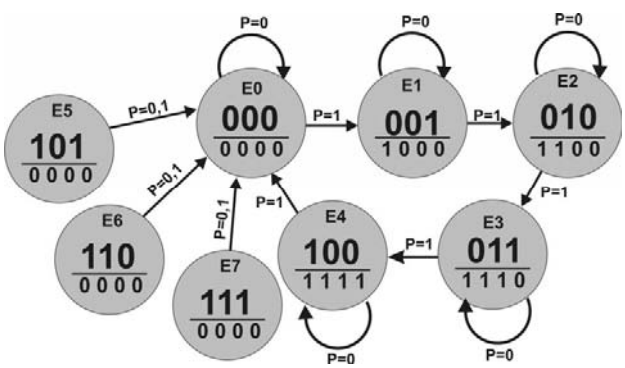
equations
Q2:=!P&Q2#P&Q1&Q0;
Q1:=!P&Q1#Q1&!Q0#P&!Q1&Q0;
Q0:=!P& Q0 # P& !Q2& Q0;
L3= Q2#Q1#Q0;
L2=Q1#Q0;
L1=Q2#Q1&Q0;
L0=Q2;
 	
END	

Simulación

MODULE botondt **C,X =.c.,.x.;** "Entradas Clk, P Pin 1,2;	**Si se incluye una línea después de module, con C, X = .c.,.x.; se simplifican los caracteres usados en la simulación.**
Test_Vectors ([Clk,P]- >[L3,L2,L1,L0,Q2,Q1,Q0]) [C,0]->[X,X,X,X,X,X,X]; [C,1]->[X,X,X,X,X,X,X]; [C,0]->[X,X,X,X,X,X,X]; [C,1]->[X,X,X,X,X,X,X]; [C,0]->[X,X,X,X,X,X,X]; [C,1]->[X,X,X,X,X,X,X]; [C,0]->[X,X,X,X,X,X,X]; [C,1]->[X,X,X,X,X,X,X]; [C,0]->[X,X,X,X,X,X,X]; [C,1]->[X,X,X,X,X,X,X];	

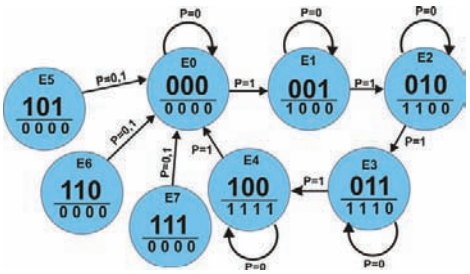
Es posible que el sistema inicie en un estado cuyo valor de Q2, Q1 y Q0 no se haya tomado en cuenta, por lo que el funcionamiento del sistema podría ser incierto. Una recomendación sería tomar en cuenta los estados no incluidos y asignar como estado próximo, con cualquier valor de entrada P, el estado E0, como lo indica la figura con los estados E5, E6 y E7:

m	Estado presente	Estado próximo	
		P=0	P=1
0	**E0**	**E0**	**E1**
1	**E1**	**E1**	**E2**
2	**E2**	**E2**	**E3**
3	**E3**	**E3**	**E4**
4	**E4**	**E4**	**E0**
5	**E5**	**E0**	**E0**
6	**E6**	**E0**	**E0**

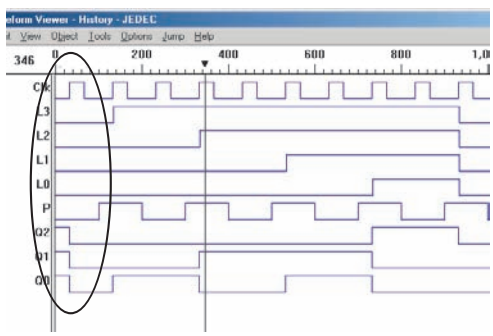


Archivo en formato ABEL-HDL incluyendo los estados E5, E6 y E7

```
      MODULE botón
      “Entradas
      C,X=.c.,.x.;
            Clk, P Pin 1,2;
      “ Salidas Combinacionales (luces)
            L3,L2,L1,L0 Pin 16,15,14,13
            istype’com’;
      “ Salidas Registradas
            Q2,Q1,Q0 pin 19,18,17 istype’reg’;
“ Sincronizar los Flip Flops
      DECLARATIONS
            sreg=[ Q2,Q1,Q0];
      EQUATIONS
sreg.clk=Clk;
      “ Asignar Valores a los estados
      DECLARATIONS
            E0=[0, 0,0];
            E1=[0,0,1];
            E2=[0,1,0];
            E3=[0,1,1];
      E4=[1,0,0];
            E5=[1,0,1];
      E6=[1,1,0];
      E7=[1,1,1];
      state_diagram sreg;
      state E0:      L3=0;L2=0;L1=0;L0=0;
            if P then E1 else E0;
      state E1:      L3=1;L2=0;L1=0;L0=0;
            if P then E2 else E1;
      state E2:      L3=1;L2=1;L1=0;L0=0;
            if P then E3 else E2;
      state E3: L3=1;L2=1;L1=1;L0=0;
            if P then E4 else E3;
      state E4: L3=1;L2=1;L1=1;L0=1;
            if P then E0 else E4;
      state E5:      L3=0;L2=0;L1=0;L0=0;
            Goto E0;
      state E6:      L3=0;L2=0;L1=0;L0=0;
            Goto E0;
      state E7:      L3=0;L2=0;L1=0;L0=0;
            Goto E0;
```



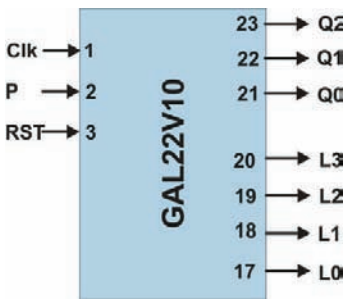
```
Test_Vectors
([Clk,P]->[L3,L2,L1,L0,Q2,Q1,Q0])
[C,0]->[X,X,X,X,X,X,X];
[C,1]->[X,X,X,X,X,X,X];
[C,0]->[X,X,X,X,X,X,X];
[C,1]->[X,X,X,X,X,X,X];
[C,0]->[X,X,X,X,X,X,X];
[C,1]->[X,X,X,X,X,X,X];
[C,0]->[X,X,X,X,X,X,X];
[C,1]->[X,X,X,X,X,X,X];
[C,0]->[X,X,X,X,X,X,X];
[C,1]->[X,X,X,X,X,X,X];
[C,0]->[X,X,X,X,X,X,X];
[C,1]->[X,X,X,X,X,X,X];
end
```



Observe que el sistema inició con los valores de 1, 1 y 1 correspondientes a Q2, Q1 y Q0, respectivamente (E7), en el primer pulso de reloj Clk el estado siguiente es 0, 0 y 0 (E0).

Una segunda opción para evitar que el sistema llegue a un estado no contemplado y se vuelva inestable es incluir una entrada de reset RST, cuya función sea que al activarse (RST = 1) las salidas Q de los Flip Flops sean iguales a cero, llevando el sistema al estado E0. Al usar el comando **ASYNC_RESET** (.ar) de ABEL-HDL se logra que todos los Flip Flops queden sincronizados con la señal de reset.

Cuando se utiliza el comando **ASYNC_RESET** (.ar), es posible que un dispositivo GAL16V8 no tenga la capacidad suficiente para implementarlo; en tal caso se debe recurrir al GAL22V10 porque tiene más capacidad.



```
MODULE botondt
C,X =.c.,.x.;
“Entradas
Clk, P, RST Pin 1,2,3;
“Salidas Combinacionales (luces)
L3,L2,L1,L0 Pin 20,19,18,17 istype’com’;
“Salidas Registradas
Q2,Q1,Q0 pin 23,22,21 istype’reg’;
“ Sincronizar Flip Flops
DECLARATIONS
sreg=[ Q2,Q1,Q0];
EQUATIONS
sreg.clk=Clk;
sreg.ar=RST;
“ Asignar Valores a los estados
DECLARATIONS
E0=[0, 0,0];
E1=[0,0,1];
E2=[0,1,0];
E3=[0,1,1];
E4=[1,0,0];
state_diagram sreg;
state E0:
L3=0;L2=0;L1=0;L0=0;
if P then E1 else E0;
state E1:
L3=1;L2=0;L1=0;L0=0;
if P then E2 else E1;
state E2:
L3=1;L2=1;L1=0;L0=0;
if P then E3 else E2;
state E3:
L3=1;L2=1;L1=1;L0=0;
if P then E4 else E3;
state E4:
L3=1;L2=1;L1=1;L0=1;
if P then E0 else E4;
Test_Vectors
([Clk,P]->[L3,L2,L1,L0,Q2,Q1,Q0])
[C,0]->[X,X,X,X,X,X,X];
[C,1]->[X,X,X,X,X,X,X];
[C,0]->[X,X,X,X,X,X,X];
[C,1]->[X,X,X,X,X,X,X];
[C,0]->[X,X,X,X,X,X,X];
[C,1]->[X,X,X,X,X,X,X];
[C,0]->[X,X,X,X,X,X,X];
[C,1]->[X,X,X,X,X,X,X];
[C,0]->[X,X,X,X,X,X,X];
[C,1]->[X,X,X,X,X,X,X];
[C,0]->[X,X,X,X,X,X,X];
[C,1]->[X,X,X,X,X,X,X];
END
```

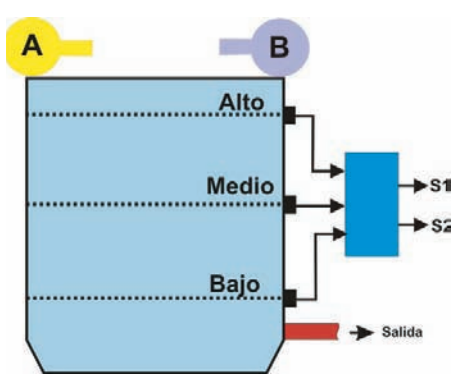
Ejemplo 9.3

Máquina de Mealy

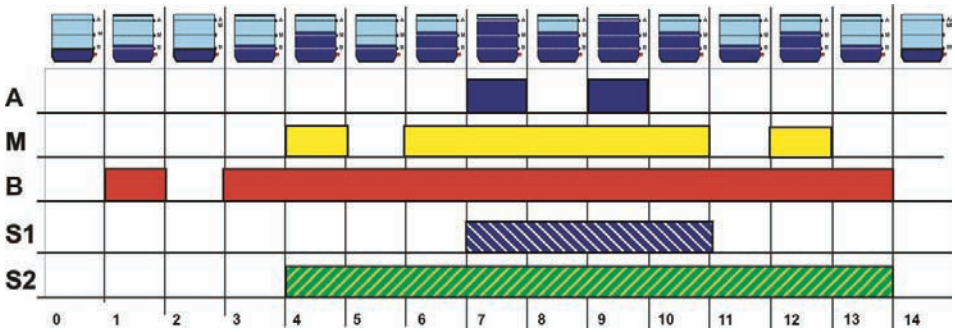
El tanque de la figura se alimenta por medio de dos bombas llamadas **A** y **B**.

El gasto de salida nunca será mayor al que proporcionen las dos bombas operando simultáneamente.

El tanque tiene un sistema detector de niveles que consta de tres sensores de entrada nivel A (alto), M (medio) y B (bajo), y dos salidas $\mathbf{S_2}, \mathbf{S_1}$.



El sistema detector de niveles trabaja de la siguiente manera:



En T = 0 el nivel está por debajo del sensor B (bajo) y la salida es S2 = 0, S1 = 0.

En T = 1 el nivel está por encima del sensor B y la salida se mantiene en S2 = 0, S1 = 0.

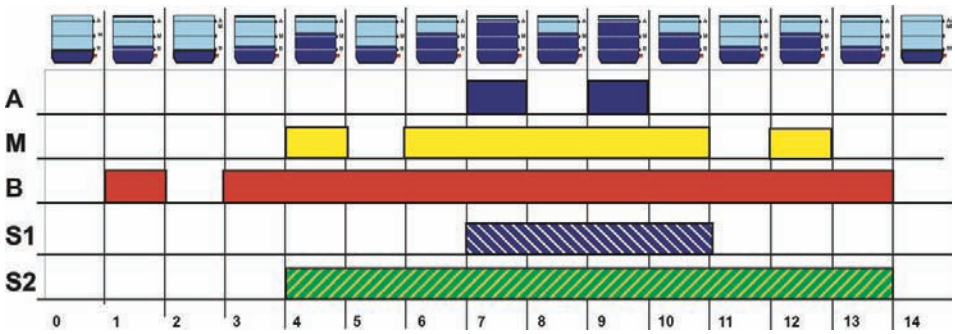
En T = 2 el nivel baja de nuevo y en T = 3 vuelve a subir (posible oleaje, manteniéndose la salida en S2 = 0, S1 = 0.

En T = 4 el nivel pasa por encima del sensor medio (M) y la salida ahora será

S2 = 1, S1 = 0.

Si se presentara una ola (baja y sube en el sensor M, T = 5 y T = 6) la salida se continuará en S2 = 1, S1 = 0.

En T = 7 el nivel pasa por encima del nivel alto (A) y la salida será ahora S2 = 1, S1 = 1, si se originara la ola (baja, sube y baja en el sensor A, T = 8, T = 9 y T = 10) la salida se mantendrá en S2 = 1, S1 = 1.



En T = 11 el nivel está por debajo del sensor M, la salida cambia a S2 = 1, S1 = 0 y se mantendrá así (T = 12 y T = 13 posible ola) hasta que el nivel esté por debajo del sensor B (T = 14), en donde la salida cambiará a S2 = 0, S1 = 0.

Trabajo solicitado

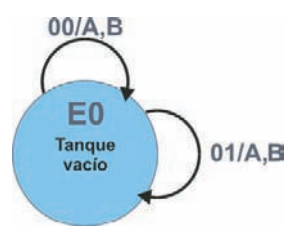
Diseñe un sistema secuencial con la máquina de Mealy, pero que, en función de la salida del sistema de detección de niveles S2, S1, controle la siguiente secuencia de operación de las bombas:

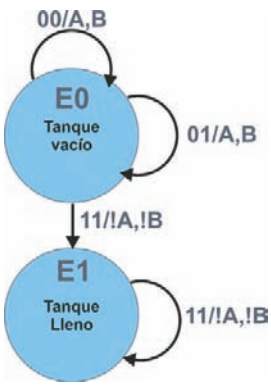
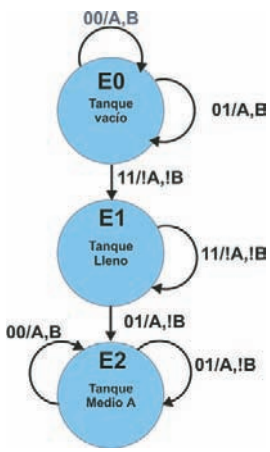
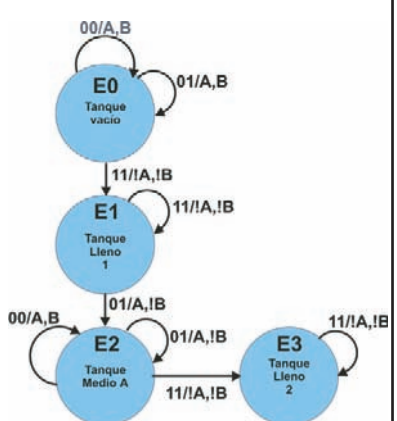
Procedimiento

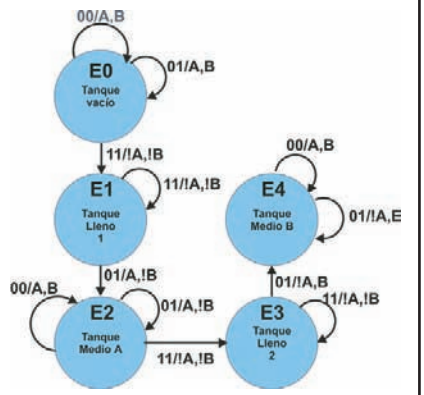
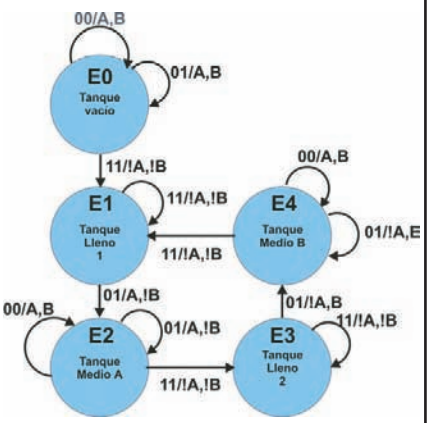
1. Partiendo de que el tanque se encuentra vacío (S2 = 0 y S1 = 0), se inicia el llenado con ambas bombas (**A** y **B**) hasta llegar al nivel alto A (S2 = 1 y S1 = 1), momento en el que las dos bombas se apagarán.
2. Si el nivel llega por debajo del sensor M (S2 = 1 y S1 = 0), deberá trabajar sólo la bomba A, si el nivel sigue bajando hasta el sensor B (S2 = 0 y S1 = 0), deberán operar ambas bombas hasta llegar el nivel al sensor M (S2 = 1 y S1 = 0), de ahí continuará sólo la bomba A hasta que llene el tanque (S2 = 1 y S1 = 1), entonces se apagarán ambas bombas.
3. Una vez llenado el tanque, si el nivel llega abajo del sensor M (S2 = 1 y S1 = 0), deberá de trabajar sólo la bomba B; si el nivel sigue bajando hasta el sensor B (S2 = 0 y S1 = 0), deberán trabajar ambas bombas hasta el nivel al sensor M (S2 = 1 y S1 = 0); de ahí continuará sólo la bomba B hasta llenar el tanque (S2 = 1 y S1 = 1), y en ese momento deberá apagarse la bomba B.
4. Cada vez que se llene el tanque (S2 = 1 y S1 = 1), y de ahí pase al nivel medio (S2 = 1 y S1 = 0), deberá alternarse el funcionamiento de la bombas A y B.
5. Cada vez que el nivel esté por debajo del sensor B (S2 = 0 y S1 = 0), deberán trabajar ambas bombas hasta llegar al nivel medio M (S2 = 1 y S1 = 0); a partir de ahí sólo trabajará una bomba, A o B, según corresponda.

Solución

1. Diagrama de transición.

El estado **E0** se llamará tanque vacío, en donde si las entradas S1 y S2 son 0,0 o 0,1, deberá permanecer en el mismo estado E0.	

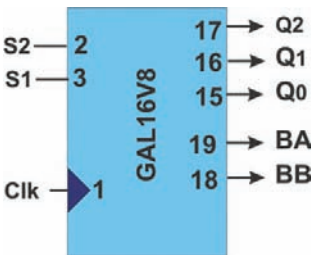
En el estado E0, si el tanque se llena (S1 = 1, S2 = 1), llegará al estado E1 con las bombas apagadas, donde, si sigue lleno, deberá seguir en el mismo estado E1 y con las bombas apagadas.	
En cambio, en el estado E1, si el nivel llega a la mitad M (S1 = 0, S2 = 1), trabajará sólo la bomba A y cambiará a un estado E2, donde, si el nivel está por encima del sensor B (S1 = 0, S2 = 1), permanecerá en el mismo estado E2 y trabajará sólo la bomba A, pero si el nivel llega al sensor B (S1 = 0, S2 = 0), deberán trabajar ambas bombas, y si se recupera el nivel a la mitad (S1 = 0, S2 = 1) trabajará sólo la bomba A.	
En el estado E2, si el tanque se llena por segunda vez (S1 = 1, S2 = 1), se apagará la bomba A y el sistema cambiará a un nuevo estado E3 (tanque lleno 2), donde, si sigue lleno, permanecerá en el mismo estado E3 con las bombas apagadas.	

En el estado E3, si el nivel llega a la mitad M (S1 = 0, S2 = 1), trabajará sólo la bomba B y cambiará a un estado E4, donde, si el nivel está por encima del sensor B (S1 = 0, S2 = 1), continuará en el mismo estado E2 y trabajará sólo la bomba B. Si el nivel llega al sensor B (S1 = 0, S2 = 0), deberán trabajar ambas bombas; si se recuperara el nivel a la mitad (S1 = 0, S2 = 1), funcionará sólo la bomba B.	
En el estado E4, si el tanque se llena por tercera vez (S1 = 1, S2 = 1), se apagará la bomba B y el sistema cambiará al estado E1 (tanque lleno 1), donde, si sigue lleno, seguirá en el mismo estado E1 con las bombas apagadas.	

2. Definir las entradas y las salidas.

Entradas Clk, S2 y S1.

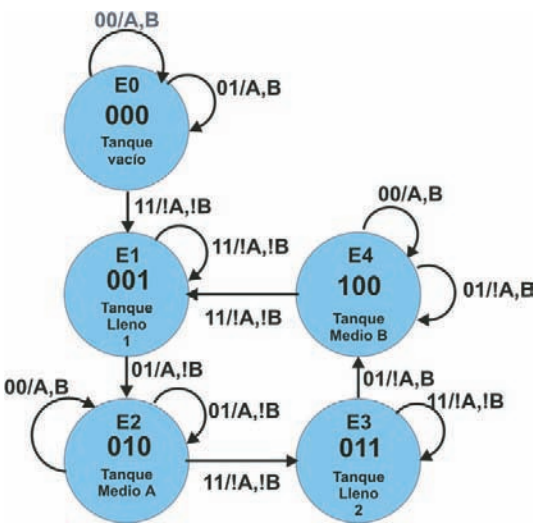
Salidas combinacionales BA y BB.
Salidas secuenciales Q2, Q1 y Q0.
Se requieren tres Flip Flops para cinco estados.



3. Asignación de valores a los estados:

Estado	Asignación		
	Q2	Q1	Q0
E0	0	0	0
E1	0	0	1
E2	0	1	0
E3	0	1	1
E4	1	0	0

4. Diagrama de transición, incluyendo los valores asignados a cada estado.



5. Tabla de estados.

Esta tabla es un resumen de lo descrito en el diagrama de transición.

Estado presente		Estado próximo S2, S1/AB			
		00	01	10	11
Tanque vacío	E0	E0/11	X	E0/11	E1/00
Tanque lleno 1	E1		X	E2/01	E1/00
Tanque medio A	E2	E3/11	X	E2/01	E4/00
Tanque lleno 2	E3		X	E3/01	E4/00
Tanque medio A	E4	E0/11	X	E5/10	E4/00

En la tabla de estados hay dos espacios no definidos, lo cual se debe a que en el diagrama de transición, en los estados E1 y E3, no está considerada la opción de entrada 00, tanque vacío, y, en condiciones normales de operación, no es posible pasar de tanque lleno a tanque vacío, sin pasar primero por tanque medio.

Para estos espacios se considera que no se pueden presentar, y es conveniente asignarles un valor de X.

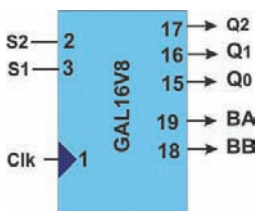
Estado presente		Estado próximo S2, S1/AB			
		00	**01**	**10**	**11**
Tanque vacío	**E0**	**E0/11**	**X**	**E0/11**	**E1/00**
Tanque lleno 1	**E1**	**X**	**X**	**E2/01**	**E1/00**
Tanque medio A	**E2**	**E3/11**	**X**	**E2/01**	**E4/00**
Tanque lleno 2	**E3**	**X**	**X**	**E3/01**	**E4/00**
Tanque medio A	**E4**	**E0/11**	**X**	**E5/10**	**E4/00**

Con la definición de las entradas y las salidas, la asignación de valores a los estados y la descripción del diagrama de transición es suficiente para formar un archivo en formato ABEL y resolver e implementar el sistema secuencial en un dispositivo lógico programable.

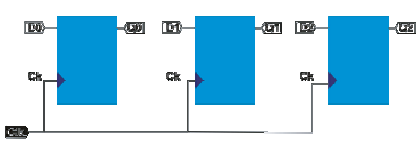
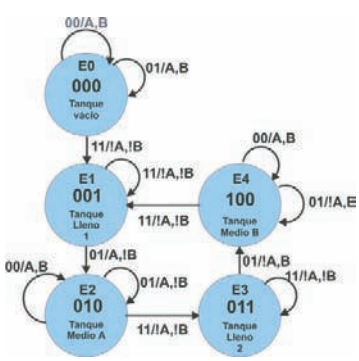
Partes del archivo en formato ABEL-HDL

1. Simplificación de las variables .c. y .x.
2. Definición de las entradas y salidas, así como de terminales.
3. Sincronización de los Flip Flops.
4. Asignación de valores a los estados.
5. Desarrollo de las salidas y estados próximos usando el comando State_Diagram.

Archivo en formato ABEL para la solución del problema

1.	MODULE dbcl “Es un sistema secuencial para el control del llenado de un tanque usando el modelo Mealy “Simplificación de la variable del pulso de reloj .c. y la .x. de la simulación c,x = .c.,.x.;	
2.	“Entradas Clk, S2, S1 pin 1,2,3; “Salidas Combinacionales BA, BB pin 19, 18 istype ‘com’; “Salidas Registradas Q2..Q0 pin 17,16,15 istype ‘REG’;	

3.	“Sincronización de los FF,s a un mismo pulso de reloj
	DECLARATIONS
	sreg=[Q0,Q1,Q2];
	EQUATIONS
	sreg.clk=Clk;	

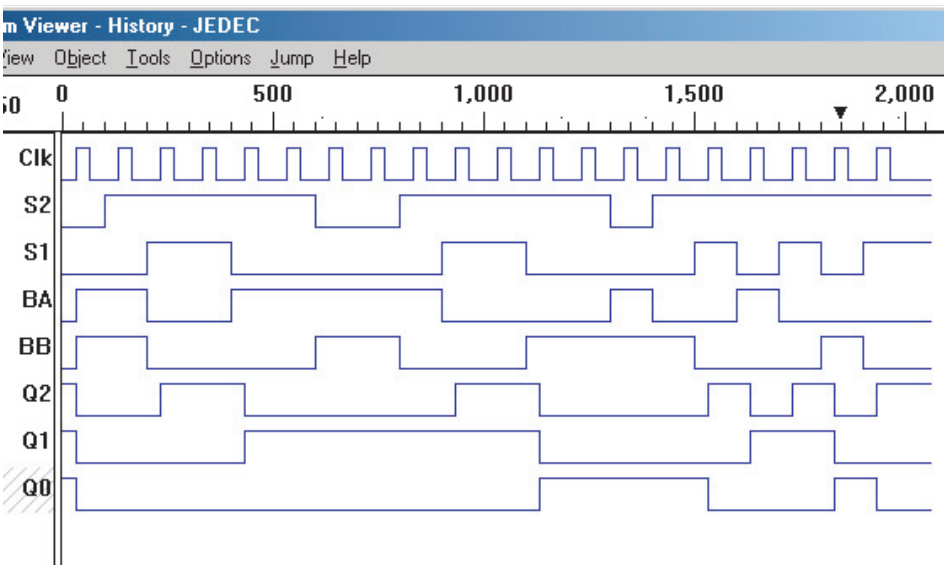
4.	“Asignación de valores a los estados DECLARATIONS E0=[0, 0, 0]; E1=[0, 0, 1]; E2=[0, 1, 0]; E3=[0, 1, 1]; E4=[1, 0, 0];	
5.	“Desarrollo de la salidas y estados próximos state_diagram sreg; STATE E0: IF !S2&!S1 Then E0 With {BA=1;BB=1;} IF S2&!S1 then E0 with {BA=1;BB=1;} IF S2& S1 then E1 with {BA=0;BB=0;} STATE E1: IF S2&!S1 then E2 with {BA=1;BB=0;} IF S2& S1 then E1 with {BA=0;BB=0;}	

```
STATE E2:
        IF !S2&!S1 then E2 with {BA=1;BB=1;}
        IF  S2&!S1 then E2 with {BA=1;BB=0;}
        IF  S2& S1 then E3 with {BA=0;BB=0;}
STATE E3:
        IF  S2&!S1 then E4 with
{BA=0;BB=1;}
        IF  S2& S1 then E3 with
{BA=0;BB=0;}
STATE E4:
        IF !S2&!S1 then E4 with {BA=1;BB=1;}
        IF  S2&!S1 then E4 with
{BA=0;BB=1;}
        IF  S2& S1 then E1 with
{BA=0;BB=0;}
End
```

Si se agrega al programa anterior la parte de simulación Test_Vector, con una secuencia de valores que prueben la mayoría de las acciones del sistema, obtendremos la simulación.

```
Test_Vectors
([Clk,S2,S1,Q2,Q1,Q0]->[BA,BB,Q2,Q1,Q0]);
[c,0,0,0,0,0]->[x,x,x,x,x];       “E0 tanque vacío E0, BA BB
[c,1,0,x,x,x]->[x,x,x,x,x];       “E0 tanque medio E0, BA BB
[c,1,1,x,x,x]->[x,x,x,x,x];       “E0 tanque medio E1
[c,1,1,x,x,x]->[x,x,x,x,x];       “E1 tanque lleno E1
[c,1,0,x,x,x]->[x,x,x,x,x];       “E1 tanque medio A E2, BA
[c,1,0,x,x,x]->[x,x,x,x,x];       “E2 tanque medio E2, BA
[c,0,0,x,x,x]->[x,x,x,x,x];       “E2 tanque vacío E2, BA BB
[c,0,0,x,x,x]->[x,x,x,x,x];       “E3 tanque vacío E2, BA BB
[c,1,0,x,x,x]->[x,x,x,x,x];       “E3 tanque medio E2 BA
[c,1,1,x,x,x]->[x,x,x,x,x];       “E2 tanque lleno E3
[c,1,1,x,x,x]->[x,x,x,x,x];       “E3 tanque lleno E3
[c,1,0,x,x,x]->[x,x,x,x,x];       “E3 tanque medio E4, BB
[c,1,0,x,x,x]->[x,x,x,x,x];       “E4 tanque medio E4, BB
[c,0,0,x,x,x]->[x,x,x,x,x];       “E4 tanque vacío E4, BA BB
[c,1,0,x,x,x]->[x,x,x,x,x];       “E4 tanque medio E0, BB
[c,1,1,x,x,x]->[x,x,x,x,x];       “E4 tanque lleno E1
[c,1,0,x,x,x]->[x,x,x,x,x];       “E1 tanque medio E2, BA
[c,1,1,x,x,x]->[x,x,x,x,x];       “E2 tanque lleno E3
[c,1,0,x,x,x]->[x,x,x,x,x];       “E3 tanque medioE4, BB
[c,1,1,x,x,x]->[x,x,x,x,x];       “E2 tanque lleno E3
```

Resultado de la simulación del Test_Vectors



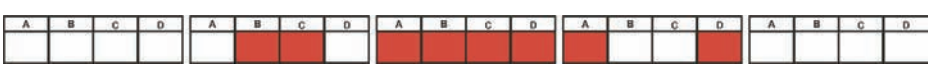
Ejemplo 9.4

Secuencia de luces

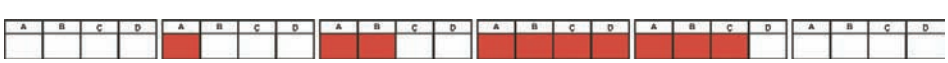
Objetivo particular. Durante el desarrollo de este ejemplo, se obtendrá el diseño de un circuito con un *display* con LED's destellantes. Dicho *display* tiene cuatro LED que encienden y apagan en una secuencia particular que dependerá de una señal de control X.

Especificaciones:

Secuencia A



Secuencia B



Si X = 0 Ocurrirá la secuencia "A" (los cuadros blancos indican que la luz está apagada, y los obscuros, que la luz está encendida).

Si X = 1 Sucederá la secuencia "B".

NOTA: *Cuando usted cambia el selector de secuencia X en medio de una secuencia, las luces continuarán con la secuencia actual hasta que se encuentre con un diseño de luces, que también esté en la otra secuencia. De ahí en adelante, la secuencia que fue seleccionada comenzará nuevamente con su correspondencia al nuevo valor de X.*

Por ejemplo, se supone que X = 0 y un selector X se hacen 1 en el momento que la secuencia de luces son (OFF-ON-ON-OFF) (secuencia "A"). La secuencia de la figura "A" continuará hasta el próximo diseño (ON-ON-ON-ON) que es también un diseño de la secuencia "B". De ahora en adelante la secuencia "B" continuará.

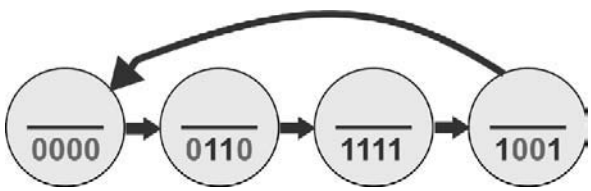
Trabajo solicitado

Diseñe el sistema secuencial usando la máquina de Moore.

Solución:

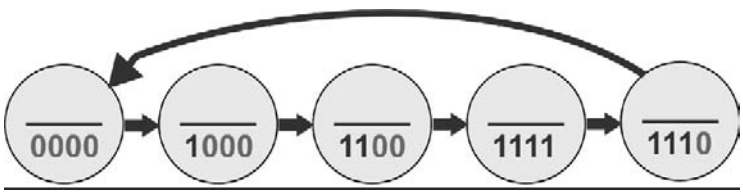
La secuencia A está compuesta por cuatro estados:

T	A	B	C	D
0	0	0	0	0
1	0	1	1	0
2	1	1	1	1
3	1	0	0	1
0	0	0	0	0



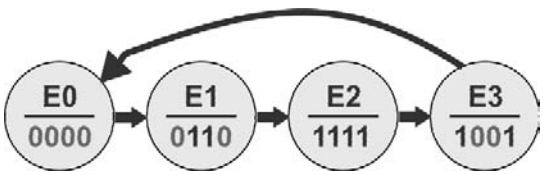
La secuencia B está formada por cinco estados descritos:

t	A	B	C	D
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	1
4	1	1	1	0
0	0	0	0	0

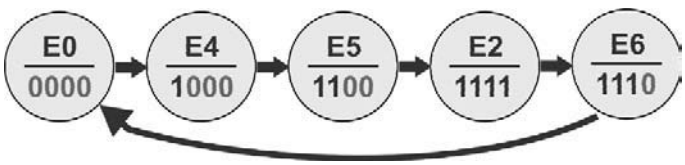


Comparando las secuencias se tiene que se repiten dos estados, donde todas las luces están apagadas (0, 0, 0, 0) y donde todas las luces están encendidas (1, 1, 1, 1), por lo que para este diseño se tendrían siete estados diferentes que se identificarán de la manera siguiente:

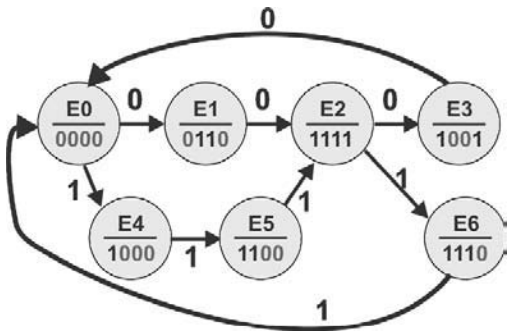
Secuencia A



Secuencia B

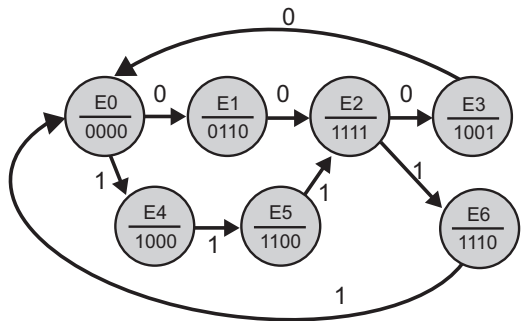


Tomando en cuenta las dos secuencias se enlazarían de la forma siguiente:



Es conveniente elaborar una tabla de estados para revisar si se tomaron en cuenta las posibles transiciones.

	X=0	X=1
E0	**E1**	**E4**
E1	**E2**	
E2	**E3**	**E6**
E3	**E0**	
E4		**E5**
E5		**E2**
E6		**E0**

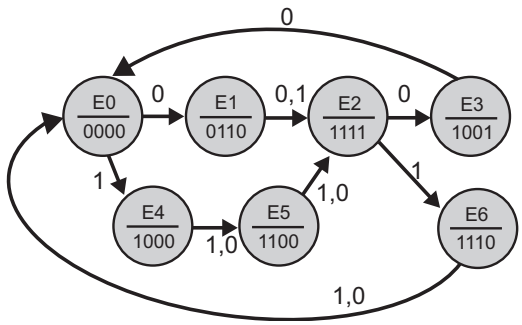


Observe que la tabla no está completa. Para determinar los estados faltantes hay que revisar una nota que dice lo siguiente:

Cuando usted cambia el selector de secuencia X en medio de una secuencia, las luces continuarán con la secuencia actual, hasta que se encuentre con un diseño de luces que también esté en la otra secuencia. De ahí en adelante, la secuencia que fue seleccionada comenzará nuevamente con su al nuevo valor de X.

Con lo anterior se concluye que sólo debe cambiar de secuencia en los estados E0 y E2, y en los demás estados deberá seguir al estado siguiente, independientemente del valor de la entrada.

	X = 0	X = 1
E0	**E1**	**E4**
E1	**E2**	**E2**
E2	**E3**	**E6**
E3	**E0**	**E0**
E4	**E5**	**E5**
E5	**E2**	**E2**
E6	**E0**	**E0**

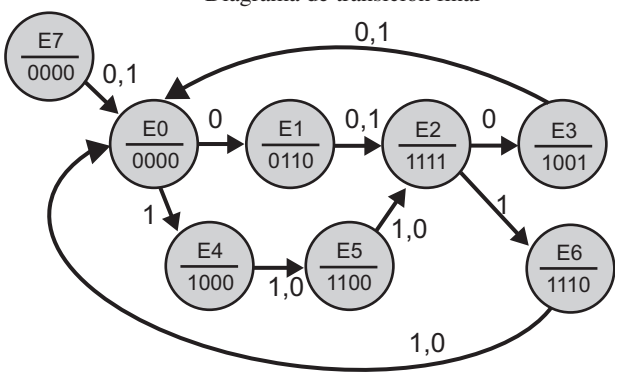


Para siete estados (de E0 a E6) se requieren por lo menos tres Flip Flops, con los cuales se podría generar hasta ocho estados. Es importante incluir un E7, cuyo estado siguiente, independientemente de la entrada, sea E0, de manera que todos los estados y las transiciones posibles sean tomadas en cuenta en el diseño y así asegurar que el funcionamiento del sistema.

Tabla final de estado siguiente

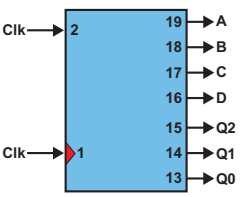
	X = 0	X = 1
E0	**E1**	**E4**
E1	**E2**	**E2**
E2	**E3**	**E6**
E3	**E0**	**E0**
E4	**E5**	**E5**
E5	**E2**	**E2**
E6	**E0**	**E0**
E7	**E0**	**E0**

Diagrama de transición final

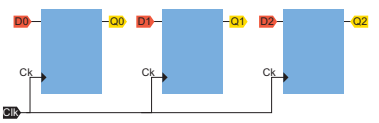


Archivo en formato ABEL-HDL

```
MODULE luc
"Entradas
Clk,X pin 1,2;
"salidas Combinacionales
A,B,C,D pin 19..16 istype 'com';
"salidas Registradas
Q2..Q0 pin 15..13 istype 'reg';
```

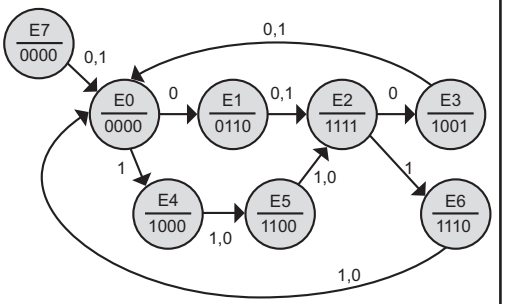



```
T=[Q2..Q0];
equations
T.clk=Clk;
```



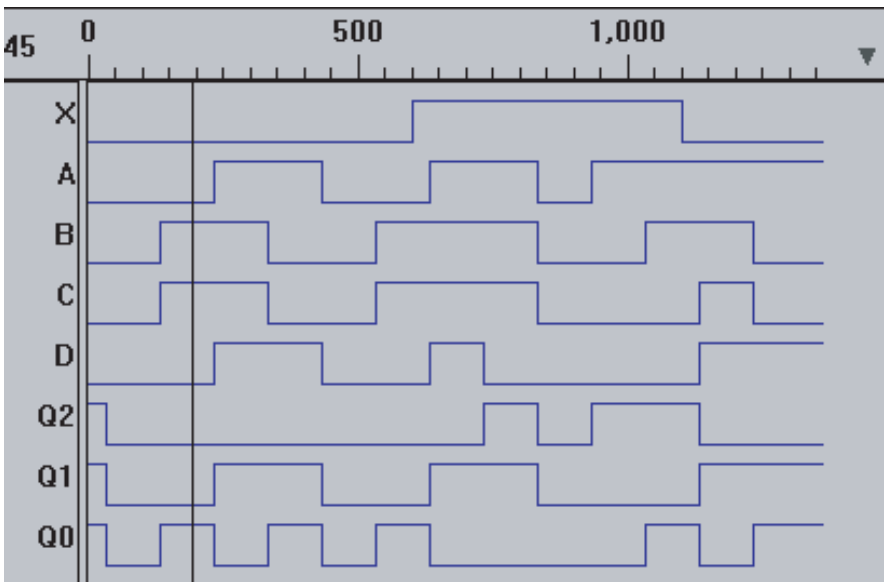

```
declarations
E0=[0,0,0];
E1=[0,0,1];
E2=[0,1,0];
E3=[0,1,1];
E4=[1,0,0];
E5=[1,0,1];
E6=[1,1,0];
E7=[1,1,1];   state_diagram T
```

```
State E0:
A=0;B=0;C=0;D=0;
IF X then E4 else E1;
State E1:
A=0;B=1;C=1;D=0;
Goto E2;
State E2:
A=1;B=1;C=1;D=1;
IF X then E6 else E3;
State E3:
A=1;B=0;C=0;D=1;
goto E0;
State E4:
A=1;B=0;C=0;D=0;
goto E5;
State E5:
A=1;B=1;C=0;D=0;
goto E2;
State E6:
A=1;B=1;C=1;D=0;
goto E0;
State E7:
A=0;B=0;C=0;D=0;
goto E0;
```



Para la simulación se agrega lo siguiente:

```
Test_Vectors
([Clk,X]->[Q2])
[.c.,0]->[.x.];
[.c.,0]->[.x.];
[.c.,0]->[.x.];
[.c.,0]->[.x.];
[.c.,0]->[.x.];
[.c.,0]->[.x.];
[.c.,1]->[.x.];
[.c.,1]->[.x.];
[.c.,1]->[.x.];
[.c.,1]->[.x.];
[.c.,1]->[.x.];
[.c.,0]->[.x.];
[.c.,0]->[.x.];
END
```



Ejemplo 9.5

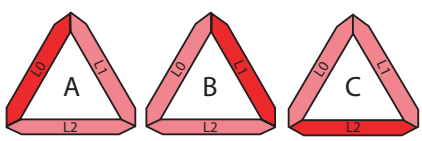
Señal de alerta

Diseñe un sistema secuencial usando la máquina de Moore como el control de una señal de alerta, la cual consta de tres luces en forma de triángulo llamadas L0, L1 y L2.

Se requieren dos secuencias diferentes que son seleccionadas por medio de un botón S, de manera que:

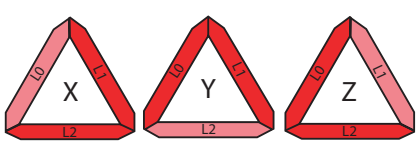
a) Ambas parten de condiciones iniciales en donde todas las lámparas están apagadas.

b) Si S = 0, ocurrirá la secuencia uno (A, B, C, A, B, C) repetidamente.



Secuencia Uno

c) Si S = 1, sucederá la secuencia dos (X, Y, Z, X, Y, Z) repetidamente.



Secuencia Dos

NOTA: *Cuando usted cambia el selector S en medio de una secuencia, las luces continuarán con la secuencia actual hasta terminar (C o Z) encendiendo todas las luces, posteriormente regresará a condiciones iniciales; de ahí en adelante seguirá con la secuencia que fue seleccionada, correspondiendo al nuevo valor de S.*

Solución:

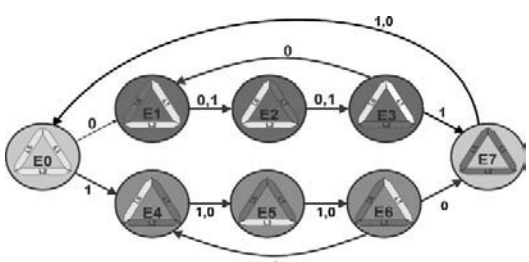
Para este diseño se tienen ocho eventos diferentes, descritos en la figura anterior, por lo cual se necesitan tres Flip Flops. El diagrama de transición se describe a continuación:

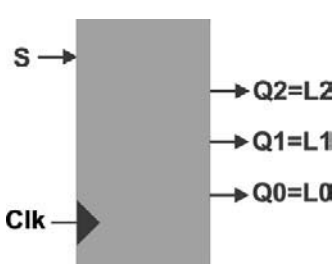
Tabla de estado siguiente

	Estado Siguiente	
	S=0	S=1
E0	**E1**	**E4**
E1	**E2**	**E2**
E2	**E3**	**E3**
E3	**E1**	**E7**
E4	**E5**	**E5**
E5	**E6**	**E6**
E6	**E7**	**E4**
E7	**E0**	**E0**

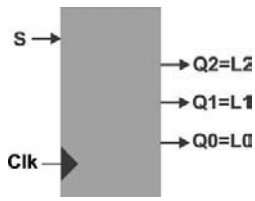
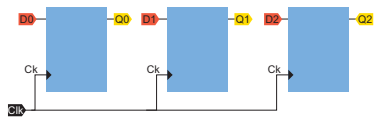
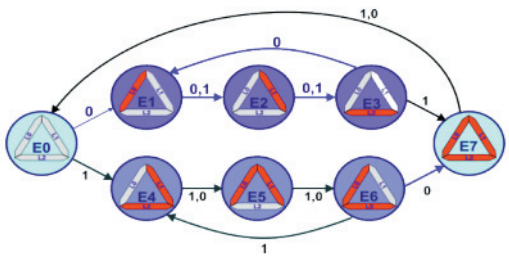
En este ejemplo, por medio de la asignación de valores a los estados, se evitan las salidas combinacionales, pero sólo si se hacen coincidir los valores de L2, L1 y L0 con Q2, Q1, Q0, como se muestra:

	Q2	Q1	Q0
E0	**0**	**0**	**0**
E1	**0**	**0**	**1**
E2	**0**	**1**	**0**
E3	**1**	**0**	**0**
E4	**1**	**1**	**0**
E5	**0**	**1**	**1**
E6	**1**	**0**	**1**
E7	**1**	**1**	**1**

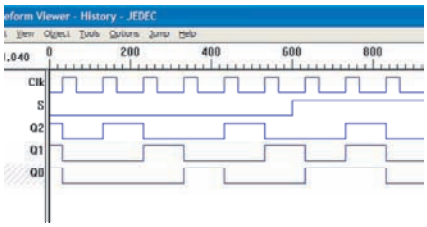




Archivo en formato ABEL-HDL

MODULE alerta **“Entradas** **Clk,S Pin 1,2;** **“salidas Registradas** **Q2..Q0 pin 19..17 istype ‘reg’;**	
“sincronización de los flip flops **Sinc=[Q2..Q0];** **Equations** **Sinc.Clk=Clk;**	
declarations **E0=[0,0,0];** **E1=[0,0,1];** **E2=[0,1,0];** **E3=[1,0,0];** **E4=[1,1,0];** **E5=[0,1,1];** **E6=[1,0,1];** **E7=[1,1,1];**	
State_diagram Sinc **State E0:** **If S then E4 else E1;** **State E1:** **goto E2;** **State E2:** **goto E3;** **State E3:** **If S then E7 else E1;** **State E4:** **goto E5;** **State E5:** **goto E6;** **State E6:** **If S then E4 else E7;** **State E7:** **goto E0;**	

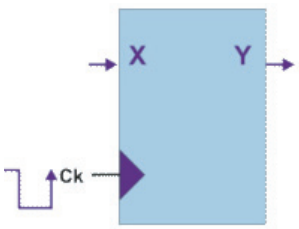
```
Test_vectors
([Clk,S]->[Q2,Q1,Q0])
[.c.,0]->[.x.,.x.,.x.];
[.c.,0]->[.x.,.x.,.x.];
[.c.,0]->[.x.,.x.,.x.];
[.c.,0]->[.x.,.x.,.x.];
[.c.,0]->[.x.,.x.,.x.];
[.c.,0]->[.x.,.x.,.x.];
[.c.,1]->[.x.,.x.,.x.];
[.c.,1]->[.x.,.x.,.x.];
[.c.,1]->[.x.,.x.,.x.];
[.c.,1]->[.x.,.x.,.x.];
[.c.,1]->[.x.,.x.,.x.];
[.c.,1]->[.x.,.x.,.x.];
[.c.,0]->[.x.,.x.,.x.];
[.c.,0]->[.x.,.x.,.x.];
[.c.,0]->[.x.,.x.,.x.];
[.c.,1]->[.x.,.x.,.x.];
[.c.,1]->[.x.,.x.,.x.];
END
```



9

Ejemplo 9.6

Detector de secuencia (máquina de Mealy)



Es un circuito con una entrada de datos X en serie y una o varias salidas, que indican si una o varias secuencias son correctas o no.

El detector tendrá una entrada X que estará sincronizada con la señal de reloj.

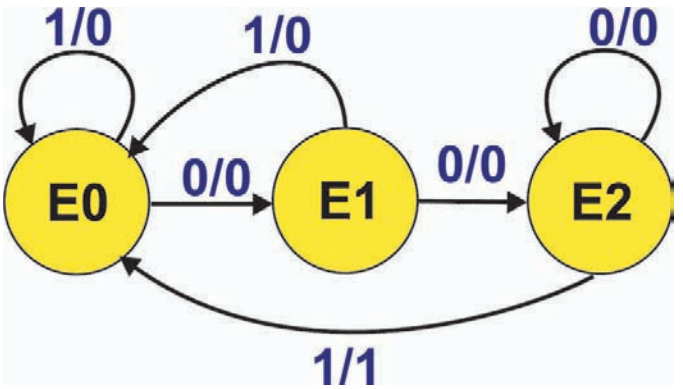
Diseño de un sistema secuencial que detecta la secuencia 001

A medida que pasen los pulsos de reloj, la entrada será 0 o 1, con lo que se "escribirá" una secuencia en binario (algo como 01100**001**1**001**0**001**10...).

Cada vez que el circuito detecte la secuencia 001, su salida se pondrá en alta y regresará a condiciones iniciales para detectar de nuevo la secuencia.

Diagrama de transición

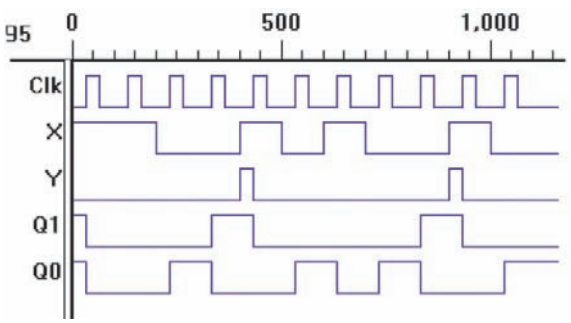
En el modelo Mealy la salida está asociada a la transición y está en función de la entrada y el estado como se indica a continuación:



Archivo en formato ABEL-HDL

```
MODULE dsec
“entradas
Clk, X pin 1,2;
“salida Combinacional
Y pin 19 istype ‘com’;
“Salida registradas
Q1,Q0 pin 18,17 istype ‘reg’;
S=[Q1,Q0];
Equations
S.clk=Clk;
declarations
E0=[0,0];
E1=[0,1];
E2=[1,0];
E3=[1,1];
State_diagram S
State E0:
If X then E0 else E1 With Y=0;
State E1:
If !X then E2 else E0 With Y=0;
State E2:
If !X then E2 else E0 With Y=1;
State E3:
```

```
goto E0;
Test_Vectors
([Clk,X]->Y)
[.c.,1]->.x.;
[.c.,1]->.x.;
[.c.,0]->.x.;
[.c.,0]->.x.;
[.c.,1]->.x.;
[.c.,0]->.x.;
[.c.,1]->.x.;
[.c.,0]->.x.;
[.c.,0]->.x.;
[.c.,1]->.x.;
[.c.,0]->.x.;
END
```



Ejemplo 9.7

Detector de secuencia (máquina de Moore)

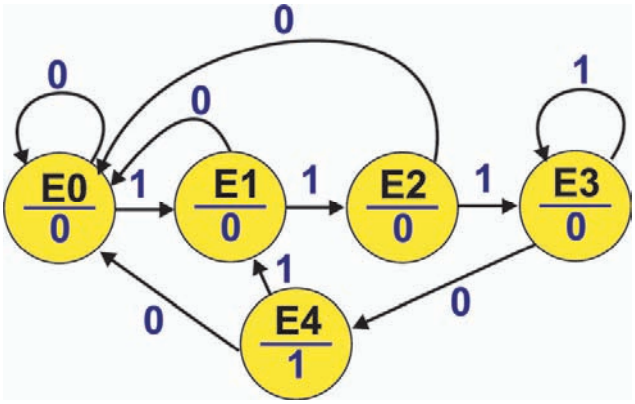
Diseño de un sistema secuencial que detecta la secuencia 1110

A medida que pasen los pulsos de reloj, la entrada será 0 o 1, con lo que se "escribirá" una secuencia en binario (algo como 01100**1110**1**1110**1000110...).

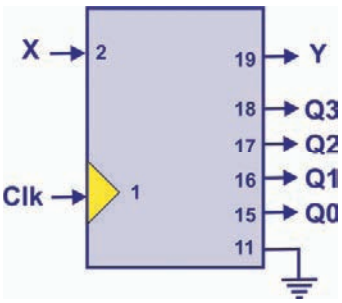
El detector tendrá una entrada X que estará sincronizada con la señal de reloj.

Cada vez que el circuito detecte la secuencia **1110**, su salida se pondrá en alta y regresará a condiciones iniciales para detectar de nuevo la secuencia.

En el modelo de Moore, la salida está asociada al estado, de modo que después de detectar la secuencia correcta 1110 en E4, la entrada siguiente es considerada como el primer dato de una nueva secuencia; por esa razón, si en E4, la entrada X = 1, el estado siguiente es el E1.



Para cinco estados se requieren por lo menos de cuatro Flip Flops (Q3 a Q0) representados como salidas en el siguiente diagrama de bloques.

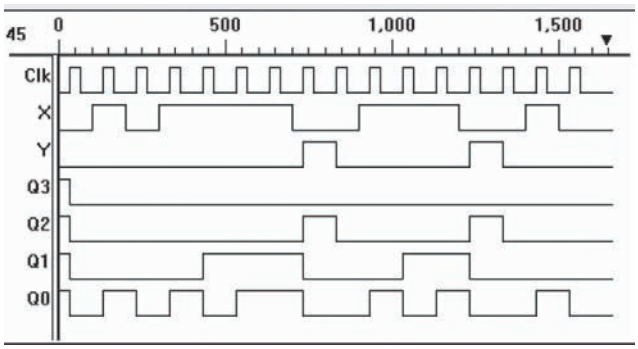


Archivo en formato ABEL-HDL

```
“Entradas
Clk, X pin 1,2;
“salida combinacional
Y pin 19 istype ‘com’;
“salidas de los Flip Flops
Q3..Q0 pin 18..15 istype ‘reg’;
S=[Q3..Q0];
equations
S.clk=Clk;
“asignacion de valores a los estados
declarations
E0=[0,0,0,0];
E1=[0,0,0,1];
E2=[0,0,1,0];
E3=[0,0,1,1];
E4=[0,1,0,0];
E5=[0,1,0,1];
E6=[0,1,1,0];
E7=[0,1,1,1];
E8=[1,0,0,0];
E9=[1,0,0,1];
E10=[1,0,1,0];
E11=[1,0,1,1];
E12=[1,1,0,0];
E13=[1,1,0,1];
E14=[1,1,1,0];
E15=[1,1,1,1];
State_diagram S
State E0:
Y=0;
If X then E1 else E0;
State E1:
Y=0;
If X then E2 else E0;
State E2:
Y=0;
If X then E3 else E0;
State E3:
Y=0;
If !X then E4 else E3;
State E4:
Y=1;
If X then E1 else E0;
State E5:
Y=0;
goto E0;
State E6:
Y=0;
goto E0;
State E7:
Y=0;
goto E0;
State E8:
Y=0;
goto E0;
State E9:
Y=1;
goto E0;
State E10:
Y=0;
goto E0;
State E11:
Y=0;
goto E0;
State E12:
Y=0;
goto E0;
State E13:
Y=0;
goto E0;
State E14:
Y=0;
goto E0;
State E15:
Y=0;
goto E0;

END
```

Para comprobar el funcionamiento se incluye la simulación:

```
Test_Vectors
([Clk,X]->Y)
[.c.,0]->.x.;
[.c.,1]->.x.;
[.c.,0]->.x.;
[.c.,1]->.x.;
[.c.,1]->.x.;
[.c.,1]->.x.;
[.c.,1]->.x.;
[.c.,0]->.x.;
[.c.,0]->.x.;
[.c.,1]->.x.;
[.c.,1]->.x.;
[.c.,1]->.x.;
[.c.,0]->.x.;
[.c.,0]->.x.;
[.c.,1]->.x.;
[.c.,0]->.x.;
```



Ejemplo 9.8

Diseñe un sistema secuencial que controle las luces traseras de un automóvil. Se tienen tres entradas que son freno **F**, direccional derecha **DD**, y direccional izquierda **DI**, además de seis luces de salida llamadas **IZ0**, **IZ1** y **IZ2**, **DE0**, **DE1** y **DE2.**

Al oprimir el freno (**F**), se deberán encender todas las luces, sin importar si están activadas las direccionales.

	IZ2	IZ1	IZ0	DE0	DE1	DE2
F	■	■	■	■	■	■

La direccional derecha tiene tres estados que se repiten en los tiempos T0, T1, T2, T3 y de nuevo T0, mientras la palanca esté en éste en **DD**.

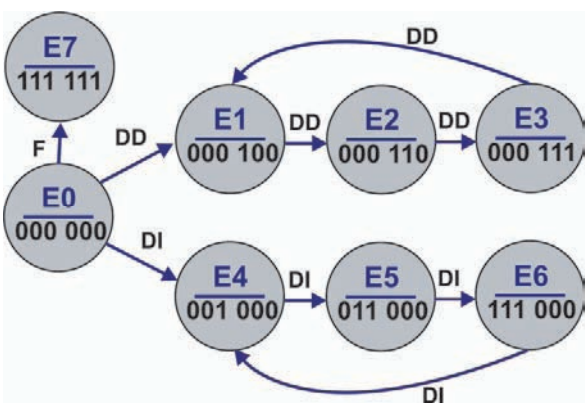
	IZ2	IZ1	IZ0	DE0	DE1	DE2
T0	□	□	□	□	□	□
T1	□	□	□	■	□	□
T2	□	□	□	■	■	□
T3	□	□	□	■	■	■

La direccional izquierda también tiene tres estados que se repiten en los tiempos T0, T1, T2, T3 y de nuevo T0, mientras la palanca esté en éste en **DI**.

	IZ2	IZ1	IZ0	DE0	DE1	DE2
T0	□	□	□	□	□	□
T1	□	□	■	□	□	□
T2	□	■	■	□	□	□
T0	■	■	■	□	□	□

Solución:

El diagrama de transición propuesto es el siguiente:



Es conveniente elaborar una tabla de estado siguiente para definir completamente las transiciones:

	Estado Siguiente				
F	0	0	0	0	1
DD	0	0	1	1	X
DI	0	1	0	1	X
E0	E0	E4	E1	X	E7
E1	E0	E4	E2	X	E7
E2	E0	E4	E3	X	E7
E3	E0	E4	E1	X	E7
E4	E0	E5	E1	X	E7
E5	E0	E6	E1	X	E7
E6	E0	E4	E1	X	E7
E7	E0	E4	E1	X	E7

Para la implementación de este diseño, se requieren tres salidas para los Flip Flops y seis para las luces; además, el GAL16V8 sólo tiene ocho salidas disponibles, por lo que se puede utilizar el GAL22V10 o ajustar una salida de los Flip Flops como salida de las luces. Para hacer el ajuste se analizará la tabla de estados y sus salidas.

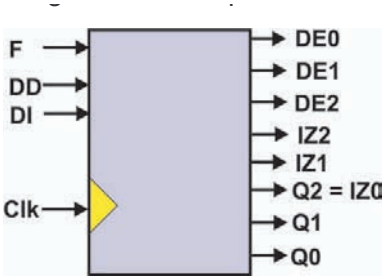
	IZ2	IZ1	IZ0	DE0	DE1	DE2
E0	0	0	0	0	0	0
E1	0	0	0	1	0	0
E2	0	0	0	1	1	0
E3	0	0	0	1	1	1
E4	0	0	1	0	0	0
E5	0	1	1	0	0	0
E6	1	1	1	0	0	0
E7	1	1	1	1	1	1

En la tabla anterior se observa que tanto IZ0 como DE0 tienen cuatro ceros y cuatro unos, por lo que es posible que cualquiera de las salidas de los Flip Flops pueda reemplazar su función. Para que esto suceda hay que ajustar la asignación de valores a los estados, de manera que una de las salidas de los Flip Flops cumpla con los valores de la salida combinacional.

Se puede sustituir la Q2 por IZ0.

	Q2	Q1	Q0
E0	0	0	0
E1	0	0	1
E2	0	1	0
E3	0	1	1
E4	1	0	0
E5	1	0	1
E6	1	1	0
E7	1	1	1

Diagrama de bloques



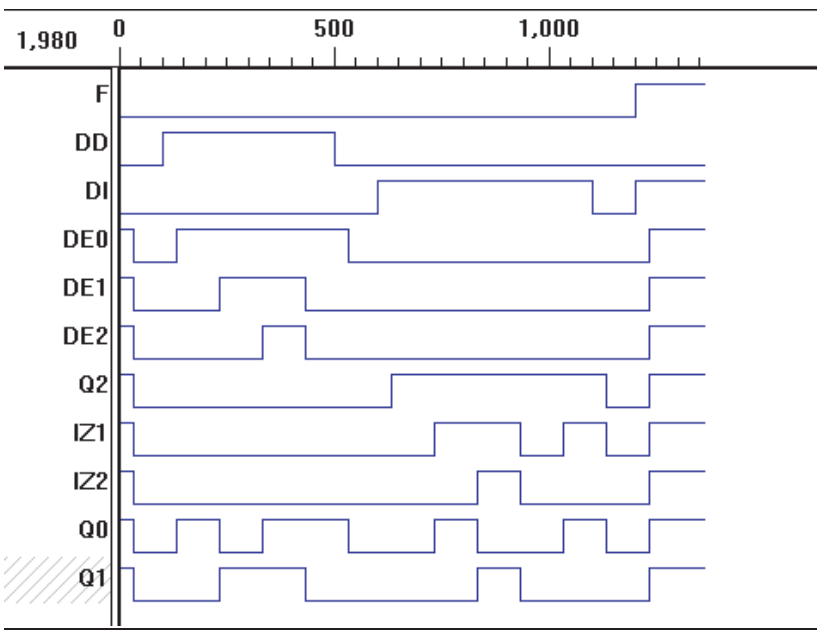
Archivo en formato ABEL-HDL

```
MODULE ltra
“Entradas
Clk,F,DD,DI pin 1..4;
“salidas Combinacionales
DE0,DE1,DE2,IZ1,IZ2 pin 19..15 istype
‘com’;
“salidas de los Flip Flops
Q2..Q0 pin 14..12 istype ‘reg’;
Sx=[Q2..Q0];
Equations
Sx.clk=Clk;
Declarations
E0=[0,0,0];
E1=[0,0,1];
E2=[0,1,0];
E3=[0,1,1];
E4=[1,0,0];
E5=[1,0,1];
E6=[1,1,0];
E7=[1,1,1];
 STATE_DIAGRAM Sx
STATE E0:
DE0=0;DE1=0;DE2=0;IZ1=0;IZ2=0;
IF !F&!DD&!DI then E0;
IF !F&!DD&DI then E4;
IF !F&DD&!DI then E1;
IF !F&DD&DI then E0;
IF F then E7;
STATE E1:
DE0=1;DE1=0;DE2=0;IZ1=0;IZ2=0;
IF !F&!DD&!DI then E0;
IF !F&!DD&DI then E4;
IF !F&DD&!DI then E2;
IF !F&DD&DI then E0;
IF F then E7;
STATE E2:
DE0=1;DE1=1;DE2=0;IZ1=0;IZ2=0;
IF !F&!DD&!DI then E0;
IF !F&!DD&DI then E4;
IF !F&DD&!DI then E3;
IF !F&DD&DI then E0;
IF F then E7;
STATE E3:
DE0=1;DE1=1;DE2=1;IZ1=0;IZ2=0;
IF !F&!DD&!DI then E0;
IF !F&!DD&DI then E4;
IF !F&DD&!DI then E1;
IF !F&DD&DI then E0;
IF F then E7;
STATE E4:
DE0=0;DE1=0;DE2=0;IZ1=0;IZ2=0;
IF !F&!DD&!DI then E0;
IF !F&!DD&DI then E5;
IF !F&DD&!DI then E1;
IF !F&DD&DI then E0;
IF F then E7;
STATE E5:
DE0=0;DE1=0;DE2=0;IZ1=1;IZ2=0;
IF !F&!DD&!DI then E0;
IF !F&!DD&DI then E6;
IF !F&DD&!DI then E1;
IF !F&DD&DI then E0;
IF F then E7;
STATE E6:
DE0=0;DE1=0;DE2=0;IZ1=1;IZ2=1;
IF !F&!DD&!DI then E0;
IF !F&!DD&DI then E4;
IF !F&DD&!DI then E1;
IF !F&DD&DI then E0;
IF F then E7;
STATE E7:
DE0=1;DE1=1;DE2=1;IZ1=1;IZ2=1;
IF !F&!DD&!DI then E0;
IF !F&!DD&DI then E4;
IF !F&DD&!DI then E1;
IF !F&DD&DI then E0;
IF F then E7;
```

NOTA: *En el caso en donde están activadas las dos direccionales DD, DI (IF !F&DD-&DI then E0;), mecánicamente no es posible por tratarse de una palanca de posición de tres posiciones, por lo que es conveniente definir como estado siguiente el estado inicial E0.*

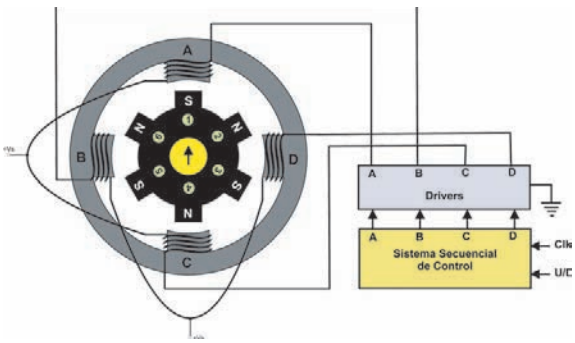
Para comprobar su funcionamiento se realiza la simulación:

```
Test_Vectors
([Clk,F,DD,DI]->[Q2,Q1,Q0])
[.c.,0,0,0]->[.x.,.x.,.x.];
[.c.,0,1,0]->[.x.,.x.,.x.];
[.c.,0,1,0]->[.x.,.x.,.x.];
[.c.,0,1,0]->[.x.,.x.,.x.];
[.c.,0,1,0]->[.x.,.x.,.x.];
[.c.,0,0,0]->[.x.,.x.,.x.];
[.c.,0,0,1]->[.x.,.x.,.x.];
[.c.,0,0,1]->[.x.,.x.,.x.];
[.c.,0,0,1]->[.x.,.x.,.x.];
[.c.,0,0,1]->[.x.,.x.,.x.];
[.c.,0,0,1]->[.x.,.x.,.x.];
[.c.,0,0,0]->[.x.,.x.,.x.];
[.c.,1,0,1]->[.x.,.x.,.x.];
END
```



Ejemplo 9.9

Secuenciador para un motor de pasos



Un motor de pasos (*Steping Motor*) es un motor de corriente continua, que realiza avances angulares constantes en ambos sentidos de rotación. Los motores de paso son fundamentalmente diferentes de los demás motores de CD, ya que no tienen escobillas ni conmutador mecánico; en su lugar, la acción de conmutación necesaria para rotación es lograda por señales externas, el rotor no tiene devanado de armadura, sólo imanes permanentes salientes como se muestra en la figura.

En algunos modelos, la flecha del motor avanza en el giro a 200 pasos por revolución ($360^{\circ}/200 = 1.8^{\circ}$ por paso).

Para manejar el motor de pasos, se usa una interfase que consta de un sistema secuencial y un *driver*, o manejador de potencia en la salida, que tenga la capacidad de conducir la corriente necesaria en las bobinas del motor de pasos.

Las señales que recibe esta interfase son:

Clk: es una señal activada por un flanco positivo (transición positiva), que le indica a la interfaz que rote al motor un sólo paso, esta entrada debe estar al menos activa por 20 microsegundos.

U/D: con esta entrada se determina el sentido de la rotación; si es uno, se rota en sentido de las manecillas del reloj; si es cero, se rota en sentido contrario de las manecillas del reloj.

Diseñe un secuenciador bidireccional para un motor de cuatro pasos usando el modelo de la máquina de Moore con las siguientes características:

1. Una entrada que controle el sentido de giro **U/D**.
2. El sistema deberá contar con una entrada que controle el paso **Clk**.
3. Cuatro salidas, llamadas A, B, C y D, una para cada bobina.

La secuencia de funcionamiento para la rotación a favor de las manecillas del reloj con U/D = 1 se presenta en la siguiente tabla:

Paso	A	B	C	D	Estado
1	1	0	1	0	E0
2	1	0	0	1	E1
3	0	1	1	0	E2
4	1	0	1	0	E3
1	1	0	1	0	E0

La secuencia de funcionamiento para la rotación en contra de las manecillas del reloj con U/D= 0 se señala en la siguiente tabla:

Paso	A	B	C	D	Estado
1	1	0	1	0	E0
2	1	0	1	0	E3
3	0	1	1	0	E2
4	1	0	0	1	E1
1	1	0	1	0	E0

En cualquiera de los cuatro estados, el sistema podrá cambiar el sentido del giro como lo muestra el diagrama de transición.

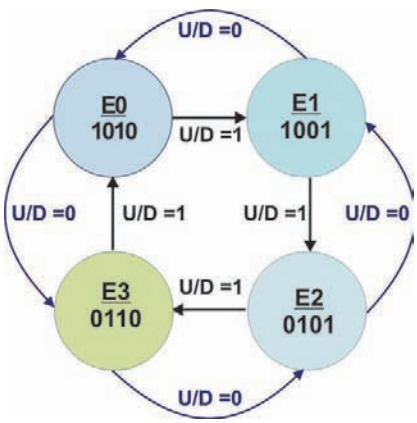
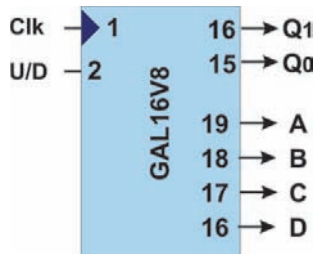


Diagrama de entradas y salidas



Asignación de valores a los estados

Estado	Q1	Q0	Salida A, B, C, D
E0	**0**	**0**	**1010**
E1	**0**	**1**	**1001**
E2	**1**	**0**	**0101**
E3	**1**	**1**	**0110**

Tabla de estados

Estado presente	Estado próximo		Salida A, B, C, D
	U/D =0	U/D =1	
E0	**E3**	**E1**	**1010**
E1	**E0**	**E2**	**1001**
E2	**E1**	**E3**	**0101**
E3	**E2**	**E0**	**0110**

Archivo en formato ABEL

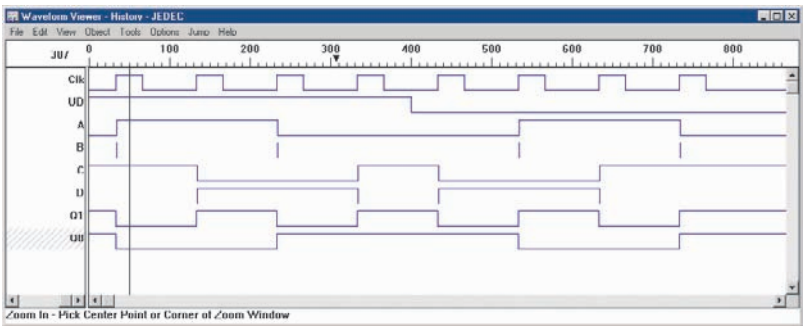
```
MODULE stepp
“Secuenciador bidireccional para un motor de pasos
c,x=.c.,.x.;
“Entradas
    Clk, UD pin 1,2;
“Salidas Combinacionales
    A,B,C,D pin 15..12 istype ‘com’;
“Salidas registradas
    Q1,Q0 pin 17,16 istype ‘reg’;
```

```
“Sincronización de los FF,s a un mismo pulso de reloj
DECLARATIONS
    sreg=[Q0,Q1];
EQUATIONS
    sreg.clk=Clk;
“Asignación de valores a los estados
DECLARATIONS
E0=[ 0, 0];
    E1=[ 0, 1];
    E2=[ 1, 0];
    E3=[ 1, 1];
state_diagram sreg;
STATE E0:
A=1;B=0;C=1;D=0;
 if UD then E1 else E3;
STATE E1:
A=1;B=0;C=0;D=1;
 if UD then E2 else E0;
STATE E2:
A=0;B=1;C=0;D=1;
 if UD then E3 else E1;
STATE E3:
A=0;B=1;C=1;D=0;
 if UD then E0 else E2;
END
```

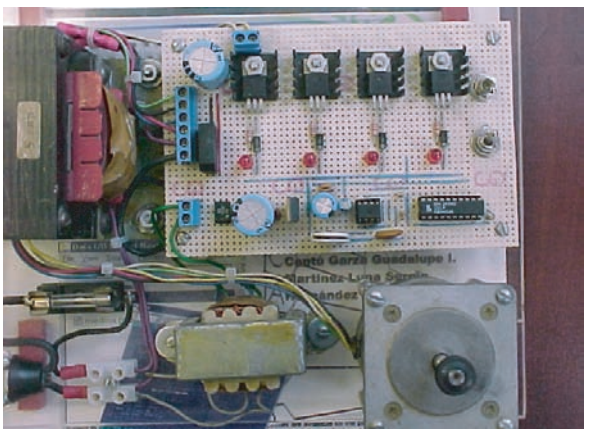
Al agregar al archivo anterior la parte de Test_Vector, se obtiene la simulación que se muestra a continuación:

```
Test_Vectors
([Clk,UD,Q1,Q0]->[A,B,C,D,Q1,Q0])
 [c,1,0,0]->[x,x,x,x,x,x];”Partiendo del estado E0 giro UD=1
 [c,1,x,x]->[x,x,x,x,x,x];
 [c,1,x,x]->[x,x,x,x,x,x];
 [c,1,x,x]->[x,x,x,x,x,x];
 [c,0,x,x]->[x,x,x,x,x,x];”Cambio de giro
 [c,0,x,x]->[x,x,x,x,x,x];
 [c,0,x,x]->[x,x,x,x,x,x];
 [c,0,x,x]->[x,x,x,x,x,x];
```

Diagrama de tiempos



Circuito secuenciador para un motor de pasos ya implementado



Problemas propuestos

1. Diseñe uno de los siguientes sistemas secuenciales y obtenga:

a) Diagrama de transición.
b) La cantidad de Flip Flops.
c) Los valores a los estados.
d) Diagrama de bloques (entradas y salidas).
e) Tabla de estados.
f) Archivo en formato ABEL-HDL.
g) Implementación en un dispositivo lógico programable (GAL16V8).

2. Tres bombas.

Diseñe un sistema secuencial usando la máquina de Moore para el llenado de un tanque con las siguientes características:

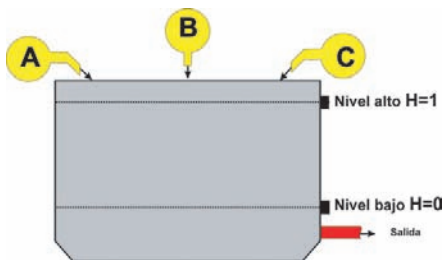
El sistema consta de tres bombas llamadas **"A"**, **"B"** y **"C"** y un sensor de nivel **"H"** que indica con **H = 1** tanque lleno, y con **H = 0** tanque vacío, el sistema deberá de trabajar bajo la siguiente secuencia:

a) Partiendo de que el tanque está vacío, el llenado deberá iniciarse con la bomba **"A"** hasta llenar el tanque (H = 1) y entonces desconectarlo.

b) Si de nuevo se vacía el tanque (H = 0), el llenado deberá hacerse con la bomba **"B"** hasta llenar el tanque y proceder a desconectarlo.

c) Si una vez más se vacía el tanque (H = 0), el llenado deberá realizarse con la bomba **"C"** hasta llenar el tanque y desconectarlo.

d) Si de nuevo se vacía el tanque (H = 0), el llenado deberá hacerse con la bomba **"A"** y, así sucesivamente, con la finalidad de que las tres bombas alternen su funcionamiento.



3. Llave electrónica (alarma).

El sistema cuenta con cuatro botones de entrada llamados A, B, C y D.

Se requieren tres salidas, **abrir puerta** (AP), **alarma** (AL) y **condiciones iniciales** (CI).

La salida condiciones iniciales, cuando es igual a uno (CI = 1), indica que el sistema está listo para aceptar un código de entrada.

La salida abrir puerta deberá valer uno (AP = 1) solamente cuando haya validado el código de entrada.

La salida alarma será uno (AL=1) solamente cuando el código de entrada no es el adecuado.

Funcionamiento:

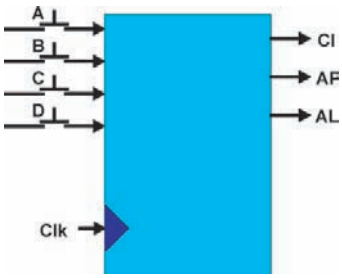
a) Partiendo de condiciones iniciales (CI= 1), si se oprimen los botones en secuencia **A**, **C**, **D**, **B** (uno a la vez), el sistema deberá activar la señal de abrir puerta (AP = 1).

b) Una vez abierta la puerta, con cualquier botón que se presione, la puerta se cerrará (AP = 0) y el sistema regresará a condiciones iniciales.

c) Con cualquier secuencia, diferente de **A**, **C**, **D, B**, que oprima el sistema, se activará una alarma (AL = 1).

d) Una vez activada la alarma, para desactivarla (AL= 0) y regresar a condiciones iniciales (CI = 1), se deberá oprimir la secuencia **D**, **B**, **C**.

El sistema deberá contar con una entrada de sincronía de los Flip Flops (Clk).

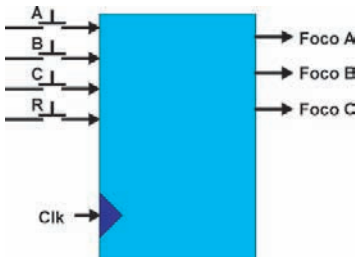


Diseñe el sistema secuencial usando la máquina de Moore.

4. Concurso del Jeopardy.

Diseñe un sistema secuencial, usando la máquina de Moore, que contenga cuatro botones de entrada llamados **A**, **B**, **C** y **R**, y tres salidas denominados **Foco A**, **Foco B**, **Foco C**.

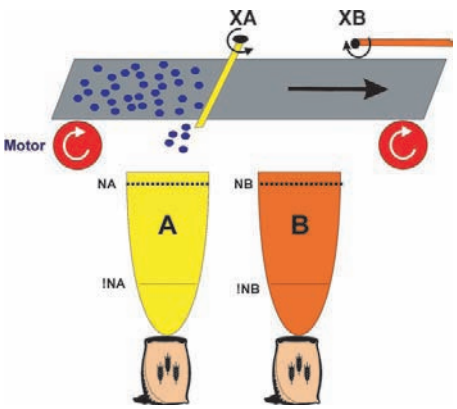
El sistema secuencial deberá indicar, por medio de un foco (ya sea **Foco A**, **Foco B** o **Foco C**), cuál de los tres participantes en un concurso de preguntas y respuestas es el primero en oprimir el botón (**Botón A**, **Botón B** o **Botón C**); además contendrá un cuarto botón (**Botón R**) para que el conductor del programa, una vez terminada la respuesta, regrese el sistema a condiciones iniciales (**Focos apagados**).



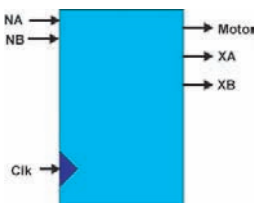
Considere que con cuatro entradas para cada estado existen 16 posibles opciones de estado siguiente.

5. Tolvas de ensacado.

Se desea mantener llenas de material dos tolvas de ensacado (llenado se sacos) llamadas **A** y **B**, las cuales son alimentadas por una banda transportadora que pasa por encima de éstas, por la que, a través de desviadores del material (**XA** y **XB**) siguen el orden en que se vaciarán los sacos, de acuerdo a los sensores de nivel alto y bajo de cada tolva (**NA**, **¡NA**, **NB**,**¡NB**).



NOTA: *En el caso de que las tolvas estén llenas, el motor de la banda transportadora deberá parar y arrancar cuando cualquiera de las tolvas se vacíe.*



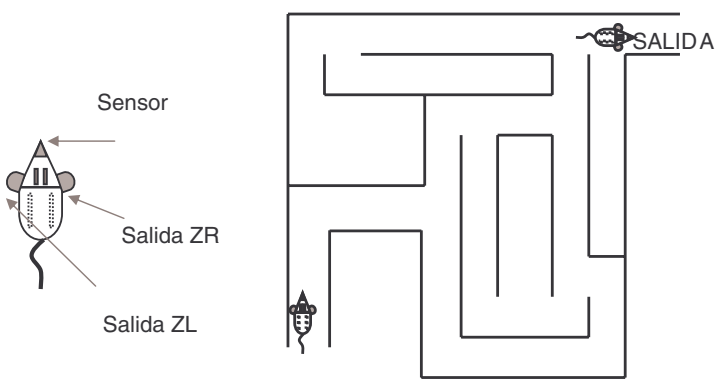
Diseñe el sistema secuencial usando la máquina de Moore.

6. Laberinto.

Diseñe el "cerebro" de un ratón. El objetivo es dotar al aparato con la suficiente inteligencia para que consiga salir de un laberinto tan rápido como sea posible.

Especificaciones:

El laberinto se muestra en la siguiente figura:



El ratón deberá tener las siguientes habilidades o capacidades: *a*) Un sensor sobre la nariz que detecte una pared enfrente de él al ser tocada. *b*) Girar a la izquierda o a la derecha hasta tocar la pared, sin importar lo largo o extenso que sea el recorrido. De esta forma, el ratón tiene dos señales de salida llamadas ZL (giro hacia la izquierda) y ZR (giro hacia la derecha), aunque sólo puede efectuar una de estas acciones a la vez (o ninguna, cuando el ratón vaya hacia delante).

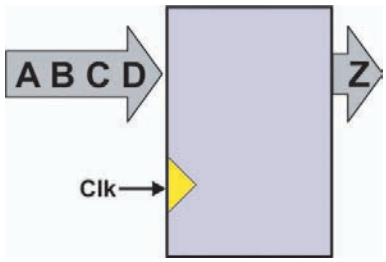
NOTA: *Después de un vistazo rápido al laberinto, ha notado una manera rápida para salir de él, es por medio de giros a la izquierda y a la derecha. Así que cuando el ratón alcance una pared, lo primero que hará será girar a la derecha; la próxima vez que se encuentre con la pared, lo que hará será girar a la izquierda, la siguiente vez girará a la derecha, etc., hasta que llegue a la salida. Verifique que esta estrategia funcione, para lo cual puede dibujar el camino que seguirá el ratón.*

Diseñe el sistema secuencial usando la máquina de Moore.

7. Diseñe un circuito secuencial síncrono con las siguientes características:

a) Una sola entrada y una sola salida.
b) Que reconozca números decimales codificados en binario del 0 al 9 (BCD).
c) La salida sea uno cuando la secuencia es diferente a un número en BCD.
d) El bit más significativo (D) va a la cabeza de la secuencia.

D	C	B	A	Z
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

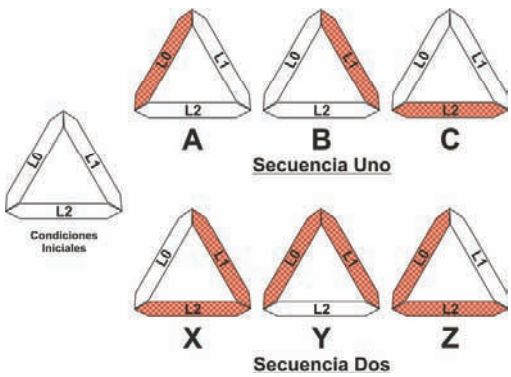


8. Control de señal de alerta.

Diseñe un sistema secuencial usando la máquina de Moore como control de una señal de alerta que consta de tres luces, en forma de triángulo llamadas $\mathbf{L_0}$, $\mathbf{L_1}$ y $\mathbf{L_2}$ en dos diferentes secuencias, las cuales parten de condiciones iniciales en donde todas las lámparas están apagadas y dependen de una señal de entrada **S** como selector, de modo que:

a) Si **S=0** ocurrirá la *secuencia Uno* (A, B, C, A, B, C, etc.).

b) Si **S=1** se originará la *secuencia Dos* (X, Y, Z, X, Y, Z, etc.), ver figura.



Nota: *Cuando usted cambia el selector de secuencia **S** en medio de una secuencia, las luces continuarán con la secuencia actual hasta terminar (C o Z) y pasará a condiciones iniciales; de ahí en adelante seguirá con la secuencia que fue seleccionada para el nuevo valor de **S**.*

Por ejemplo,

se asume que **S** = 0 y un selector **S** se hace 1 en el momento que la secuencia de luces es (A = 0, B = 1 y C = 0), correspondiente a la condición B de la secuencia ***Uno***, continuará hasta C y de ahí a condiciones iniciales (A = 0, B = 0 y C = 0), a partir de ese instante seguirá con la secuencia ***Dos***.

9. Barreras de seguridad de una vía de ferrocarril

En una vía de ferrocarril, con tráfico en ambos sentidos, corta una carretera en donde se colocan barreras activadas por una salida Z a través de un sistema secuencial (ver figura inferior).

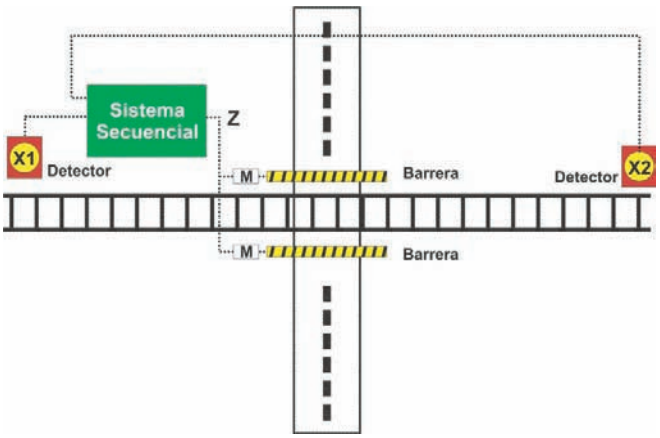
A 500 m del punto de cruce se colocan detectores (X1 y X2 respectivamente) en ambas direcciones. El estado inicial es Z = 0, este valor debe pasar al estado uno (Z =1) cuando se acerca un ferrocarril, en cualquier sentido; el cual, a los 500 m del cruce debe volver al estado cero, esto sucede justo cuando el último vagón se aleja más de dicha distancia, independientemente de la longitud del ferrocarril.

Considere que no está permitido hacer maniobras, es decir, no hay cambio de dirección, y que sólo pasa un tren a la vez.

Las posibilidades que pueden presentarse son:

a) Tren corto de X1 a X2 (tren menor a 1,000 metros).
b) Tren corto de X2 a X1 (tren menor a 1,000 metros).
c) Tren largo de X1 a X2 (tren mayor a 1,000 metros).
d) Tren largo de X2 a X1 (tren mayor a 1,000 metros).

Diseñe el sistema secuencial usando la máquina de Moore.



10. Resuelva el Ejemplo 9.3 de esta práctica utilizando la máquina de Moore.

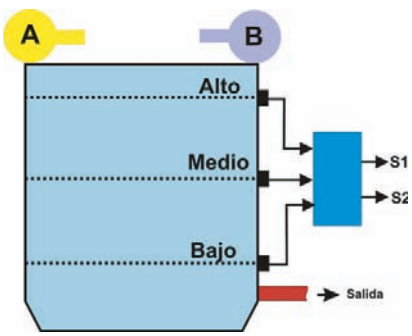
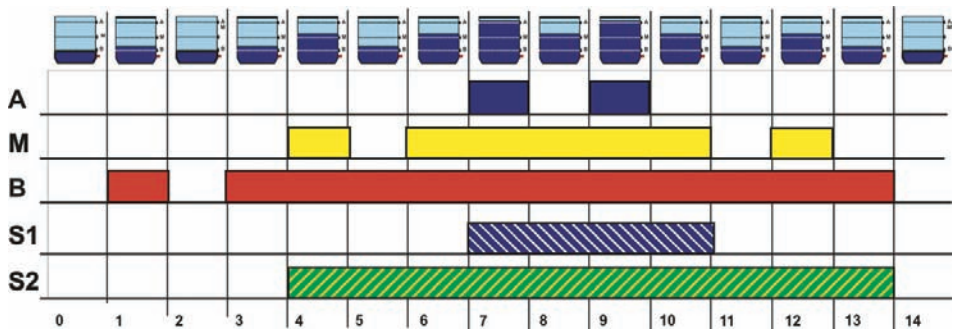


Diagrama de tiempos del sistema de nivel con tres sensores de entrada A, B y C, y dos salidas S2, S1:



Reporte

Realice el reporte correspondiente a esta práctica con las siguientes especificaciones:

1. Portada.
   a) Nombre de la práctica.
   b) Fecha de realización.
   c) Nombre y número de matrícula.
   d) Nombre del instructor.
2. Introducción, explicar el objetivo de la práctica.
3. Procedimiento y metodología.
4. Representación de la función por medio de diagramas de alambrado, diagrama esquemático, circuito y ecuación, según sea el caso.
5. Resultados, conclusiones y recomendaciones.
6. Agregar el cuestionario resuelto que aparece al final de la práctica.
7. Referencias bibliográficas.

Nota: *Tanto el procedimiento como la metodología deben ser explicados, y los resultados analizados y comentados. Un reporte con diagramas sin explicaciones ni conclusiones carece de valor.*

PRÁCTICA 10

Contadores

Objetivo particular

Durante el desarrollo de esta práctica se diseñarán diferentes tipos de contadores binarios, de décadas, ascendentes y/o descendentes, etcétera, usando ABEL-HDL.

El tiempo estimado para el desarrollo de esta práctica es de dos horas.

Fundamento teórico

Los contadores son sistemas secuenciales que tienen el propósito de contar sucesos electrónicos, como los impulsos, avanzando a través de una secuencia de estados binarios.

Los contadores se clasifican en:

a) Binarios

b) De décadas

c) Especiales

Todos ellos pueden ser ascendentes y/o descendentes.

Los contadores binarios, a la vez, se clasifican por el número de bits; por ejemplo:

Bits	Conteo ascendente	Conteo descendente
1	0 a 1	1 a 0
2	0 a 3	3 a 0
3	0 a 7	7 a 0
4	0 a 15	15 a 0
5	0 a 31	31 a 0
6	0 a 63	63 a 0

Los contadores decimales, también llamados de décadas en BCD, son de cuatro bits, porque sólo cuentan de 0 a 9 o de 9 a 0.

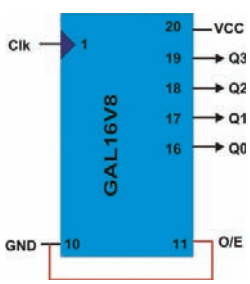
Un contador especial puede ser el de un minutero o segundero, ya sea de un reloj digital o de un marcador deportivo de 0 a 59 en forma ascendente, o de 59 a 0 en forma descendente.

Ejemplo 10.1

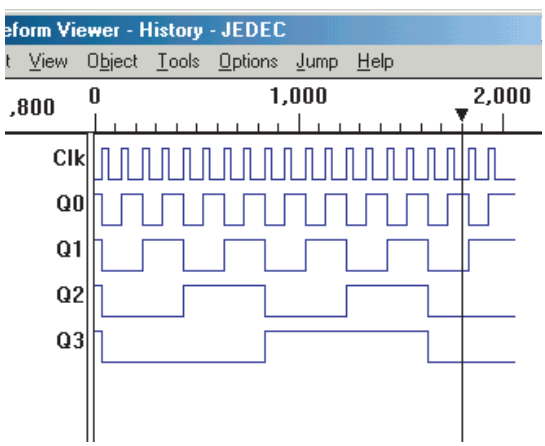
Diseño de un contador binario de cuatro bits ascendente (de 0 a 15)

Para simplificar el programa en ABEL-HDL se usará la instrucción COUNT := (COUNT + 1);. Para activar las salidas de los Flip Flops es necesario conectar la terminal 11 (Output/Enable) a GND como se indica en la figura.

```
MODULE conta
"Constantes la señal de reloj en la
simulación .c. se sustituye por C
"La salida .x. se sustituye por sólo x
     C,x = .c.,.x.;
```



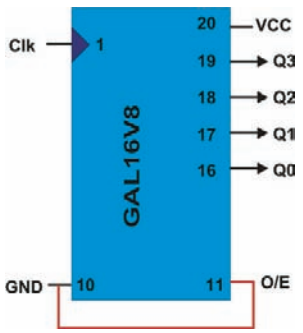
```
  "Entrada de reloj
        Clk pin 1;
"Salidas de los Flip Flops
        Q3..Q0 pin 19..16 istype 'reg,buf-
fer';

Declarations
        COUNT = [Q3..Q0];
Equations
COUNT.Clk=Clk;
COUNT := (COUNT + 1);

Test_Vectors
([Clk ] -> COUNT)
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
END
```

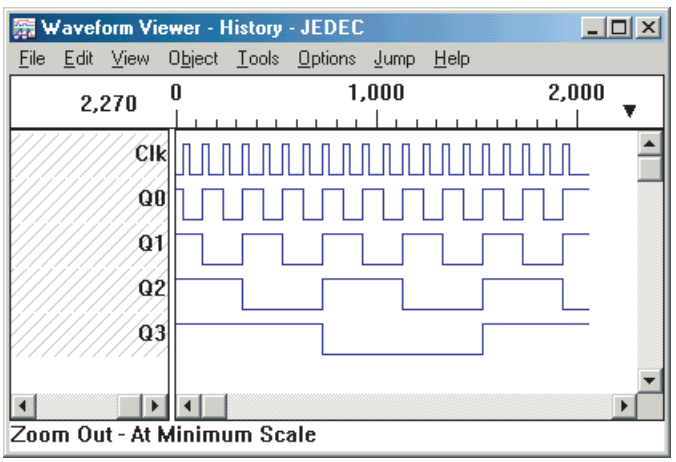


Ejemplo 10.2

Diseño de un contador binario de cuatro bits descendente (de 15 a 0)

```
MODULE countd
“Constantes
      C,x = .c.,.x.;
“Entrada de reloj
      Clk pin 1;
“Salidas de los Flip Flops
      Q3..Q0 pin 19..16 istype ‘reg,buffer’;
Declarations
      COUNT = [Q3..Q0];

Equations
COUNT.Clk=Clk;

COUNT := (COUNT - 1);

Test_Vectors
([Clk ] -> COUNT)
  [ C ] ->   x  ;
  [ C ] ->   x  ;
  [ C ] ->   x  ;
  [ C ] ->   x  ;
  [ C ] ->   x  ;
  [ C ] ->   x  ;
  [ C ] ->   x  ;
  [ C ] ->   x  ;
  [ C ] ->   x  ;
  [ C ] ->   x  ;
  [ C ] ->   x  ;
  [ C ] ->   x  ;
  [ C ] ->   x  ;
  [ C ] ->   x  ;
  [ C ] ->   x  ;
  [ C ] ->   x  ;
  [ C ] ->   x  ;
  [ C ] ->   x  ;

END
```



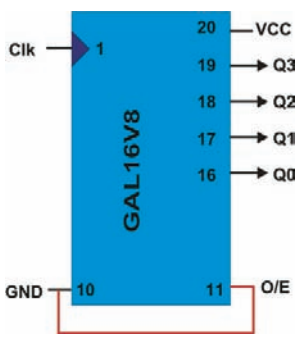


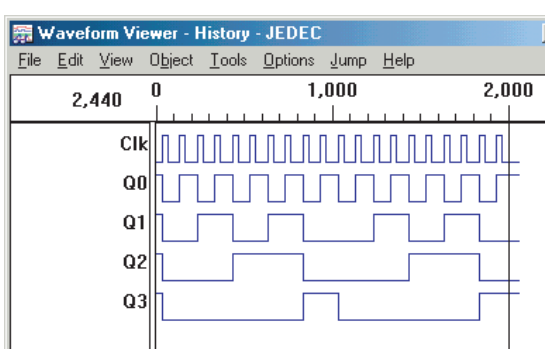
Ejemplo 10.3

Diseño de un contador de décadas o BCD ascendente (0 a 9)

En este ejemplo se utilizará el comando TRUTH_TABLE en el modo secuencial.

```
MODULE contdeca
“Constantes
      C,x = .c.,.x.;
“Entrada de reloj
      Clk pin 1;
“Salidas de los Flip Flops
      Q3..Q0 pin 19..16 istype ‘reg,buffer’;
Declarations

      DECA = [Q3..Q0];

Equations
DECA.Clk=Clk;
TRUTH_TABLE
(DECA :> DECA)
   0:>1;
   1:>2;
   2:>3;
   3:>4;
   4:>5;
   5:>6;
   6:>7;
   7:>8;
   8:>9;
   9:>0;
Test_Vectors
([Clk ] -> [DECA])
  [ C ] ->   x  ;
  [ C ] ->   x  ;
  [ C ] ->   x  ;
  [ C ] ->   x  ;
  [ C ] ->   x  ;
  [ C ] ->   x  ;
  [ C ] ->   x  ;
  [ C ] ->   x  ;
```



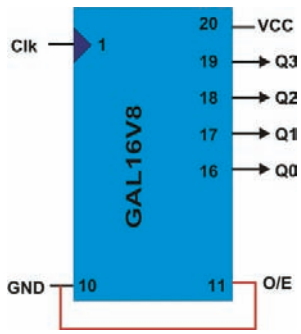


```
[ C ] ->   x  ;
[ C ] ->   x  ;
[ C ] ->   x  ;
[ C ] ->   x  ;
[ C ] ->   x  ;
[ C ] ->   x  ;
[ C ] ->   x  ;
[ C ] ->   x  ;
[ C ] ->   x  ;
[ C ] ->   x  ;

END
```

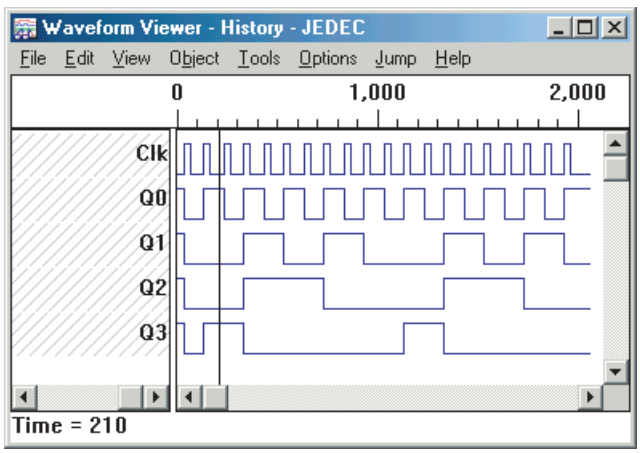
Ejemplo 10.4

Diseño de un contador de décadas o BCD descendente (9 a 0)

```
MODULE contdecd
“Constantes
      C,x = .c.,.x.;
“Entrada de reloj
      Clk pin 1;
“Salidas de los Flip Flops
      Q3..Q0 pin 19..16 istype ‘reg,buffer’;
Declarations
      DECA = [Q3..Q0];
Equations
DECA.Clk=Clk;
TRUTH_TABLE
(DECA :> DECA)
   9:>8;
   8:>7;
   7:>6;
   6:>5;
   5:>4;
   4:>3;
   3:>2;
   2:>1;
   1:>0;
```



```
    0:>9;
Test_Vectors
([Clk ] -> [DECA])
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
   [ C ] ->   x  ;
END
```



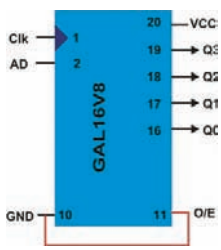
Ejemplo 10.5

Contador ascendente y descendente

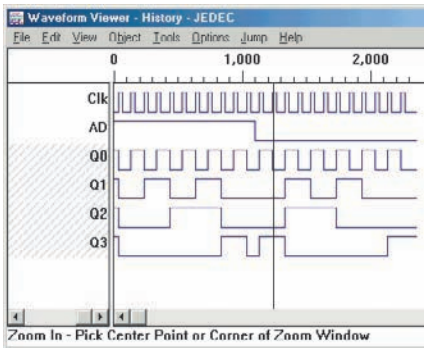
Diseñe un contador de décadas que contenga una entrada A/D de modo que:

a) Si la entrada A/D es igual a uno, el conteo es ascendente (0 a 9).

b) Si la entrada A/D es igual a cero, el conteo es descendente (9 a 0).



```
MODULE cdecad
“Constantes
        C,x = .c.,.x.;
“Entrada de reloj
        Clk,AD pin 1,2;
“Salidas de los Flip Flops
        Q3..Q0 pin 19..16 istype ‘reg,buffer’;
Declarations
        DECA = [Q3..Q0];
Equations
DECA.Clk=Clk;
```

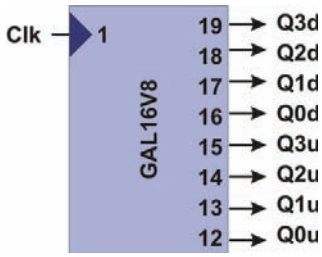



```
TRUTH_TABLE
([AD,DECA] :> DECA)
   [1,0]:>1;
   [1,1]:>2;
   [1,2]:>3;
   [1,3]:>4;
   [1,4]:>5;
   [1,5]:>6;
   [1,6]:>7;
   [1,7]:>8;
   [1,8]:>9;
   [1,9]:>0;
   [0,9]:>8;
   [0,8]:>7;
   [0,7]:>6;
   [0,6]:>5;
   [0,5]:>4;
   [0,4]:>3;
   [0,3]:>2;
   [0,2]:>1;
   [0,1]:>0;
   [0,0]:>9;
```

```
Test_Vectors
([AD,Clk ] -> [DECA])
   [ 1,C ] ->   x  ;
   [ 1,C ] ->   x  ;
   [ 1,C ] ->   x  ;
   [ 1,C ] ->   x  ;
   [ 1,C ] ->   x  ;
   [ 1,C ] ->   x  ;
   [ 1,C ] ->   x  ;
   [ 1,C ] ->   x  ;
   [ 1,C ] ->   x  ;
   [ 1,C ] ->   x  ;
   [ 1,C ] ->   x  ;
   [ 0,C ] ->   x  ;
   [ 0,C ] ->   x  ;
   [ 0,C ] ->   x  ;
   [ 0,C ] ->   x  ;
   [ 0,C ] ->   x  ;
   [ 0,C ] ->   x  ;
   [ 0,C ] ->   x  ;
   [ 0,C ] ->   x  ;
   [ 0,C ] ->   x  ;
   [ 0,C ] ->   x  ;
END
```

Ejemplo 10.6

Diseño de un contador de décadas ascendente de 0 a 99

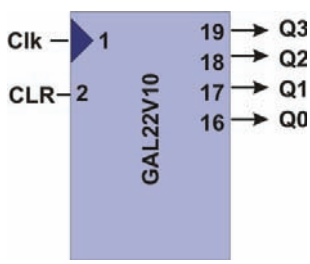


Ejemplo 10.7

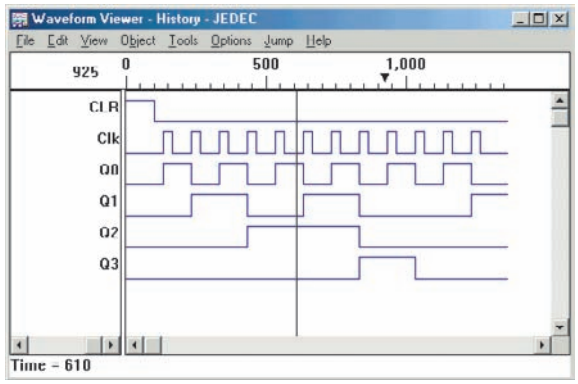
Contador con CLR

Diseñe un contador de décadas ascendente de 0 a 9 que incluya una señal asíncrona CLR, de manera que si CLR = 1, el contador debería regresar al valor cero. Implémentelo en un GAL22V10.

```
MODULE cdeclr
“Constantes
      C,x = .c.,.x.;
“Entrada de reloj
      Clk,CLR pin 1,2;
“Salidas de los Flip Flops
      Q3..Q0 pin 19..16 istype ‘reg,buffer’;
Declarations
      DECA = [Q3..Q0];

Equations
DECA.Clk=Clk;
DECA.ar=CLR;

TRUTH_TABLE
([CLR,DECA] :> DECA)
   [1,x]:>0;
   [0,0]:>1;
   [0,1]:>2;
```



```
   [0,2]:>3;
   [0,3]:>4;
   [0,4]:>5;
   [0,5]:>6;
   [0,6]:>7;
   [0,7]:>8;
   [0,8]:>9;
   [0,9]:>0;

Test_Vectors
([CLR,Clk ] -> [DECA])
   [ 1,x ] ->   x  ;
   [ 0,C ] ->   x  ;
   [ 0,C ] ->   x  ;
   [ 0,C ] ->   x  ;
   [ 0,C ] ->   x  ;
   [ 0,C ] ->   x  ;
   [ 0,C ] ->   x  ;
   [ 0,C ] ->   x  ;
   [ 0,C ] ->   x  ;
   [ 0,C ] ->   x  ;
   [ 0,C ] ->   x  ;
END
```

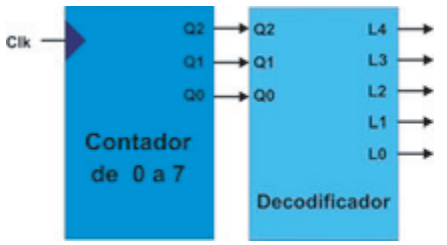


Ejemplo 10.8

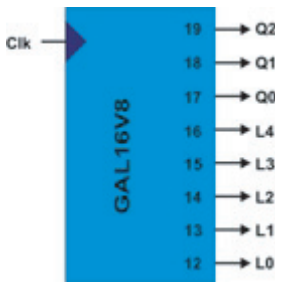
Diseñe un sistema secuencial síncrono que contenga sólo la entrada de Ck, y cinco LED de salida, que sea capaz de encender los LED con la siguiente secuencia de ocho tiempos de t = 0 a t = 7. Repítala indefinidamente.

Ck	t	Lo	L1	L2	L3	L4
↑	0	**1**	**0**	**0**	**0**	**0**
↑	1	**0**	**1**	**0**	**0**	**0**
↑	2	**0**	**0**	**1**	**0**	**0**
↑	3	**0**	**0**	**0**	**1**	**0**
↑	4	**0**	**0**	**0**	**0**	**1**
↑	5	**0**	**0**	**0**	**1**	**0**
↑	6	**0**	**0**	**1**	**0**	**0**
↑	7	**0**	**1**	**0**	**0**	**0**

Para este diseño se propone elaborar dos bloques. Uno secuencial, con el objetivo de efectuar la función de un contador de tres bits, o de 0 a 7 en binario, para que la cuenta se incremente por cada pulso de reloj. Y otro combinacional, cuyo objetivo sea decodificar y proporcionar la salida adecuada para cada conteo.



Estas dos funciones, la del contador y el decodificador, se podrían integrar en un solo archivo de ABEL-HDL usando, en ambos casos, el comando TRUTH_TABLE y un solo circuito integrado.

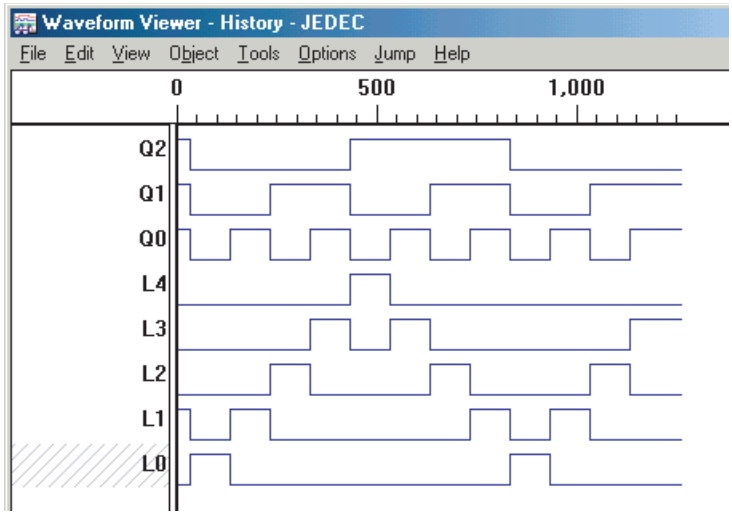


10

Archivo ABEL-HDL

```
MODULE luces
DECLARATIONS
“Constantes
        c,x=.c.,.x.;
“Nombre del reloj (clock)
CLK PIN 1;
“ Salidas del decodificador
L4..L0  PIN 16..12  ISTYPE ‘com’;

“ Salidas del contador
Q2..Q0  PIN 19..17 ISTYPE ‘reg’;

S=[Q2,Q1,Q0];
equations
S.clk=CLK;
“Tabla de estados
truth_table
([S]:>[S]);
[0]:>[1];
[1]:>[2];
[2]:>[3];
[3]:>[4];
[4]:>[5];
[5]:>[6];
[6]:>[7];
[7]:>[0];
```

```
“Tabla de Verdad
truth_table
([S]->[L0,L1,L2,L3,L4]);
[0]->[1,0,0,0,0];
[1]->[0,1,0,0,0];
[2]->[0,0,1,0,0];
[3]->[0,0,0,1,0];
[4]->[0,0,0,0,1];
[5]->[0,0,0,1,0];
[6]->[0,0,1,0,0];
[7]->[0,1,0,0,0];

Test_Vectors
([CLK]->[L0,L1,L2,L3,L4]);
[c]->[x,x,x,x,x];
[c]->[x,x,x,x,x];
[c]->[x,x,x,x,x];
[c]->[x,x,x,x,x];
[c]->[x,x,x,x,x];
[c]->[x,x,x,x,x];
[c]->[x,x,x,x,x];
[c]->[x,x,x,x,x];
[c]->[x,x,x,x,x];
[c]->[x,x,x,x,x];
[c]->[x,x,x,x,x];
[c]->[x,x,x,x,x];
END
```

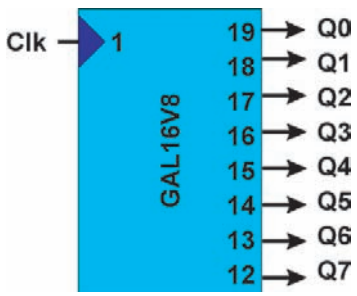


Ejemplo 10.*9*

Diseñe un contador binario de ocho bits (0 a 255) en forma ascendente:

Archivo ABEL-HDL.

```
MODULE COUNTER
“Constants
        C,x = .c.,.x.;
“Entrada de reloj
        Clk pin 1;
“Salidas de los Flip Flops
        Q7..Q0 pin 19,18,17,16,15,14,13,12 istype ‘reg,buffer’;
Declarations
        COUNT = [Q7..Q0];
Equations
COUNT := (COUNT + 1)
    END
```



Problema propuesto

1. Diseñe un reloj digital para un marcador deportivo que contenga minutos y segundos en diferentes modalidades (seleccione sólo uno):

a) Marcador de fútbol sóccer ascendente de 0 a 45 minutos.

b) Marcador de baloncesto descendente de 12 a 0 minutos.

c) Marcador de fútbol americano descendente de 15 a 0 minutos.

d) Marcador de fútbol americano que indique el límite de tiempo para iniciar la jugada, tiene que ser de 24 a 0 segundos con entrada de reset para el regreso a cero.

PRÁCTICA 11

Sistemas secuenciales asíncronos

Objetivo particular

Durante el desarrollo de esta práctica se aplicará la metodología para diseñar un sistema secuencial asíncrono y se implementará mediante captura esquemática o lenguaje de descripción de hardware en un dispositivo lógico programable.

Fundamento teórico

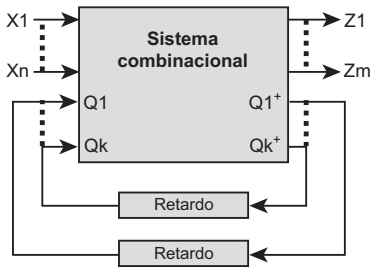
Diseño de sistemas secuenciales asíncronos (modo nivel)

Definición de sistema secuencial asíncrono

El sistema secuencial asíncrono es un sistema donde los valores de salidas

(Z1 . . . Zn) y de los estados siguientes (Q1+ . . . Qk+) no dependen únicamente de las combinaciones de las entradas (X1 . . . Xn), sino también de los estados actuales (Q1 . . . Qk) .

En éste, los estados cambian casi inmediatamente después del cambio de una entrada, lo cual sucede sin esperar a que se presente una señal de sincronía o Clk.



Descripción a bloques de un sistema secuencial asíncrono.

NOTA: *Los retardos pueden representar tiempos de propagación entre los elementos del sistema, o retardos intencionales para asegurar el funcionamiento adecuado del mismo.*

Otra forma de describir estos sistemas es que son circuitos lógicos combinacionales con retroalimentación, como se muestra en la figura.

Para el diseño de un sistema secuencial asíncrono es conveniente contar con un diagrama de tiempos o un diagrama de transición con la descripción del problema.

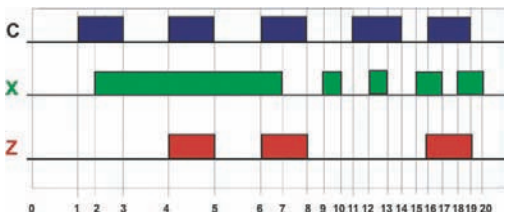


Diagrama de tiempos

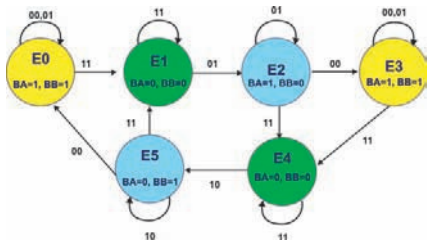


Diagrama de transición

Características de los los sistemas secuenciales asíncronos:

- No necesitan del reloj como señal de control.
- La salida cambia cuando una variable de entrada cambia (por eventos).
- Se implementan a partir de operadores **And**, **Or** y **Not**, utilizando retroalimentación, o a partir de FF RS básico (sin reloj).
- No requieren de FF estándar (JK, T o D).
- Son más rápidos en su respuesta porque no dependen de una señal adicional de sincronía, sólo de un cambio de nivel.
- Son directamente adaptables a la implementación de PLC mediante un circuito escalera.

Definición de estados

Estado o evento: Condición o situación durante la cual se satisface una condición, se realiza una actividad o se espera un evento.

Estado total: Descrito por la combinación de los valores de las variables de entrada (X, Y) y los valores de las variables que determinan el estado presente (Q1, Q2).

Estado interno: Descrito por los valores de los estados presentes (Q1, Q2).

Estado estable: Estado interno próximo o siguiente, que es igual al estado interno presente $Q1^+ Q2^+ = Q1\ Q2$.

Estado inestable: El estado interno próximo o siguiente, es distinto al estado interno presente $Q1^+ Q2^+ \neq Q1\ Q2$.

Tabla de flujo

En una tabla de flujo (o transiciones) se tiene la misma información que en un diagrama de estados, pero organizada de forma tabular. Para su construcción se establecen las siguientes condiciones:

a) Sólo debe haber un estado estable por fila.

b) Sólo cambia una variable de entrada a la vez.

Entonces, la tabla recibe el nombre de tabla de flujo primitiva.

En una tabla de flujo primitiva se pueden considerar eventos no descritos en el diagrama de tiempo, o en el diagrama de transición; de tal suerte, que con ésta se asegura la consideración de todos los eventos posibles en el funcionamiento u operación de un diseño.

Ejemplo de tabla de flujo con tres estados E0 a E3, una entrada P y una salida Y:

	P		Y
	0	1	
1	**E0**	E1	0
2	E2	**E1**	1
3	**E2**	E3	1
4	E0	**E3**	0

Metodología de diseño de un sistema secuencial asíncrono

A continuación se muestran los 13 pasos sugeridos para el desarrollo y diseño de sistemas secuenciales asíncronos:

1. Especificar el sistema.
2. Construir la tabla de flujo primitiva.
3. Eliminar los estados redundantes.
4. Efectuar la mezcla de filas.
5. Expandir la tabla de salidas (si se requiere).
6. Construir la tabla de estados internos.
7. Asignar valores a los estados.
8. Construir la tabla de estados final.
9. Completar tabla de salidas.
10. Obtener las ecuaciones mínimas.
11. Realizar la simulación.
12. Efectuar la representación gráfica.
13. Realizar la implementación.

Procedimiento

1. Especificar el sistema.

   Usando un diagrama de tiempos o diagrama de transición se pueden describir los eventos del sistema.

2. Construir la tabla de flujo primitiva

   Se trata de una tabla que tiene únicamente un estado estable por renglón, con salidas especificadas sólo para estados estables, donde se incluyan todos los posibles eventos.

3. Eliminar los estados redundantes o equivalentes.

   Dos estados son equivalentes si:

   a) Son estados estables en la misma columna (misma combinación de entradas).

   b) Tienen la misma salida.

   c) Sus estados siguientes son equivalentes.

4. Efectuar la mezcla de filas.

En este proceso la tabla de flujo primitiva se transforma en una tabla de estados totales.

5. Expandir la tabla de salidas.

Si para una hilera se tiene la posibilidad de mezclar filas, pero los estados estables tienen salidas diferentes, es necesario expandir las salidas especificando el valor correspondiente a cada estado estable, de acuerdo con la condición de entrada en que se presente.

6. Construir la tabla de estados internos.

Convertir la tabla de estados totales, obtenida al mezclar las filas, en una tabla de estados internos. Se asigna el mismo nombre a todos los estados estables de una misma fila con el propósito de clarificar las transiciones que se presenten entre un estado y otro.

7. Asignar valores a los estados.

Asignar un valor a un estado consiste en proporcionar un valor binario a cada estado que lo haga único. Además, esta asignación debe asegurar que haya un sólo cambio entre los valores de una asignación y otra cuando entre éstas se dé una transición, como las descritas en la tabla de estados internos.

8. Construir la tabla de estados final.

En esta tabla se sustituyen los estados internos por el valor binario de la asignación propuesta en el paso anterior.

9. Completar tabla de salidas.

En algunos casos, las salidas no están completamente definidas, lo cual podría generar valores transitorios no convenientes para el sistema; por tal razón, es necesario asignar un valor a la salida, de manera que no se presente el transitorio.

10. Obtener las ecuaciones mínimas.

Es conveniente obtener las ecuaciones mínimas para optimizar la implementación física del sistema, para lo cual se tienen los siguientes recursos:

a) Manipulación algebraica.

b) Mapas de Karnaugh.

c) Uso de software.

11. Realizar la simulación.

Es recomendable asegurarse de que el diseño realizado cumpla con las especificaciones propuestas antes de implementarlo físicamente. Para ello se propone la simulación, la cual se efectúa con algún software o con el lenguaje ABEL-HDL (que incluye el Test_ Vectors). Por lo general, la forma de presentarse es usando

un diagrama de tiempo, donde se incluyan las entradas y las salidas, así como los valores de los estados (Q).

12. Efectuar la representación gráfica.

Esta representación ofrece una visión panorámica de los elementos y su interconexión, ya sea para su análisis o implementación física. Tradicionalmente se utilizan los siguientes elementos:

a) Diagrama esquemático.

b) Diagrama escalera.

13. Realizar la implementación.

Es la realización física del proyecto, el cual puede llevarse a cabo mediante:

a) Circuitos de función fija TTL y CMOS.

b) Dispositivos lógicos programables (PLD).

b) Controlador lógico programable (PLC).

Para ayudar a la comprobación de la metodología del diseño, se seleccionaron varios ejemplos con distinto grado de dificultad, en los cuales se describen claramente todos los pasos del método.

Ejemplo 11.1

Botón

Diseñe un sistema secuencial asíncrono que contenga un botón de entrada **P** y una salida **Y**, de manera que al oprimir por primera vez **P**, la salida **Y** deberá cambiar de un valor inicial de 0 a 1. Al soltar el botón, la salida se mantendrá en 1; al oprimir por segunda vez **P**, la señal de salida **Y** deberá cambiar a 0, y al soltarlo continuará con en ese valor, como lo indica la siguiente gráfica:

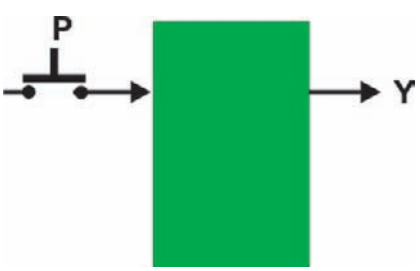


Diagrama de bloques

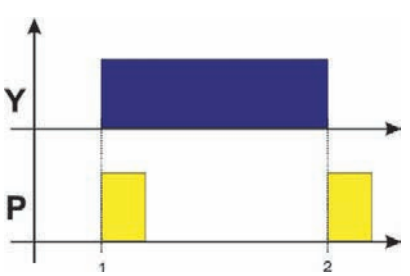
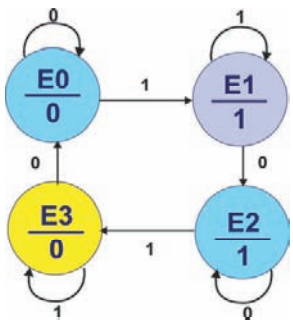


Diagrama de tiempos

Procedimiento

1. Especificar el sistema.

Con el diagrama de transición se podría explicar el funcionamiento del sistema, indicando para cada uno de los estados las posibles entradas y los estados siguientes:



Partiendo de un estado **E0** con salida

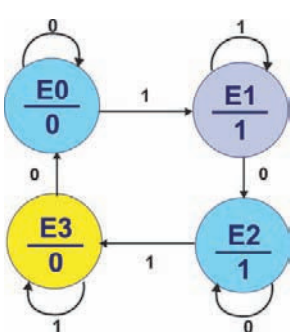
Y = 0, si no se oprime el botón (P = 0), permanece en el mismo estado. Al presionar el botón (P = 1) cambia al estado **E1**, donde la salida es Y = 1.

Permanece en ese estado si el botón se mantiene oprimido (P = 1) y cambia a un estado **E2** al soltar el botón (P = 0), donde la salida sigue siendo Y = 1.

Ya en **E2** se mantiene, si no se oprime el botón (P = 0), y cambia a un estado **E3** al presionar el botón (P =1), donde la salida cambia a Y = 0. Continúa en **E3** mientras el botón se encuentra oprimido (P = 1), y al soltarlo (P = 0) regresa al estado inicial **E0.**

2. Construir la tabla de flujo primitiva.

Para construir la tabla de flujo primitiva se recurre al diagrama de transición. En el estado **E0** con entrada cero (P = 0), es estable, ya que se mantiene mientras el valor de la entrada no cambie. Esto se indica en la hilera 1 como **E0**; además, la salida Y para esa hilera es cero, y con entrada uno (P = 1) tendría una transición a un estado **E1** que es estable en la hilera 2 (**E2**) (sólo se permite un estado estable por hilera) con salida Y = 1. Observe la figura.



	P		Y
	0	1	
1	E0	E1	0
2		E1	1

Continuando con el análisis de los demás estados, se obtiene la tabla de flujo primitiva que se muestra a continuación:

	P		Y
	0	1	
1	**E0**	E1	0
2	E2	**E1**	1
3	**E2**	E3	1
4	E0	**E3**	0

3. Eliminar los estados redundantes o equivalentes.

Los estados establecidos en la tabla de flujo primitiva no son indispensables, ya que podría haber estados redundantes; por ello, es necesario identificarlos (si los hay) y eliminarlos.

Un estado es redundante si existe otro equivalente. Dos estados son equivalentes si:

a) Son estados estables en la misma columna (misma combinación de entradas).

b) Tienen la misma salida.

c) Sus estados siguientes son equivalentes.

	P		Y
	0	1	
1	**E0**	E1	0
2	E2	**E1**	1
3	**E2**	E3	1
4	E0	**E3**	0

‘ En la tabla anterior se observa que el valor de entrada P = 0 **E0** y **E2** son estables, pero no tienen la misma salida, y para la entrada P = 1 **E1** y **E3** son estables, pero tampoco tienen la misma salida, por lo que se concluye que no hay estados redundantes y todos son necesarios.

4. Efectuar la mezcla de filas.

Una vez eliminados los estados redundantes, las filas o hileras se mezclan para reducir la tabla.

Dos filas o más se pueden mezclar, siempre y cuando no haya ningún conflicto sobre qué estado debe ocupar cada columna (se entiende por conflicto la ocupación simultánea de una columna por dos estados diferentes).

La salida no se considera como un factor de conflicto en la mezcla de filas. Esto es, dos filas con salidas diferentes pueden mezclarse.

Se construye un diagrama de mezcla con el propósito de tener una visualización completa sobre las posibilidades de mezcla de las filas.

El diagrama de mezcla consiste en asignar un punto por cada fila, los cuales se unen por líneas cuando pueden mezclarse.

El procedimiento propuesto para encontrar la posibilidad de mezcla es la comparación de una fila con todas las demás.

Comparación de las filas 1 y 2

En la columna 0 se tiene que **E0 ≠ E2**, por lo tanto, estas filas no se pueden mezclar.

	P	
	0	1
1	E0	E1
2	E2	E1

Comparación de las filas 1 y 3

En la columna 0 se tiene que **E0 ≠ E2**, y en la columna 1, **E1 ≠ E3**; por lo que no es posible mezclar las filas.

	P	
	0	1
1	E0	E1
3	E2	E3

Comparación de las filas 1 y 3

En la columna 1 se tiene que **E1 ≠ E3**, el resultado es que estas filas no se pueden mezclar.

	P	
	0	1
1	E0	E1
4	E0	E3

Comparación de las filas 2 y 3

En la columna 1 se tiene que **E1 ≠ E3**, por lo tanto estas filas no se mezclan.

	P	
	0	1
2	E2	E1
3	E2	E3

Comparación de las filas 2 y 4

En la columna 0 se tiene que E2 ≠ E0 y en la columna 1, E1 ≠ E3; por lo tanto estas filas no se pueden mezclar.

	P	
	0	1
2	E2	E1
4	E0	E3

Comparación de las filas 3 y 4

Por último, en la columna 0 se tiene que **E2 ≠ E0**, lo cual da como resultado que las filas no se mezclan.

	P	
	0	1
3	E2	E3
4	E0	E3

En este problema no es posible la mezcla de filas.

5. Expandir la tabla de salidas.

No es necesario este paso, ya que no se realizó mezcla de filas.

6. Construir la tabla de estados internos.

Para obtener la tabla de estados internos, a cada fila se le identifica con un nombre y a los estados estables de esa fila se les asigna el mismo nombre. Por ejemplo, la primera hilera se sustituye por la letra "a", la segunda por la letra "b", y así hasta la última hilera por "d". Observe la tabla.

	P		Y
	0	1	
a	E0	E1	0
b	E2	E1	1
c	E2	E3	1
d	E0	E3	0

Sustituyendo los nombres de los estados estables de cada fila se tiene:

E0 → a

E1 → b

E2 → c

E3 → d

	P		Y
	0	1	
a	a	b	0
b	c	b	1
c	c	d	1
d	a	d	0

Y así se obtiene la tabla de estados internos:

	P		Y
	0	1	
a	a	b	0
b	c	b	1
c	c	d	1
d	a	d	0

7. Asignar valores a los estados.

Para la asignación de estados se analiza cada columna, indicando las transiciones desde la fila en la cual está un estado inestable, hacia la fila en la que se vuelve estable.

Columna 0) (b → c), (d → a),

1) (a → b), (c → d),

Una vez obtenidas las transiciones, es conveniente asegurarse de que el valor binario de la asignación propuesta para cada estado cambie en una sola variable.

Para este ejemplo, se tiene que **b** es contigua con **c**, y además con **a**; de la misma forma a debe ser contigua con **d** y **b**. Una forma de visualizar fácilmente la condición es usar la estructura de un mapa de Karnaugh, en este caso de dos variables, de manera que entre cuadros contiguos sólo haya un cambio de variable, como se muestra a continuación:

		Q2 = 0	Q2 = 1
Q1	0	a	d
	1	b	c

Como resultado de la asignación se tiene que:

	Q1	Q2
a	0	0
b	0	1
c	1	1
d	1	0

NOTA: *Esta asignación no es la única que cumple con las condiciones de transición, hay otras que también darían una solución al problema.*

8. Construir la tabla de estados final.

Sustituyendo los valores de la asignación se obtiene:

	P		Y
	0	1	
a	a	b	0
b	c	b	1
c	c	d	1
d	a	d	0

Q1 Q2	P		Y
	0	1	
00	00	01	0
01	11	01	1
11	11	10	1
10	00	10	0

9. Completar tabla de salidas

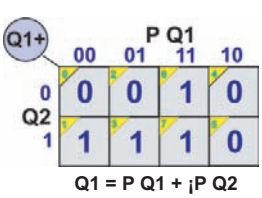
No es necesario completar la tabla de salidas, puesto que no se requiere la expansión de ellas.

10. Obtener las ecuaciones mínimas.

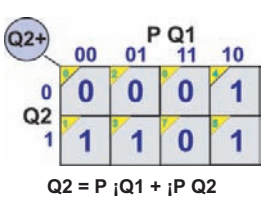
De lo anterior se tiene una tabla de verdad, con el propósito de obtener las ecuaciones mínimas.

m	P	Q1	Q2	Q1+	Q2+	Y
0	0	0	0	0	0	0
1	0	0	1	1	1	1
2	0	1	0	0	0	0
3	0	1	1	1	1	1
4	1	0	0	0	1	0
5	1	0	1	0	1	1
6	1	1	0	1	0	0
7	1	1	1	1	0	1

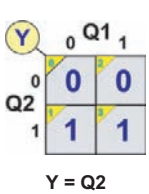
De la tabla de verdad se obtienen los siguientes mapas de Karnaugh:



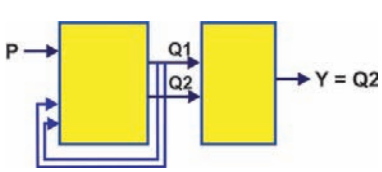
Q1 = P Q1 + ¡P Q2



Q2 = P ¡Q1 + ¡P Q2



Y = Q2



NOTA: *La salida Y no depende de la variable P, por lo que el mapa se expresa sólo en función de Q1 y Q2.*

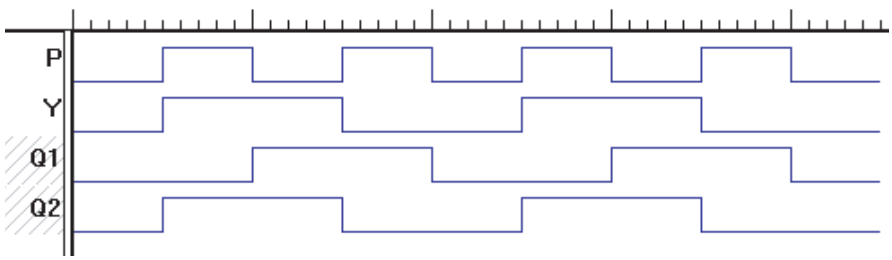
11. Realizar la simulación.

Archivo en formato ABEL-HDL utilizado para la simulación:

```
MODULE boton
"entrada
P pin 1;
"salida
Y, Q1, Q2 pin 19..17 istype 'com';
equations
Q1=P&Q1#!P&Q2;
Q2=P&!Q1#!P&Q2;
Y=Q2;
```

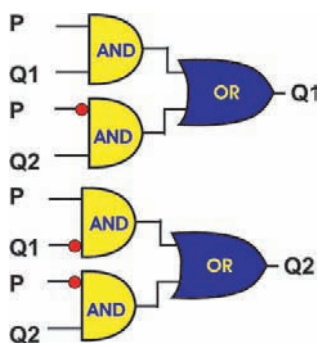
```
Test_Vectors
(P->Y)
0->.x.;
1->.x.;
0->.x.;
1->.x.;
0->.x.;
1->.x.;
0->.x.;
1->.x.;
0->.x.;
END
```

Diagrama de tiempos del *IspStarter*:

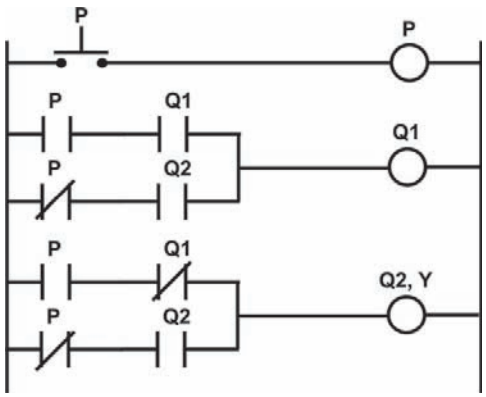


12. Efectuar la representación gráfica.

a) Diagrama esquemático:

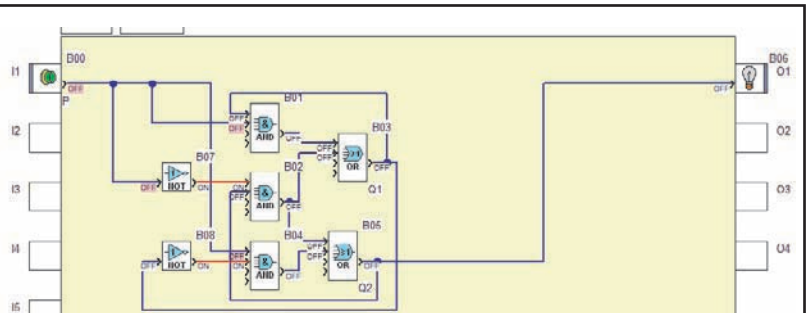
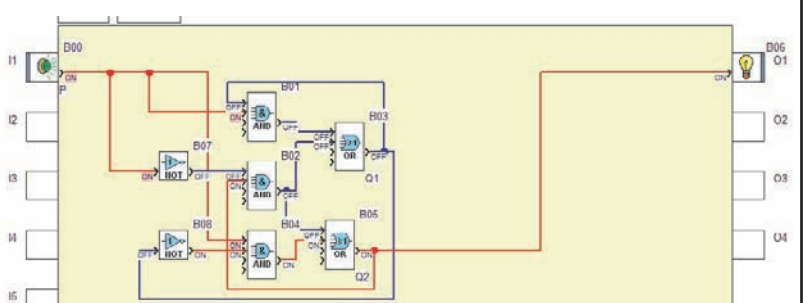


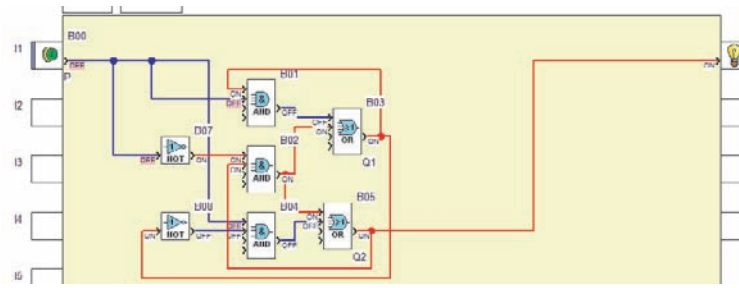
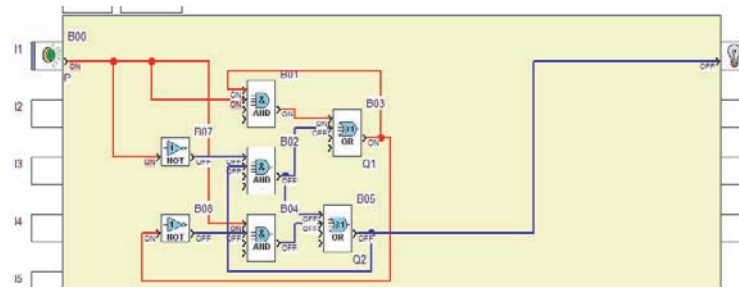
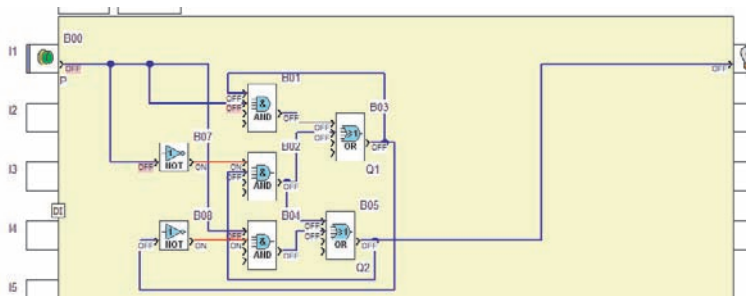
b) Diagrama escalera:



13. Realizar la implementación.

Implementación y simulación en un PLD marca Crouzet, partiendo del diagrama esquemático:

Evento 1 P = 0, Y = 0	
Evento 2 P = 1, Y = 1	

Evento 3 P = 0, Y = 1	
Evento 4 P = 1, Y = 0	
Evento 5 P = 0, Y = 0 Regresa a condiciones iniciales	

Ejemplo 11.2

Diseñe un sistema secuencial asíncrono que contenga un botón de entrada **P** y una salida **Y**, de modo que al oprimir por primera vez la salida **Y** deberá cambiar de un valor inicial 0 a 1, y al soltar el botón la salida se mantendrá en 1. Al oprimirlo por segunda vez, la señal de salida continuará en 1 y al soltarlo cambiará a 0, como lo indica la siguiente gráfica:

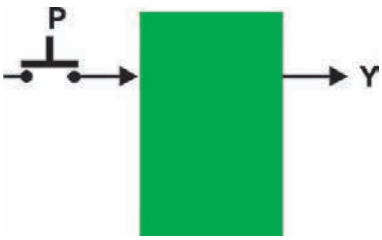


Diagrama de bloques

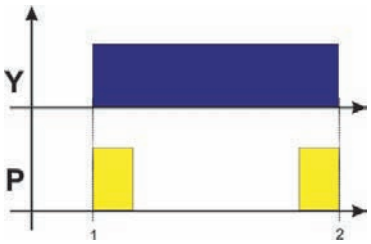
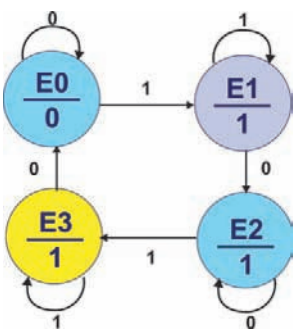


Diagrama de tiempos

Procedimiento

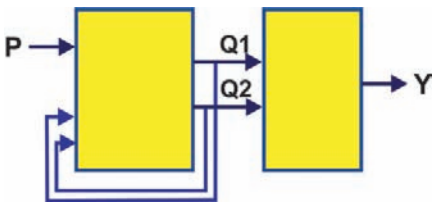
1. Especificar el sistema.
Diagrama de transición:



2. Construir la tabla de flujo primitiva.

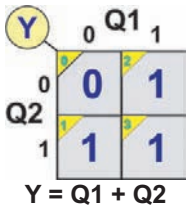
	P		Y
	0	1	
a	E0	E1	0
b	E2	E1	1
c	E2	E3	1
d	E0	E3	1

Note que en la tabla, respecto del ejemplo anterior, el único cambio es la salida **Y** por tratarse del modelo de Moore, donde la salida depende de los valores de **Q1** y **Q2,** que no cambian. Observe la figura.



10. Obtener las ecuaciones mínimas.[1]

La salida **Y** está en función de **Q1** y **Q2**, de modo que, tomando en cuenta la tabla anterior, se llega a:

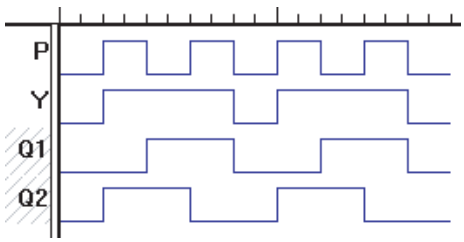


Archivo en formato ABEL-HDL:

```
Y = Q1 + Q2
MODULE boton
“entrada
P pin 1;
“salida
Y, Q1, Q2 pin 19..17 istype ‘com’;
equations
Q1=P&Q1#!P&Q2;
Q2=P&!Q1#!P&Q2;
Y=Q1#Q2;
```

```
Test_Vectors
(P->Y)
0->.x.;
1->.x.;
0->.x.;
1->.x.;
0->.x.;
1->.x.;
0->.x.;
1->.x.;
0->.x.;
END
```

11. Realizar la simulación.



[1]Como no es necesario seguir todo el procedimiento propuesto, se omiten algunos pasos.

12. Efectuar la representación gráfica.

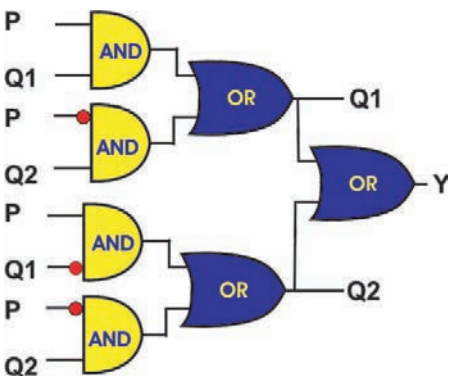


Diagrama esquemático

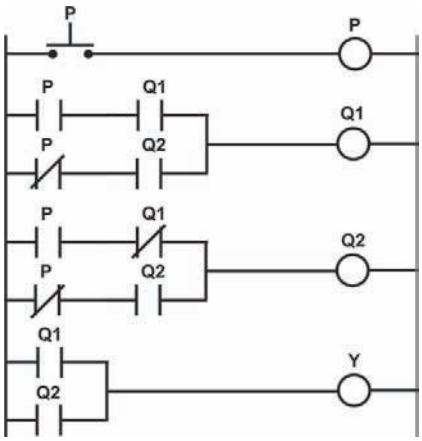
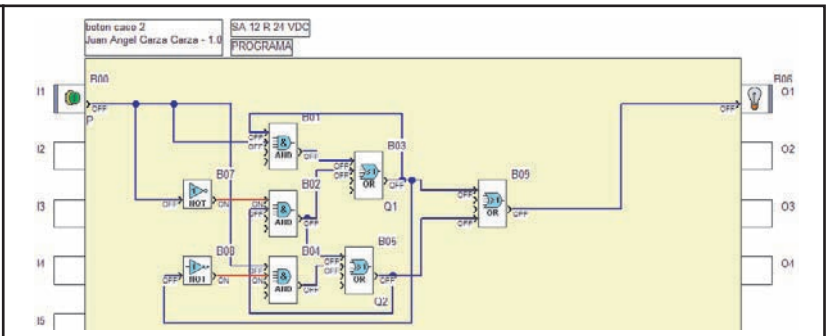
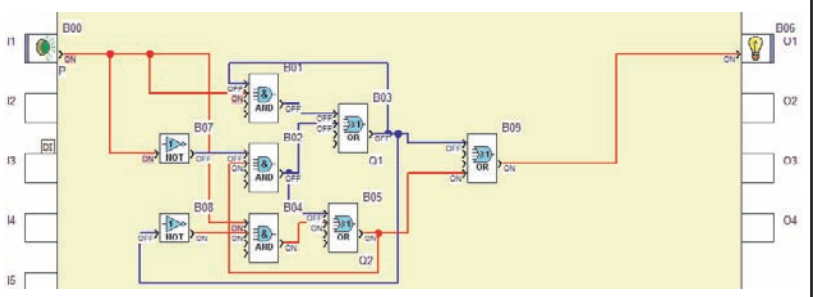


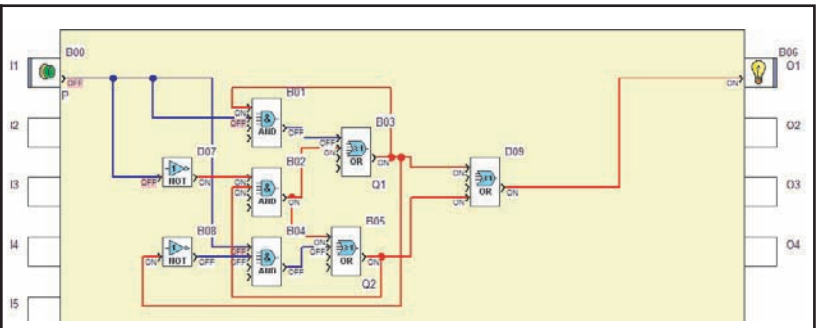
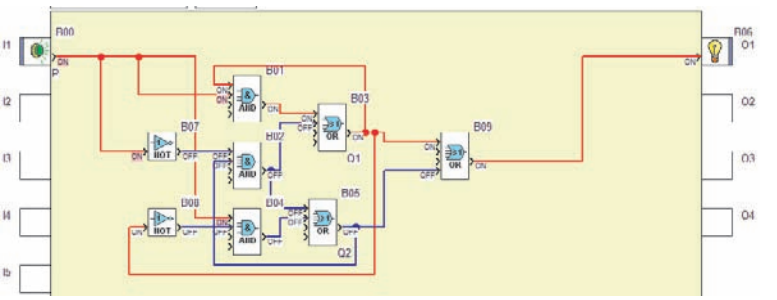
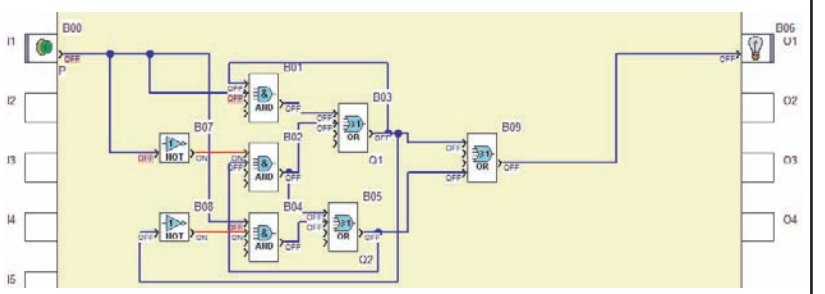
Diagrama escalera

13. Realizar la implementación.

Implementación y simulación en un PLD marca Crouzet por medio del diagrama esquemático:

Evento 1 P = 0, Y = 0	
Evento 2 P = 1, Y = 1	

11

Evento 3 P = 0, Y = 1	
Evento 4 P = 1, Y = 1	
Evento 5 P = 0, Y = 0 Regresa a condiciones iniciales	

Ejemplo 11.3

Diseñe un sistema secuencial asíncrono que tenga una entrada **P**, de manera que al oprimir el botón por primera vez, la salida **Y** permanezca en un valor de 0 y al soltar el botón, la salida cambie a 1. Al oprimirlo por segunda vez, la señal de salida continuará en 1 y al soltarlo cambiará a 0, como lo indica la siguiente gráfica:

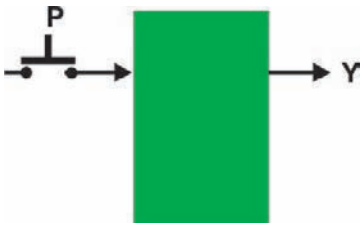


Diagrama de bloques

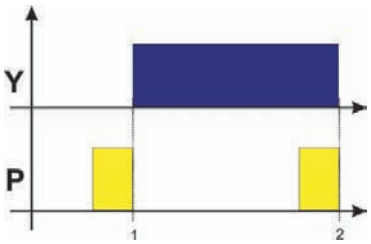
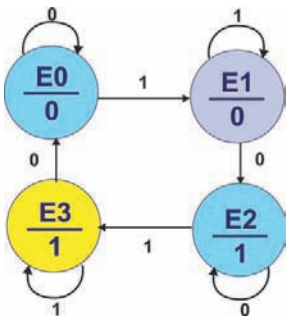


Diagrama de tiempos

Procedimiento

1. Especificar el sistema.

Diagrama de transición:



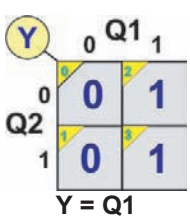
2. Construir la tabla de flujo primitiva.

	P		Y
	0	1	
a	E0	E1	0
b	E2	E1	0
c	E2	E3	1
d	E0	E3	1

Observe que, en esta ocasión, el único cambio es la salida **Y**, de modo que no es necesario seguir todo el procedimiento propuesto.

10. Obtener las ecuaciones mínimas.[2]

La salida **Y** está en función de **Q1** y **Q2**, de modo que, de la tabla anterior, se tiene que:

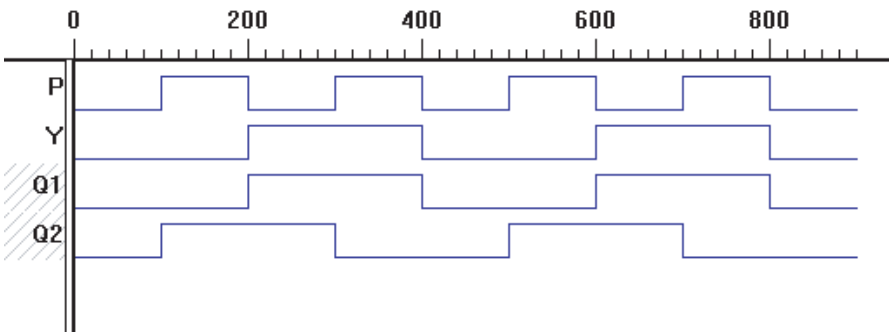


[2]Como no es necesario seguir todo el procedimiento propuesto, se omiten algunos pasos.

Archivo en formato ABEL-HDL:

```
MODULE boton
“entrada
P pin 1;
“salida
Y, Q1, Q2 pin 19..17 istype ‘com’;
equations
Q1=P&Q1#!P&Q2;
Q2=P&!Q1#!P&Q2;
Y=Q1;

Test_Vectors
(P->Y)
0->.x.;
1->.x.;
0->.x.;
1->.x.;
0->.x.;
1->.x.;
0->.x.;
1->.x.;
0->.x.;
END
```

11. Realizar la simulación.



12. Efectuar la representación gráfica.

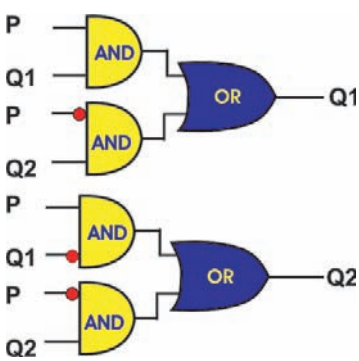


Diagrama esquemático

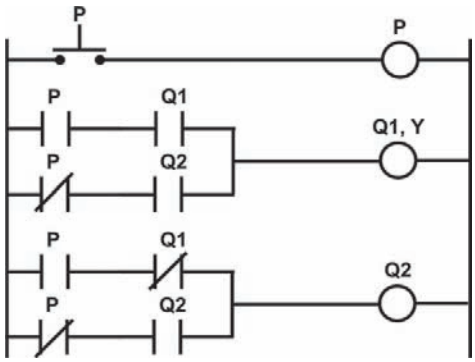
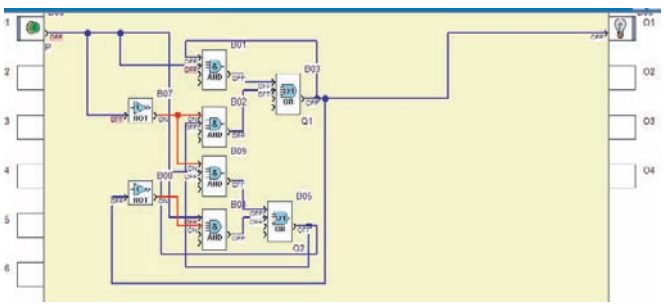
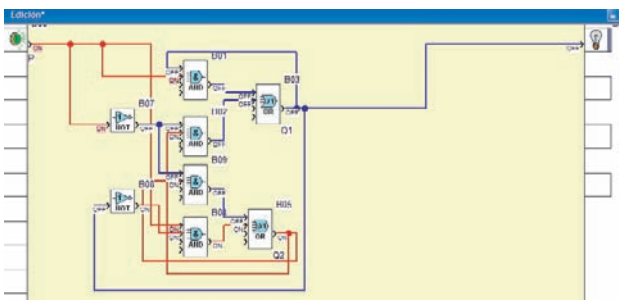
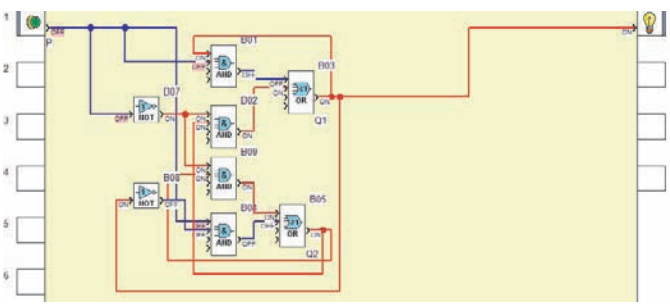
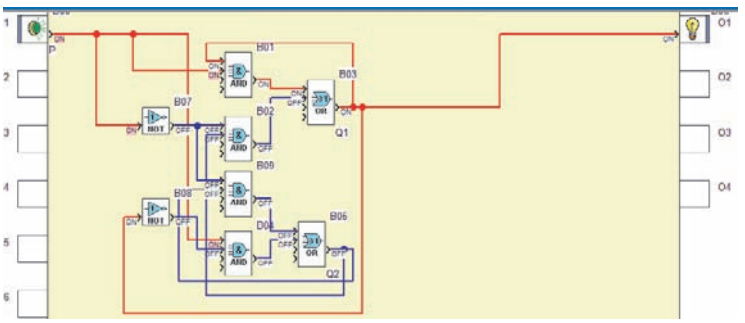
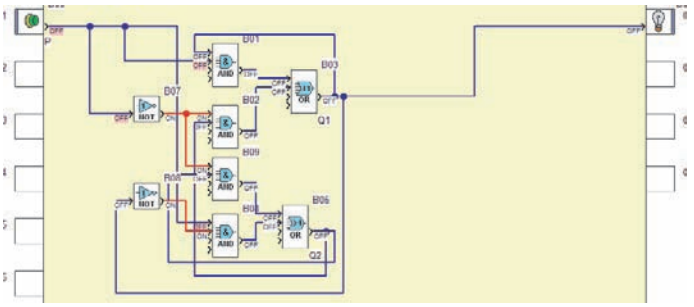


Diagrama escalera

13. Efectuar la implementación.

Implementación y simulación en un PLD marca Crouzet mediante el diagrama esquemático:

Evento 1 P = 0, Y = 0	
Evento 2 P = 1, Y = 0	
Evento 3 P = 0, Y = 1	

Evento 4 P = 1, Y = 1	
Evento 5 P = 0, Y = 0 Regresa a condiciones iniciales	

Observe que en la implementación no fue necesaria la simplificación de la **AND** común de **P' Q2**, como en los casos anteriores, ni agregar una **AND** más.

Ejemplo 11.4

Diseñe un sistema secuencial asíncrono con una entrada **P**, de manera que al oprimir el botón por primera vez, la salida **Y** permanezca con un valor de 0 y al soltar el botón la salida cambie a 1. Al presionarlo por segunda vez, la señal de salida cambiará a 0 y al soltarlo continuará con el mismo valor, como lo indica la siguiente gráfica:

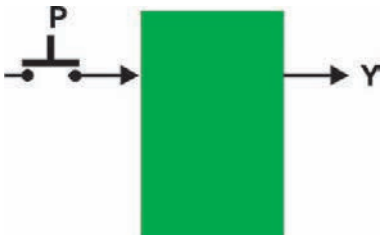


Diagrama de bloques

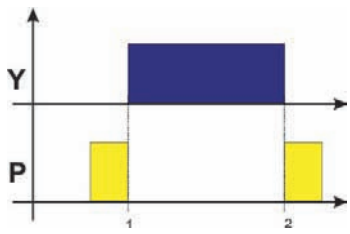
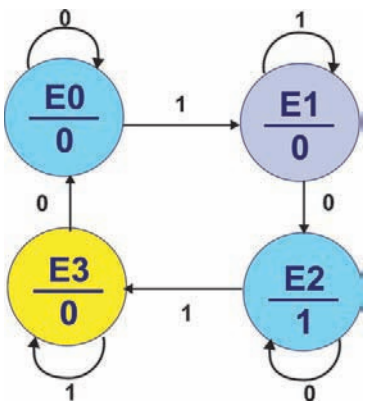


Diagrama de tiempos

Procedimiento

1. Especificar el sistema.

Diagrama de transición:



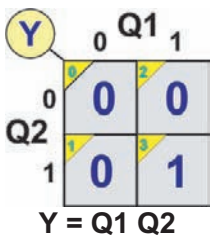
2. Construir la tabla de flujo primitiva.

	P		Y
	0	1	
a	E0	E1	0
b	E2	E1	0
c	E2	E3	0
d	E0	E3	1

Note que en la tabla, respecto del ejemplo anterior, el único cambio es la salida **Y**, de modo que no es necesario seguir todo el procedimiento.

10. Obtener las ecuaciones mínimas.

La salida **Y** está en función de **Q1** y **Q2**, así que de la tabla anterior se tiene que:



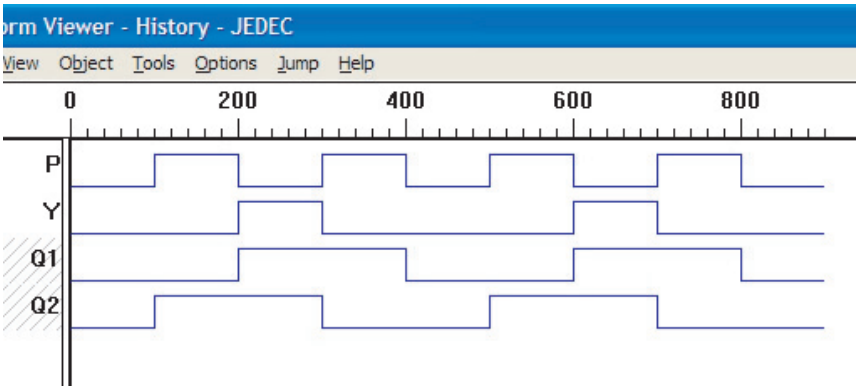
Y = Q1 Q2

11

Archivo en formato ABEL-HDL:

```
MODULE boton
“entrada
P pin 1;
“salida
Y, Q1, Q2 pin 19..17 istype ‘com’;
equations
Q1=P&Q1#!P&Q2;
Q2=P&!Q1#!P&Q2;
Y=Q1&Q2;

Test_Vectors
(P->Y)
0->.x.;
1->.x.;
0->.x.;
1->.x.;
0->.x.;
1->.x.;
0->.x.;
1->.x.;
0->.x.;
END
```

11. Realizar la simulación.



12. Efectuar la representación gráfica.

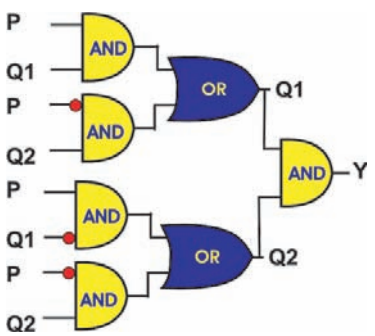


Diagrama esquemático

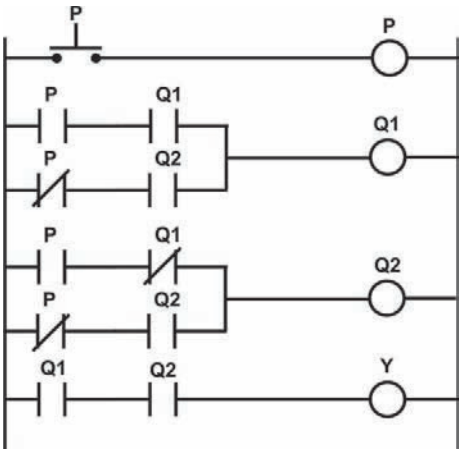


Diagrama escalera

Arranque y paro

Diseñe un sistema secuencial asíncrono que contenga dos botones de entrada llamados **A** (arranque) y **P** (paro), de manera que al oprimir el botón **A** la salida deberá tomar el valor de 1, mientras que con el botón **P**, la salida **Y**, tendrá el valor de 0. Observe la gráfica.

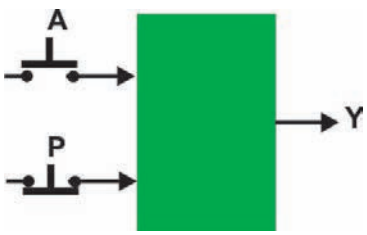


Diagrama de bloques

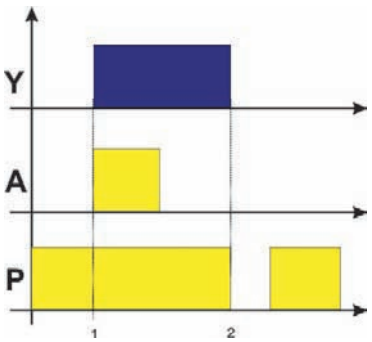
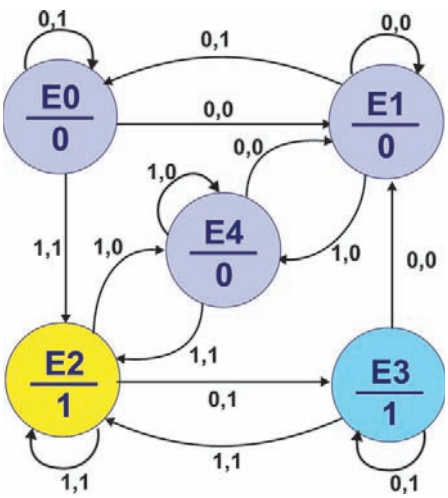


Diagrama de tiempos

Procedimiento

1. Especificar el sistema.

   Diagrama de transición:



2. Construir la tabla de flujo primitiva.

	A P 00	01	10	11	Y
1	E2	**E0**	-	E1	0
2		E3	E4	**E1**	1
3	**E2**	E0	E4		0
4	E2	**E3**	-	E1	1
5	E2	-	**E4**	E1	0

3. Eliminar los estados redundantes.

a) Son estados estables en la misma columna (misma combinación de entradas) E0 y E3.

b) Tienen la misma salida.
E0 y E3 tienen diferente salida, por lo que no hay reducción de estados.

4. Efectuar la mezcla de filas.

Mezclando las filas 1 con 2 y con 5, además de 3 con 4, se obtiene la siguiente tabla:

	A P 00	01	10	11	Y
0,2,4	**E2**	**E0**	**E4**	E1	0
1,3	E2	**E3**	E4	**E1**	1

5. Expandir la tabla de salidas.

No es necesario expandir la tabla de salidas porque 1, 2 y 5 tienen salida Y = 0, en tanto que 3 y 4 tienen salida Y = 1.

6. Construir la tabla de estados internos.

Asignando el valor a la primera hilera de **a** y a la segunda de **b**, el resultado es el siguiente:

	A P 00	01	10	11	Y
a	**E2**	**E0**	**E4**	E1	0
b	E2	**E3**	E4	**E1**	1

Al sustituir los nombres de (E0, E2, E4) → a, (E3, E1) → b, se realiza la tabla de estados internos:

	A P 00	01	10	11	Y
a	**a**	**a**	**a**	b	0
b	a	**b**	a	**b**	1

7. Asignar valores a los estados.

En este ejemplo, la asignación es muy simple: a = 0 y b = 1.

8. Construir la tabla de estados final.

Sustituyendo los valores de la asignación se tiene que:

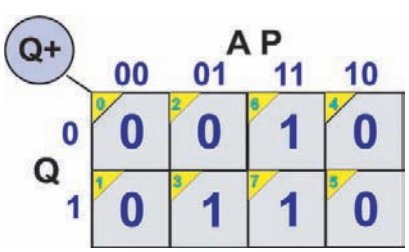
	A P 00	01	10	11	Y
0	**a**	**a**	**a**	b	0
1	a	**b**	a	**b**	1

	A P				
Q1	00	01	10	11	Y
0	0	0	0	1	0
1	0	1	0	1	1

9. Completar tabla de salidas.

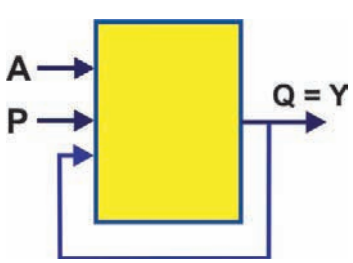
No es necesario completar tabla de salidas.

10. Obtener las ecuaciones mínimas (mapa de Karnaugh).



Y = Q

Q+= P(A+Q)



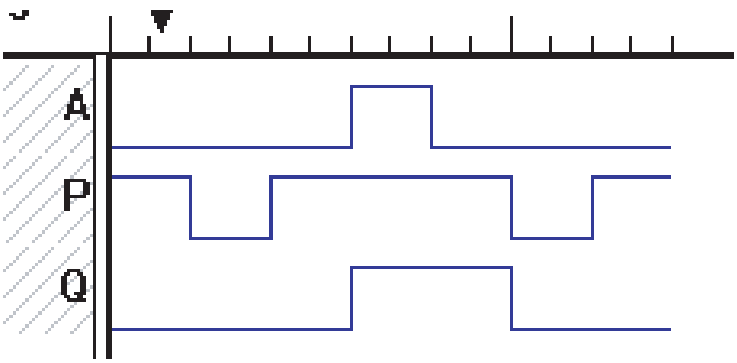
11

Archivo en formato ABEL-HDL:

```
"Q= P(A+Q)
MODULE ap
"entradas
A,P pin 1,2;
"Salidas
Q pin 19 istype 'com';
equations
Q=P&(A#Q);
```

```
Test_Vectors
([A,P]->Q)
[0,1]->.x.;
[0,0]->.x.;
[0,1]->.x.;
[1,1]->.x.;
[0,1]->.x.;
[0,0]->.x.;
[0,1]->.x.;
end
```

11. Realizar la simulación.



13. Efectuar la representación.

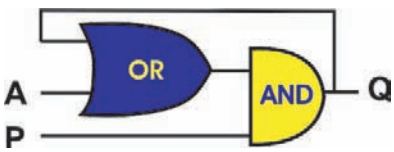


Diagrama esquemático

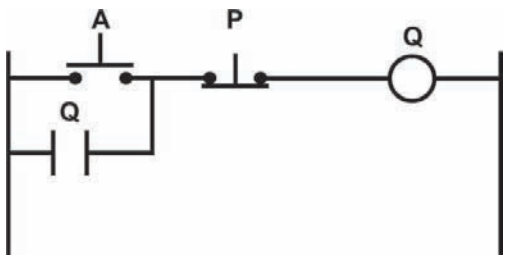
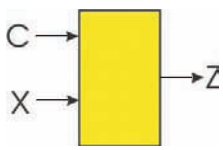


Diagrama escalera

Ejemplo 11.5

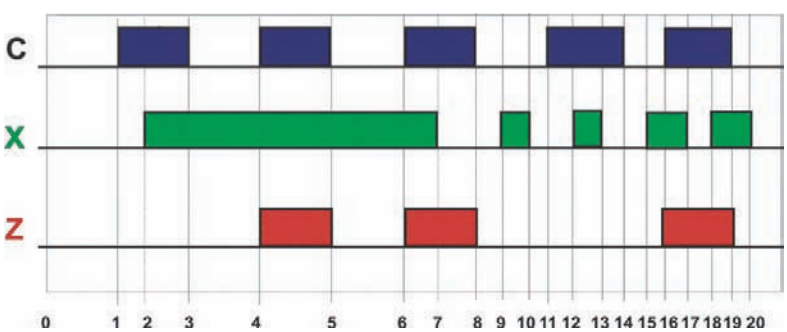
Pulsos completos

Procedimiento



1. Especificar el sistema.

Diseñe un sistema secuencial asíncrono, donde en la salida **Z** aparecerán sólo pulsos completos de una señal **C** cuando la señal **X** tenga un valor de 1, como lo muestra la gráfica del diagrama de tiempos.



2. Construir la tabla de flujo primitiva.

Para obtener la tabla de flujo primitiva se consideran los posibles valores de las entradas **C** y **X**: 00, 01, 11, 10], en el orden descrito; ya que como se puede apreciar, sólo cambia un valor a la vez de C o X, pero no ambos, en columnas consecutivas.

Se inicia la tabla de flujo primitiva con el análisis para el tiempo t = 0.

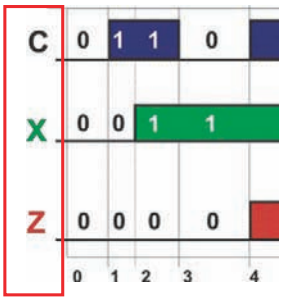
Partiendo de un estado estable inicial **E1**, cuando C X = 00 con salida Z = 0.

Hilera	CX				Salida
	00	**01**	**11**	**10**	**Z**
1	**E1**		**-**		**0**

También es posible incluir un guión (-) en la columna 11, ya que ésta no puede partir de una entrada 00, pues tendría que pasar primero por 01 o 10, pero no 11 (no se permiten dos cambios a la vez).

En *t = 1*:

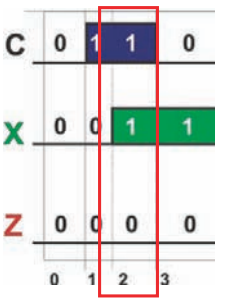
Si del **E1** se presenta una entrada 10, se cambiaría a un estado transitorio E2 en la misma hilera 1 y un estado estable **E2** en una hilera nueva 2, cuya salida también es Z = 0, ya que no se permite en una tabla de flujo primitiva tener dos estados estables en la misma columna.



Hilera	CX				Salida
	00	**01**	**11**	**10**	**Z**
1	**E1**		-	E2	**0**
2		-		**E2**	**0**

En *t = 2:*

Si del **E2** se presenta una entrada 11, se cambiaría a un estado transitorio E3 en la misma hilera 2, y a un estado estable **E3** en una hilera nueva 3, cuya salida es Z = 0.



Hilera	CX				Salida
	00	**01**	**11**	**10**	**Z**
1	**E1**		-	E2	**0**
2		-	E3	**E2**	**0**
3	-		**E3**		**0**

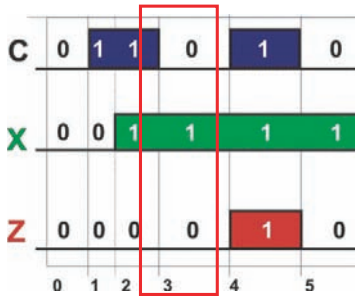
En *t = 3:*

Si del **E3** se presenta una entrada 01, se cambiaría a un estado transitorio E4 en la misma hilera 3, y un estado estable **E4** en una hilera nueva 4, cuya salida también es Z = 0.

Hilera	CX				Salida
	00	**01**	**11**	**10**	**Z**
1	**E1**		-	E2	**0**
2		-	E3	**E2**	**0**
3	-	E4	**E3**		**0**
4		**E4**		-	

En *t = 4:*

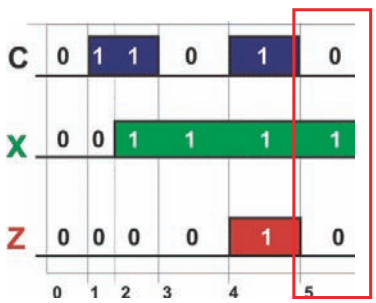
Si del **E4** se presenta una entrada 11, se podría regresar a E3; sin embargo, la salida para este caso es Z = 0 y se requiere un valor de Z =1, por lo que se necesita un estado diferente como transitorio E5 en la misma hilera 4, y un estado estable **E5** en una hilera nueva 5, cuya salida es Z = 1.



Hilera	CX				Salida
	00	**01**	**11**	**10**	**Z**
1	**E1**		-	E2	**0**
2		-	E3	**E2**	**0**
3	-	E4	**E3**		**0**
4		**E4**	E5	-	**0**
5	-		**E5**		**1**

En *t = 5:*

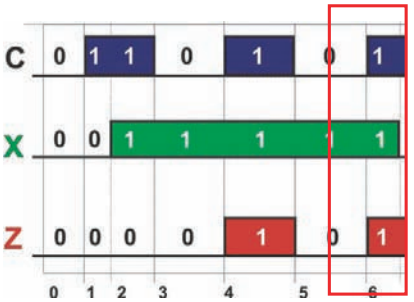
Si del **E5** se presenta una entrada 01, se cambiaría a un estado transitorio E6 en la misma hilera 5 y un estado estable **E6** en una hilera nueva 6, cuya salida es Z = 0.



Hilera	CX				Salida
	00	**01**	**11**	**10**	**Z**
1	**E1**		-	E2	**0**
2		-	E3	**E2**	**0**
3	-	E4	**E3**		**0**
4		**E3**	E5	-	**0**
5	-	E6	**E5**		**1**
6		**E6**		-	**0**

En *t = 6:*

Si del **E6** se presenta una entrada 11 con salida Z = 1, regresará mediante un estado transitorio E5 en la misma hilera 6 a un estado estable **E5** en una hilera 5, cuya salida es Z = 1.

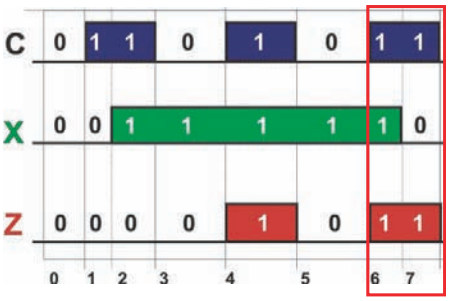


Hilera	CX				Salida
	00	**01**	**11**	**10**	**Z**
1	**E1**		-	E2	**0**
2		-	E3	**E2**	**0**
3	-	E4	**E3**		**0**
4		**E4**	E5	-	**0**
5	-	E6	**E5**		**1**
6		**E6**	E5	-	**0**

En *t = 7:*

Si del **E5** se presenta una entrada 10 con salida Z = 1, cambiará a un estado transitorio E7 en la misma hilera 5 a un estado estable **E7** en una nueva hilera 7, cuya salida es Z = 1.

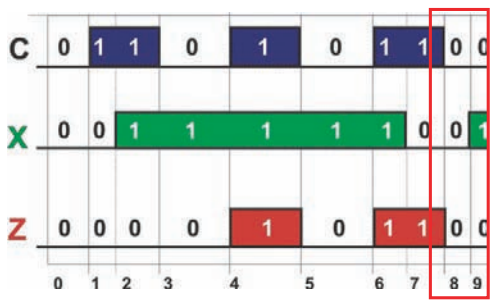
(No es posible regresar a E2 porque la salida es diferente).



Hilera	CX				Salida
	00	01	11	10	Z
1	**E1**		-	E2	0
2		-	E3	**E2**	0
3	-	E4	**E3**		0
4		**E4**	E5	-	0
5	-	E6	**E5**	E7	1
6		**E6**	E5	-	0
7		-		**E7**	1

En *t* = *8:*

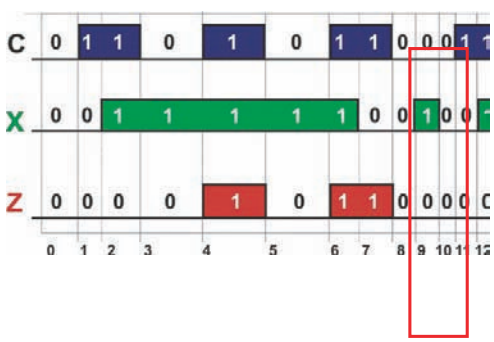
Si del **E7** se presenta una entrada 00 con salida Z = 0, regresará por medio de un estado transitorio E1 en la misma hilera 7 a un estado estable **E1** en una hilera 1, cuya salida es Z = 0.



Hilera	CX				Salida
	00	01	11	10	Z
1	**E1**		-	E2	0
2		-	E3	**E2**	0
3	-	E4	**E3**		0
4		**E4**	E5	-	0
5	-	E6	**E5**	E7	1
6		**E6**	E5	-	0
7	E1	-		**E7**	1

En *t* = *9:*

Si del **E1** se presenta una entrada 01 con salida Z = 0, regresará por medio de un estado transitorio E4 en la misma hilera 1 a un estado estable **E4**, ya existente en una hilera 4.

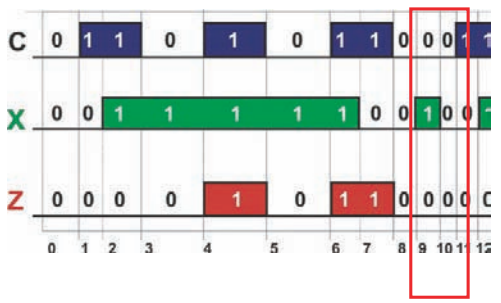


11

Hilera	CX				Salida
	00	**01**	**11**	**10**	**Z**
1	**E1**	E4	-	E2	**0**
2		-	E3	**E2**	**0**
3	-	E4	**E3**		**0**
4		**E4**	E5	-	**0**
5	-	E6	**E5**	E7	**1**
6		**E6**	E5	-	**0**
7	E1	-		**E7**	**1**

En *t* = *10*:

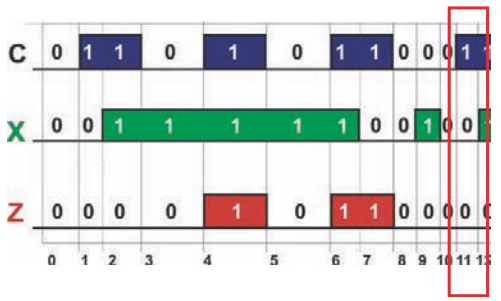
Si del **E4** se presenta una entrada 00 con salida Z = 0, regresará por medio de un estado transitorio E1 en la misma hilera 4 a un estado estable **E1**, ya existente en una hilera 1, cuya salida es Z = 0.



Hilera	CX				Salida
	00	**01**	**11**	**10**	**Z**
1	**E1**	E4	-	E2	**0**
2		-	E3	**E2**	**0**
3	-	E4	**E3**		**0**
4	E1	**E4**	E5	-	**0**
5	-	E6	**E5**	E7	**1**
6		**E6**	E5	-	**0**
7	E1	-		**E7**	**1**

En *t* = *11:*

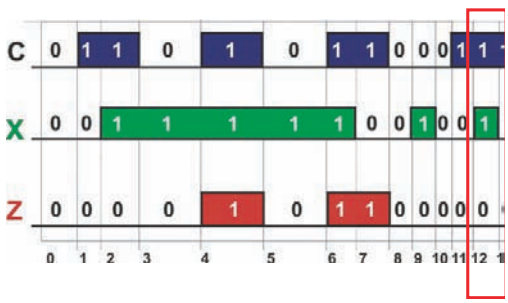
Si del **E1** se presenta una entrada 10 con salida Z = 0, pasaría a un estado estable **E2** ya considerado.



Hilera	CX				Salida
	00	**01**	**11**	**10**	**Z**
1	**E1**	E4	-	E2	**0**
2		-	E3	**E2**	**0**
3	-	E4	**E3**		**0**
4	E1	**E4**	E5	-	**0**
5	-	E6	**E5**	E7	**1**
6		**E6**	E5	-	**0**
7	E1	-		**E7**	**1**

En *t = 12:*

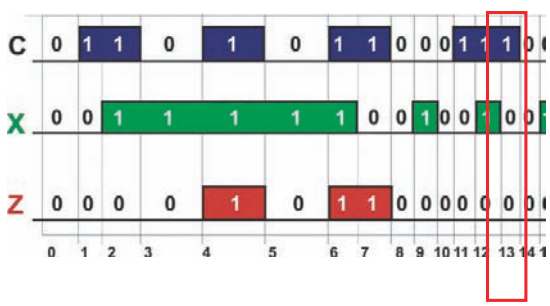
Si del **E2** se presenta una entrada 11 con salida Z = 0, pasaría a un estado estable **E3** ya considerado.



Hilera	CX				Salida
	00	**01**	**11**	**10**	**Z**
1	**E1**	E4	-	E2	**0**
2		-	E3	**E2**	**0**
3	-	E4	**E3**		**0**
4	E1	**E4**	E5	-	**0**
5	-	E6	**E5**	E7	**1**
6		**E6**	E5	-	**0**
7	E1	-		**E7**	**1**

En *t = 13:*

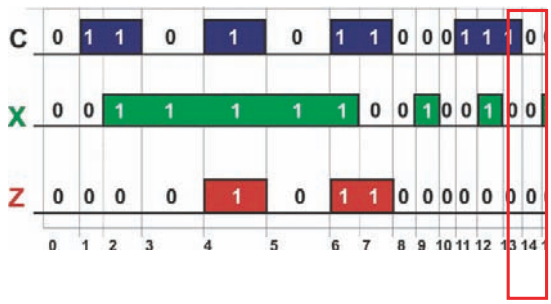
Si del **E3** se presenta una entrada 10 con salida Z = 0, regresará por medio de un estado transitorio E2 en la misma hilera 3 a un estado estable **E2** ya existente en una hilera 2.



Hilera	CX				Salida
	00	**01**	**11**	**10**	**Z**
1	**E1**	E4	-	E2	**0**
2		-	E3	**E2**	**0**
3	-	E4	**E3**	E2	**0**
4	E1	**E4**	E5	-	**0**
5	-	E6	**E5**	E7	**1**
6		**E6**	E5	-	**0**
7	E1	-		**E7**	**1**

En *t = 14:*

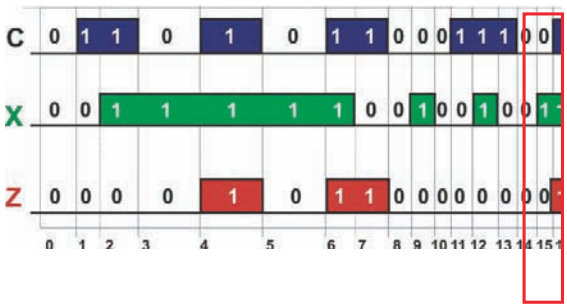
Si del **E2** se presenta una entrada 00 con salida Z = 0, regresará por medio de un estado transitorio E1 en la misma hilera 2 a un estado estable **E1** ya existente en una hilera 1.



Hilera	CX				Salida
	00	**01**	**11**	**10**	**Z**
1	**E1**	E4	-	E2	**0**
2	E1	-	E3	**E2**	**0**
3	-	E4	**E3**	E2	**0**
4	E1	**E4**	E5	-	**0**
5	-	E6	**E5**	E7	**1**
6		**E6**	E5	-	**0**
7	E1	-		**E7**	**1**

En *t = 15:*

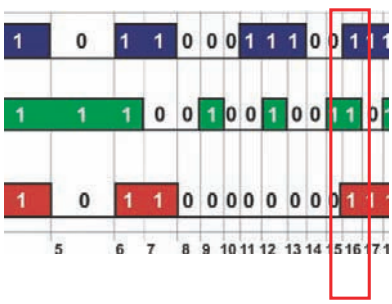
Del **E1** se presenta una entrada 01 con salida Z = 0 a un estado estable **E4** ya considerado en la hilera 4.



Hilera	CX				Salida
	00	**01**	**11**	**10**	**Z**
1	**E1**	E4	-	E2	**0**
2	E1	-	E3	**E2**	**0**
3	-	E4	**E3**	E2	**0**
4	E1	**E4**	E5	-	**0**
5	-	E6	**E5**	E7	**1**
6		**E6**	E5	-	**0**
7	E1	-		**E7**	**1**

En *t = 16:*

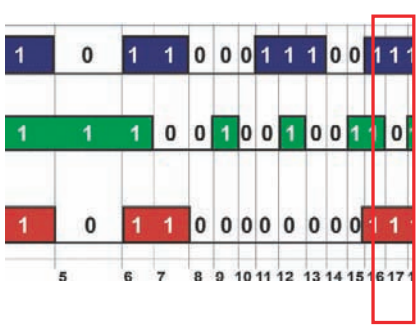
Del **E4** se presenta una entrada 11 con salida Z = 1 a un estado estable **E5** ya considerado en la hilera 5.



Hilera	CX				Salida
	00	**01**	**11**	**10**	**Z**
1	**E1**	E4	-	E2	**0**
2	E1	-	E3	**E2**	**0**
3	-	E4	**E3**	E2	**0**
4	E1	**E4**	E5	-	**0**
5	-	E6	**E5**	E7	**1**
6		**E6**	E5	-	**0**
7	E1	-		**E7**	**1**

En *t = 17:*

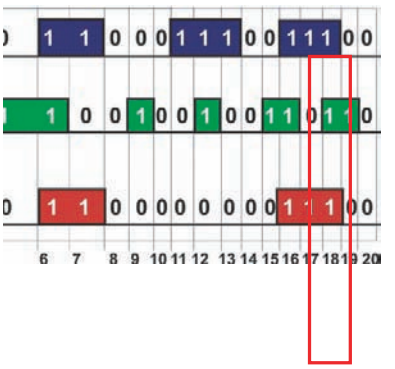
Del **E5** se presenta una entrada 10 con salida Z = 1 a un estado estable **E7** ya considerado en la hilera 7.



Hilera	CX				Salida
	00	**01**	**11**	**10**	**Z**
1	**E1**	E4	-	E2	**0**
2	E1	-	E3	**E2**	**0**
3	-	E4	**E3**	E2	**0**
4	E1	**E4**	E5	-	**0**
5	-	E6	**E5**	E7	**1**
6		**E6**	E5	-	**0**
7	E1	-		**E7**	**1**

En *t = 18:*

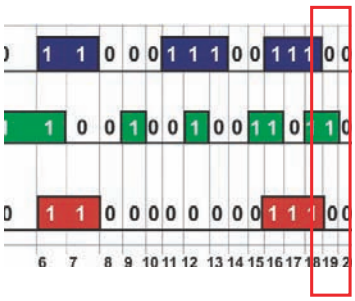
Si del **E7** se presenta una entrada 11 con salida Z = 1 a un transitorio E5 en la misma hilera 7, regresará a un estado estable **E5** ya considerado en la hilera 5.



Hilera	CX				Salida
	00	**01**	**11**	**10**	**Z**
1	**E1**	E4	-	E2	**0**
2	E1	-	E3	**E2**	**0**
3	-	E4	**E3**	E2	**0**
4	E1	**E4**	E5	-	**0**
5	-	E6	**E5**	E7	**1**
6		**E6**	E5	-	**0**
7	E1	-	E7	**E7**	**1**

En *t = 19:*

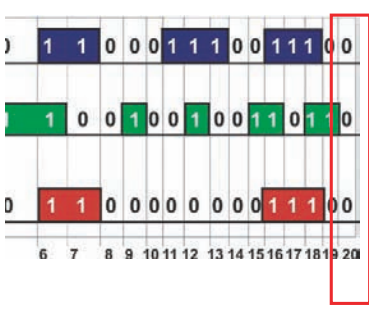
Si del **E5** se presenta una entrada 01 con salida Z = 0, regresará a un estado estable **E6** ya considerado en la hilera 6.



Hilera	CX				Salida
	00	**01**	**11**	**10**	**Z**
1	**E1**	E4	-	E2	**0**
2	E1	-	E3	**E2**	**0**
3	-	E4	**E3**	E2	**0**
4	E1	**E4**	E5	-	**0**
5	-	E6	**E5**	E7	**1**
6		**E6**	E5	-	**0**
7	E1	-	E7	**E7**	**1**

En *t = 20:*

Si del **E6** se presenta una entrada 00 con salida Z = 0, pasaría a un estado transitorio E1 en la misma hilera y después al estado estable **E1** ya considerado en la hilera 6.



Hilera	CX				Salida
	00	**01**	**11**	**10**	**Z**
1	**E1**	E4	-	E2	**0**
2	E1	-	E3	**E2**	**0**
3	-	E4	**E3**	E2	**0**
4	E1	**E4**	E5	-	**0**
5	-	E6	**E5**	E7	**1**
6	E1	**E6**	E5	-	**0**
7	E1	-	E5	**E7**	**1**

Con esto se completa la tabla de flujo primitiva.

3. Eliminar los estados redundantes.

Los estados establecidos en la tabla de flujo primitiva no son necesariamente indispensables, puede haber estados redundantes; por ello, es necesario identificarlos (si los hay) y eliminarlos.

Un estado es redundante si existe uno equivalente. Dos estados son equivalentes si:

a) Son estados estables en la misma columna (misma combinación de entradas).
b) Tienen la misma salida.
c) Sus estados siguientes son equivalentes.

Al analizar el inciso *a*) (son estados estables en la misma columna) se tiene que:

En la columna 00, sólo hay un estado estable (**E1**).
En la columna 01, **E4** y **E6** son estados estables.
En la columna 11, **E3** y **E5** son estados estables.
En la columna 00, **E2** y **E7** son estados estables.

	CX 00	CX 01	CX 11	CX 10	Salida Z
1	**E1**	E4	-	E2	0
2	E1	-	E3	**E2**	0
3	-	E4	**E3**	E2	0
4	E1	**E4**	E5	-	0
5	-	E6	**E5**	E7	1
6	E1	**E6**	E5	-	0
7	E1	-	E7	**E7**	1

Tienen la misma salida:

E4, Z = 0 y **E6**, Z = 0 si tienen la misma salida.
E3, Z = 0 y **E5**, Z = 1 no cumplen por tener salida diferente.
E2, Z = 0 y **E7**, Z = 1 no cumplen por tener salida diferente.

Hilera	CX 00	CX 01	CX 11	CX 10	Salida Z
1	**E1**	E4	-	E2	0
2	E 1	-	E3	**E2**	0
3	-	E4	**E3**	E2	0
4	E 1	**E4**	E5	-	0
5	-	E6	**E5**	E7	1
6	E 1	**E6**	E5	-	0
7	E1	-	E7	**E7**	1

Sus estados siguientes son equivalentes:

El estado siguiente de **E4**, es **E5** para entrada 11.
El estado siguiente de **E4**, es **E1** para entrada 00.
El estado siguiente de **E6**, es **E5** para entrada 11.
El estado siguiente de **E6**, es **E1** para entrada 00.

Por lo tanto, **E4** es equivalente a **E6**.

Al eliminar **E6** se tiene la siguiente tabla de flujo primitiva.

Hilera	CX				Salida
	00	**01**	**11**	**10**	**Z**
1	**E1**	E4	-	E2	**0**
2	E1	-	E3	**E2**	**0**
3	-	E4	**E3**	E2	**0**
4	E1	**E4**	E5	-	**0**
5	-	E6	**E5**	E7	**1**
6	E1	**E6**	E5	-	**0**
7	E1	-	E5	**E7**	**1**

En donde el E6 inestable de la hilera 5, columna 01, sería E4.

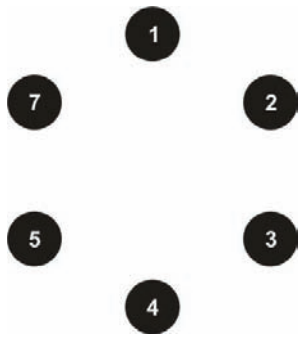
Hilera	CX				Salida
	00	**01**	**11**	**10**	**Z**
1	**E1**	E4	-	E2	**0**
2	E1	-	E3	**E2**	**0**
3	-	E4	**E3**	E2	**0**
4	E1	**E4**	E5	-	**0**
5	-	E4	**E5**	E7	**1**
7	E1	-	E5	**E7**	**1**

4. Efectuar la mezcla de filas.

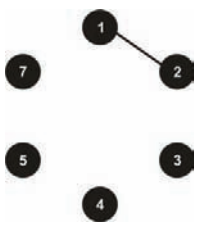
Una vez eliminados los estados redundantes, las filas o hileras pueden mezclarse para reducir la tabla. Dos filas o más se pueden mezclar, siempre y cuando no haya ningún conflicto sobre qué estado debe ocupar cada columna (entendiéndose por conflicto la ocupación simultánea de una columna por dos estados diferentes).

La salida no se considera como un factor de conflicto en la mezcla de filas. Esto es, dos filas con salidas diferentes pueden mezclarse.

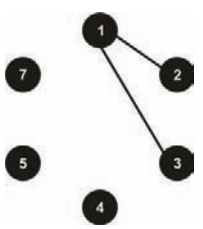
Con el propósito de tener una visualización completa sobre las posibilidades de mezcla de las filas, se construye un diagrama de mezcla en el que se asigna un punto por cada fila, los cuales se unen por líneas, cuando éstos pueden mezclarse.



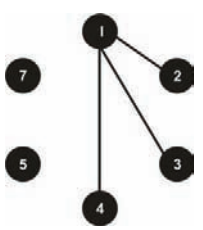
Por ejemplo, las hileras 1 y 2 pueden mezclarse porque en la columna:

00 01 11 10	Tienen igual E1. Sólo existe E4. Sólo hay E3. Es el mismo E2.	

Las hileras 1 y 3 pueden mezclarse porque en la columna:

00 01 11 10	Sólo existe E1. Tienen el mismo E4. Sólo hay E3. Es el mismo E2.	

Las hileras 1 y 4 pueden mezclarse porque en la columna:

00 01 11 10	Tienen igual E1. Tienen el mismo E4. Sólo hay E5. Sólo existe E2.	

Las hileras 1 y 5 no se mezclan porque para la columna 10 tienen diferente estado.
Las hileras 1 y 7 no se mezclan porque para la columna 10 tienen diferente estado.
Las hileras 2 y 3 sí se pueden mezclar.

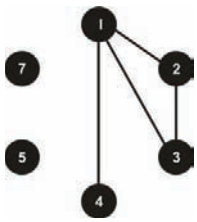
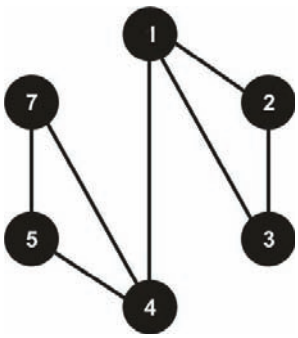


Diagrama de mezcla completo:



Hay dos opciones de mezcla:

a) En grupos de dos E1 con E4, E2 con E3 y E5 con E7.
b) En grupos de tres E1 con E2 y E3, E4 con E5 y E7.

Se tomará en cuenta la segunda opción:

Hilera	CX			
	00	**01**	**11**	**10**
1, 2 y 3	**E1**	E4	**E3**	**E2**
4, 5 y 7	E1	**E4**	**E5**	**E7**

Expandiendo la tabla e incluyendo una columna para cada salida Z, se tiene:

Hilera	CX				Z			
	00	**01**	**11**	**10**	**00**	**01**	**11**	**10**
a	**E1**	E4	**E3**	**E2**	**0**	**-**	**0**	**0**
b	E1	**E4**	**E5**	**E7**	**-**	**0**	**1**	**1**

Se asignará la letra “a” a la hilera 1, 2 y 3, y la letra “b” a la hilera compuesta por 4, 5 y 7.

5. Expandir la tabla de salidas.

Para asignar los valores a las salidas que quedaron sin definir, se toma el siguiente criterio:

Si el paso del estado estable E4 de la hilera “b” columna 01, al estado estable E1 hilera “a” columna 00, se refleja en la parte de salidas y el cambio es de un valor 0 a otro valor 0, por lo que no afecta asignarle el mismo valor de 0.

Hilera	CX				Z			
	00	01	11	10	00	01	11	10
a	**E1**	E4	**E3**	**E2**	0	-	0	0
b	E1	~~E4~~	**E5**	**E7**	**0**	~~0~~	1	1

Hilera	CX				Z			
	00	01	11	10	00	01	11	10
a	**E1**	E4	**E3**	**E2**	0	**0**	0	0
b	E1	**E4**	**E5**	**E7**	**0**	0	1	1

6. Construir la tabla de estados internos.

Para obtener la tabla de estados internos, se sustituyen los estados estables de la hilera “a” por la letra, al igual que los de la hilera “b”.

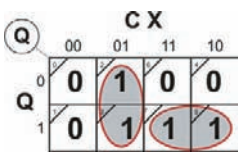
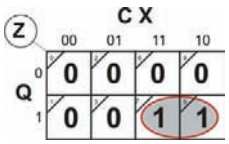
Hilera	CX				Z			
	00	01	11	10	00	01	11	10
a	**a**	b	**a**	**a**	0	0	0	0
b	a	**b**	**b**	**b**	0	0	1	1

7. Asignar valores a los estados.

Asignación de a = 0 y b = 1:

Q	CX				Z			
	00	01	11	10	00	01	11	10
0	**0**	1	**0**	**0**	0	0	0	0
1	0	**1**	**1**	**1**	0	0	1	1

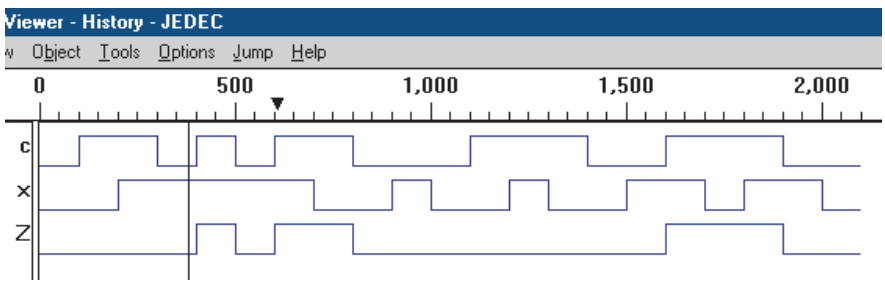
10. Obtener las ecuaciones mínimas.

	**Q = Q C + C' X**
	**Z = Q C**

Archivo en formato ABEL-HDL:

```
MODULE ejeuno
"Entradas
        c,x pin;
"Salidas
        Q, Z pin istype 'com';
Equations
Q= !c & x # c & Q;
Z =c & Q;
Test_Vectors
([c,x,Q]->[Q,Z])
[0,0,0]->[.x.,.x.];
[1,0,.x.]->[.x.,.x.];
[1,1,.x.]->[.x.,.x.];
[0,1,.x.]->[.x.,.x.];
[1,1,.x.]->[.x.,.x.];
[0,1,.x.]->[.x.,.x.];
[1,1,.x.]->[.x.,.x.];
[1,0,.x.]->[.x.,.x.];
[0,0,.x.]->[.x.,.x.];
[0,1,.x.]->[.x.,.x.];
[0,0,.x.]->[.x.,.x.];
[1,0,.x.]->[.x.,.x.];
[1,1,.x.]->[.x.,.x.];
[1,0,.x.]->[.x.,.x.];
[0,0,.x.]->[.x.,.x.];
[0,1,.x.]->[.x.,.x.];
[1,1,.x.]->[.x.,.x.];
[1,0,.x.]->[.x.,.x.];
[1,1,.x.]->[.x.,.x.];
[0,1,.x.]->[.x.,.x.];
[0,0,.x.]->[.x.,.x.];
END
```

11. Realizar la simulación.



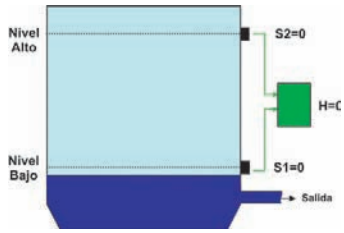
13. Realizar la implementación.

Ejemplo 11.6

Detector de nivel de un tanque

A continuación se muestra un ejemplo de un sistema secuencial asíncrono para detectar el nivel de un tanque con dos sensores llamados S1 (nivel bajo), S2 (nivel alto), que contiene una salida H, de modo que: H = 0 cuando el nivel va de S1 hacia S2 (subida), hasta llegar a S2 y H = 1 cuando el nivel va de S2 hacia S1 (bajada), hasta que llegue a S1.

Observe el diagrama de tiempos.



Tanque con sensores de nivel

Procedimiento

1. Especificar el sistema.

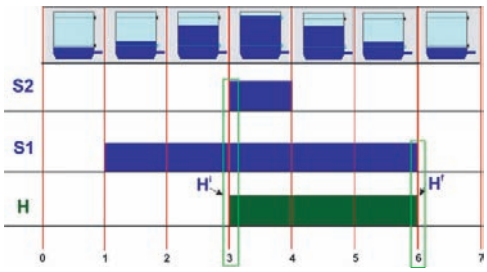


Diagrama de tiempos

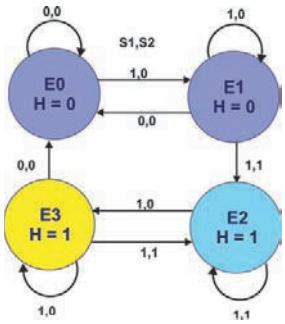


Diagrama de transición

En el diagrama de transición se observa que en el estado E1, si el nivel baja hasta S1 (0,0), el sistema regresa al estado E0; y en el estado E3, si el nivel sube a S2 (1,1), el sistema regresa a E2. Estas transiciones no están contempladas en el diagrama de tiempos.

2. Construir la tabla de flujo primitiva.

En la tabla se considera que el valor de entrada 01 no se puede presentar, ya que implicaría que sólo el sensor S2 detecta el nivel, lo cual no es posible dentro de las condiciones normales de funcionamiento.

	S_1S_2 00	01	11	10	H
1	**E_0**	-	-	E_1	0
2	E_0	-	E_2	**E_1**	0
3	-	-	**E_2**	E_3	1
4	E_0	-	E_2	**E_3**	1

3. Eliminar los estados redundantes.

Para este ejemplo no hay reducción de estados.

4. Efectuar la mezcla de filas.

Al mezclar los renglones 1 con 2, y 3 con 4, además de expandir la salida, se obtiene:

	S_1S_2				H
	00	01	11	10	00
1, 2	**E_0**	-	E_2	**E_1**	0
3, 4	E_0	-	**E_2**	**E_3**	1

5. Expandir la tabla de salidas.

No es necesario expandir tabla de salidas.

6. Construir la tabla de estados internos.

Sustituyendo E0 y E1 por a, y, E2 y E3 por b, ya que en ambos casos son estables en el mismo renglón, da como resultado lo siguiente:

	S_1S_2				H
	00	01	11	10	00
a	**E_0**	-	E_2	**E_1**	0
b	E_0	-	**E_2**	**E_3**	1

	S_1S_2				H
	00	01	11	10	00
a	**a**	-	b	**a**	0
b	a	-	**b**	**b**	1

7. Asignar valores a los estados.

Como a→b y b→a, asignamos los valores a = 0 y b = 1:

8. Construir la tabla de estados final.

Q	S_1S_2				H
	00	01	11	10	00
0	0	-	**1**	0	**0**
1	0	-	**1**	**1**	**1**

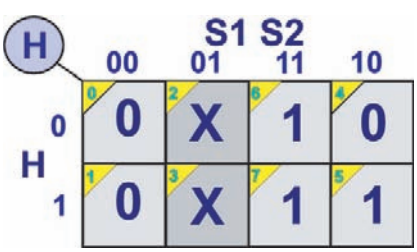
9. Completar tabla de salidas.

No es necesario completar la tabla de salidas.

10. Obtener las ecuaciones mínimas.

Observe en la tabla anterior el valor de $\mathbf{Q^+ = H}$.

Para obtener la ecuación mínima utilice el mapa de Karnaugh.



Ecuaciones: H = S2 + S1H (1) (agrupando unos).

H = (S2 + H) S1 (2) (agrupando ceros).

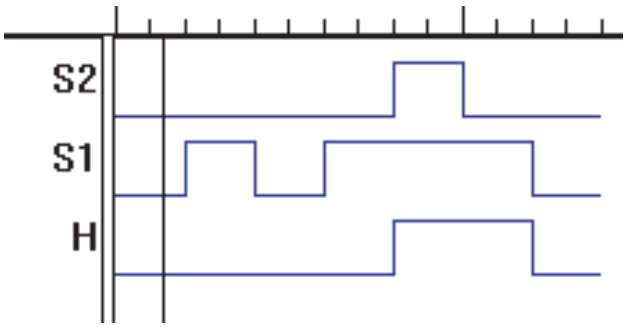
Para elaborar el archivo en formato ABEL-HDL, el costo de implementación de ambos resultados es el mismo, por lo que se elaborará el archivo ABEL-HDL con la ecuación 2.

Archivo en formato ABEL-HDL:

```
MODULE sensor
“Entradas
S1, S2 pin 1,2;
“Salida
H pin 19 istype ‘com’;
equations
H =(S2 # H)& S1;
```

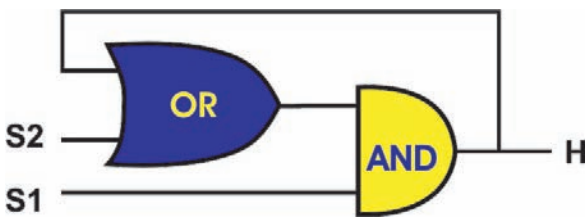
```
Test_Vectors
([S1,S2]->[H])
[0,0]->[.x.];
[1,0]->[.x.];
[0,0]->[.x.];
[1,0]->[.x.];
[1,1]->[.x.];
[1,0]->[.x.];
[0,0]->[.x.];
END
```

11. Realizar la simulación.

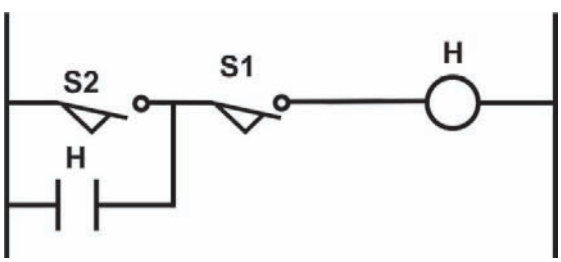


12. Efectuar la representación gráfica.

a) Diagrama esquemático:

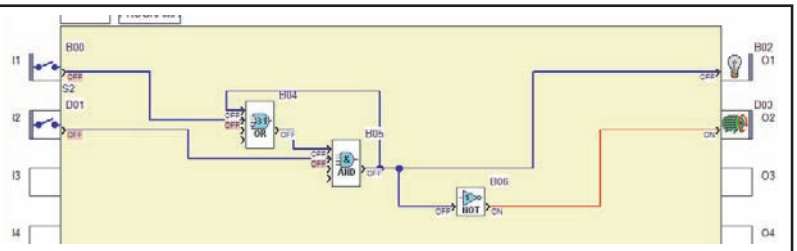
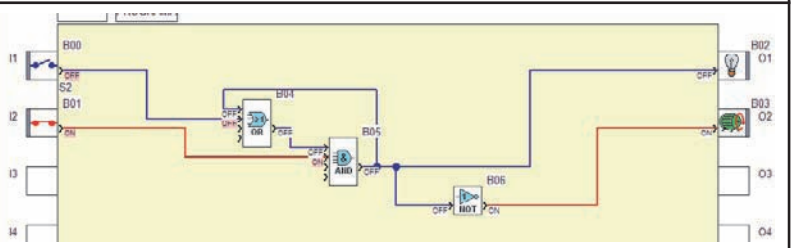
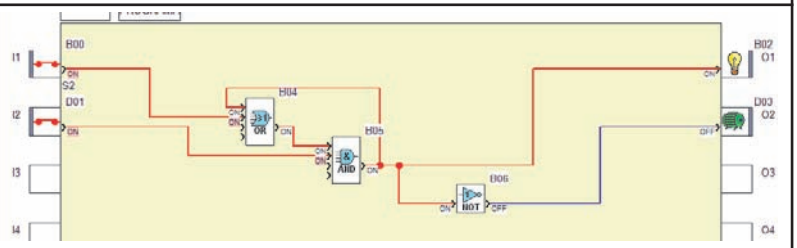
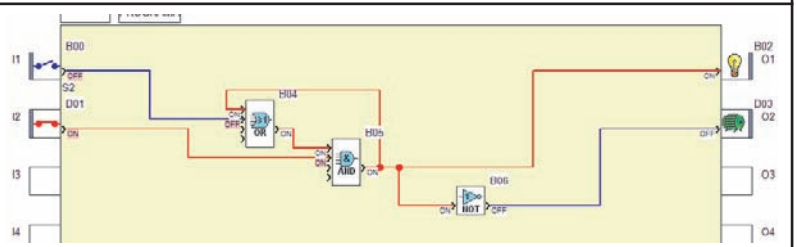


b) Diagrama escalera:



13. Realizar la implementación.

Para la implementación y simulación en un PLD marca Crouzet partiendo del diagrama esquemático, se incluyó una salida para el motor de la bomba que es igual al complemento de H; de modo que cuando H = 0, el motor está trabajando, y cuando H = 1, el motor está apagado.

Evento 1 S1 = 0 y S2 = 0 Motor encendido y H = 0	
Evento 2 S1 = 1 y S2 = 0 Motor encendido y H = 0	
Evento 3 S1 = 1 y S2 = 1 Motor apagado y H = 1	
Evento 4 S1 = 1 y S2 = 0 Motor apagado y H = 1	

Ejemplo 11.7

Sistema alternativo de dos bombas no simultáneas

Diseñe un sistema secuencial que controle el llenado de un tanque con las siguientes características:

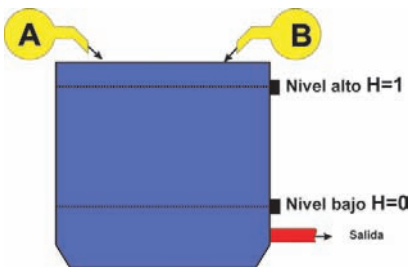
a) El sistema consta de dos bombas: **"A"** y **"B**".

b) Tiene un sensor de nivel **"H"** que indica con H = 1, tanque lleno; y con H = 0, tanque vacío (como se describió en el ejercicio anterior).

c) Partiendo de que el tanque está vacío (H = 0), el llenado deberá iniciarse encendiendo la bomba **"A"** hasta llenar el tanque (H = 1), y entonces se apagará.

d) Si de nuevo se vacía el tanque (H = 0), el llenado deberá hacerse encendiendo la bomba **"B"** hasta que se llene el tanque (H = 1), y posteriormente se apaga.

e) Si se vacía nuevamente el tanque, el llenado deberá hacerse con la bomba **"A"**, y así sucesivamente, con la finalidad de que las bombas alternen su funcionamiento.



Procedimiento

1. Especificar el sistema.

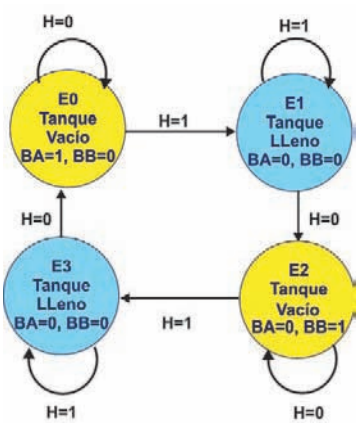


Diagrama de transición

2. Construir la tabla de flujo primitiva.

	Entrada H		Salidas	
	0	1	A	B
1	$\underline{E_0}$	E_1	1	0
2	E_2	$\underline{E_1}$	0	0
3	$\underline{E_2}$	E_3	0	1
4	E_0	$\underline{E_3}$	0	0

3. Eliminar los estados redundantes.

No hay reducción de estados.

4. Efectuar la mezcla de filas.

No es posible mezclar filas.

	Entrada H		Salidas		
	0	**1**	**A**	**B**	
1	**$\underline{E_0}$**	E_1	**1**	**0**	a
2	E_2	**$\underline{E_1}$**	**0**	**0**	b
3	**$\underline{E_2}$**	E_3	**0**	**1**	c
4	E_0	**$\underline{E_3}$**	**0**	**0**	d

5. Expandir la tabla de salidas.

No es necesario expandir la tabla de salidas.

6. Construir la tabla de estados internos.

Sustituyendo los nombres de los estados por las variables propuestas se obtiene:

	Entrada H		Salidas	
	0	**1**	**A**	**B**
a	**a**	b	**1**	**0**
b	c	**b**	**0**	**0**
c	**c**	d	**0**	**1**
d	a	**d**	**0**	**0**

7. Asignar valores a los estados.

Requisitos: Columna 0) (b → c), (d → a)

Columna 1) (a → b), (c → d),

		Q2	
		0	**1**
Q1	**0**	**a**	**d**
	1	**b**	**c**

Asignación:

a = 00, b = 01, c = 11, d = 10

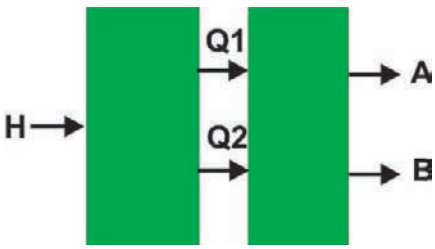
8. Construir la tabla de estados final.

Sustituyendo las variables por los valores asignados se tiene:

Q1 Q2	Entrada H		Salidas	
	0	1	A	B
00	00	01	1	0
01	11	01	0	0
11	11	10	0	1
10	00	10	0	0

9. Completar tabla de salidas.

No es necesario completar tabla de salidas.



10. Obtener las ecuaciones mínimas.

a) Tabla de estado siguiente:

m	H	Q1	Q2	Q1+	Q2+
0	0	0	0	0	0
1	0	0	1	1	1
2	0	1	0	0	0
3	0	1	1	1	1
4	1	0	0	0	1
5	1	0	1	0	1
6	1	1	0	1	0
7	1	1	1	1	0

Tabla de salidas:

m	Q1	Q2	A	B
0	0	0	1	0
1	0	1	0	0
2	1	0	0	0
3	1	1	0	1

Ecuaciones obtenidas por medio de LogicAid:

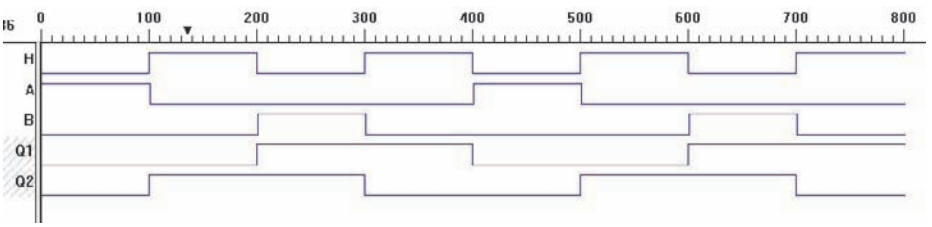
Q1+ = H'Q2 + H Q1
Q2+ = H'Q2 + H Q1'
A = Q1' Q2'
B = Q1 Q2

Archivo en formato ABEL-HDL:

```
MODULE dbayb
"Entrada
H pin 1;
"Salidas
Q1,Q2,A,B pin 19..16 istype 'com';
equations
Q1=!H&Q2#H&Q1;
Q2=!H&Q2#H&!Q1;
A=!Q1&!Q2;
B=Q1&Q2;
```

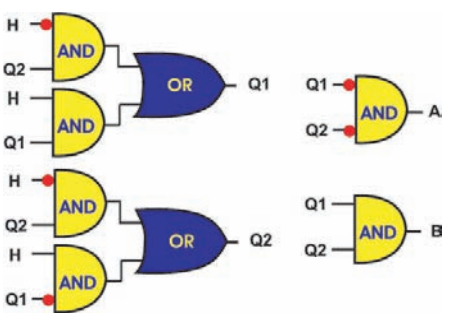
```
Test_Vectors
(H->[A,B])
0->[.x.,.x.];
1->[.x.,.x.];
0->[.x.,.x.];
1->[.x.,.x.];
0->[.x.,.x.];
1->[.x.,.x.];
0->[.x.,.x.];
1->[.x.,.x.];
END
```

11. Realizar la simulación.



12. Efectuar la representación gráfica.

a) Diagrama esquemático:



b) Diagrama escalera:

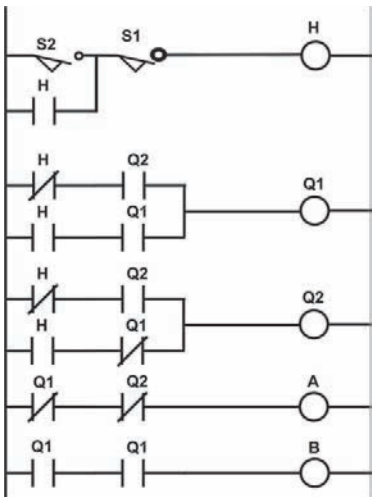
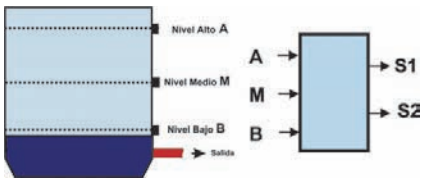


Diagrama escalera incluyendo sensor de nivel

Ejemplo 11.8

Detector de tres niveles

Diseñe un sistema secuencial asíncrono para la detección del nivel de un tanque, para que por medio de tres sensores, llamados **B** (bajo), **M** (medio) y **A** (alto), se obtengan las salidas S1 y S2 con los valores presentados en la siguiente gráfica. Descripción del tanque y sus sensores de nivel



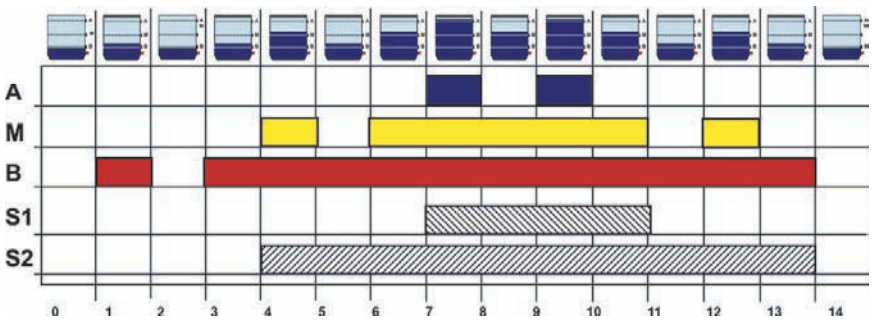
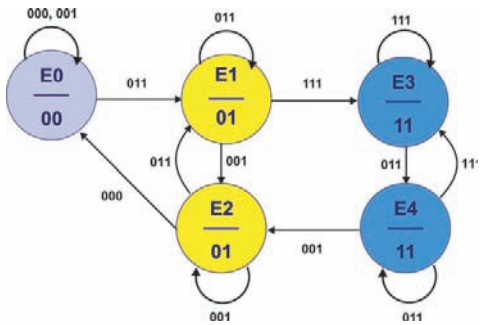


Diagrama de tiempos

Procedimiento

1. Especificar el sistema.

Del comportamiento, descrito en el diagrama de tiempos, se propone el siguiente diagrama de transición:



2. Construir la tabla de flujo primitiva.

	Entradas AMB								Salidas	
	000	001	010	011	100	101	110	111	S1	S2
1	**E0**	**E0**	-	E1	-	-	-	-	**0**	**0**
2	-	E2	-	**E1**	-	-	-	E3	**0**	**1**
3	E0	**E2**	-	E1	-	-	-	-	**0**	**1**
4	-	-	-	E4	-	-	-	**E3**	**1**	**1**
5	-	E2		**E4**				E3	**1**	**1**

3. Eliminar los estados redundantes.

No es posible la reducción de estados.

4. Efectuar la mezcla de filas.

Al mezclar las filas 2 con 3, y 4 con 5 da como resultado:

	Entradas AMB								Salidas	
	000	001	010	011	100	101	110	111	S1	S2
1	**E0**	**E0**	-	E1	-	-	-	-	**0**	**0**
2, 3	E0	**E2**	-	**E1**	-	-	-	E3	**0**	**1**
4, 5	-	E2	-	**E4**	-	-	-	**E3**	**1**	**1**

Como las salidas son las mismas en cada fila, no es necesario expandir la tabla de salidas.

5. Expandir la tabla de salidas.

Como vimos, no es necesario expandir salidas, ya que Q1 = S1 y Q2 = S2.

6. Construir la tabla de estados internos.

Sustituyendo E0 por "a", E1 y E2 por "b" y, E3 y E4 por "c" se obtiene:

	Entradas AMB								Salidas	
	000	001	010	011	100	101	110	111	S1	S2
a	**a**	**a**	-	b	-	-	-	-	**0**	**0**
b	a	**b**	-	**b**	-	-	-	c	**0**	**1**
c	-	b	-	**c**	-	-	-	**c**	**1**	**1**

7. Asignar valores a los estados.

Requisitos: (a → b), (b → c), (c → b)

Q1 \ Q2	0	1
0	a	
1	b	c

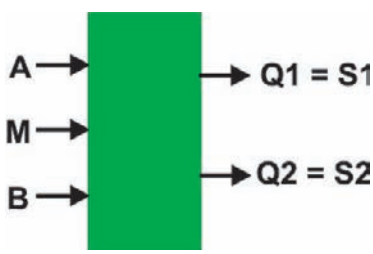
Asignación: a = 00, b = 01, c = 11.

8. Construir la tabla de estados.

	Entradas AMB								Salidas	
Q1 Q2	000	001	010	011	100	101	110	111	S1	S2
00	**00**	**00**	-	01	-	-	-	-	**0**	**0**
01	00	**01**	-	**01**	-	-	-	11	**0**	**1**
11	-	01	-	**11**	-	-	-	**11**	**1**	**1**
10	-	-	-	-	-	-	-	-	-	-

9. Completar la tabla de salidas.

No es necesario completar la tabla de salidas.



10. Obtener las ecuaciones mínimas.

	Entradas					Salidas	
m	A	M	B	Q1	Q2	Q1+	Q2+
0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0
2,3	0	0	0	0	X	X	X
4	0	0	1	0	0	0	0
5	0	0	1	0	1	0	1
6	0	0	1	1	0	X	X
7	0	0	1	1	1	0	1
8 a 11	0	1	0	X	X	X	X
12	0	1	1	0	0	0	1
13	0	1	1	0	1	0	1
14	0	1	1	1	0	X	X
15	0	1	1	1	1	1	1
16 a 27	1	1	0	X	X	X	X
28	1	1	1	0	0	X	X
29	1	1	1	0	1	1	1
30	1	1	1	1	0	X	X
31	1	1	1	1	1	1	1

Ecuaciones:

Q1+ = M Q1 + A
Q2+ = B Q2 + M

m	Q1	Q2	S1	S2
0	0	0	0	0
1	0	1	0	1
2	1	0	X	X
3	1	1	1	1

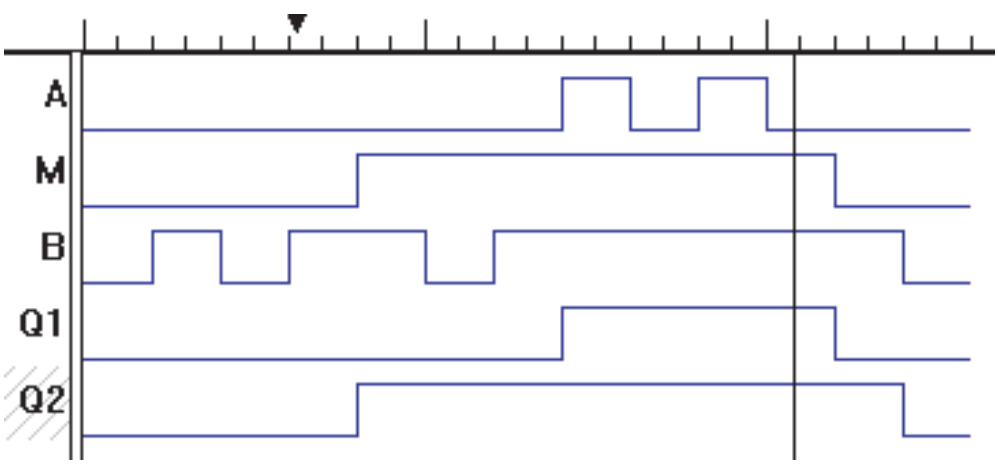
S1 = Q1

S2 = Q2

Archivo en formato ABEL-HDL:

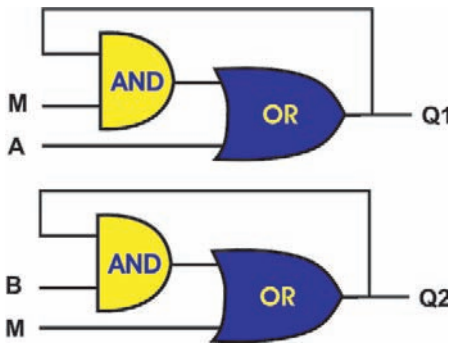
```
MODULE tnivasy
“entradas
A,M,B pin 1..3;
“Salidas
Q1,Q2 pin 19,18 istype ‘com’;
equations
Q1=M&Q1#A;
Q2=B&Q2#M;

Test_Vectors
([A,M,B]->[Q1,Q2])
[0,0,0]->[.x.,.x.];
[0,0,1]->[.x.,.x.];
[0,0,0]->[.x.,.x.];
[0,0,1]->[.x.,.x.];
[0,1,1]->[.x.,.x.];
[0,1,0]->[.x.,.x.];
[0,1,1]->[.x.,.x.];
[1,1,1]->[.x.,.x.];
[0,1,1]->[.x.,.x.];
[1,1,1]->[.x.,.x.];
[0,1,1]->[.x.,.x.];
[0,0,1]->[.x.,.x.];
[0,0,0]->[.x.,.x.];
END
```

11. Realizar la simulación.

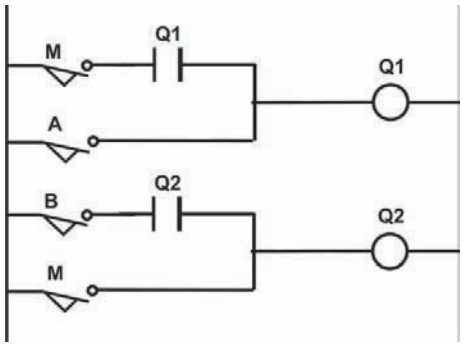


12. Realizar la representación.

a) Diagrama esquemático:



b) Diagrama escalera:



Ejemplo 11.9

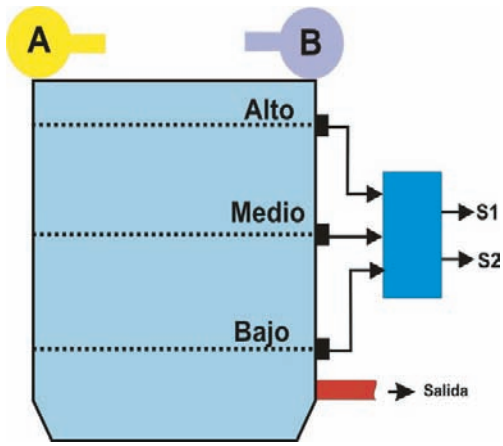
Dos bombas simultáneas (1)

El tanque de la figura se alimenta con dos bombas llamadas **A** y **B**.

El gasto de salida nunca será mayor al que proporcionen las dos bombas operando simultáneamente.

El tanque tiene un sistema detector de niveles, que consta de tres sensores de entrada: nivel **A** (alto), **M** (medio) y **B** (bajo); y dos salidas, **S2, S1**, que indican lo siguiente:

	S2	S1
Bajo	0	0
Medio	0	1
Alto	1	1



Diseñar un sistema secuencial asíncrono que controle la siguiente secuencia de operación de las bombas, en función de la salida del sistema de detección de niveles S2, S1.

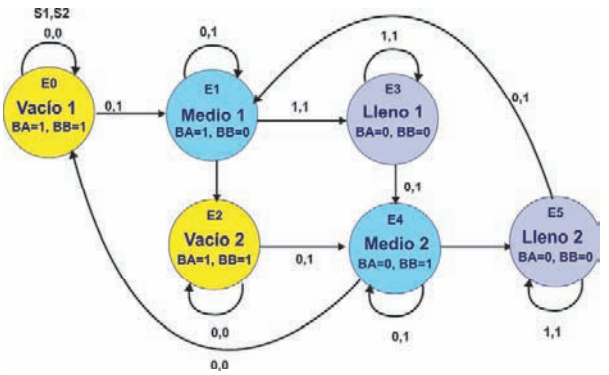
Trabajo solicitado

1. Partiendo de que el tanque está vacío (S2 = 0 y S1 = 0), se inicia el llenado al hacer funcionar ambas bombas, **A** y **B**, hasta llegar al nivel medio (S2 = 0 y S1 = 1) y de ahí sólo trabajará la bomba **A**. Si se vacía de nuevo (S2 = 0 y S1 = 0) encenderán otra vez ambas bombas, y al llegar al nivel medio (S2 = 0 y S1 = 1) solamente trabajará la bomba **B**.
2. Cada vez que se vacíe el tanque (S2 = 0 y S1 = 0) y pase al nivel medio (S2 = 1 y S1=0), deberá de alternarse el funcionamiento de las bombas **A** y **B**.
3. Si el tanque se llena (S2 = 1 y S1= 1), las bombas deberán apagarse.
4. Cada vez que se llene el tanque (S2 = 1 y S1 = 1) y continúe al nivel medio (S2 = 1 y S1 = 0), deberá alternarse el funcionamiento de las bombas A y B.

Procedimiento

1. Especificar el sistema.

Diagrama de transición:



2. Construir la tabla de flujo primitiva.

	Entradas S2, S1				Salidas	
Hileras	00	01	11	10	A	B
1	**E0**	E1	-	-	1	1
2	E2	**E1**	E3	-	1	0
3	**E2**	E4	-	-	1	1
4	-	E4	**E3**	-	0	0
5	E0	**E4**	E5	-	0	1
6	-	E1	**E5**	-	0	0

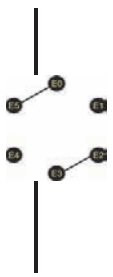
3. Eliminar los estados redundantes.

a) (E0, E2) (E1, E4)(E3, E5) son estados estables en la misma columna.

b) (E0, E2) (E3, E5) tienen la misma salida.

c) Sus estados siguientes no son equivalentes, por lo tanto, no hay reducción de estados.

4. Efectuar la mezcla de filas y **5.** expandir la tabla de salidas.



	S1, S2				A, B			
Hileras	00	01	11	10	00	01	11	10
1,6	E0	E1	E5	-	11	--	00	--
2	E2	E1	E3	-	--	10	--	--
3,4	E2	E4	E3	-	11	--	00	--
5	E0	E4	E5	-	--	01	--	--

6. Construir la tabla de estados internos.

	S1, S2				A, B			
Hileras	00	01	11	10	00	01	11	10
a	a	b	a	-	11	--	00	--
b	c	b	c	-	--	10	--	--
c	c	d	c	-	11	--	00	--
d	a	d	a	-	--	01	--	--

7. Asignar valores a los estados.

Requisitos: 00) (b → c), (d → a)
01) (a → b), (c → d)
11) (b → c), (d → a)

Q1 \ Q2	0	1
0	a	d
1	b	c

	Q1	Q2
a	0	0
b	0	1
c	1	1
d	1	0

11

8. Construir la tabla de estados final.

	S2 ,S1				A, B			
Q1 Q"	00	01	11	10	00	01	11	10
00	**00**	01	**00**	- -	11	- -	00	--
01	11	**01**	11	- -	- -	10	- -	--
11	**11**	10	**11**	- -	11	- -	00	--
10	00	**10**	00	- -	- -	01	- -	--

9. Completar la tabla de salidas.

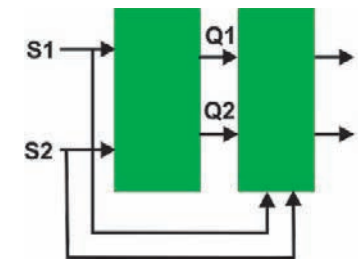
	S2, S1				A, B			
Q1 Q"	00	01	11	10	00	01	11	10
00	**00**	01	**00**	- -	11	**10**	00	--
01	11	**01**	11	- -	**1 -**	10	**- 0**	--
11	**11**	10	**11**	- -	11	**01**	00	--
10	00	**10**	00	- -	**- 1**	01	**0 -**	--

10. Obtener las ecuaciones mínimas (utilizando LogicAid).

Se obtiene:

Ecuaciones:

Q1+ = S1'Q2 + S2'S1 Q1 + S2 Q2
Q2+ = S1'Q2 + S2'S1 Q1' + S2 Q2
A = S1' + S2' Q1'
B = S1' + S2' Q1

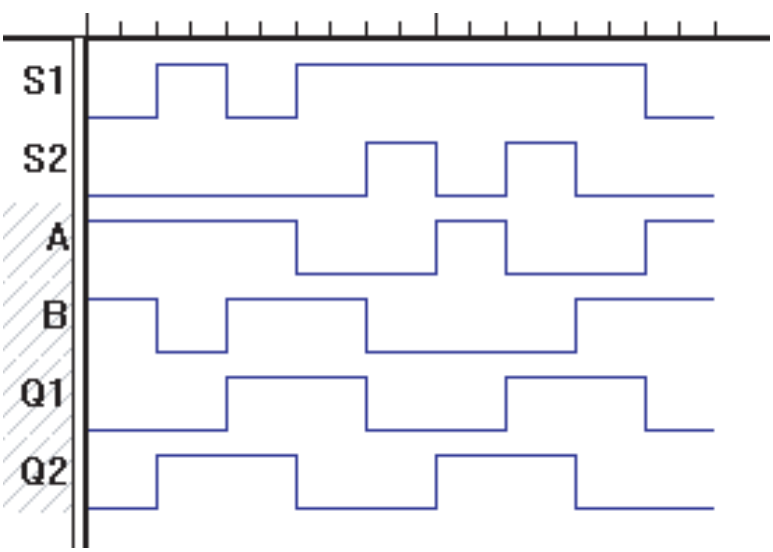


Archivo en formato ABEL-HDL, incluyendo la simulación:

```
MODULE basinc
"Entradas
S1,S2 pin 1,2;
"salidas
A,B,Q1,Q2 pin 19..16 istype 'com';
```

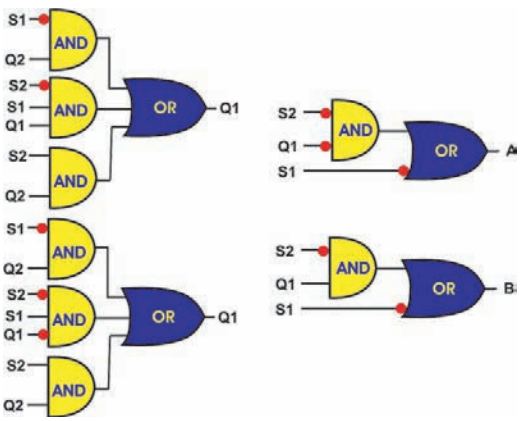
```
equations
Q1=!S1&Q2#!S2&S1&Q1#S2&Q2;
Q2=!S1&Q2#!S2&S1&!Q1#S2&Q2;
A=!S1#!S2&!Q1;
B=!S1#!S2&Q1;
Test_Vectors
([S1,S2]->[A,B])
[0,0]->[.x.,.x.];
[1,0]->[.x.,.x.];
[0,0]->[.x.,.x.];
[1,0]->[.x.,.x.];
[1,1]->[.x.,.x.];
[1,0]->[.x.,.x.];
[1,1]->[.x.,.x.];
[1,0]->[.x.,.x.];
[0,0]->[.x.,.x.];
END
```

11. Realizar la simulación.

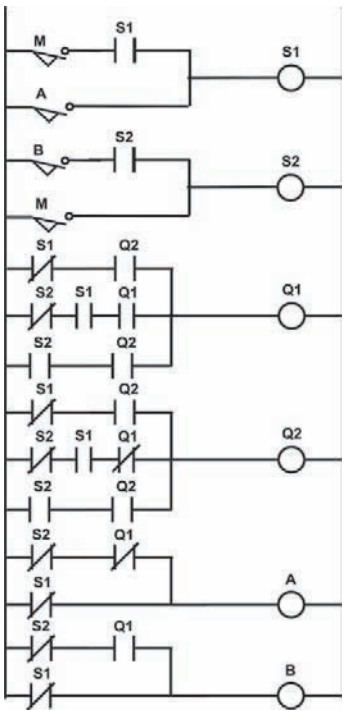


12. Realizar la representación gráfica:

a) Diagrama esquemático:



b) Diagrama escalera:



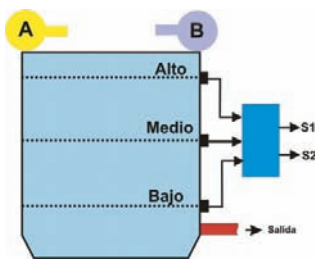
Dos bombas simultáneas (2)

El tanque de la figura se alimenta con dos bombas: **A** y **B**.

El gasto de salida nunca será mayor al que proporcionen las dos bombas operando simultáneamente.

El tanque tiene un sistema detector de niveles que consta de tres sensores de entrada nivel A (alto), M (medio) y B (bajo), y dos salidas **S2**, **S1** que indican lo siguiente:

	S2	S1
Bajo	0	0
Medio	0	1
Alto	1	1



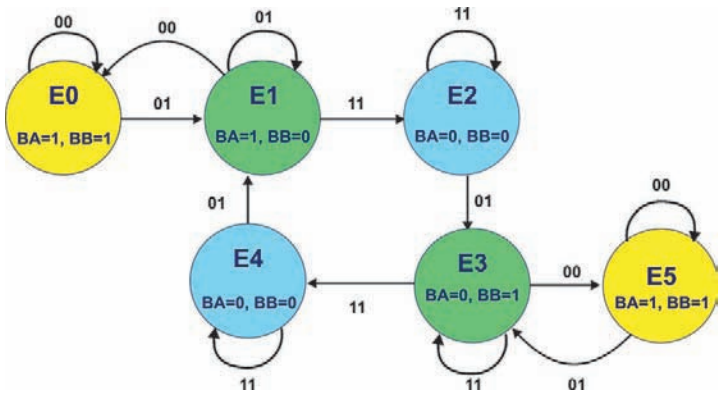
Trabajo solicitado

Diseñar un sistema secuencial asíncrono que controle la siguiente secuencia de operación de las bombas, pero en función de la salida del sistema de detección de niveles S2, S1.

1. Partiendo de que el tanque está vacío (S2 = 0 y S1 = 0), se inicia el llenado con ambas bombas **A** y **B** hasta llegar al nivel medio (S2 = 0 y S1 = 1). A partir de este momento sólo trabajará la bomba **A**; si el tanque se vacía encenderán ambas bombas, y al llegar al nivel medio (S2 = 0 y S1 = 1) operará nuevamente la bomba **A**. Cuando el tanque se llene (S2 = 1 y S1 = 1) se deberán apagar ambas bombas. Al llegar otra vez al nivel medio (S2 = 0 y S1 = 1), trabajará solamente la bomba B, y si no es suficiente, vaciándose el tanque, operarán ambas bombas hasta llegar de nuevo al nivel medio (S2 = 0 y S1 = 1), y de ahí trabajará de nuevo sólo la bomba B hasta llenarlo (S2 = 1 y S1 = 1).
2. Cada vez que se llene el tanque (S2 = 1 y S1 = 1), y pase al nivel medio (S2 = 1 y S1 = 0), deberá trabajar una sola bomba alternándose en su funcionamiento.
3. Las bombas no se alternarán en su funcionamiento cuando el nivel pase de bajo a medio.

Procedimiento

1. Especificar el sistema.



2. Construir la tabla de flujo primitiva.

	Entradas S2, S1				Salidas	
Hileras	00	01	11	10	A	B
1	**E0**	E1	-	-	1	1
2	E0	**E1**	E2	-	1	0
3	-	E3	**E2**	-	0	0
4	E5	**E3**	E4	-	0	1
5	-	E1	**E4**	-	0	0
6	**E5**	E3	-		1	1

3. Eliminar los estados redundantes.

No hay reducción de estados.

4. Efectuar la mezcla de filas y **5.** expandir la tabla de salidas.

	Entradas S2, S1				Salidas A, B			
Hileras	00	01	11	10	00	01	11	10
1, 2	**E0**	**E1**	E2	-	11	10	-	-
3, 6	**E5**	E3	**E2**	-	11	-	00	-
4	E5	**E3**	E4	-	-	01	-	-
6	-	E1	**E4**		-	-	00	

6. Construir la tabla de estados internos.

	Entradas S2, S1				Salidas A, B			
Hileras	00	01	11	10	00	01	11	10
a	E0	E1	E2	-	11	10	-	-
b	E5	E3	E2	-	11	-	00	-
c	E5	E3	E4	-	-	01	-	-
d	-	E1	E4		-	-	00	

	Entradas S2, S1				Salidas A, B			
Hileras	00	01	11	10	00	01	11	10
a	a	a	b	-	11	10	-	-
b	b	c	b	-	11	-	00	-
c	b	c	d	-	-	01	-	-
d	-	a	d		-	-	00	

7. Asignar valores a los estados.

Requisitos: 00) (b → c)

01) (b → c), (d → a)

11) (a → b), (c → d)

		Q2	
		0	1
Q1	0	b	a
	1	c	d

	Q1	Q2
b	0	0
c	0	1
d	1	1
a	1	0

8. Construir la tabla de estados final.

	Entradas S2, S1				Salidas A, B			
Hileras	00	01	11	10	00	01	11	10
10	10	10	00	-	11	10	-	-
00	00	01	00	-	11	-	00	-
01	00	01	11	-	-	01	-	-
11	-	10	11		-	-	00	

Al acomodar en orden las hileras se tiene que:

	Entradas S2, S1				Salidas A, B			
Hileras	00	01	11	10	00	01	11	10
00	00	01	00	-	11	-	00	-
01	00	01	11	-	-	01	-	-
11	-	10	11		-	-	00	
10	10	10	00	-	11	10	-	-

9. Completar la tabla de salidas.

	Entradas S2, S1				Salidas A, B			
Hileras	00	01	11	10	00	01	11	10
00	00	01	00	-	11	01	00	-
01	00	01	11	-	11	01	0X	-
11	-	10	11		-	X0	00	
10	10	10	00	-	11	10	X0	-

10. Obtener las ecuaciones mínimas.

	Entradas				Salidas			
m	S2	S1	Q1	Q2	Q1+	Q2+	A	B
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	0	X	1
2	0	0	1	0	1	0	1	1
3	0	0	1	1	X	X	X	X
4	0	1	0	0	0	1	0	1
5	0	1	0	1	0	1	0	1
6	0	1	1	0	1	0	1	0
7	0	1	1	1	1	0	X	0
8	1	0	0	0	X	X	X	X
9	1	0	0	1	X	X	X	X
10	1	0	1	0	X	X	X	X
11	1	0	1	1	X	X	X	X
12	1	1	0	0	0	0	0	0
13	1	1	0	1	1	1	0	X
14	1	1	1	0	0	0	0	0
15	1	1	1	1	1	1	X	0

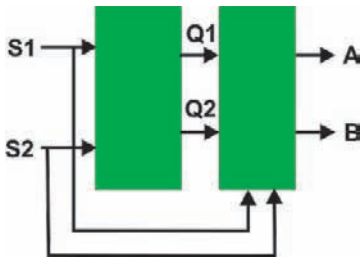
Ecuaciones:

Q1+ = S1'Q1 + S1 Q2

Q2+ = S1'S2 Q1' + S1 Q2

A = S2' + S1'Q1

B = S2' + S1'Q1'

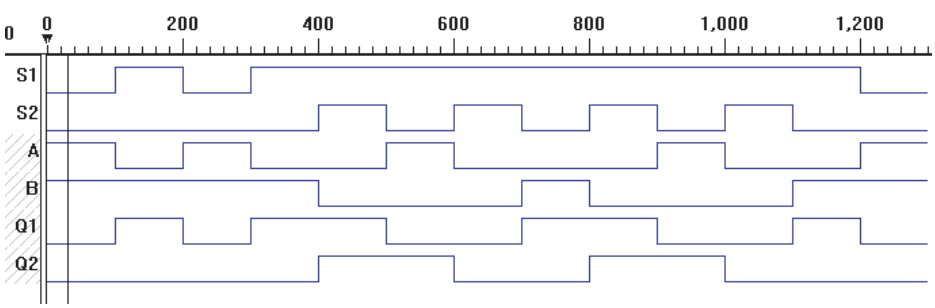


Archivo en formato ABEL-HDL:

```
MODULE ASINCT
"Entradas
S1,S2 pin 1,2;
"salidas
A,B,Q1,Q2 pin 19..16 istype 'com';
equations
Q2=!S2&Q2#S2&Q1;
Q1=!S2&S1&!Q2#S2&Q1;
A=!S1#!S2&Q2;
B=!S1#!S2&!Q2;
```

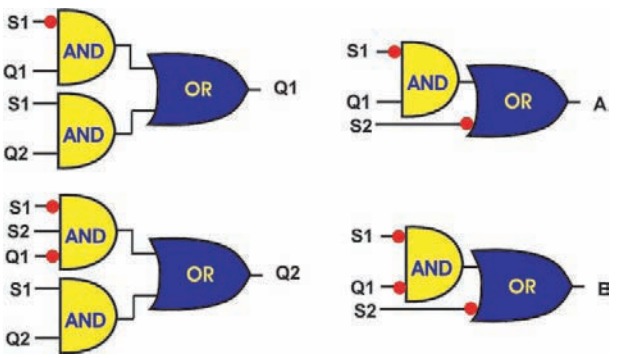
```
Test_Vectors
([S1,S2]->[A,B])
[0,0]->[.x.,.x.];
[1,0]->[.x.,.x.];
[0,0]->[.x.,.x.];
[1,0]->[.x.,.x.];
[1,1]->[.x.,.x.];
[1,0]->[.x.,.x.];
[1,1]->[.x.,.x.];
[1,0]->[.x.,.x.];
[1,1]->[.x.,.x.];
[1,0]->[.x.,.x.];
[1,1]->[.x.,.x.];
[1,0]->[.x.,.x.];
[0,0]->[.x.,.x.];
END
```

11. Realizar la simulación.

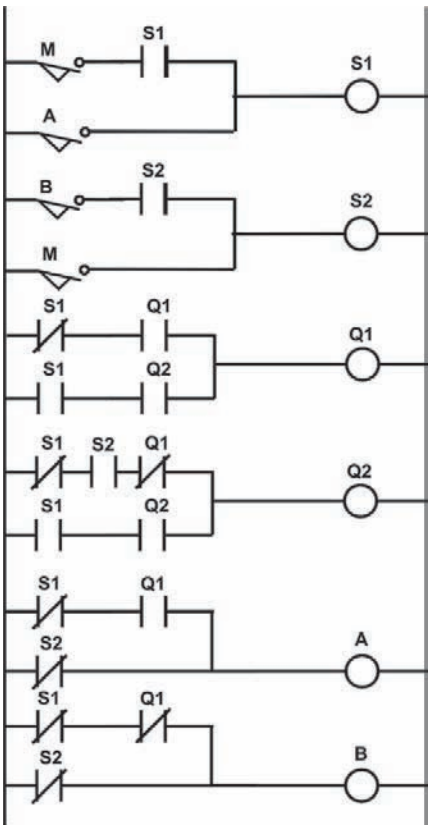


12. Realizar la representación gráfica.

a) Diagrama esquemático:



b) Diagrama escalera:



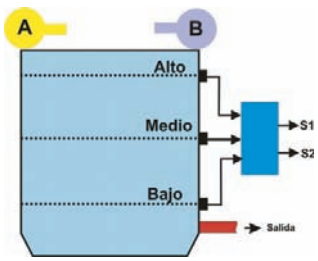
Dos bombas simultáneas (3)

El tanque de la figura se alimenta por medio de dos bombas llamadas **A** y **B**.

El gasto de salida nunca será mayor al que proporcionen las dos bombas operando simultáneamente.

El tanque tiene un sistema detector de niveles que consta de tres sensores de entrada: nivel **A** (alto), **M** (medio) y **B** (bajo); y dos salidas, **S2, S1** que indican lo siguiente:

	S2	S1
Bajo	0	0
Medio	0	1
Alto	1	1



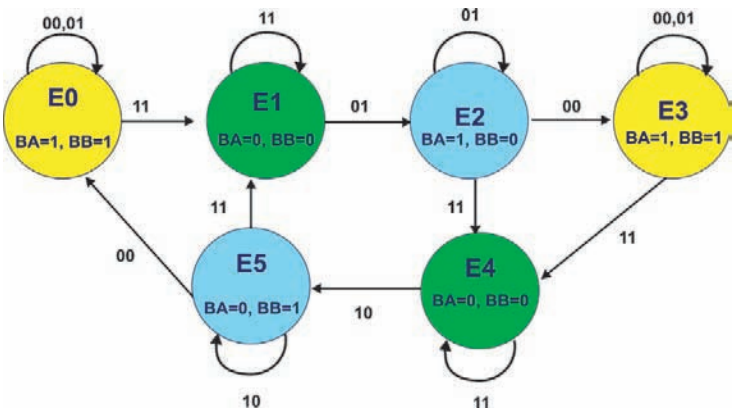
Trabajo solicitado

Diseñar un sistema secuencial asíncrono que controle la siguiente secuencia de operación de las bombas, en función de la salida del sistema de detección de niveles S2, S1.

1. Partiendo de que el tanque está vacío (S2 = 0 y S1 = 0), se inicia el llenado con ambas bombas **A** y **B,** hasta llenar el tanque (S2 = 1 y S1 = 1), para después desconectarlas.
2. Una vez lleno, si el nivel llega a medio (S2 = 0 y S1 = 1), sólo operará la bomba A; si el nivel sigue bajando hasta el nivel bajo (S2 = 0 y S1 = 0) operarán de nuevo ambas bombas **A** y **B**, hasta llenar el tanque (S2 = 1 y S1 = 1), para después desconectarlas.
3. Una vez lleno, si el nivel llega de nuevo a medio (S2 = 0 y S1 = 1), trabajará B; si el nivel sigue bajando hasta el nivel bajo (S2 = 0 y S1 = 0), nuevamente operarán ambas bombas **A** y **B** hasta llenar el tanque (S2 = 1 y S1 = 1), para después desconectarlas.
4. Cada vez que el nivel pase de lleno a medio, deberá trabajar una sola bomba alternándose en su funcionamiento.
5. Y cada vez que se vacía, trabajarán ambas bombas hasta llenar el tanque.

Procedimiento

1. Especificar el sistema.



2. Construir la tabla de flujo primitiva.

	Entradas S2, S1				
Hileras	00	01	11	10	AB
1	**E0**	**E0**	E1	-	11
2	-	E2	**E1**	-	00
3	E3	**E2**	E4	-	10
4	**E3**	**E3**	E4		11
5	-	E5	**E4**		00
6	E0	**E5**	E1		01

3. Eliminar los estados redundantes.

No hay reducción de estados.

4. Efectuar la mezcla de filas.

	Entradas S2, S1				
Hileras	00	01	11	10	AB
1	**E0**	**E0**	E1	-	11
2	-	E2	**E1**	-	00
3	E3	**E2**	E4	-	10
4	**E3**	**E3**	E4		11
5	-	E5	**E4**		00
6	E0	**E5**	E1		01

5. Expandir la tabla de salidas.

En este caso no es necesario expandir las salidas.

6. Construir la tabla de estados internos.

	Entradas S2, S1				
Hileras	00	01	11	10	AB
a	E0	E0	E1	-	11
b	-	E2	E1	-	00
c	E3	E2	E4	-	10
d	E3	E3	E4		11
e	-	E5	E4		00
f	E0	E5	E1		01

Si se sustituyen los estados de E0 a E5 por **a** hasta **f**, respectivamente, con lo cual se obtiene:

	Entradas S2,S1				
Hileras	00	01	11	10	AB
a	a	a	b	-	11
b	-	c	b	-	00
c	d	c	e	-	10
d	d	d	e		11
e	-	f	e		00
f	a	f	b		01

7. Asignar valores a los estados.

	Entradas S2, S1				
Hileras	00	01	11	10	AB
a	a	a	b	-	11
b	-	c	b	-	00
c	d	c	e	-	10
d	d	d	e		11
e	-	f	e		00
f	a	f	b		01

Se requiere que:

Columna 00, (c → d) y (f →a)
Columna 01, (b → c) y (e → f)
Columna 11, (a → b), (c → e), (d → e) y (f → b)

Al analizar el caso b, tiene un sólo cambio con a, c y f; en tanto que f tiene un solo cambio con a, por lo que resulta imposible la asignación.

		Q1, Q2			
		00	01	11	10
Q3	0	a	b	f	
	1		c		

Se propone que, para que sea posible la asignación, se incluyan dos estados transitorios, llamados α y β, de la siguiente manera: pasar de f → α→ a y de e → β → d. Observe la tabla.

		Q1, Q2			
		00	01	11	10
Q3	0	a	b	f	α
	1	d	c	e	β

	Entradas S2, S1				
Hileras	00	01	11	10	AB
a	a	a	b	-	11
b	-	c	b	-	00
c	d	c	e	-	10
d	d	d	β		11
e	-	f	e		00
f	α	f	b		01
α	a	-	-		
β	-	-	e		

Asignación:

	Q1	Q2	Q3
a	0	0	0
d	0	0	1
b	0	1	0
c	0	1	1
α	1	0	0
β	1	0	1
f	1	1	0
e	1	1	1

8. Construir la tabla de estados final.

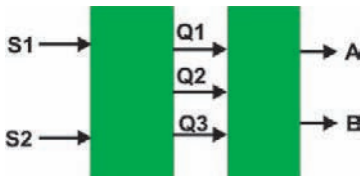
	Entradas S2, S1				
Hileras	00	01	11	10	AB
000	000	000	010	-	11
010	-	011	010	-	00
011	001	011	111	-	10
001	001	001	101		11
111	-	110	111		00
110	100	110	010		01
100	000	-	-		
101	-	-	111		

Acomodando hileras,

	Entradas S2, S1				
Hileras	00	01	11	10	AB
000	000	000	010	-	11
001	001	001	101		11
010	-	011	010	-	00
011	001	011	111	-	10
100	000	-	-		
101	-	-	111		
110	100	110	010		01
111	-	110	111		00

9. Completar la tabla de salidas.

No es necesario completar la tabla de salidas.



10. Obtener las ecuaciones mínimas.

Tabla para obtener los valores de Q1+, Q2+ y Q3+:

m	S1 S2 Q1 Q2 Q3	Q1+ Q2+ Q3+
0	00000	000
1	00001	001
2	00010	---
3	00011	001
4	00100	000
5	00101	---
6	00110	100
7	00111	---
8	01000	000
9	01001	001
10	01010	011
11	01011	011
12	01100	---
13	01101	---
14	01110	110
15	01111	110
16	10000	---
17	10001	---
18	10010	---
19	10011	---
20	10100	---
21	10101	---
22	10110	---
23	10111	---
24	11000	010
25	11001	101
26	11010	010
27	11011	111
28	11100	---
29	11101	111
30	11110	010
31	11111	111

Q1+ = S1 Q3 + S1'Q1 Q2

Q2+ = S2 Q2 + S1 Q3' + Q1 Q3

Q3+ = S1'Q1'Q2 + S1 Q3 + Q2'Q3

Q1 Q2 Q3	A B
000	**11**
001	**11**
010	**00**
011	**10**
100	**--**
101	**--**
110	**01**
111	**00**

A = Q2' + Q1'Q3

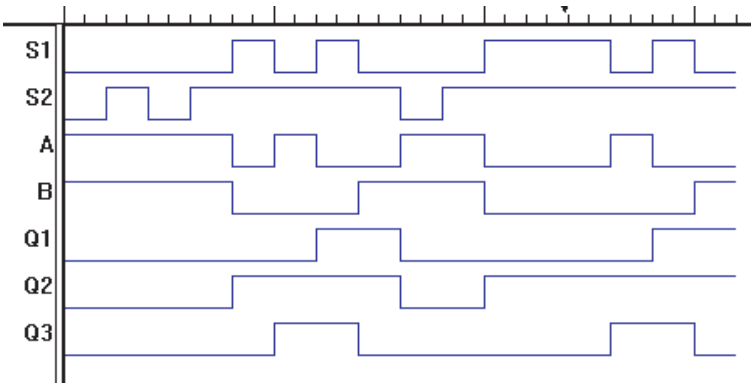
B = Q2' + Q1 Q3'

Archivo ABEL-HDL:

```
MODULE dbasy
"Entradas
S1,S2 pin 1,2;
"salidas
A,B,Q1,Q2,Q3 pin 19..15 istype 'com';
equations
Q1=S1&Q3#!S1&Q1&Q2;
Q2=S2&Q2#S1&!Q3#Q1&Q3;
Q3=!S1&!Q1&Q2#S1&Q3#!Q1&Q3;
A=!Q2#!Q1&Q3;
B=!Q2#Q1&!Q3;
```

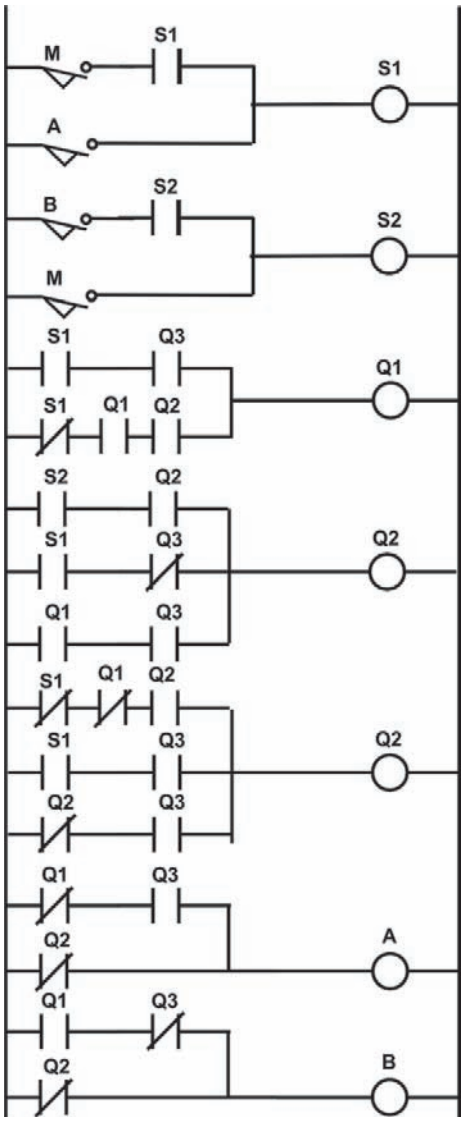
```
Test_Vectors
([S1,S2]->[A,B])
[0,0]->[.x.,.x.];
[0,1]->[.x.,.x.];
[0,0]->[.x.,.x.];
[0,1]->[.x.,.x.];
[1,1]->[.x.,.x.];
[0,1]->[.x.,.x.];
[1,1]->[.x.,.x.];
[0,1]->[.x.,.x.];
[0,0]->[.x.,.x.];
[0,1]->[.x.,.x.];
[1,1]->[.x.,.x.];
[1,1]->[.x.,.x.];
[1,1]->[.x.,.x.];
[0,1]->[.x.,.x.];
[1,1]->[.x.,.x.];
[0,1]->[.x.,.x.];
END
```

11. Realizar la simulación.



12. Realizar la representación gráfica.

a) Diagrama escalera:



Ejemplo 11.10

Barreras de seguridad de una vía de ferrocarril

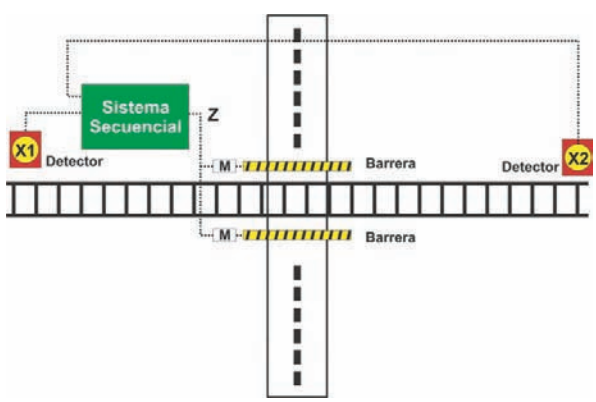
A una vía de ferrocarril, con tráfico en ambos sentidos, la corta una carretera, en la cual se colocan barreras activadas por una salida **Z** mediante un sistema secuencial, como se muestra en la figura de abajo.

A 500 m del punto de cruce en ambas direcciones, se colocan detectores **X1** y **X2**, respectivamente. A partir de un estado inicial donde **Z** = 0, este valor debe pasar al estado uno (**Z** =1) cuando se acerca un ferrocarril, en cualquier sentido, al rebasar la máquina los 500 m del cruce. Debe volver al estado cero cuando el último vagón se aleja de dicha distancia, independientemente de la longitud del ferrocarril.

Considere que no está permitido hacer maniobras; es decir, no hay cambio de dirección y sólo pasa un tren a la vez.

Trabajo solicitado

Diseñe el sistema secuencial asíncrono.



Descripción del problema y sus sensores

Con las condiciones descritas anteriormente se pueden presentar cuatro casos:

- Caso 1, tren corto con dirección X1 a X2.
- Caso 2, tren corto con dirección X2 a X1 .
- Caso 3, tren largo con dirección X1 a X2.
- Caso 4, tren largo con dirección X2 a X1 .

Secuencia de cada uno de los casos:

Caso 1	
X1 X2	Z
00	0
10	1
00	1
01	1
00	0

Caso 2	
X1 X2	Z
00	0
01	1
00	1
10	1
00	0

Caso 3	
X1 X2	Z
00	0
10	1
11	1
01	1
00	0

Caso 4	
X1 X2	Z
00	0
01	1
11	1
10	1
11	0

Procedimiento

1. Especificar el sistema.

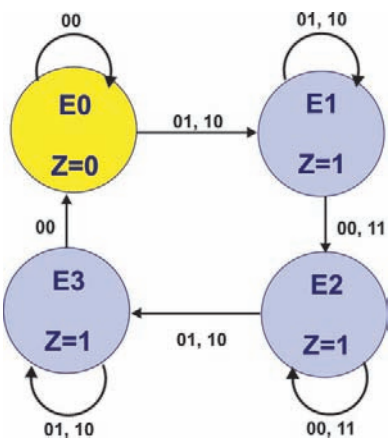


Diagrama de transición

2. Construir la tabla de flujo primitiva.

	Entradas X1, X2				Salida
Hileras	00	01	11	10	Z
1	**E0**	E1	-	E1	0
2	E2	**E1**	E2	**E1**	1
3	**E2**	E3	**E2**	E3	1
4	E0	**E3**		**E3**	1

3. Eliminar los estados redundantes.

No hay posibilidad de reducción de estados.

4. Efectuar la mezcla de filas.

No es posible la mezcla de filas.

5. Expandir la tabla de salidas.

En este caso no hay expansión de tablas de salida.

6. Construir la tabla de estados internos.

	Entradas X1, X2				Salida
Hileras	00	01	11	10	Z
a	**E0**	E1	-	E1	0
b	E2	**E1**	E2	**E1**	1
c	**E2**	E3	**E2**	E3	1
d	E0	**E3**		**E3**	1

	Entradas X1, X2				Salida
Hileras	00	01	11	10	Z
a	**a**	b	-	b	0
b	c	**b**	c	**b**	1
c	**c**	d	**c**	d	1
d	a	**d**		**d**	1

7. Asignar valores a los estados.

Análisis por columna:

00, b → c, d → a

01, a → b, c → d

11, b → c

10, a → b, c → d

		Q2 = 0	Q2 = 1
Q1	0	a	d
	1	b	c

Asignación:

a = 00, b = 01, c = 11, d = 10

8. Construir la tabla de estados final.

	Entradas X1, X2				Salida
Q1 Q2	00	01	11	10	Z
00	00	01	-	01	0
01	11	01	11	01	1
11	11	10	11	10	1
10	00	10	-	10	1

9. Completar la tabla de salidas.

No es necesario completar la tabla de salidas.

10. Obtener las ecuaciones mínimas.

m	X1	X2	Q1	Q2	Q1+	Q2+
0	0	0	0	0	0	0
1	0	0	0	1	1	1
2	0	0	1	0	0	0
3	0	0	1	1	1	1
4	0	1	0	0	0	1
5	0	1	0	1	0	1
6	0	1	1	0	1	0
7	0	1	1	1	1	0
8	1	0	0	0	0	1
9	1	0	0	1	0	1
10	1	0	1	0	1	0
11	1	0	1	1	1	0
12	1	1	0	0	X	X
13	1	1	0	1	1	1
14	1	1	1	0	X	X
15	1	1	1	1	1	1

Q1 = X1'X2'Q2 + X1X2Q2 + X2Q1 + X1Q1

Q2 = X1'X2'Q2 + X1X2 + X2Q1' + X1Q1'

Z = Q1 + Q2

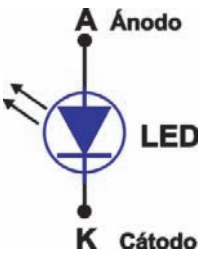
Archivo en formato ABEL-HDL:

```
MODULE tasinc
“entradas
X1,X2 pin 1,2;
“salidas
Q1,Q2,Z pin 19..17 istype ‘com’;
equations
Q1=!X1&!X2&Q2#X1&X2&Q2#X2&Q1#X1&Q1;
Q2=!X1&!X2&Q2#X1&X2#X2&!Q1#X1&!Q1;
Z=Q1#Q2;
```

APÉNDICE A

El diodo emisor de luz (LED)

En los sistemas digitales es conveniente tener una salida visual que comúnmente es a base de LED o *Display* ya sea luminoso o de cristal líquido.

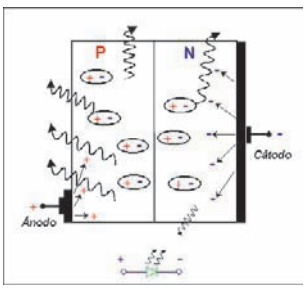


El LED es un diodo que produce luz visible (o invisible, infrarroja) cuando se encuentra polarizado directamente.

El símbolo del LED es similar al de un diodo de unión.

El voltaje de polarización de un LED varía desde 1.4 V hasta 2.5 V y la corriente necesaria para que emita la luz va desde 8 mA hasta los 20 mA.

Principio de funcionamiento

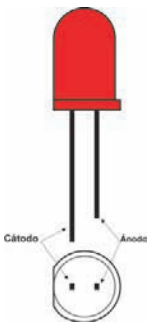


En cualquier unión P-N polarizada directamente, dentro de la estructura y principalmente cerca de la unión, ocurre una recombinación de huecos y electrones (al paso de la corriente). Esta recombinación requiere que la energía que posee un electrón libre no ligado se transfiera a otro estado.

En todas las uniones P-N una parte de esta energía se convierte en calor y otro tanto en fotones. En el *Si* y el *Ge* el mayor porcentaje se transforma en calor y la luz emitida es insignificante. Por esta razón se utiliza otro tipo de materiales para fabricar los LED, como fosfuro arseniuro de galio (GaAsP) o fosfuro de galio (GaP).

Tabla de características eléctricas y luminosas de algunos LED de propósito general

Tipo	Color	Voltaje en directa VF (V)	Voltaje en inversa VR (V)	Máxima corriente en directa IF (ma)	Máxima potencia Disipada PD (mW)	Ángulo típico de vista grados	Intensidad luminosa típica MCD
ECG3007	Rojo	1.60	5.0	35	105	25	2.0
ECG3007	Rojo	1.68	5.0	50	100	70	2.5
ECG3008	Rojo	2.00	5.0	35	105	90	5.0
ECG3009	Naranja	2.00	5.0	35	105	90	5.0
ECG3010	Verde	2.00	5.0	35	105	90	1.0
ECG3011	Amarillo	2.12	5.0	35	105	90	3.0

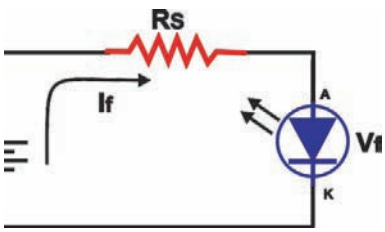


Cálculo de la resistencia limitadora de corriente de un LED

Rs = (**Vs** – **Vf**)/ **If**

Suponiendo un **Vs** = 5Vcd para un LED cuya If = 15 ma y con un Vf = 1.6 volts

Rs = (5 – 1.6)/ .015 = 226 Ω

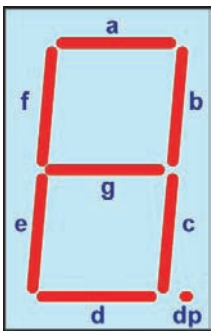


Existen dos valores comerciales cercanos a 226 Ω que son 220 Ω y 330 Ω si elegimos la de 220 Ω la corriente será ligeramente superior a los 15 (ma) con una intensidad de luz mayor, y si elegimos la de 330 Ω la de corriente será menor con intensidad de luz menor.

La intensidad producida por un LED se llama intensidad luminosa y se mide en unidades de candela.

Display numérico de 7 segmentos

Tabla de características eléctricas

	tipo	Corriente por segmento (ma)	Voltaje inverso por segmento (V)	Potencia disipada Pt (mW)	
ECG3050	Ánodo Común	30	10	750	
ECG3052	Cátodo Común	30	5	700	
ECG3053	CA Naranja	20	3	400	
ECG3054	CA Verde	20	3	400	
ECG3056	CC Rojo	30	5	700	

APÉNDICE B

La página de Internet para descargar el software y la licencia de uso para el desarrollo de estas prácticas es:

http://latticesemi.com/ftp/ispstarter.html

Requerimientos mínimos del sistema para trabajar en una PC con ispEXPERT Starter Software:

486/Pentium-compatible PC

Mouse y mouse driver

Windows NT 4.0, Windows 98, Windows 95 o Windows Me

32 megabytes de RAM

Resolución de pantalla (800 × 600) SVGA

108 megabytes de espacio disponible.

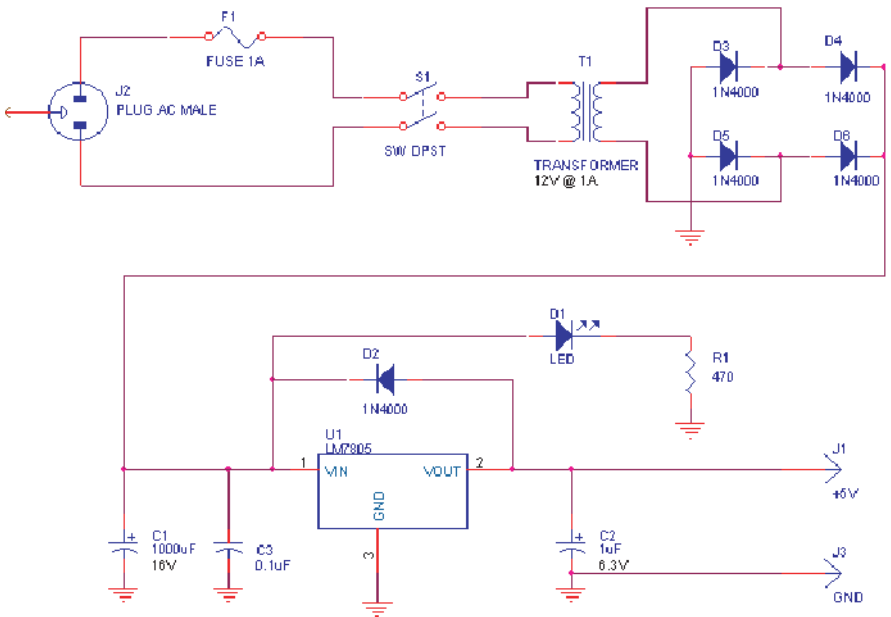
Para la instalación desde el CD que viene en este libro:

1. Introduzca el CD en la computadora.
2. Localize el directorio software/lattice/cd dentro del CD.
3. Ejecute el programa *setup* que viene en el directorio mencionado.
4. Seleccione la opción *Instalar software*.

5. Una vez terminado el proceso de instalación, solicite la licencia de uso en la página de internet www.latticesemi.com (es necesario conocer el número de serie de su disco duro y además contar con un correo electrónico (e-mail) para recibir por parte de la compañía Lattice el archivo licence.dat que deberá instalar dentro del directorio ISPTOOLS/ISPCOMP/LICENSE/.
6. Una vez instalado el archivo license.dat apague y encienda la computadora.

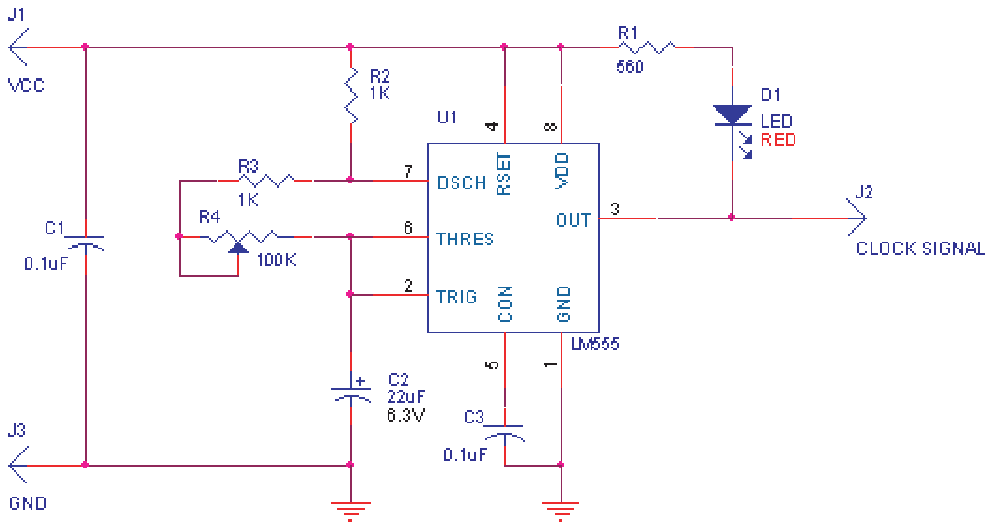
APÉNDICE C

Fuente regulada de 5Vcd.



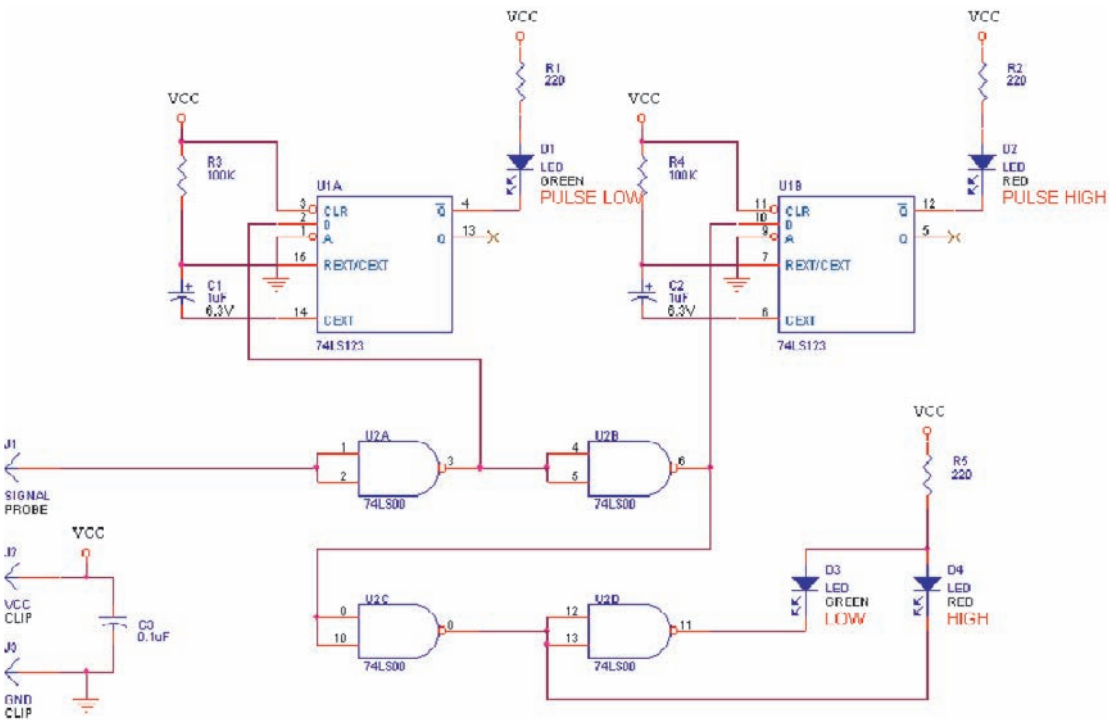
APÉNDICE D

Generador de pulsos con un LM555



APÉNDICE E

Punta Lógica

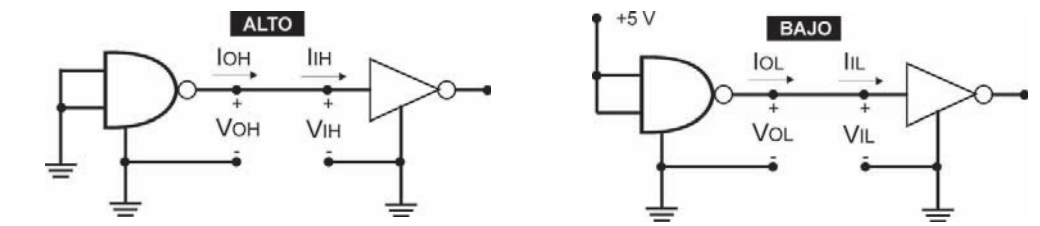


APÉNDICE F

Circuitos integrados digitales

Aunque existen muchos fabricantes de circuitos integrados digitales, parte de la nomenclatura y terminología está prácticamente estandarizada.

Parámetros de corriente y voltaje



Corrientes y Voltajes en los estados Lógicos

V_{IH}(mín) voltaje de entrada de nivel alto. Nivel de voltaje que se requiere para uno lógico en la entrada. Cualquier voltaje debajo de este nivel no será aceptado como ALTO por el circuito lógico.

V_{IL}(máx) voltaje de entrada de nivel bajo. Nivel de voltaje que se necesita para un cero lógico en la entrada. Cualquier voltaje que esté por encima de este nivel no será aceptado como BAJO por el circuito lógico.

V_{OH}(mín) voltaje de salida de alto nivel. Nivel de voltaje mínimo a la salida de un circuito lógico en estado ALTO bajo condiciones de carga definidas.

V_{OL}(máx) voltaje de salida de nivel bajo. Máximo nivel a la salida de un circuito en cero lógico bajo condiciones de carga definidas.

I_{IH} corriente de entrada de nivel alto. Corriente que fluye en una entrada cuando se aplica un voltaje de nivel alto específico a dicha entrada.

I_{IL} corriente de entrada de nivel bajo. Corriente que fluye en una entrada cuando se aplica un voltaje de nivel bajo específico a dicha entrada.

I_{OH} corriente de salida de nivel alto. Corriente que fluye desde una salida en el estado uno lógico en condiciones de carga específica.

I_{OL} corriente de salida de nivel bajo. Corriente que fluye desde una salida en estado cero lógico en condiciones de carga específicas.

Especificaciones de voltaje de entrada/salida de una familia **TTL** estándar

Símbolo	Parámetro	Mínimo	Normal	Máximo	Unidades
Vcc	Voltaje de alimentación	**4.75**	**5**	**5.25**	**V**
V_{OH}	Voltaje de salida de alto nivel	**2.4**	**3.4**		**V**
V_{OL}	Voltaje de salida de nivel bajo		**0.2**		**V**
V_{IH}	Voltaje de entrada de nivel alto	**2.0**			**V**
V_{IL}	Voltaje de entrada de nivel bajo			**0.8**	**V**
I_{OH}	Corriente de salida de nivel alto			**-0.4**	**mA**
I_{OL}	Corriente de salida de nivel bajo			**16**	**mA**
TA	Temperatura de operación al aire libre	**0**		**70**	**^{0}C**

Factor de carga de salida (Fan-Out). En general, la salida de un circuito lógico debe manejar varias entradas lógicas. El factor de carga de salida se define como el número máximo de entradas lógicas estándar que una salida puede manejar confiablemente. Por

ejemplo, si una compuerta lógica que se especifica con un factor de carga de 10 puede manejar 10 entradas lógicas normales. Si este número es excedido no se pueden garantizar los voltajes de nivel lógico de salida.

Retrasos de la propagación. Una señal lógica siempre experimenta un retraso al recorrer un circuito. Los dos tiempos de retraso de propagación se definen como sigue:

t_{PLH}: Tiempo de retraso al pasar de estado lógico 0 a 1 lógico (de BAJO a ALTO).

t_{PHL}: Tiempo de retraso al pasar de estado lógico 1 a 0 lógico (de ALTO a BAJO).

Considere que **t_{PHL}** es el retraso en la respuesta de la salida cuando pasa de ALTO a BAJO. Se mide entre los puntos de 50% en las transiciones de entrada y de salida. El valor **t_{PLH}** es el retraso en la respuesta de la salida cuando pasa de BAJO a ALTO.

En términos generales **t_{PLH}** y **t_{PHL}** no son el mismo valor y ambos variarán según las condiciones de carga. Los valores de los tiempos de propagación se utilizan como una medida de la velocidad relativa de los circuitos lógicos. Por ejemplo, un circuito lógico con valores de 10 ns es un circuito más rápido que uno con valores de 20 ns en condiciones especificadas de carga.

Requerimientos de potencia. Cada uno de los circuitos integrados requiere de cierta cantidad de potencia para poder funcionar. Esta potencia es suministrada por uno o más voltajes de alimentación conectados a las terminales del CI. Generalmente sólo hay una terminal de suministro de potencia en el encapsulado y se etiqueta como V_{CC} (para TTL) y como V_{DD} (para dispositivos CMOS).

La cantidad de potencia que requiere un circuito integrado se determina por la corriente ICC, que consume de la fuente de alimentación V_{CC}; y la potencia real es el producto $I_{CC} \times V_{CC}$.

Para muchos CI el consumo de corriente de la fuente de alimentación variará según los estados lógicos de los circuitos en el encapsulado. Por ejemplo, para la figura muestra un circuito integrado NAND donde todas las salidas de las compuertas son ALTAS. El consumo de corriente de la fuente de alimentación VCC en este caso recibe el nombre de I_{CCH}. De igual manera la figura muestra el consumo de corriente cuando todas las salidas de la compuerta son BAJAS. Esta corriente se conoce como I_{CCL}.

En términos generales I_{CCH} e I_{CCL} serán valores diferentes. La corriente promedio se calcula con base en el supuesto de que las salidas compuertas estarán BAJAS la mitad del tiempo y ALTAS la otra mitad.

$$I_{CC}(prom) = \frac{I_{CCH} + I_{CCL}}{2}$$

F

Y se puede emplear para calcular el consumo de potencia como:

$$P_D(prom) = I_{CC}(prom) * V_{CC}$$

Inmunidad al ruido. Los campos eléctricos y magnéticos aleatorios pueden inducir voltajes en los alambres de conexión entre los circuitos lógicos. Estas señales espurias, no deseadas, se denominan ruido, y algunas veces pueden ocasionar que el voltaje en la entrada de un circuito lógico caiga por debajo (mín) o exceda VIL(máx), lo que podría producir una operación poco confiable. La inmunidad al ruido de un circuito lógico se refiere a la capacidad del circuito para tolerar voltajes de ruido en sus entradas. A una medida cuantitativa de inmunidad al ruido se le denomina **margen de ruido.**

El margen de ruido en estado alto V_{NH} se define como:

$$V_{NH} = V_{OH}(\text{mín}) - V_{IH}(\text{mín})$$

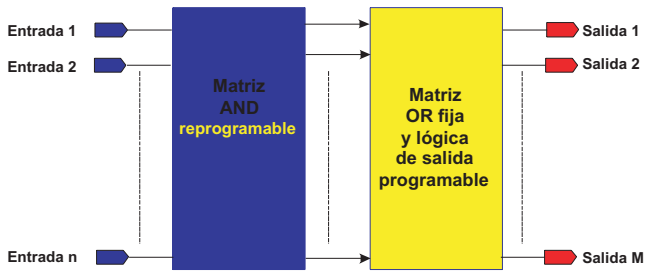
V_{NH} es la diferencia entre la menor salida ALTA posible y el voltaje mínimo de entrada posible y el voltaje mínimo de entrada requerido para un ALTO. Cuando una salida lógica ALTA está manejando una entrada del circuito lógico, cualquier espiga de ruido negativa, mayor que V_{NH} que aparezca en la línea de señales puede hacer que el voltaje disminuya a un rango indeterminado, donde puede ocurrir una operación impredecible.

El **margen de ruido en estado bajo** V_{NL} se define como:

$$V_{NL} = V_{IL}(\text{máx}) - V_{OL}(\text{máx})$$

APÉNDICE G

La matriz lógica genérica (GAL): Está formada por una matriz AND reprogramable y una matriz OR fija con configuraciones de salidas programables.



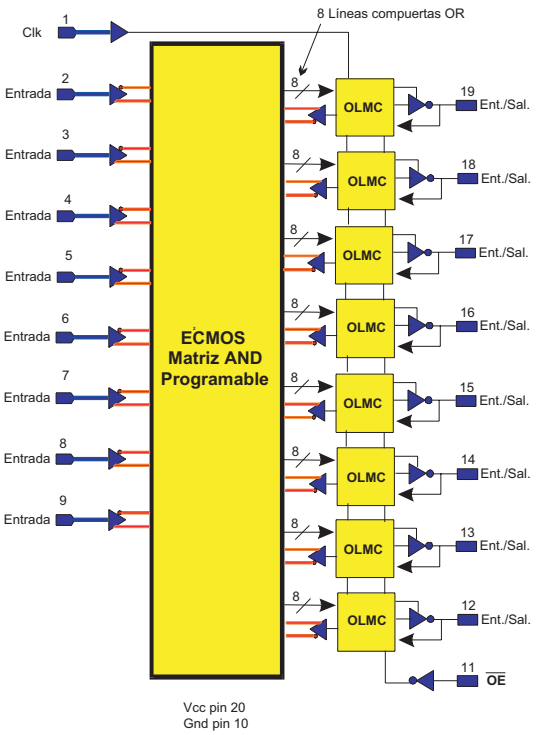
GAL16V8D

GAL	**Matriz Lógica Genérica.**
16	**Hasta 16 entradas.**
V	**Configuración de salidas variable.**
8	**Hasta 8 salidas.**

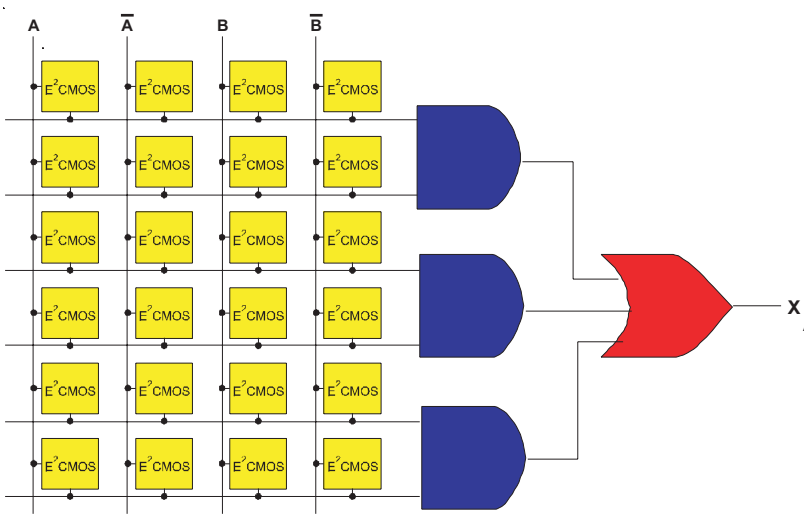
Características del GAL16V8D

- 3.5 ns máximo tiempo de propagación.
- Fmáx = 250 Mhz.
- 2.5 ns máximo tiempo de propagación de la entrada de reloj al dato de salida.

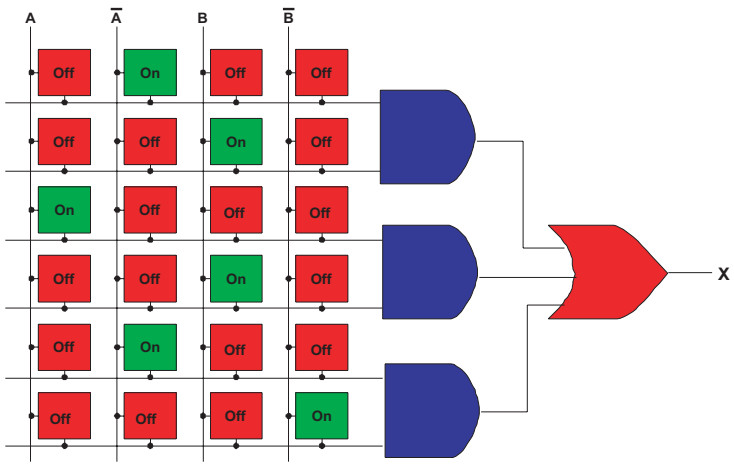
- Celdas reprogramables.
- Vcc = 5 Volts ± 5%
- Consumo de corriente 90 mA.
- Rapidez en el borrado < 100 ms.
- 20 años de retención de los datos.
- 8 Output Logic MacroCells (OLMC)
- Polaridad de salida programable.
- Temperatura de operación de 0 a 75° C.
- Firma electrónica para identificación.



Estructura interna



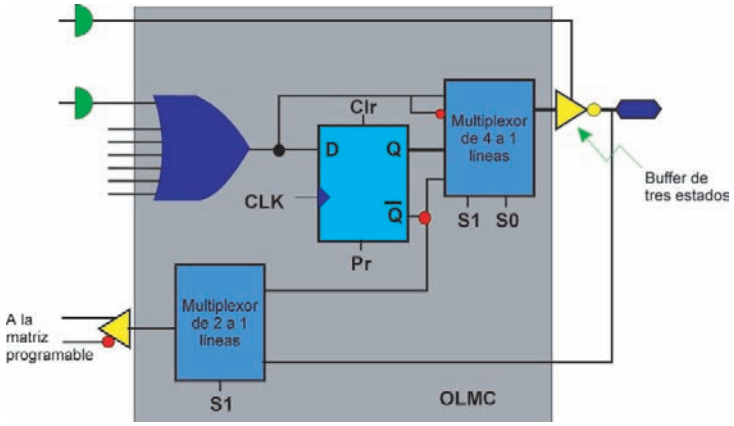
Implementación de una Suma de Productos X = **A'** B + A B + **A' B'**



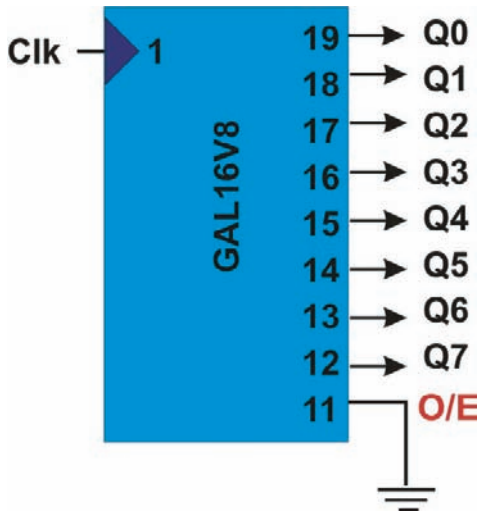
OLMC para utilizar el Flip Flop D del GAL16V8

Al usar Flip Flops en captura esquemática conecte la terminal 11 OE (*Output Enable*) a tierra (Gnd) con la finalidad de que las salidas del GAL16V8 sean consideradas como salidas registradas o FF's.

La **OLMC** (Output Logic Macro Cell) del GAL16V8 es el elemento que permite que la terminal se pueda configurar como entrada, salida combinacional o salida registrada, este dispositivo GAL16V8 cuenta con ocho terminales OLMC, que le permiten una gran versatilidad.



En los sistemas secuenciales o con el uso de Flip Flops es necesario que la terminal **Output/Enable** (terminal 11 en el GAL16V8) se conecte a tierra para tener las salidas de los Flip Flops activadas.



Lattice Semiconductor Corporation

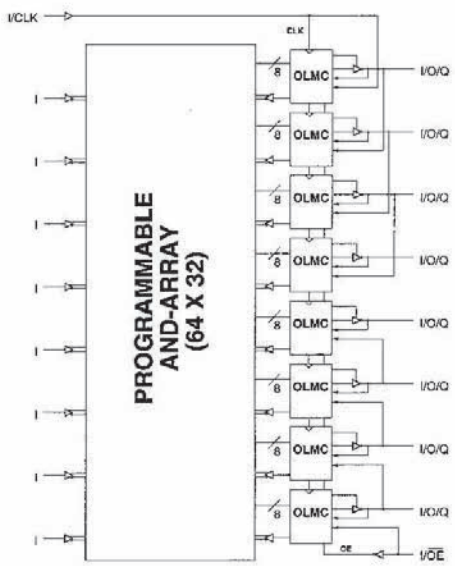
GAL16V8

High Performance E²CMOS PLD Generic Array Logic™

Features

- HIGH PERFORMANCE E²CMOS® TECHNOLOGY
  - 3.5 ns Maximum Propagation Delay
  - Fmax = 250 MHz
  - 3.0 ns Maximum from Clock Input to Data Output
  - UltraMOS® Advanced CMOS Technology
- 50% to 75% REDUCTION IN POWER FROM BIPOLAR
  - 75mA Typ Icc on Low Power Device
  - 45mA Typ Icc on Quarter Power Device
- ACTIVE PULL-UPS ON ALL PINS
- E² CELL TECHNOLOGY
  - Reconfigurable Logic
  - Reprogrammable Cells
  - 100% Tested/100% Yields
  - High Speed Electrical Erasure (<100ms)
  - 20 Year Data Retention
- EIGHT OUTPUT LOGIC MACROCELLS
  - Maximum Flexibility for Complex Logic Designs
  - Programmable Output Polarity
  - Also Emulates 20-pin PAL® Devices with Full Function/Fuse Map/Parametric Compatibility
- PRELOAD AND POWER-ON RESET OF ALL REGISTERS
  - 100% Functional Testability
- APPLICATIONS INCLUDE:
  - DMA Control
  - State Machine Control
  - High Speed Graphics Processing
  - Standard Logic Speed Upgrade
- ELECTRONIC SIGNATURE FOR IDENTIFICATION

Functional Block Diagram



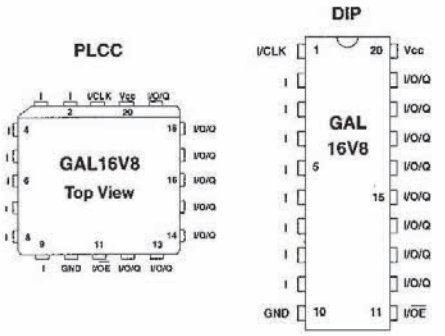
Description

The GAL16V8, at 3.5 ns maximum propagation delay time, combines a high performance CMOS process with Electrically Erasable (E²) floating gate technology to provide the highest speed performance available in the PLD market. High speed erase times (<100ms) allow the devices to be reprogrammed quickly and efficiently.

The generic architecture provides maximum design flexibility by allowing the Output Logic Macrocell (OLMC) to be configured by the user. An important subset of the many architecture configurations possible with the GAL16V8 are the PAL architectures listed in the table of the macrocell description section. GAL16V8 devices are capable of emulating any of these PAL architectures with full function/fuse map/parametric compatibility.

Unique test circuitry and reprogrammable cells allow complete AC, DC, and functional testing during manufacture. As a result, Lattice Semiconductor delivers 100% field programmability and functionality of all GAL products. In addition, 100 erase/write cycles and data retention in excess of 20 years are specified.

Pin Configuration



LATTICE SEMICONDUCTOR CORP., 5555 Northeast Moore Ct., Hillsboro, Oregon 97124, U.S.A.
Tel. (503) 681-0118; 1-888-ISP-PLDS; FAX (503) 681-3037; http://www.latticesemi.com

October 1998

Lattice Semiconductor Corporation

Specifications *GAL16V8D*

ABSOLUTE MAXIMUM RATINGS[1]

Supply voltage V_{CC} ..... −0.5 to +7V
Input voltage applied ..... −2.5 to V_{CC} +1.0V
Off-state output voltage applied..... −2.5 to V_{CC} +1.0V
Storage Temperature..... −65 to 150°C
Ambient Temperature with
Power Applied ..... −55 to 125°C

1. Stresses above those listed under the "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress only ratings and functional operation of the device at these or at any other conditions above those indicated in the operational sections of this specification is not implied (while programming, follow the programming specifications).

RECOMMENDED OPERATING COND.

Commercial Devices:
Ambient Temperature (T_A) ..... 0 to 75°C
Supply voltage (V_{CC})
with Respect to Ground ..... +4.75 to +5.25V

Industrial Devices:
Ambient Temperature (T_A) ..... −40 to 85°C
Supply voltage (V_{CC})
with Respect to Ground ..... +4.50 to +5.50V

DC ELECTRICAL CHARACTERISTICS

Over Recommended Operating Conditions (Unless Otherwise Specified)

SYMBOL	PARAMETER	CONDITION	MIN.	TYP.[3]	MAX.	UNITS
V_{IL}	Input Low Voltage		V_{SS} − 0.5	—	0.8	V
V_{IH}	Input High Voltage		2.0	—	V_{CC}+1	V
I_{IL}[1]	Input or I/O Low Leakage Current	$0V \le V_{IN} \le V_{IL}$ (MAX.)	—	—	−100	µA
I_{IH}	Input or I/O High Leakage Current	$3.5V \le V_{IN} \le V_{CC}$	—	—	10	µA
V_{OL}	Output Low Voltage	I_{OL} = MAX. Vin = V_{IL} or V_{IH}	—	—	0.5	V
V_{OH}	Output High Voltage	I_{OH} = MAX. Vin = V_{IL} or V_{IH}	2.4	—	—	V
I_{OL}	Low Level Output Current	L-3/-5 & -7 (Ind. PLCC)	—	—	16	mA
		L-7 (Except Ind. PLCC)/-10/-15/-25 Q-10/-15/-20/-25	—	—	24	mA

Lattice Semiconductor Corporation

Specifications *GAL16V8D*

AC SWITCHING CHARACTERISTICS

Over Recommended Operating Conditions

PARAMETER	TEST COND'[1].	DESCRIPTION	COM -3 MIN.	COM -3 MAX.	COM -5 MIN.	COM -5 MAX.	COM / IND -7 MIN.	COM / IND -7 MAX.	UNITS
tpd	A	Input or I/O to Comb. Output	1	3.5	1	5	1	7.5	ns
tco	A	Clock to Output Delay	1	3	1	4	1	5	ns
tcf[2]	—	Clock to Feedback Delay	—	2.5	—	3	—	3	ns
tsu	—	Setup Time, Input or Feedback before Clock↑	2.5	—	3	—	7	—	ns
th	—	Hold Time, Input or Feedback after Clock↑	0	—	0	—	0	—	ns
fmax[3]	A	Maximum Clock Frequency with External Feedback, 1/(tsu + tco)	182	—	142.8	—	83.3	—	MHz
	A	Maximum Clock Frequency with Internal Feedback, 1/(tsu + tcf)	200	—	166	—	100	—	MHz
	A	Maximum Clock Frequency with No Feedback	250	—	166	—	100	—	MHz
twh	—	Clock Pulse Duration, High	2	—	3	—	5	—	ns
twl	—	Clock Pulse Duration, Low	2	—	3	—	5	—	ns
ten	B	Input or I/O to Output Enabled	—	4.5	—	6	—	9	ns
	B	$\overline{OE}$ to Output Enabled	—	4.5	—	6	—	6	ns
tdis	C	Input or I/O to Output Disabled	—	4.5	—	5	—	9	ns
	C	$\overline{OE}$ to Output Disabled	—	4.5	—	5	—	6	ns

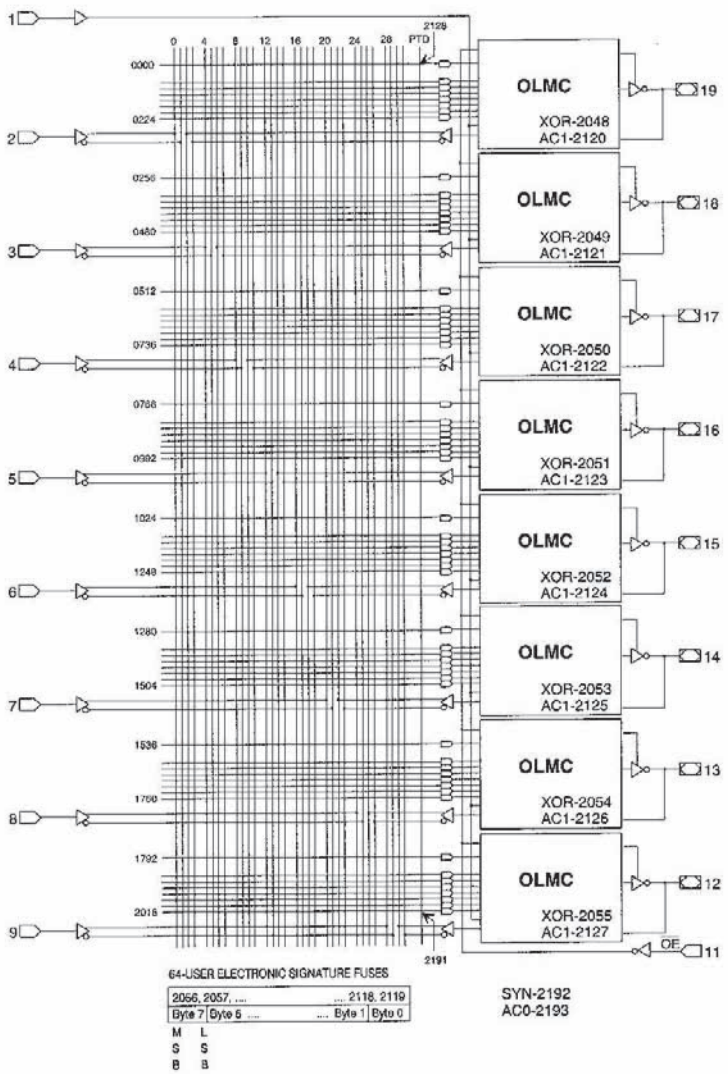
1) Refer to **Switching Test Conditions** section.
2) Calculated from fmax with internal feedback. Refer to **fmax Descriptions** section.
3) Refer to **fmax Descriptions** section. Characterized but not 100% tested.

Lattice Semiconductor Corporation

Specifications **GAL16V8**

Registered Mode Logic Diagram

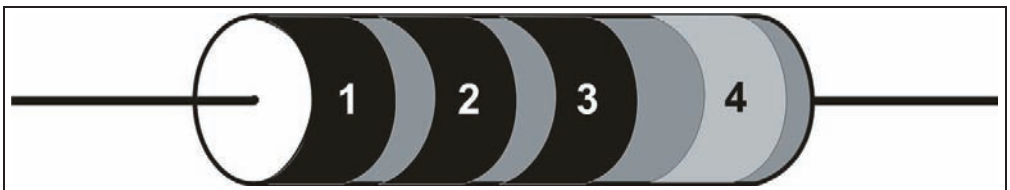
DIP & PLCC Package Pinouts



G

APÉNDICE H

Código de colores para obtener el valor de las resistencias.



Color	1 Banda	2 Banda	3 Banda	
Negro	**0**	**0**	**X**	1
Café	**1**	**1**	**X**	10
Rojo	**2**	**2**	**X**	100
Naranja	**3**	**3**	**X**	1,000
Amarillo	**4**	**4**	**X**	10,000

Verde	**5**	**5**	**X**	100,000
Azul	**6**	**6**	**X**	1;000,000
Violeta	**7**	**7**	**X**	10;,000,000
Gris	**8**	**8**	**X**	100;000,000
Blanco	**9**	**9**		

La cuarta banda indica la tolerancia.

Oro	**+/-**	**5%**
Plata	**+/-**	**10%**
Sin Banda	**+/-**	**20%**

GLOSARIO

! : Cuando se emplea dentro de un símbolo en lenguaje ABEL-HDL, indica una función **Not**, por ejemplo: **!A.**

 # : Cuando se emplea dentro de un símbolo en lenguaje ABEL-HDL, indica una función **Or;** por ejemplo, **A#B**.

& : Cuando se emplea dentro de un símbolo IEEE/ANSI, o en lenguaje ABEL-HDL, indica una función **And,** por ejemplo, **A&B**.

ABEL (*Advanced Boolean Expression Language*): Lenguaje de descripción de hardware universal para el diseño con PLD.

Activación (*activation*): La ejecución de una acción.

Álgebra Booleana: Proceso algebraico utilizado como herramienta para el análisis y diseño de sistemas digitales; en el álgebra booleana sólo son posibles dos valores 0 y 1.

Análisis (analysis): Parte del proceso de desarrollo cuyo propósito principal es realizar un modelo del dominio del problema.

Antifusible: Contrario del fusible; es un circuito abierto que se puede programar para ser una baja impedancia. Es, al igual que el fusible, OTP (*one time programing*).

Arquitectura (*architecture*): Estructura organizacional de un sistema. Una arquitectura puede ser descompuesta recursivamente en partes que interactúan entre sí por medio de interfaces, relaciones que conectan las partes y restricciones para ensamblar las mismas.

Arreglo de compuertas: Grupo de transistores que se configura por el usuario en los niveles de conexión metálicos, formando funciones lógicas.

ASICS (*Application Specific Integrated Circuits*): Circuitos integrados de aplicación específica.

Atributo (*attribute*): Propiedad de un tipo, identificada mediante un nombre.

BCD: Código decimal expresado en binario, usado para representar cada dígito de un número decimal mediante su equivalente binario de cuatro bits.

Bit: Contracción de Digito Binario (*Binary Digit*).

Bloque: Un bloque es una parte de un PLD, el cual está formado por varios elementos lógicos con interconexión programable entre sí. Varios bloques interconectados forman el dispositivo.

Boole, George (1815-1864): Lógico y matemático británico, en gran medida autodidacta. Boole fue nombrado profesor de matemáticas en el Queen's College de Cork en Irlanda (hoy el University College) en 1849. En 1854, escribió sobre las leyes del pensamiento, en donde describe un sistema algebraico que más tarde se conoció como el álgebra de Boole. En él, las proposiciones lógicas se indican por símbolos y pueden relacionarse mediante operadores matemáticos abstractos que corresponden a las leyes de la lógica. El álgebra de Boole es fundamental para el estudio de las matemáticas puras y para el diseño de los modernos ordenadores o computadoras.

Bus global: Rutas dentro de un integrado que permiten conectar todos los elementos lógicos.

BYTE: Grupo de ocho bits.

Capa (*layer*): Forma específica de agrupar paquetes en un modelo al mismo nivel de abstracción.

Circuito: Disposición de componentes eléctricos y/o electrónicos interconectados de manera que realizan una función específica.

Circuito integrado (CI): Circuito en el que todos sus componenetes se encuentran integrados en un único chip semiconductor de muy pequeño tamaño.

CMOS (*Complementary Metal-Oxide Semiconductor*): Un tipo de circuito de transistores que utiliza transistores MOSFET**.**

Codificador: Circuito digital que convierte información de línea a un formato codificado.

Código: Un conjunto de bits ordenados según un patrón único y utilizados para representar información, tal como números, letras y otros símbolos.

Compilar (*compiling*): Rutina que transforma un programa escrito en un seudocódigo o en un lenguaje de programación automática con una serie de instrucciones en lenguaje básico de maquina.

Comportamiento (*behavior*): Efectos visibles de una operación o evento, incluyendo sus resultados.

Concurrencia (*concurrency*): Ocurrencia de dos o más actividades durante el mismo intervalo de tiempo.

Contacto: Dispositivo que abre o cierra un circuito eléctrico.

CPLD (*Complex Programmable Logic Device*): Es un integrado donde se tienen varios PLDs con una red de rutas que permite interconectarlos y realizar funciones lógicas más complejas.

Dispositivo: Se refiere a un circuito integrado (CI).

DR FPGA (*FPGA reconfigurable dinámicamente*): FPGA que puede ser reprogramado durante la operación del sistema. Algunos permiten reconfigurar algunas partes y otros deben ser reprogramados completamente.

E^2CMOS (*Electrically Erasable Complementary Metal Oxide Semiconductor*): Memoria que se puede borrar eléctricamente.

EDA (*Electronic Design Automation*): Nombre que se le da a todas las herramientas (tanto hardware como software) para la ayuda al diseño de sistemas electrónicos.

EEPLD: PLD que utiliza celdas de memoria EEPROM para guardar la lógica programada. Es mucho más complejo que un PLD simple.

EEPROM o E2PROM (*Electrically Erasable Programmable Read-Only Memory*): Memoria programable de sólo lectura eléctricamente borrable. Un tipo de memoria semiconductora.

EPLD: PLD que utiliza celdas de memoria EPROM en vez de fusibles para guardar la lógica programada.

EPROM (*Erasable Programmable Read-Only Memory*): Memoria de sólo lectura programable y borrable. Un tipo de memoria semiconductora.

Especificación (*Specification*): Descripción declarativa de lo que algo es o hace. Contraste: implementación.

Estado (*State*): Condición o situación en la vida de un objeto, durante la cual satisface una condición, realiza una actividad o está esperando un evento.

Evento (*Event*): Un acontecimiento significativo. Un evento tiene una ubicación en el tiempo y en el espacio y puede tener parámetros. En el contexto de un diagrama de estado, un evento es un acontecimiento que puede disparar una transición de estados.

***Flash*:** Tecnología de memorias no volátiles, que permite bajos costos y altos desempeños. Los dispositivos con esta tecnología son borrados y programados eléctricamente.

Flip Flop (FF): Dispositivo de memoria con capacidad de almacenar un solo bit.

FPGA (*Field Programmable Gate Array*): Consiste de un arreglo de bloques lógicos, rodeado de bloques de entrada/salida programables y conectados a través de interconexiones programables.

FPLA (*Field Programmable Logic Array*): PLD que posee tanto las AND como las OR programables, pero con la complejidad de un PLD simple.

Fusible (*Fuse*): Elemento de baja resistencia que puede ser modificado en un circuito abierto. La programación del fusible se denomina "quemar" el fusible y suele hacerse térmicamente mediante corrientes elevadas para éste. Es OTP (sólo se puede programar una vez).

HDL (*Hardware Description Language*): Lenguaje que permite describir un diseño lógico usando ecuaciones booleanas, tablas de verdad y de estados, así como la descripcion lógica.

Implementación (*Implementation*): La definición de cómo está construido o compuesto algo. Por ejemplo, una clase es una implementación de un tipo.

JEDEC (*Joint Electron Device Engineering Concil*): Los archivos JEDEC contienen el mapa de fusibles del PLD listo para ser programado.

***Link* (enlazar):** Parte de un subprograma que lo vincula con el programa principal, mediante la correlación entre dos o mas partes.

LogicAid: Programa de aplicación de la computadora para simplificar funciones Booleanas a partir de Ecuaciones, Minitérminos, Maxitérminos, Tablas de Verdad y Tablas de Estados.

Macrocelda: Circuito en bloque que contiene compuertas OR para sumar los productos (resultados del arreglo de AND. Además contiene Flip-Flops, un *buffer* tres estados, y varios multiplexores para seleccionar las señales de control.

Mapas de Karnaugh (Kmap): Formato bidimensional de una tabla de verdad empleado para simplificar Funciones Booleanas en forma suma de productos o productos de sumas.

Máquina de estados (*state machine*): Comportamiento que especifica las secuencias de estados por los que atraviesa un objeto o una interacción durante su ciclo de vida en respuesta a eventos.

***Mask-programmable*:** Dispositivos, por lo general arreglos de compuerta que son programados en fábrica, poniendo conexiones de metal entre los elementos lógicos.

Maxitérmino: Término Or que contiene todas las variables de la función ya sea en su forma normal o complementada.

Método (*method*): La implementación de una operación. El algoritmo o procedimiento que permite llegar al resultado de una operación. Sinónimo: method [OMA].

Minimización de lógica: Es un proceso en el cual una expresión booleana se simplifica para que requiera menos compuertas (espacio).

Minitérmino: Término producto (And) que contiene todas las variables de la función, ya sea en su forma normal o complementada.

Módulo (*module*): Unidad de manipulación y almacenamiento de un software. Incluye, módulos de código fuente, módulos de código binario, módulos de código ejecutable.

MOS (*Metal-Oxide Semiconductor*): Tecnología para crear transistores controlados por voltaje.

No-volátil: Memoria que no necesita estar alimentada para conservar la información programada.

OTP (*One Time Programmable*): Sólo puede ser programado una vez.

PAL (*Programmable Array Logic*): Arquitectura que simplifica la de los PLAS. En ésta los arreglos de OR son fijos y los de AND son programables.

PIC (*Programmable Integrated Circuit*): Cualquier circuito integrado que puede ser programado después de la fabricación de las capas de silicio.

PLA (*Programmable Logic Array*): Arquitectura que utiliza un arreglo de AND programable, en serie con un arreglo de OR programable.

PLD (*Programmable Logic Device*): Circuito que puede ser configurado por el usuario para que realice una función lógica. Éstos suelen estar constituidos por un arreglo de compuertas ANDs seguidos por un arreglo de compuertas ORs. Normalmente se utiliza para pequeños PLDs como PALS y FPLAS.

Producto de sumas: Expresión lógica igual a la salida de un arreglo de compuertas OR seguido por un arreglo de compuertas AND.

Producto de términos: Es igual a la salida de un arreglo de compuertas AND.

Programable: Que se puede configurar de una manera deseada.

Registro de entrada: Flip-Flop o un *Latch* en algunos CPLDs que mantiene las señales de entrada, utilizado cuando se multiplexa el bus.

Retroalimentación: Camino que conecta una señal generada internamente a una entrada. Suele ser programada y permite funciones lógicas.

Señal (*signal*): Evento, identificado mediante un nombre, que puede ser invocado explícitamente. Las señales pueden tener parámetros. Una señal puede ser difundida (*broadcast*) o dirigida a un objeto o grupo de objetos, en particular.

Signatura (*signature*): Nombre y parámetros de una operación, mensaje o evento. Opcionalmente los parámetros pueden incluir un parámetro devuelto como resultado de la operación, mensaje o evento. Sinónimo: signature [OMA].

Sistema (*system*): Colección de unidades conectadas entre sí, que están organizadas para llevar a cabo un propósito específico. Un sistema puede describirse mediante uno o más modelos, posiblemente desde puntos de vista distintos.

Sistemas combinacionales: Son aquéllos en donde los valores de salida únicamente dependen de las combinaciones de entrada.

Sistemas secuenciales: Son aquéllos en donde los valores de salida no dependen únicamente de las combinaciones de entrada sino también de las salidas mismas.

SSI (*Small Scale Integration*): Medida de complejidad de un circuito integrado que es equivalente a 10 compuertas.

***Standard Cell*:** Un método de diseño de circuitos *semicustom* o *full custom* en el cual se juntan las células predefinidas para obtener una función predeterminada.

Tabla de verdad (*Truth Table*): Forma de representación tabular de una función en la que se indica el valor de la salida o salidas para cada una de las posibles combinaciones que las variables de entrada pueden tomar.

Terminal: Extremo de un conductor preparado para facilitar su conexión con un aparato.

***Three State*:** Es un tipo de salida de un dispositivo lógico que puede tomar el valor de uno, cero y alta impedancia.

Transición (*transition*): Una relación entre dos estados que indica que un objeto que está en el primer estado realizará una acción especificada y entrará en el segundo estado cuando un evento especificado ocurra y unas condiciones especificadas sean satisfechas. En dicho cambio de estado se dice que la transición es disparada.

TTL (*Transistor Transistor Logic*): Familia de dispositivos lógicos bipolares más usada.

VERILOG: Lenguaje de diseño donde se introduce la descripción de hardware de alto nivel.

VHDL *(VHSIC Hardware Description Language*): Es uno de los lenguajes de programación de dispositivos lógicos más utilizados. Creado por el Departamento de Defensa de Estados Unidos.

VHSIC (*Very High Speed Integrated Circuit*): Circuito integrado de muy alta velocidad. Se comenzó a desarrollar por el Departamento de Defensa de los Estados Unidos en 1979.

ZIF (*Zero Insertion Force Socket*): *Socket* del programador para insertar dispositivos sin necesiad de ejercer fuerza.

BIBLIOGRAFÍA

César A. Leal Chapa, FIME, UANL.

Charles H. Roth, Jr., WEST, *Digital Design.*

Charles R. Kime, Prentice Hall, *Fundamentals of logic Design*.

Gerald R. Peterson, LIMUSA, *Electrónica Industrial Moderna.*

Ronald J. Toccci, Prentice Hall, *Teoría de conputación y diseño lógico.*

Thomas L. Floyd, Prentice Hall, *Sistemas Digitales Principios y Aplicaciones.*

Timothy J. Maloney, Prentice Hall, *Fundamentos de Diseño Digital.*

Wakerly, Principles & Practices, Prentice Hall, *Fundamentos de Sistemas Digitales.*

Páginas de Internet

EETOOLS http://www.eetools.com/product.htm

Lattice Semiconductors http://www.latticesemi.com/

M.C. Juan Ángel Garza Garza http://jagarza.fime.uanl.mx

University of Pennsylvania http://www.ee.upenn.edu/rca/software/abel/abel.primer.html http://fling.seas.upenn.edu

XELTEK http://www.xeltek.com