UNIVERSIDAD AUTONOMA DE NUEVO LEON

FACULTAD DE INGENIERIA MECANICA Y ELECTRICA

DIVISION DE ESTUDIOS DE POSGRADO

APLICACION DE HERRAMIENTAS COMPUTACIONALES AL DISEÑO SECUENCIAL ASINCRONO

POR

ING. BIANCA MIRALDA MEDINA LOTT

T E S I S

EN OPCION AL GRADO DE MAESTRO EN CIENCIAS DE LA INGENIERIA ELECTRICA CON ESPECIALIDAD EN ELECTRONICA

SAN NICOLAS DE LOS GARZA, N. L.
CIUDAD UNIVERSITARIA SEPTIEMBRE DE 2005

APLICACION DE HERRAMIENTAS COMPUTACIONALES AL DISEÑO SECUENCIAL ASINCRONO

UNIVERSIDAD AUTONOMA DE NUEVO LEON
FACULTAD DE INGENIERIA MECANICA
Y ELECTRICA
DIVISION DE ESTUDIOS DE POSGRADO

APLICACION DE HERRAMIENTAS COMPUTACIONALES AL DISEÑO SECUENCIAL ASINCRONO

POR

ING. BIANCA MIRALDA MEDINA LOTT

T E S I S

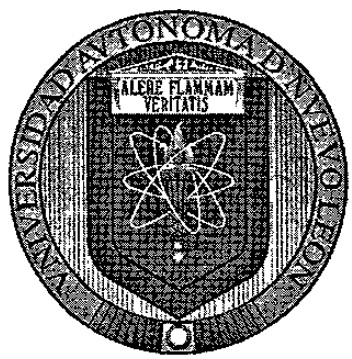
EN OPCION AL GRADO DE MAESTRO EN
CIENCIAS DE LA INGENIERIA ELECTRICA
CON ESPECIALIDAD EN ELECTRONICA

SAN NICOLAS DE LOS GARZA, N. L.
CIUDAD UNIVERSITARIA SEPTIEMBRE DE 2005

UNIVERSIDAD AUTONOMA DE NUEVO LEON

FACULTAD DE INGENIERIA MECANICA Y ELECTRICA

DIVISION DE ESTUDIOS DE POST-GRADO



APLICACIÓN DE HERRAMIENTAS COMPUTACIONALES AL DISEÑO SECUENCIAL ASINCRONO

POR

ING. BIANCA MIRALDA MEDINA LOTT

TESIS

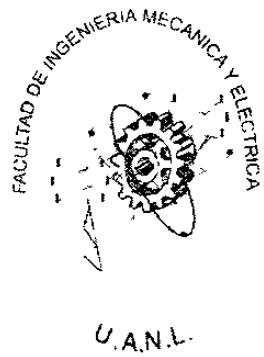
EN OPCION AL GRADO DE MAESTRO EN CIENCIAS DE LA INGENIERIA ELECTRICA CON ESPECIALIDAD EN ELECTRONICA

CIUDAD UNIVERSITARIA, SEPTIEMBRE DEL 2005

UNIVERSIDAD AUTONOMA DE NUEVO LEON

FACULTAD DE INGENIERIA MECANICA Y ELECTRICA

DIVISION DE ESTUDIOS DE POST-GRADO



APLICACIÓN DE HERRAMIENTAS COMPUTACIONALES AL DISEÑO SECUENCIAL ASINCRONO

POR

ING. BIANCA MIRALDA MEDINA LOTT

TESIS

EN OPCION AL GRADO DE MAESTRO EN CIENCIAS DE LA INGENIERIA ELECTRICA CON ESPECIALIDAD EN ELECTRONICA

CIUDAD UNIVERSITARIA, SEPTIEMBRE DEL 2005

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica
División de Estudios de Post-grado

Los miembros del Comité de Tesis recomendamos que la Tesis "Aplicación de Herramientas Computacionales al Diseño Secuencial Asíncrono", realizada por la Ing. Bianca Miralda Medina Lott con número de matricula 0898436 sea aceptada para su defensa como opción al grado de Maestro en Ciencias de la Ingeniería Eléctrica con Especialidad en Electrónica.

El Comité de Tesis

M.C. Juan Angel Garza Garza
Asesor

M.C. José Manuel Rocha Núñez
Revisor

Dr. César Elizondo González
Revisor

VoBo

Dr. Guadalupe Alan Castillo Rodríguez
División de Estudios de Postgrado

Ciudad Universitaria, Septiembre del 2005

AGRADECIMIENTOS

Mi más sincero agradecimiento a M.C. Juan Angel Garza, quien generosamente brindó su tiempo para la elaboración de ésta Tesis.

A mis asesores M.C. José Manuel Rocha Núñez y al Dr. Cesar Elizondo González por su apoyo en el desarrollo de esta tesis.

Principalmente a mi familia, por el ánimo que me infundieron y su apoyo inquebrantable.

INDICE

SINTESIS

Esta tesis tiene como objetivo proponer una metodología que nos permita resolver secuencias lógicas asíncronas. También provee una descripción de las nuevas herramientas computacionales propuestas para el tema, así como una introducción a estas herramientas, introducción con la cual se pretende que los interesados sean capaces de hacer uso de ellas.

La metodología propuesta se basa en una serie de pasos que con la ayuda de las nuevas herramientas computacionales nos permite resolver las secuencias lógicas asíncronas complejas en un tiempo menor y con una mayor eficiencia que la forma tradicional.

Esta metodología abarca desde la especificación del sistema hasta la implementación de las secuencias lógicas asíncronas en dispositivos del tipo GAL o PLD's, o en un controlador lógico programable llamado también PLC.

CAPITULO 1

INTRODUCCIÓN

Las secuencias lógicas son utilizadas ampliamente en nuestro tiempo como medio de control dentro diferentes procesos industriales, ya que permiten ser implementadas en herramientas especializadas como PLC's (Controladores Lógicos Programables), o dentro de chips programables de bajo costo llámense PLD's (Dispositivo de Lógica Programable), GAL's (Generic Array Logic), etc.

Descripción del Problema

El diseño tradicional de un sistema secuencial asíncrono es sumamente complejo y a la vez requiere de experiencia y del dominio de métodos y técnicas para la simplificación y optimización del diseño, y normalmente se lleva mucho tiempo para su elaboración.

La metodología de elaboración de las secuencias lógicas que ha sido utilizada, consta de serios problemas ya que tradicionalmente se manejan bloques ya previamente diseñados que no necesariamente están optimizados, por que anteriormente no se contaba con las herramientas modernas que ahora existen, por consecuencia, esto implica tener componentes innecesarios, con una respuesta de tiempo elevada, y de contar con variables auxiliares redundantes dentro de una secuencia, lo que repercute directamente en el costo del diseño.

Objetivo de la Tesis

Demostrar que con las actuales herramientas computacionales para la resolución de secuencias lógicas apoyadas con las diversas metodologías de diseño, se simplifica el diseño de un circuito secuencial asíncrono con respecto al método tradicional.

Hipótesis

La propuesta es obtener un resultado óptimo de secuencias lógicas empleando métodos sofisticados por medio de software, proporcionando un manejo adecuado a una cierta cantidad considerable de variables, con la finalidad de dosificar el tiempo y esfuerzo en la elaboración de estas para el interesado en cuestión.

Limites de Estudio

Esta tesis abarcará el estudio y análisis de diferentes métodos del diseño secuencial asíncrono utilizando herramientas computacionales así como sus diferentes formas de implementación.

Justificación

A partir de que comenzó a ser utilizado el diseño secuencial en el control de procesos dentro de la industria y algunos otros campos, se ha podido observar las serias limitantes que existen en la elaboración de secuencias lógicas, limitantes como son el tiempo utilizado en su resolución y la inseguridad de lograr una secuencia lógica minimizada y optima en su diseño. Ya que es necesario contar con una cierta experiencia para poder elaborar las secuencias lógicas.

Con las herramientas desarrolladas últimamente es posible obtener secuencias lógicas óptimas en un menor tiempo y contando únicamente con los conocimientos necesarios

de los métodos existentes, lo que resulta en la sencillez de la elaboración e implementación de secuencias lógicas con costos mínimos de entrenamiento para ello.

Metodología

Se desarrollaran las secuencias lógicas, a partir de las últimas herramientas de softwarc disponibles.

Se resolverán ejemplos con estas herramientas para verificar las ventajas y desventajas de su uso dentro del diseño secuencial.

Se determinaran los indicadores adecuados para la valoración de este método como lo es el tiempo, cantidad de variables, etc.

Revisión Bibliográfica

El soporte tecnológico y bibliográfico que se requiera para el desarrollo de esta tesis, se relacionará.

Además de aprovechar la información que se encuentra en la red y mantener contacto directo utilizando este medio de comunicación con tecnólogos que se relacionan con el tema.

CAPITULO 2

DISPOSITIVOS LOGICOS

2.1.- ASICS

Desde los finales de la década de 1970, los equipos electrónicos digitales utilizan Circuitos Integrados (*CI* o *CHIPS*) de función lógica fija, realizados en pequeña o mediana escala de integración (SSI, MSI).

Para la implementación de aplicaciones muy complejas, que requieren de una gran cantidad de circuitos de función fija, por lo que resulta más conveniente integrarlos en un solo dispositivo fabricado a la medida, los cuales son llamados: ASICS, *Application Specific Integrated Circuits.*(Circuitos Integrados de Aplicación Especifica o circuitos a la medida).

Entre las ventajas que presenta el uso de los ASIC's podemos mencionar que: ahorran espacio, reducen el número de dispositivos, tienen menor costo, reducen el tiempo de ensamble, bajo consumo de potencia, menor calentamiento, facilidad en la verificación (control de calidad) y mejor confiabilidad.

Los ASIC se pueden clasificar por su tecnología de fabricación en cuatro categorías (ver figura 2.1.1): Arreglos de Compuertas, Celdas Estándar, Full Custom y Lógica Programable

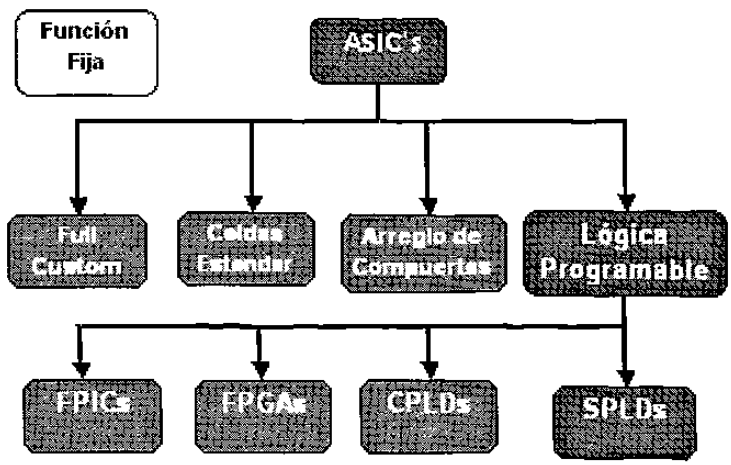


Figura 2.1.1.- Clasificación de circuitos integrados ASICS

Las tecnologías de Arreglos de Compuertas, Celdas Estándar y Full Custom, están encaminadas a la producción industrial de alto volumen y requieren de equipo especializado para la fabricación del ASIC.

Por otro lado, con la Lógica Programable es posible diseñar e implementar funciones desde un solo circuito con el uso de solamente una computadora, un programador y software de Diseño Electrónico.

2.1.1.- Dispositivos de Lógica Programable (PLD)

Un dispositivo de lógica programable (PLD) es un Circuito Integrado cuya estructura lógica final es directamente configurada por el usuario, sin necesidad de llevar a cabo ningún proceso de fabricación.

Peggy Aycinena de la revista electrónica Integrated System Design asegura que los dispositivos lógicos programables son la ola del futuro porque presentan las siguientes

características: 10,000 compuertas en 1 in^2, entradas y salidas configurables, reprogramables y programables remotamente para diferentes funciones.

Los PLDs facilitan el proceso de diseño y reducen el tiempo de desarrollo, cuando se requieren prototipos o producción de baja escala, pues todo el proceso se puede llevar a cabo con la ayuda de una computadora personal, programas de aplicación y el programador los cuales actualmente están disponibles a bajo costo.

Algunos fabricantes de PLD's son:

Actel (www.actel.com), Altera Corp. (www.altera.com),
Atmel Corp. (www.atmel.com), Chip Express (www.chipexress.com),
Cypress Sem. (www.cypress.com), Lattice Sem. (www.latticesemi.com),
Quicklogic Corp. (www.quicklogic.com), Xilinx Inc. (www.xilinx.com).

Los diferentes tipos de dispositivos de lógica programable que existen hoy en día pueden clasificarse por su tecnología o su capacidad (Figura 2) tales como:

- Simplex Programmable Logic Device SPLDs.
- Complex Programmable Logic Device CPLDs.
- Field Programmable Gate Array FPGAs.
- Field Programmable Inter Connect FPICs.

De la clasificación anterior sólo nos enfocaremos, como ejemplos a los Simples Programmable Logic Device SPLDs.

Los SPLDs están constituidos por un arreglo de compuertas AND, seguido por otro arreglo de compuertas OR, con uno o ambos arreglos programables como podemos observar en la figura 2.1.2. Algunos incluyen Flip Flops.

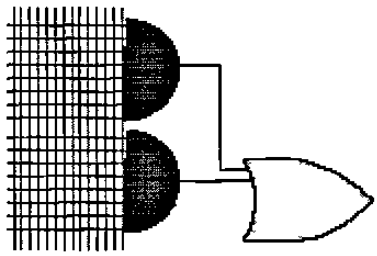


Figura 2.1.2 Arreglo And Or de un SPLD.

A su vez los SPLDs se pueden clasificar según su estructura interna en (ver figura 2.1.3):

- PAL Programmable Array Logic, VANTIS.
- GAL Generic Array Logic, Lattice Semiconductor.
- PLA Programmable Logic Array.
- PLD Programmable Logic Device.

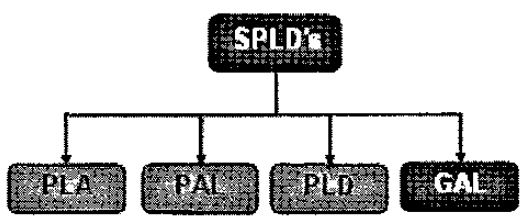


Figura 2.1.3.- Clasificación de SPLDs

De estos tipos de SPLDs, el GAL destaca por su bajo precio y versatilidad por lo que lo describiremos en el siguiente punto.

2.1.2.- Arreglo Lógico Genérico (GAL).

GAL (Generic Array Logic), en español Arreglo Lógico Genérico, es un tipo de circuito integrado, de marca registrada por Lattice Semiconductor, que ha sido diseñado con el propósito de sustituir a la mayoría de las PAL, manteniendo la compatibilidad de sus terminales.

El GAL básicamente está formado por una matriz AND reprogramable y una matriz OR fija con configuración programable de salidas y/o entradas.

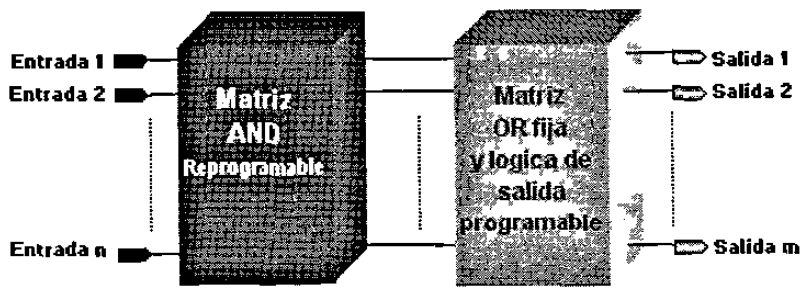


Figura 2.1.4 Estructura Básica de un GAL

Como un ejemplo de las características ofrecidas por este tipo de dispositivos, a continuación se enlistan las especificaciones más relevantes del circuito GAL16V8 de marca Lattice Semiconductor.

- fmax = 250 Mhz.
- 3.5 ns máximo tiempo de propagación.
- 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.

En el GAL16V8, las terminales tienen las siguientes funciones: La terminal 1 es la entrada de CLK, las terminales del 2 a la 9 son ocho Entradas fijas, la terminal 10 de GND, la terminal 11 como una Entrada de control O/E (Output/Enable), de la 12 a la 19 ocho terminales programables OLMC y terminal 20 de VCC.

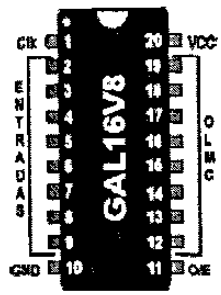


Figura 2.1.5 Distribución de Terminales del GAL 16V8.

Las terminales de la 12 a 19 correspondientes al OLMC (*Output Logic Macrocell)* pueden programarse para trabajar como entradas y/o salidas, y en el caso de ser usadas como salidas estas pueden ser combinacionales o registradas (Flip Flops), lo cual le da la versatilidad de ser programado de diferentes formas y para diferentes requerimientos.

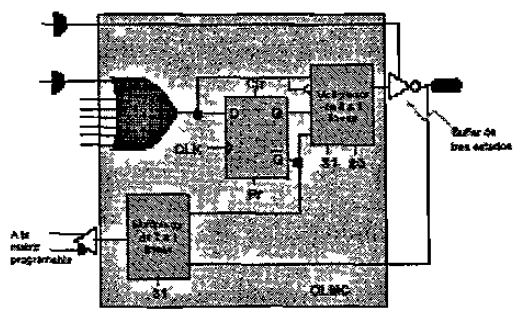


Figura 2.1.6 Configuración interna de la OLMC de un GAL 16V8.

La programación de los PLDs en general se lleva a cabo por medio de programas de aplicación especializados, siendo las dos estrategias de programación más utilizadas, la captura esquemática y la de lenguaje de descripción de hardware (HDL).

La gran ventaja de estas herramientas es hacer los diseños en la computadora, donde los errores son fácilmente detectables y corregibles.

2.2.- Controladores Lógicos Programables

Definición

Se entiende por controlador lógico programable (PLC), o autómata programable, a toda máquina electrónica diseñada para controlar en tiempo real y en medio industrial procesos secuenciales.
Esta definición se está quedando un poco desfasada, ya que han aparecido los micro-plc's, destinados a pequeñas necesidades y al alcance de cualquier persona.

Campos de aplicación

Un autómata programable suele emplearse en procesos industriales que tengan una o varias de las siguientes necesidades:

- Espacio reducido.
- Procesos de producción periódicamente cambiantes.
- Procesos secuenciales.
- Maquinaria de procesos variables.
- Instalaciones de procesos complejos y amplios.
- Chequeo de programación centralizada de las partes del proceso.

Aplicaciones generales:

- Maniobra de máquinas.
- Maniobra de instalaciones.
- Señalización y control.

Tal y como dijimos anteriormente, esto se refiere a los autómatas programables industriales, dejando de lado los pequeños autómatas para uso más personal (que se pueden emplear, incluso, para automatizar procesos en el hogar, como la puerta de un cochera o las luces de la casa).

Ventajas e inconvenientes de los PLC's

Entre las ventajas tenemos:

- Menor tiempo de elaboración de proyectos.
- Posibilidad de añadir modificaciones sin costo añadido en otros componentes.
- Mínimo espacio de ocupación.
- Menor costo de mano de obra.
- Mantenimiento económico.
- Posibilidad de gobernar varias máquinas con el mismo autómata.
- Menor tiempo de puesta en funcionamiento.
- Si el autómata queda pequeño para el proceso industrial puede seguir siendo de utilidad en otras máquinas o sistemas de producción.

Y entre los inconvenientes:

- Adiestramiento de técnicos.
- Costo.

Al día de hoy los inconvenientes se han hecho nulos, ya que la mayoría de las carreras de ingeniería incluyen la automatización como una de sus asignaturas. En cuanto al costo tampoco hay problema, ya que hay autómatas para todas las necesidades y a precios ajustados (tenemos desde pequeños autómatas por poco más de $1000. hasta PLC's que alcanzan cifras mucho mas elevadas).

Estructura externa

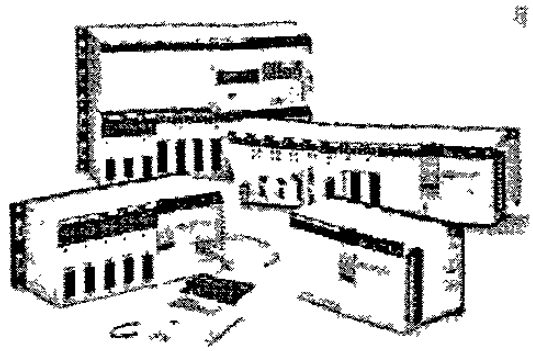


Figura 2.2.1 Estructura Externa de un PLC

Todos los autómatas programables, poseen una de las siguientes estructuras:

- Compacta: en un solo bloque están todos lo elementos.
- Modular.
- Estructura americana: separa las E/S del resto del autómata.
- Estructura europea: cada módulo es una función (fuente de alimentación, CPU, E/S, etc.).

Exteriormente nos encontraremos con cajas que contienen una de estas estructuras (como podemos observar en la figura 2.2.1), las cuales poseen indicadores y conectores en función del modelo y fabricante.

Para el caso de una estructura modular se dispone de la posibilidad de fijar los distintos módulos en railes normalizados, para que el conjunto sea compacto y resistente.

Los micro-autómatas suelen venir sin caja, en formato kit, ya que su empleo no es determinado y se suele incluir dentro de un conjunto más grande de control o dentro de la misma maquinaria que se debe controlar.

Estructura interna

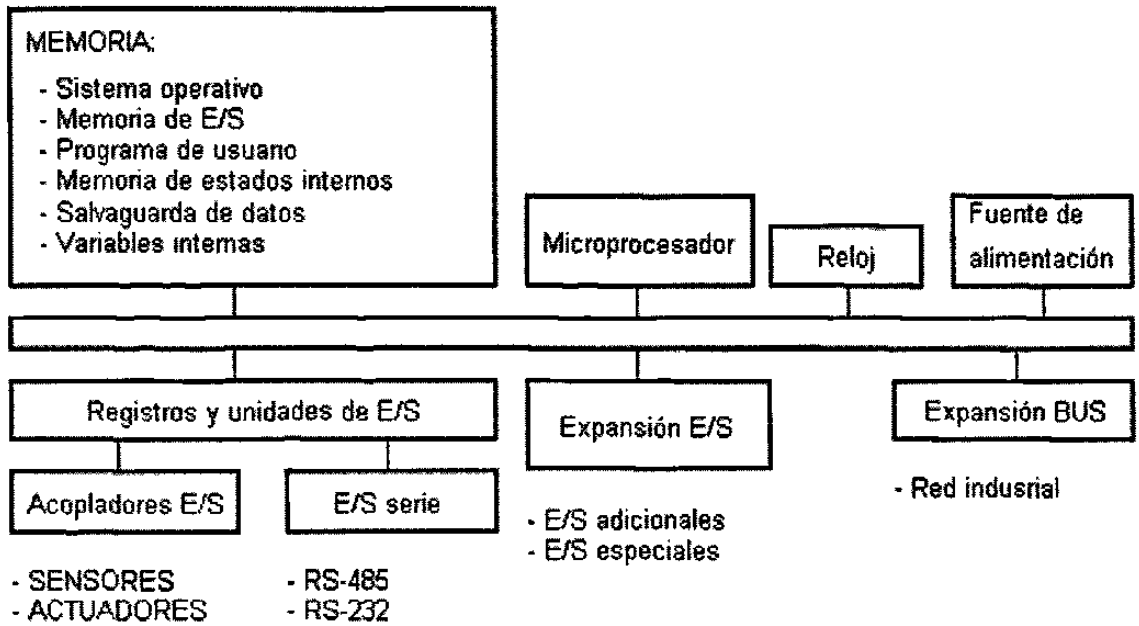


Figura 2.2.2 Estructura Interna de un PLC

Los elementos esenciales, que todo autómata programable posee como mínimo, son:

- Sección de entradas: se trata de líneas de entrada, las cuales pueden ser de tipo digital o analógico. En ambos casos tenemos unos rangos de voltaje característicos, los cuales se encuentran en las hojas de datos del fabricante. A estas líneas conectaremos los sensores.
- Sección de salidas: son una serie de líneas de salida, que también pueden ser de carácter digital o analógico. A estas líneas conectaremos los actuadores.
- Unidad central de proceso (CPU): se encarga de procesar el programa de usuario que le introduciremos. Para ello disponemos de diversas zonas de memoria, registros, e instrucciones de programa.

Adicionalmente, en determinados modelos más avanzados, podemos disponer de funciones ya integradas en la CPU; como reguladores PID, control de posición, etc.

Tanto las entradas como las salidas están aisladas de la CPU según el tipo de autómata que utilicemos. Normalmente se suelen emplear optó-acopladores en las entradas, y relévadores /optó-acopladores en las salidas.

Aparte de estos elementos podemos disponer de los siguientes (ver figura 2.2.2):

- Unidad de alimentación (algunas CPU la llevan incluida).
- Unidad o consola de programación: que nos permitirá introducir, modificar y supervisar el programa de usuario.
- Dispositivos periféricos: como nuevas unidades de E/S, más memoria, unidades de comunicación en red, etc.
- Interfaces: facilitan la comunicación del autómata mediante enlace serie con otros dispositivos (como una PC).

Memoria

Dentro de la CPU vamos a disponer de un área de memoria, la cual emplearemos para diversas funciones:

- Memoria del programa de usuario: aquí introduciremos el programa que el autómata va a ejecutar cíclicamente.
- Memoria de la tabla de datos: se suele subdividir en zonas según el tipo de datos (como marcas de memoria, temporizadores, contadores, etc.).
- Memoria del sistema: aquí se encuentra el programa en código máquina que monitoriza el sistema (programa del sistema o firmware). Este programa es ejecutado directamente por el microprocesador/microcontrolador que posea el autómata.
- Memoria de almacenamiento: se trata de memoria externa que empleamos para almacenar el programa de usuario, y en ciertos casos parte de la memoria de la tabla de datos. Suele ser de uno de los siguientes tipos: EPROM, EEPROM, o FLASH.

CPU (Unidad central de procesamiento)

La CPU es el corazón del autómata programable. Es la encargada de ejecutar el programa de usuario mediante el programa del sistema (es decir, el programa de usuario es interpretado por el programa del sistema). Sus funciones son:

- Vigilar que el tiempo de ejecución del programa de usuario no excede un determinado tiempo máximo (tiempo de ciclo máximo). A esta función se le suele denominar Watchdog (perro guardián).
- Ejecutar el programa de usuario.
- Crear una imagen de las entradas, ya que el programa de usuario no debe acceder directamente a dichas entradas.
- Renovar el estado de las salidas en función de la imagen de las mismas obtenida al final del ciclo de ejecución del programa de usuario.
- Chequeo del sistema.

Para ello el autómata va a poseer un ciclo de trabajo, que ejecutará de forma continua:

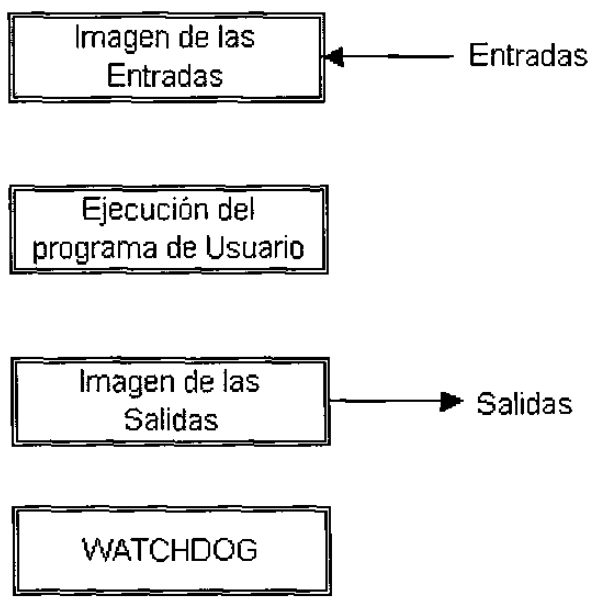


Figura 2.2.3 Ciclo de Trabajo de un Autómata Programable

Unidades de Entrada/Salida (E/S).

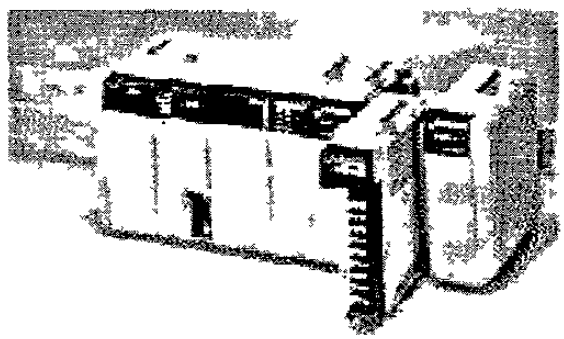


Figura 2.2.4 Modulo de Entradas y Salidas de un PLC.

Generalmente vamos a disponer de dos tipos de E/S:

- Digital
- Analógica

Las E/S digitales se basan en el principio de todo o nada, es decir o no conducen señal alguna o poseen un nivel mínimo de voltaje. Estas E/S se manejan al nivel de bit dentro del programa de usuario.

Las E/S analógicas pueden poseer cualquier valor dentro de un rango determinado especificado por el fabricante. Se basan en conversores A/D y D/A aislados de la CPU (ópticamente o por etapa de potencia). Estas señales se manejan al nivel de byte o palabra (8/16 bits) dentro del programa de usuario.

Las E/S son leídas y escritas dependiendo del modelo y del fabricante, es decir pueden estar incluidas sus imágenes dentro del área de memoria o ser manejadas a través de instrucciones específicas de E/S.

La figura 2.2.4 muestra un modulo de entradas y salidas.

Interfaces

Todo autómata, salvo casos excepcionales, posee la virtud de poder comunicarse con otros dispositivos (como una PC). Lo normal es que posea una E/S serie del tipo RS-232 / RS-422.

A través de esta línea se pueden manejar todas las características internas del autómata, incluyendo la programación del mismo, y suele emplearse para monitorización del proceso en otro lugar separado.

Equipos o unidades de programación

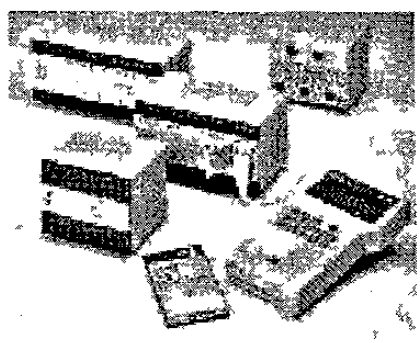


Figura 2.2.5 Equipos de Programación.

El autómata debe disponer de alguna forma de programación, la cual se suele realizar empleando alguno de los siguientes elementos (ver figura 2.2.5):

- Unidad de programación: suele ser en forma de calculadora. Es la forma más simple de programar el autómata, y se suele reservar para pequeñas modificaciones del programa o la lectura de datos en el lugar de colocación del autómata.
- Consola de programación: es una terminal a modo de ordenador que proporciona una forma más cómoda de realizar el programa de usuario y observar parámetros internos del autómata. Obsoleto actualmente.
- PC: es el modo más potente y empleado en la actualidad. Permite programar desde un ordenador personal estándar, con todo lo que ello supone: herramientas más

potentes, posibilidad de almacenamiento en soporte magnético, impresión, transferencia de datos, monitorización mediante software SCADA, etc.

Para cada caso el fabricante proporciona lo necesario, bien el equipo o el software/cables adecuados. Cada equipo, dependiendo del modelo y fabricante, puede poseer una conexión a uno o varios de los elementos anteriores. En el caso de los micro-plc se escoge la programación por PC o por unidad de programación integrada en la propia CPU.

Dispositivos periféricos.

El autómata programable, en la mayoría de los casos, puede ser ampliable. Las ampliaciones abarcan un gran abanico de posibilidades, que van desde las redes internas (LAN, etc.), módulos auxiliares de E/S, memoria adicional hasta la conexión con otros autómatas del mismo modelo.

Cada fabricante facilita las posibilidades de ampliación de sus modelos, los cuales pueden variar incluso entre modelos de la misma serie.

CAPITULO 3

METODOLOGIA DEL DISEÑO DE SISTEMAS SECUENCIALES ASINCRONOS

3.1 Método de Roth

Estos sistemas pueden ser implementados con los PLDs, ya que las características de las OLMC (Output Logic Macrocell) es permitir la retroalimentación, también las ecuaciones obtenidas se pueden representar por medio de un diagrama escalera e implementarse en un Controlador lógico programable PLC.

A continuación se muestran 14 pasos sugeridos para el desarrollo y diseño de sistemas secuenciales asíncronos:

1.- Especificar el Sistema.

Por medio de un Diagrama de Tiempos o Diagrama de transición se puede describir cada uno de los eventos del sistema.

2.- Tabla de Flujo Primitiva.

En una tabla de flujo (o transiciones) se tiene la misma información que en un diagrama de estados, pero organizada de forma tabular. Si en la construcción de ella se establecen las siguientes condiciones:

a).- Solo debe de haber un estado estable por fila.
b).- Solo 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 tiempos o en el diagrama de transición de tal suerte que se asegura la consideración de todos los eventos posibles en el funcionamiento u operación de un diseño.

3.- Eliminación de estados redundantes o equivalentes.

En el proceso de la descripción de los eventos por medio del diagrama de tiempos o el diagrama de transición en algunos casos se pueden incluir eventos innecesarios o redundantes a los que se les denomina estados equivalentes, para determinar si dos estados son equivalentes se tiene que cumplir las siguientes tres condiciones:

a) Son estados estables en la misma columna (misma combinación de entradas).

b) Tienen la misma salida.

c) Sus estados siguientes son equivalentes.

4.- Mezcla de Filas.

Las filas o hileras pueden mezclarse con el propósito de reducir la tabla y con ello utilizar una menor cantidad de variables.

Dos filas o más se pueden mezclar siempre y cuando, no haya ningún conflicto sobre que 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, este consiste en asignar un punto por cada fila y se unen esos puntos por líneas cuando estos pueden mezclarse.

5.- Expandir tabla de salidas.

Si para una hilera se tiene la posibilidad de mezclar filas pero los estados estables tienen salidas diferentes es conveniente expandir las salidas convirtiéndose en modelo de Mealy.

En el caso de que no sea necesaria la expansión entonces las salidas solo dependerían de los valores de memoria Q y se considera modelo de Moore.

6.- Tabla de estados internos.

Convertir la tabla de estados totales obtenida al mezclar las filas en una tabla de estados internos.

7.- Asignación de Valores a los Estados.

Para cumplir con las transiciones descritas en la tabla de estados internos se asignan valores a las variables de modo que estos cumplan que solo cambie un valor entre dichas transiciones.

8.- Tabla de Estados.

En esta tabla se sustituyen los estados internos por el valor de la asignación propuesta en el paso anterior.

9.- Completar Tabla de Salidas.

En algunos casos las salidas no están completamente definidas y esto puede generar valores transitorios no convenientes para el sistema por tal razón es necesario asignar un valor a la salida de modo que no se presente el transitorio.

10.- Obtención de las ecuaciones por medio de minimización.

Se puede utilizar recursos como Manipulación algebraica, Mapas de karnaugh o el uso de programas de aplicación para obtener las ecuaciones mínimas.

11.- Elaborar el Archivo en formato ABEL-HDL.

Por medio de esta herramienta nos permite efectuar la simulación para comprobar el si el funcionamiento obtenido es el deseado del sistema.

12 Simulación.

Por lo general la forma de presentarse es por medio de un diagrama de tiempo en donde se incluyen las entradas y salidas así como los valores de los estados (Q's).

13.- Representación grafica.

a) Diagrama esquemático
b) Diagrama escalera

14.- Implementación

a) PLD
b) PLC

3.2.- Herramientas de Software para Sistemas Secuenciales Asíncronos

En la actualidad se han desarrollado herramientas de software y hardware que permiten elaborar aplicaciones secuenciales asíncronas mediante el uso de dispositivos digitales, estas se dividen principalmente en las siguientes categorías:

- Compiladores
- Programas de Captura Esquemática
- Lenguajes de Descripción de Hardware
- Programador
- Logic Aid

Compilador:

Los compiladores son programas de aplicación cuya función es que a partir de archivos de captura esquemática, diagramas de tiempos o Lenguajes de descripción de hardware, generen un archivo en formato JEDEC que contiene el mapa de fusibles del GAL.

Programas de Captura Esquemática:

Por Captura Esquemática se entiende el proceso de descripción, mediante un dibujo, de un circuito eléctrico, en el que se representan a los diferentes componentes del circuito y solo se efectúan interconexiones entre ellos.

Existen varios programas con la aplicación de Captura Esquemática como el "Schematic" del IspStarter de Lattice Semiconductor o "Fundation" de XILINX entre otros.

Esta técnica permite simular en la computadora el circuito virtualmente y verificar su funcionamiento antes de su fabricación o implementación en un PLD, reduciendo así el ciclo de diseño y el tiempo de obtención de un producto.

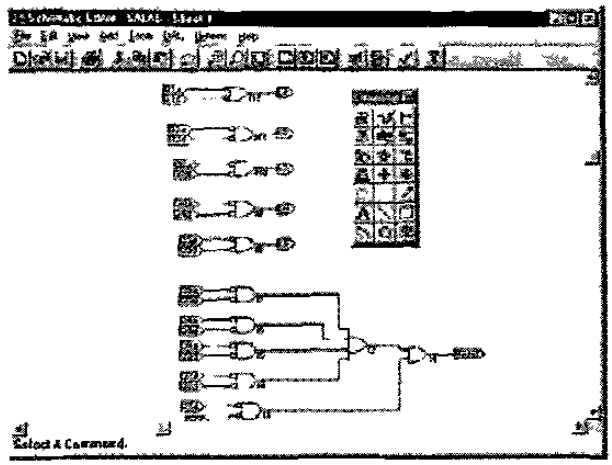
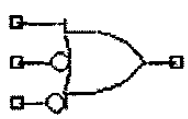
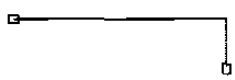


Figura 3.2.1 Programa de Captura Esquemática

Los cuatro componentes básicos de la captura esquemática son: Símbolos, Conectores, Etiquetas y Puertos de Entrada y/o Salida.

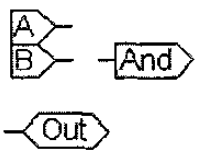


Símbolos es una representación gráfica de los componentes.



Conectores (alambre), permite la interconexión entre las terminales de los símbolos, o dispositivos de entrada/salida.

Etiquetas (Variables), Son los nombres para la identificación de las entradas o salidas.



Puertos de entrada/salida es la definición de las características de un Puerto de Entrada, Salida o Puerto Bidirecional

Utilizando el programa IspStarter de Lattice Semiconductor, el primer paso en el proceso de obtener el circuito es precisamente la captura esquemática en donde se genera un archivo de extensión .SCH, posteriormente se pasa al proceso de enlace (link) en donde se valida si los componentes son permitidos en el dispositivo seleccionado, además si el dispositivo tiene la capacidad de integrar los elementos requeridos (Fit Design). También es posible efectuar la simulación antes de obtener el circuito final con la intención de asegurar que este cumpla con los requerimientos solicitados.

En el proceso de enlace se genera un archivo reporte con extensión .REP que nos indica, entre otras cosas, la asignación de terminales (Pin Out) y el archivo JEDEC con el cual efectuaremos la programación del circuito a través de un programador.

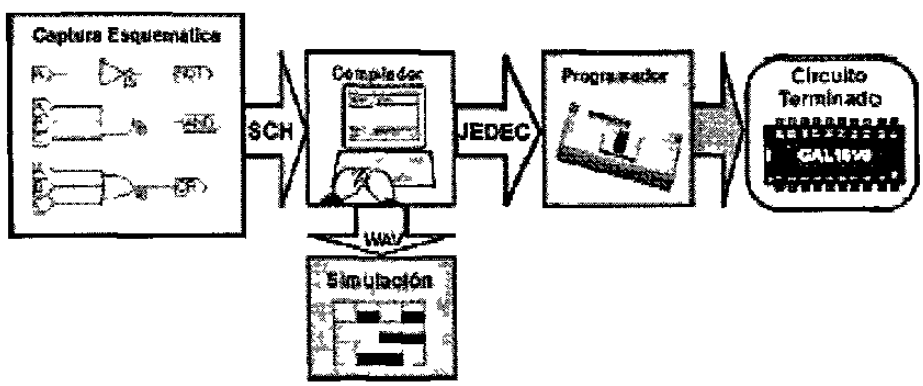


Figura 3.2.2 Diagrama del proceso de diseño digital por medio de captura esquemática.

La desventaja de la captura esquemática es que en el diseño de circuitos grandes no es posible comprenderlos debido a que hay demasiados componentes e interconexiones en la pantalla, para esos casos es recomendable el uso de lenguajes de descripción de Hardware.

Lenguajes de Descripción de Hardware:

Otra forma alternativa de diseño es usar un Lenguaje de Descripción de Hardware" (HDL), en donde se describe el funcionamiento a través de instrucciones de alto nivel en forma similar a un lenguaje de programación.

Los HDL's más usados son:

- ABEL, Advanced Boolean Expression Language.
- VHDL, en donde el acrónimo tiene dos raíces la V de VHSIC (Very High Speed Integrated Circuits) y HDL (Hardware Description Language).
- VERILOG idéntico en función al VHDL
- El diseño utilizando HDL's posee varias ventajas sobre la metodología tradicional de diseño a nivel compuerta, algunas de estas ventajas son listadas a continuación.
- Es posible verificar el funcionamiento del sistema dentro del proceso de diseño sin necesidad de implementar el circuito.
- La simulación del diseño, antes de que éste sea implementado mediante compuertas, permitiendo probar la arquitectura del sistema para tomar decisiones en cuanto a cambios en el diseño.
- Esta metodología elimina el antiguo método tedioso de diseño mediante compuertas, reduce el tiempo de diseño y la cantidad de errores producidos por el armado del circuito.
- Un circuito hecho mediante una descripción en un HDL puede ser utilizado en cualquier tipo de dispositivo programable capaz de soportar la densidad del diseño. Es decir, no es necesario adecuar el circuito a cada dispositivo porque las herramientas de síntesis se encargan de ello.

En este documento se describe la programación usando el lenguaje ABEL-HDL.

Programador:

Los programadores tienen la función de que a partir de un archivo JEDEC y auxiliados por la computadora graba el mapa de fusibles en el GAL.

3.2.1.- LOGIC AID

El programa Logic Aid es una herramienta muy útil para la simplificación de funciones booleanas, tanto en diseño secuencial como combinacional.

Esta herramienta nos permite efectuar simplificaciones a partir de minitérminos y maxiterminos, ecuaciones, tablas de verdad, tablas de estado, diagramas de transición y mapas de karnaugh.

Además otra de sus principales ventajas es que es un software que no demanda un equipo personal muy poderoso para poder ser utilizado y prácticamente puede ser utilizado en cualquier computadora personal con sistema operativo Windows.

La interfaz grafica es muy sencilla de utilizar, básicamente tiene un menú principal como lo vemos en la figura 3.2.1.1, en el cual al crear un nuevo archivo nos permite escoger entre los diferentes tipos de minimizaciones con que cuenta como se muestra en la figura 3.2.1.2.

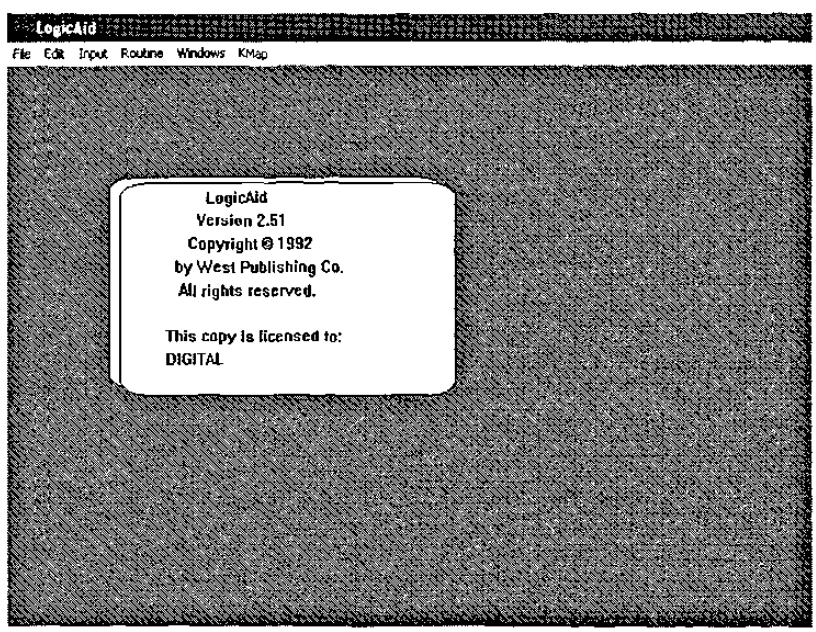


Figura 3.2.1.1- Logic Aid

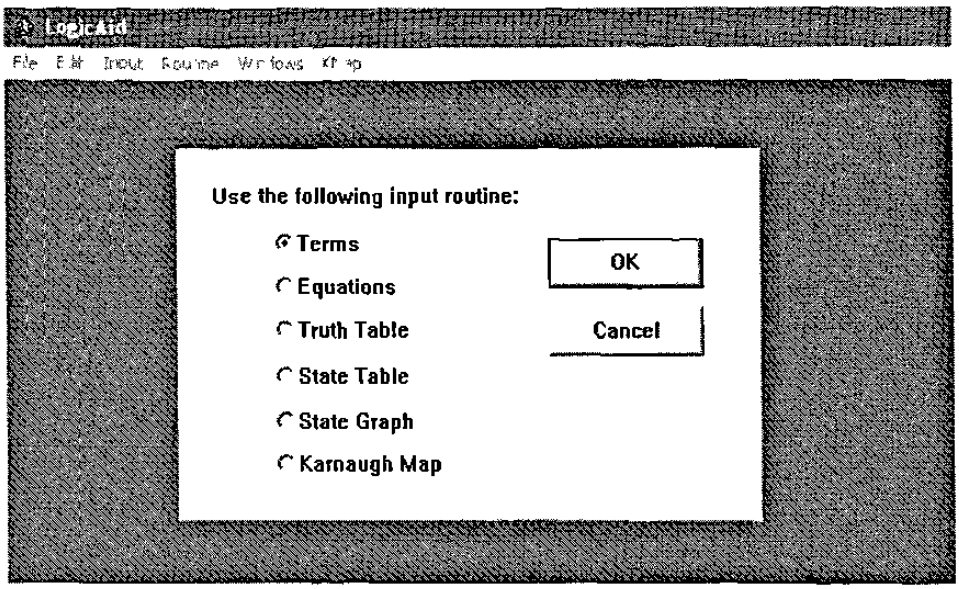
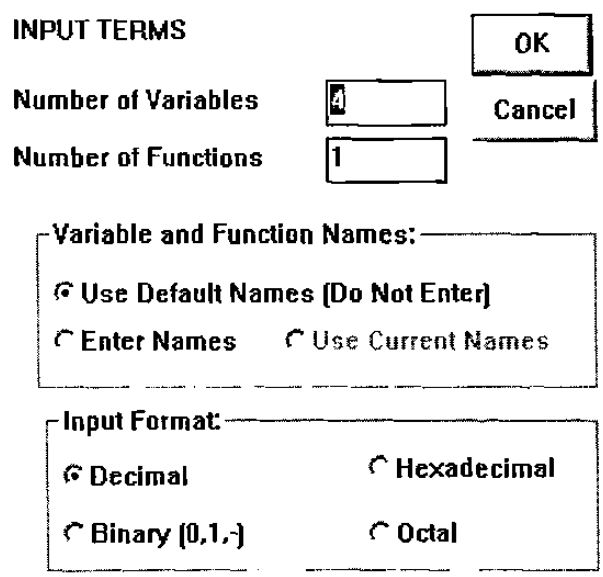


Figura 3.2.1.2-. Opciones

Si seleccionamos la primera opción "Terms" nos muestra otro menú que se muestra en la figura 3.2.1.3, donde indicamos el número de variables, las funciones, nos permite cambiar los nombres de ellas y a la vez seleccionar el formato numérico adecuado para estas.



7

Figura 3.2.1.3.- Términos de Entrada (Input Terms).

Si seleccionamos la segunda opción "Equations", nos muestra un menú donde nos pide el número de variables y de funciones, nos permite cambiar los nombres y elegir también el formato de nuestra ecuación como se muestra en la figura 3.2.1.4.

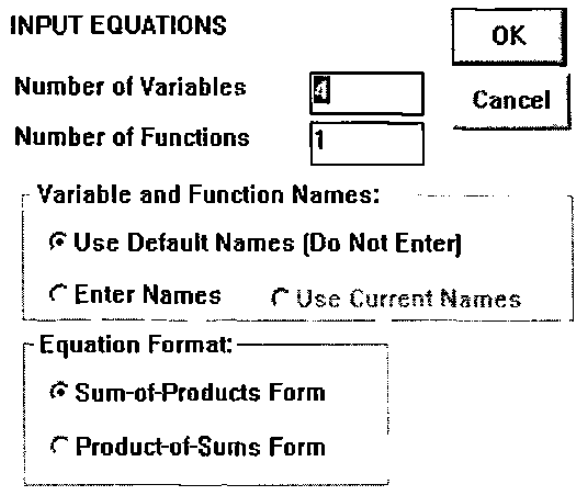


Figura 3.2.1.4.- Ecuaciones de Entrada (Input Equations).

La tercera opción, "Truth Table", nos muestra después de haberla seleccionado un menú donde nos permite seleccionar el numero de variables y de funciones, seleccionar el formato de la entrada y salida de combinaciones entre otras opciones como se muestra en la figura 3.2.1.5.

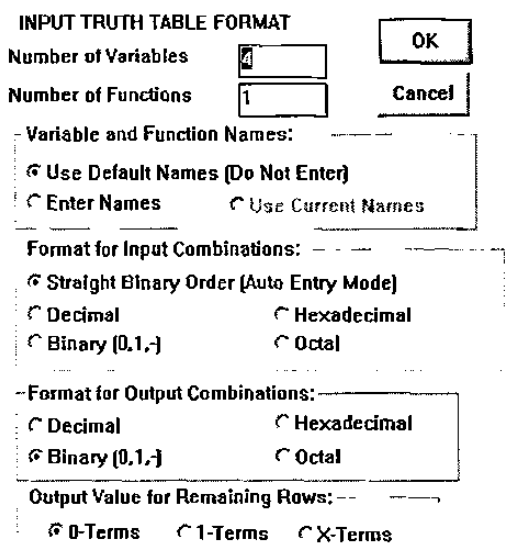


Figura 3.2.1.5.- Formato de entrada de la tabla de verdad.

La cuarta opción "State Table", nos muestra un menú donde nos permite selecciona el formato de entrada y salida de la tabla de estados, el tipo de maquina Mealy o Moore y finalmente el tipo de formato del encabezado como se muestra en la figura 3.2.1.6.

.

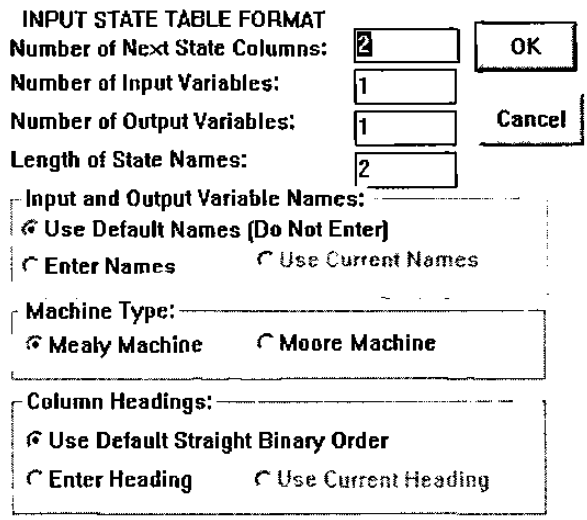
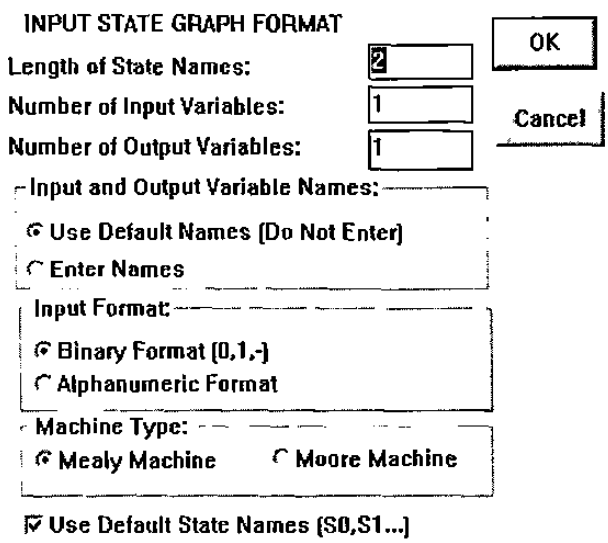


Figura 3.2.1.6.- formato de entrada de la tabla de estados.

La quinta opción "State Graph" nos permite efectuar una minimización en base a una grafica de estados, después de seleccionarla nos muestra un menú donde debemos indicar el numero de entradas y salidas, el tamaño de el nombre de nuestros estados, el formato de entrada y tipo de maquina secuencial, como se muestra en la figura 3.2.1.8.



3.2.1.8.- formato de entrada de grafica de estados.

Y finalmente la sexta opción "Karnaugh Map", que no permite efectuar minimizaciones directamente a partir de un mapa de karnaugh, nos muestra un menú donde principalmente debemos indicar el numero de variables y sus nombres como se muestra en la figura 3.2.1.9.

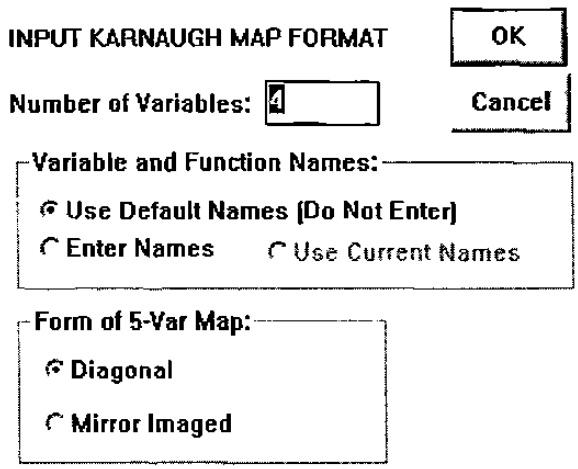


Figura 3.2.1.9.- Formato de entrada

A continuación detallamos los pasos necesarios para obtener la minimización de las ecuaciones booleanas a partir de una tabla de verdad.

Como primer paso, abrimos el logicaid y creamos un nuevo archivo como se muestra en la figura 3.2.1.10.

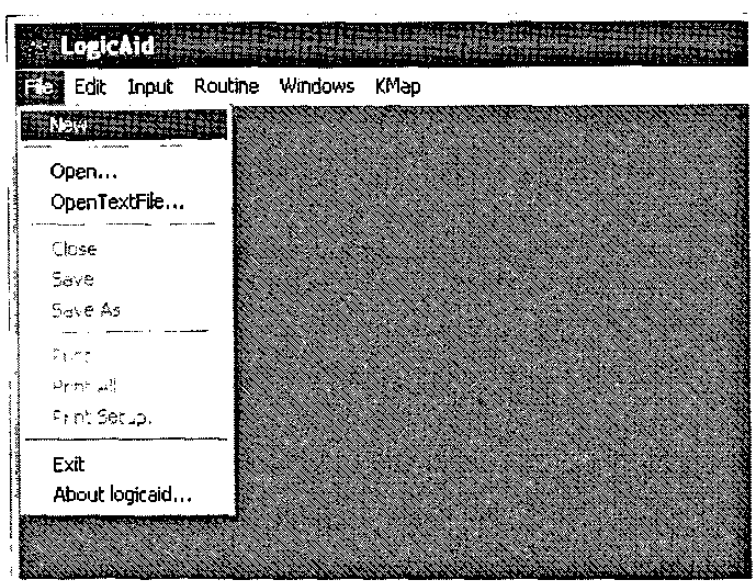


Figura 3.2.1.10.- Logicaid.

Después seleccionamos la opción numero tres para realizar la obtención de las ecuaciones a partir de la tabla de verdad, como se muestra en la figura 3.2.1.11.

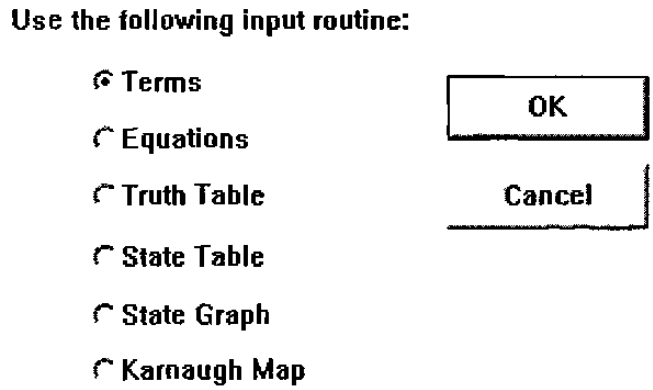


Figura 3.2.1.11.- Opción de Terms.

A continuación seleccionaremos un número de 4 variables y 2 funciones de salida en el menú que aparece en la figura 3.2.1.12.

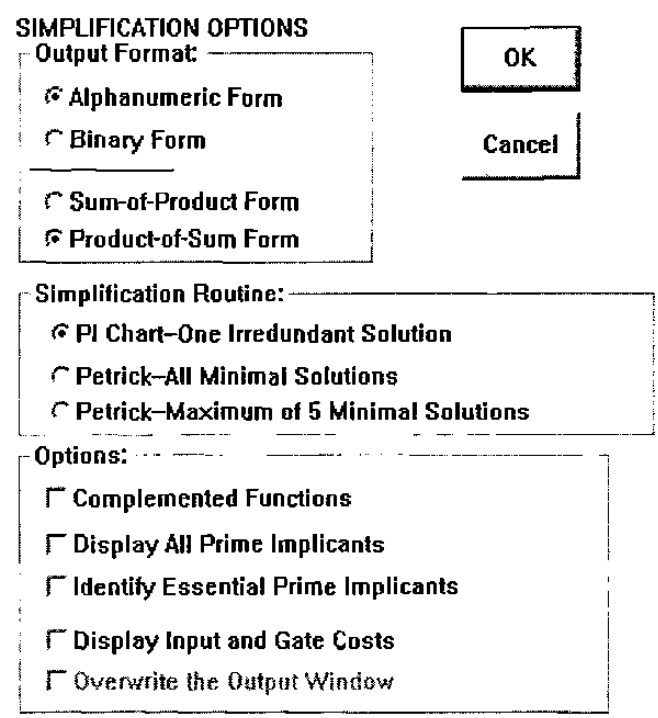


Figura 3.2.1.12.- Opciones de simplificación

Después de haber seleccionado en las variables y funciones de salida nos aparece un block de texto donde introduciremos las variables de entrada los cuales van en la segunda columna encerrada en el círculo, como se muestra en la figura 3.2.1.13.

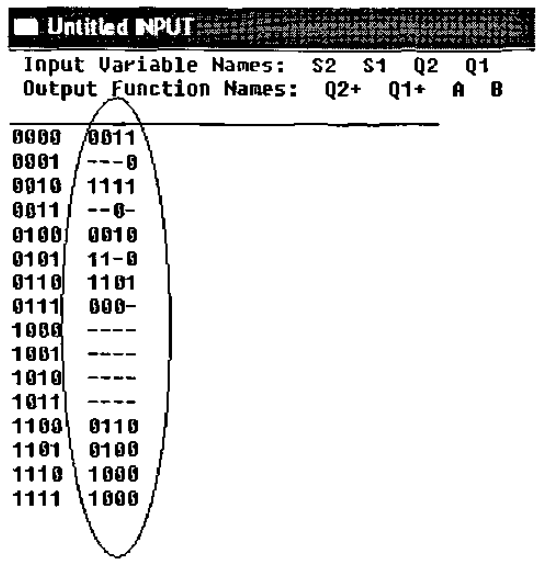


Figura 3.2.1.13.- Entrada de variables

Una vez que hemos introducido todas las diferentes combinaciones de entrada nos vamos al menú Routine y después la opción de Simplificación como se observa en la figura 3.2.1.14 y el programa nos mostrara un menú donde seleccionaremos el formato de las ecuaciones de salida y nos permitirá elegir entre suma de productos o suma de producto como se muestra en la figura 3.2.1.15.

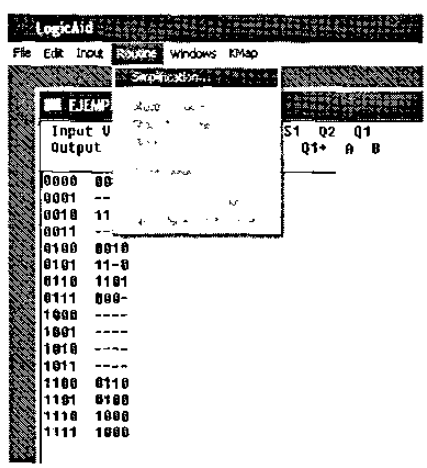


Figura 3.2.1.14.- Simplificación

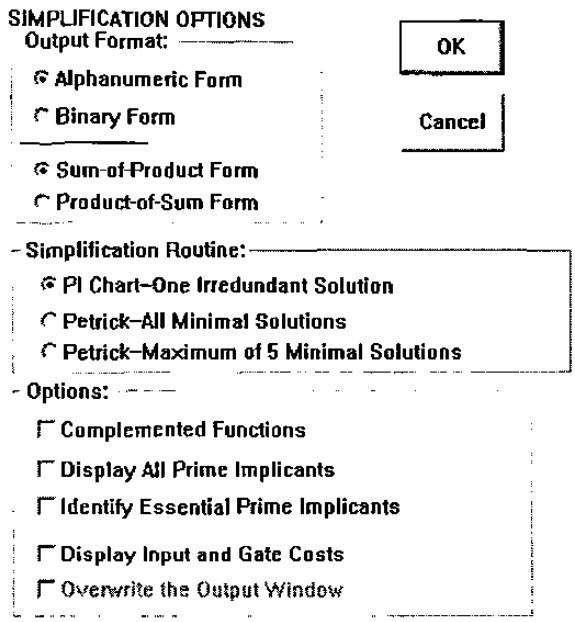


Figura 3.2.1.15.- Opciones de simplificación.

Finalmente aparece una ventana (figura 3.2.1.16) donde podemos ver el resultado de la obtención de ecuaciones.

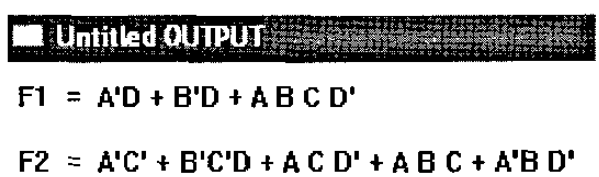


Figura 3.2.1.16.- Suma de Productos

Si seleccionamos la opción de producto de la suma como se muestra en la figura 3.2.1.17 obtenemos un resultado diferente, como lo observamos en la ventana de la figura 3.2.1.18.

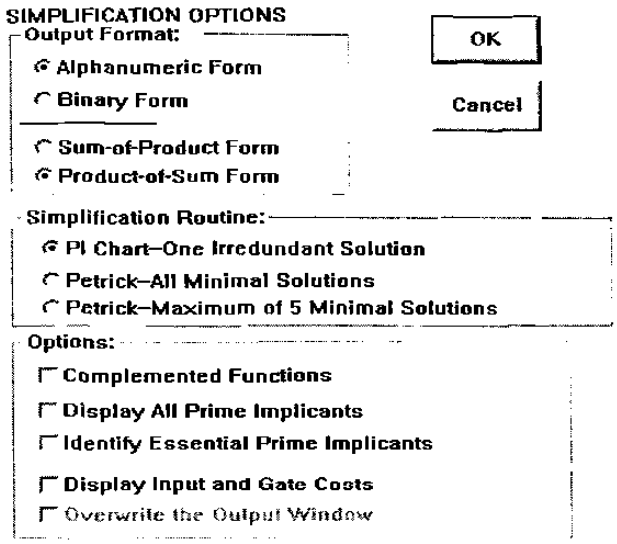


Figura 3.2.1.17.- Opciones de simplificación 2.

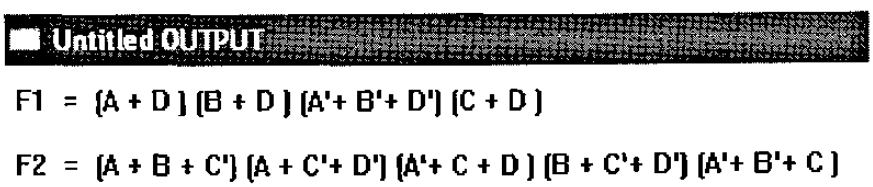


Figura 3.2.1.18.- Suma de Productos.

3.3.- Aplicaciones de Sistemas Secuenciales Asíncronos

3.3.1.- Caso 1

Diseñe un sistema secuencial asíncrono que contenga un botón de entrada **P** y una salida **Y** de modo que al oprimir por primera vez P la salida Y deberá de cambiar de un valor inicial cero a uno, al soltar el botón la salida deberá de mantenerse en uno, al oprimirlo por segunda vez P la señal de salida Y deberá de cambiar a cero y al soltarlo permanecer en ese valor como lo indica la siguiente gráfica:

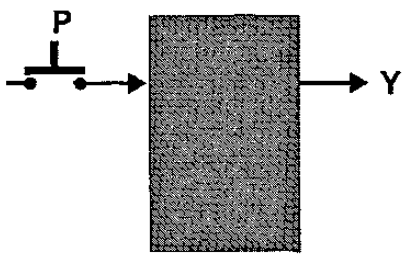


Figura 3.3.1.1-Diagrama de bloques

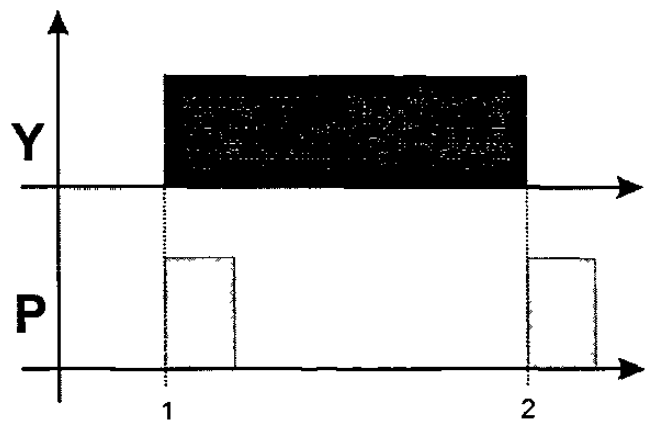


Figura 3.3.1.2.- Diagrama de tiempos

1.- Especificar el Sistema.

Por medio del diagrama de transición se puede explicar el funcionamiento del sistema indicando para cada uno de los estados las posibles entradas y los estados siguientes:

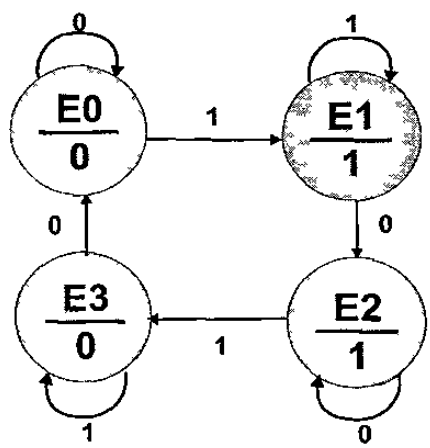


Figura 3.3.1.3.- Diagrama de Transición.

Partiendo de un estado **E0** con salida Y=0, si no se oprime el botón (P=0) permanece en el mismo estado. Al oprimir el botón (P=1) cambia al estado **E1** en donde la salida Y=1.

En ese estado permanece si el botón se mantiene oprimido (P=1) y cambia a un estado **E2** al soltar el botón (P=0) en donde la salida sigue siendo Y=1.

Una vez en **E2** se mantiene si no se oprime el botón (P=0) y cambia a un estado **E3** al oprimir el botón (P=1) en donde la salida cambia a Y=0.

Estando en **E3** se mantiene mientras el botón se encuentre oprimido (P=1), al soltar el botón (P=0) regresaremos al estado inicial **E0**

2.- Tabla de Flujo Primitiva.

Para construir la tabla de flujo primitiva recurrimos al diagrama de transición. En el estado E0 sucede que 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 (solo es permitido un estado estable por hilera) que se indica como **E2** y con salida Y=1 mostrado en la figura 3.3.3.

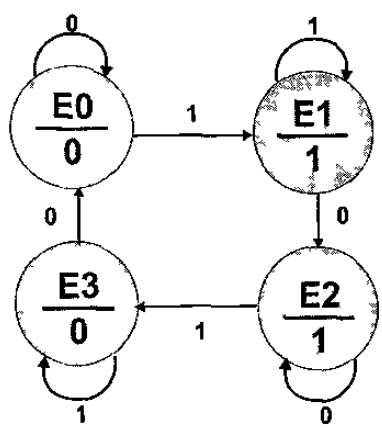


Figura 3.3.1.3.- Diagrama de Transición.

	P		Y
	0	1	
1	E0	E1	0
2		E1	1

Tabla 3.3.1.1.-Tabla de flujo primitiva.

Continuando con el análisis de los demás estados logramos obtener 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

Tabla 3.3.1.2.- Tabla de flujo primitiva

3.- Reducción de Estados.

No necesariamente los estados establecidos en la tabla de flujo primitiva son 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.

	P		Y
	0	1	
1	**E0**	E1	0
2	E2	**E1**	1
3	**E2**	E3	1
4	E0	**E3**	0

Tabla 3.3.1.2.- Tabla de flujo primitiva

En la tabla anterior observamos que para 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 no tienen la misma salida, por lo que se concluye que no hay Estados redundantes y todos son necesarios.

4.- Mezcla de Filas.

Una vez eliminados los estados redundantes, las filas o hileras pueden mezclarse para reducir la tabla. Siempre y cuando no hay ningún conflicto sobre que 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. El diagrama de mezcla consiste en asignar un punto por cada fila y se unen esos puntos por líneas cuando estos pueden mezclarse.

El procedimiento propuesto es la comparación de una fila con todas las demás para encontrar la posibilidad de mezcla:

Comparación de las filas 1 y 2

En la columna 0 tenemos 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 tenemos que E0 ≠ E2 y en la columna 1 E1 ≠ E3 por lo tanto estas filas no se pueden mezclar

	P	
	0	1
1	E0	E1
3	E2	E3

Comparación de las filas 1 y 3

En la columna 1 tenemos que E1 ≠ E3 por lo tanto 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 tenemos que E1 ≠ E3 por lo tanto estas filas no se pueden mezclar

	P	
	0	1
2	E2	E1
3	E2	E3

Comparación de las filas 2 y 4

En la columna o tenemos 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

Y por ultimo la comparación de las filas 3 y 4

En la columna 0 tenemos que E2 ≠ E0 por lo tanto estas filas no se pueden mezclar

	P	
	0	1
3	E2	E3
4	E0	E3

En nuestro problema no es posible la mezcla de filas.

5.- Expandir Tabla de Salidas

No es necesario este paso, ya que no se realizó mezcla de filas.

6.-Tabla de Estados Internos.

Para obtener la tabla de estados internos a cada fila se le identifica con un nombre y los estados estables de esa fila se les asignara el mismo nombre, por ejemplo la primera hilera la sustituimos por la letra "a", así mismo la segunda hilera por la letra b y así hasta la ultima hilera por d como se muestra en la tabla:

	P		Y
	0	1	
a	E0	E1	0
b	E2	E1	1
c	E2	E3	1
d	E0	E3	0

Tabla 3.3.1.2.-Tabla de estados internos.

Sustituyendo los nombres de los estados estables de cada fila tenemos:

E0 → a
E1 → b
E2 → c
E3 →d

Y así obtenemos 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

Tabla 3.3.1.2.1.-Tabla de estados internos.

7.- Asignación de Valores a los Estados.

Para la asignación de estados se analiza cada columna indicando las transiciones desde la fila en la cual se encuentra un estado inestable hacia la fila en la cual se vuelve estable.

	P		Y
	0	1	
a	**a**	b	0
b	c	**b**	1
c	**c**	d	1
d	a	**d**	0

Tabla 3.3.1.2.1.-Tabla de estados internos.

Columna 0) (b → c), (d → a),

1) (a → b), (c → d),

Una vez obtenidas las transiciones debemos de asegurar que el valor binario de la asignación propuesta para cada estado cambie en una sola variable.

Para este ejemplo tenemos que b tiene que ser contigua con c y además con a, de la misma forma a debe ser contigua con d y además con b, una forma de visualizar fácilmente la condición es usar la estructura de un mapa de Karnaugh, en este caso de dos variables de modo que entre cuadros contiguos solo exista un cambio de variable como se muestra a continuación:

		Q2	
		0	1
Q1	0	a	d
	1	b	c

Figura 3.3.1.3.- Asignación de valores.

Como resultado de la asignación tenemos

	Q1	Q2
a	0	0
b	0	1
c	1	1
d	1	0

Figura 3.3.1.3.1.- Asignación de valores.

Nota: Esta asignación no es la única que cumple con las condiciones de transición existen otras que también darían una solución al problema.

8.- Tabla de Estados finales

Sustituyendo los valores de la asignación obtenemos:

Tabla 3.3.1.3

	P		Y
	0	1	
a	a	b	0
b	c	b	1
c	c	d	1
d	a	d	0

Tabla 3.3.1.4

Q1 Q2	P		Y
	0	1	
00	00	01	0
01	11	01	1
11	11	10	1
10	00	10	0

Tablas 3.3.1.3 y 3.3.1.4.-Tabla de estados finales.

9.- Completar Tabla de Salidas.

No es necesario completar la tabla de salidas puesto que no requerido la expansión de ellas.

10.- Obtención de las ecuaciones por medio de minimización.

De lo anterior podemos obtener 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

Tabla 3.3.1.4.- Tabla de verdad.

De la tabla de verdad obtenemos lo siguientes mapas de karnaugh:

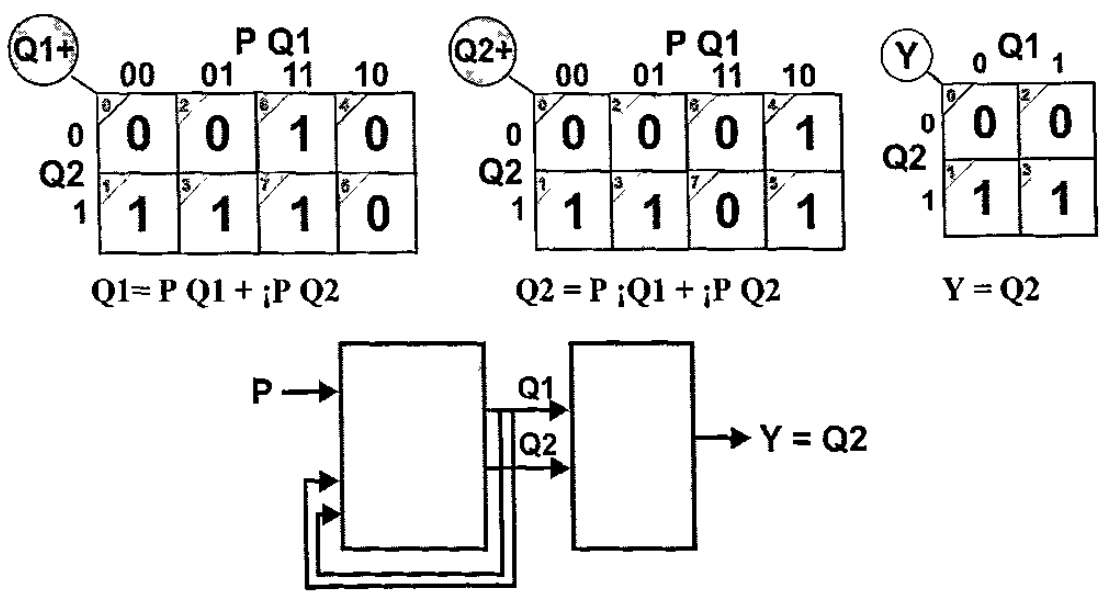


Figura 3.3.1.5.- Mapas de Karnaugh e Implementación.

Nota: La salida Y no depende de la variable P expresando el mapa en solo en función de Q1 y Q2.

11.- Simulación.

Archivo en formato ABEL-HDL utilizado para la simula

```
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
```

12.- Simulación

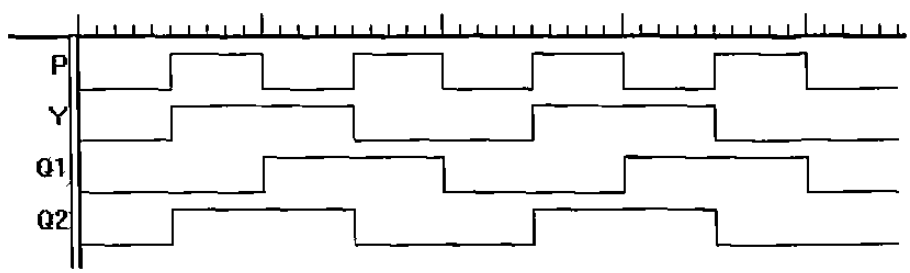


Figura 3.3.1.6.- Simulación

13.- Representación.

a) Diagrama esquemático

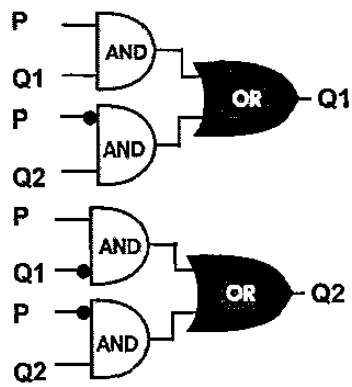


Figura 3.3.1.7.- Diagrama Esquemático

b) Diagrama Escalera

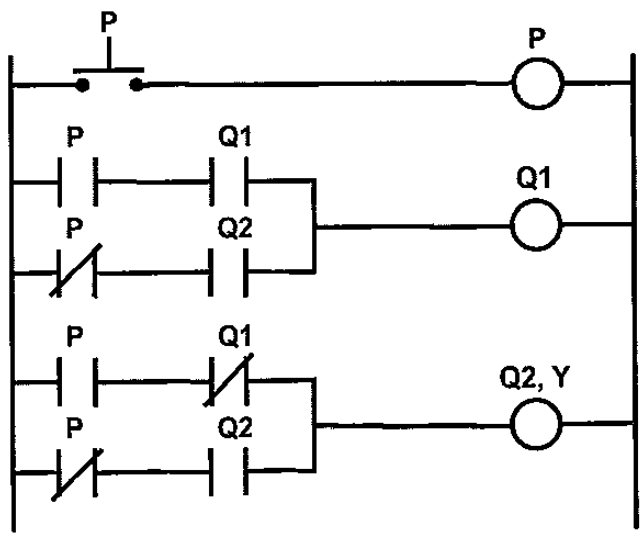
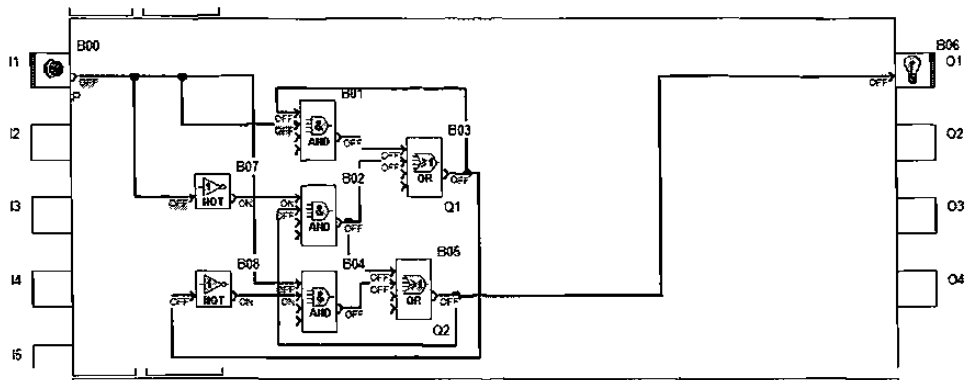


Figura 3.3.1.8.- Diagrama Escalera.

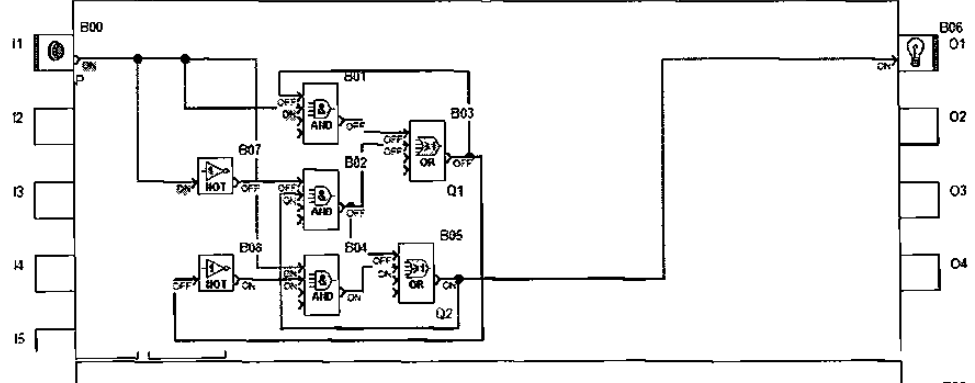
14.- Implementación.

Implementación y simulación en un PLD Marca Crouzet partiendo del diagrama esquemático.

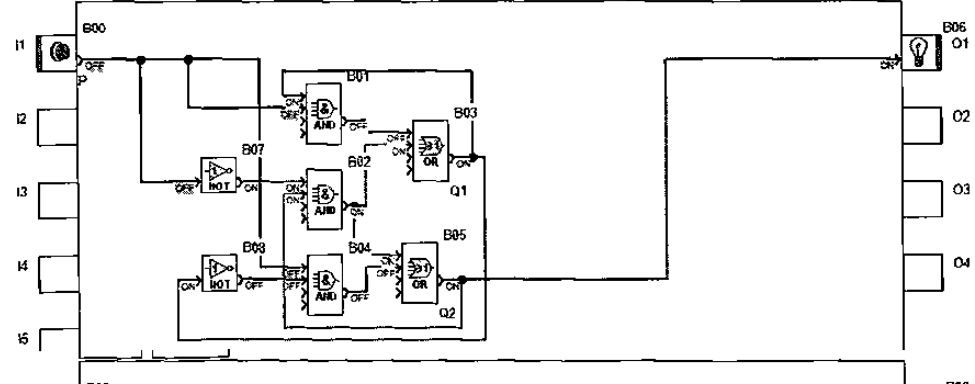
Evlento 1
P=0, Y=0



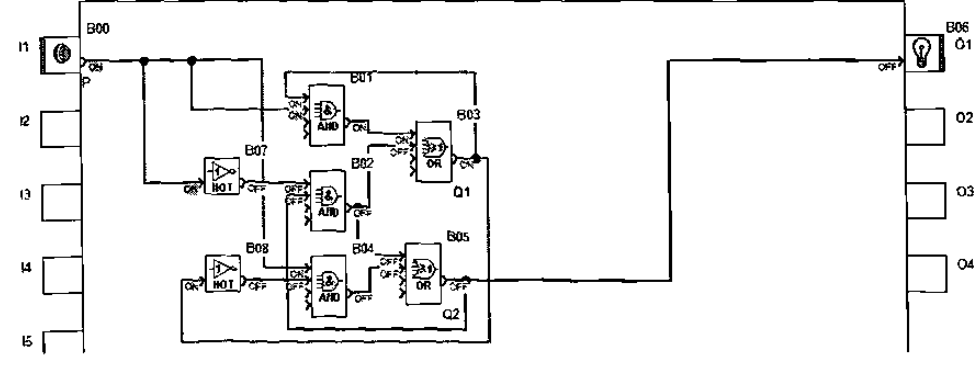
Evento 2
P=1, Y=1



Evento 3
P=0, Y=1



Evento 4
P=1, Y=0



Evento 5
P=0, Y=0
Regresa a
condiciones
iniciales

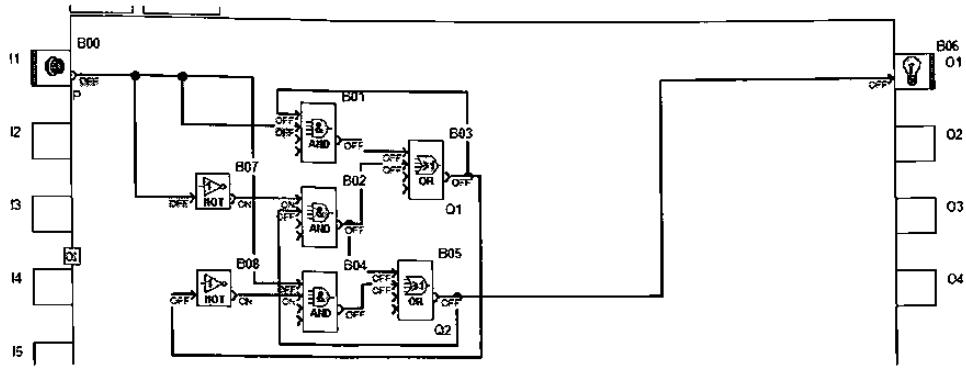


Figura 3.3.1.9.- Implementación de Eventos en un PLD.

3.3.2.- Caso 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 de cambiar de un valor inicial cero a uno, al soltar el botón la salida deberá de mantenerse en uno, al oprimirlo por segunda vez la señal de salida deberá de continuar en uno y al soltarlo cambiar a cero como lo indica la siguiente grafica:

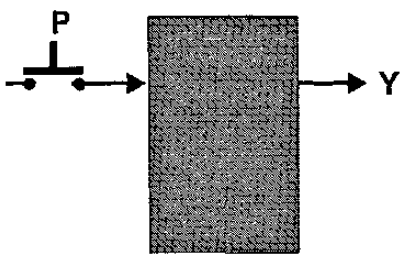


Figura 3.3.2.1.-Diagrama de bloques

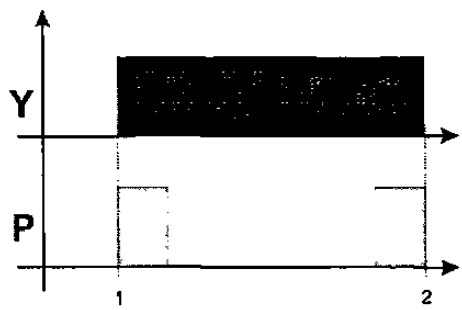


Figura 3.3.2.2.- Diagrama de tiempos

1.- Especificar el sistema

Diagrama de transición

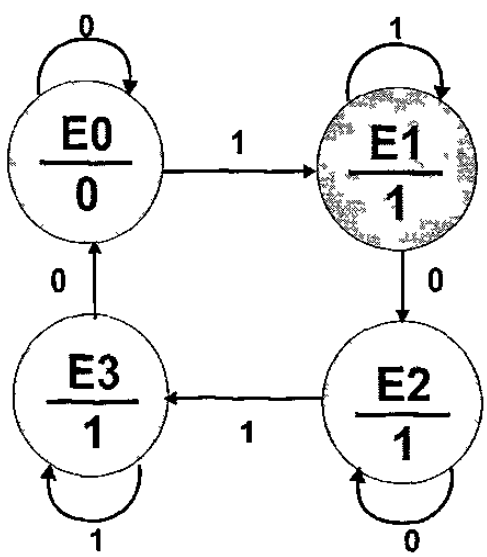


Figura 3.3.2.3.- Diagrama de Transición.

2.- Tabla de flujo primitiva

	P		Y
	0	1	
a	E0	E1	0
b	E2	E1	1
c	E2	E3	1
d	E0	E3	1

Tabla 3.3.2.1.- Tabla de flujo primitiva.

Podemos observar en la tabla mostrada que con respecto al ejemplo anterior el único cambio es la salida Y por tratarse del modelo de Moore en donde la salida depende de los valores de Q1 y Q2 que no cambian como se muestra en la siguiente figura 3.3.2.1.

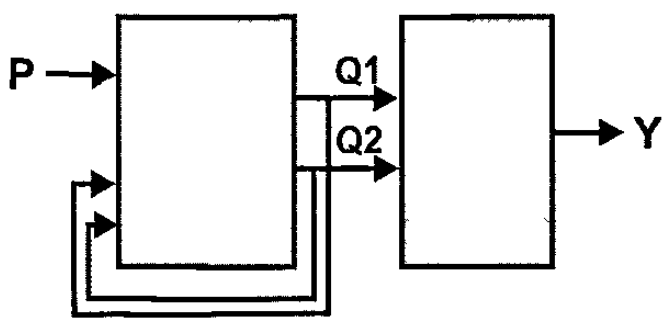


Figura 3.3.2.4

10.- Obtención de las ecuaciones por medio de minimización.

La salida Y esta en función de Q1 y Q2 de modo que de la tabla anterior tenemos el resultado que se muestra en la figura 3.3.2.2.

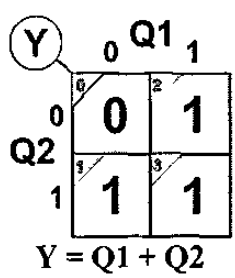


Figura 3.3.2.5.- Mapa de Karnaugh.

11.- 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
```

12.- Simulación

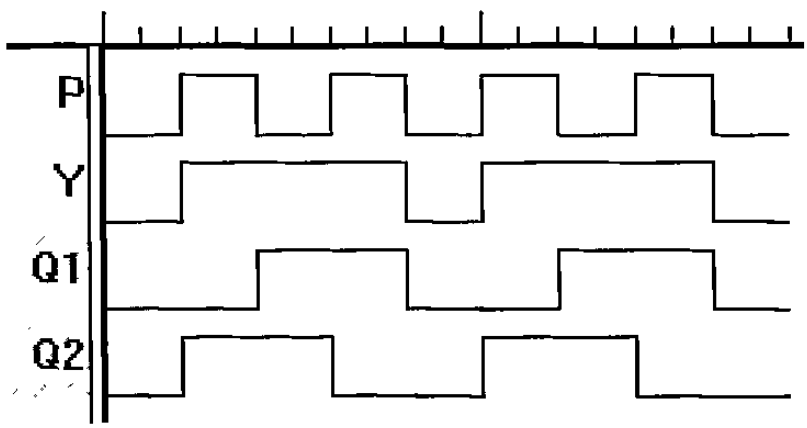


Figura 3.3.2.6.- Simulación.

13.- Representación.

Figura 3.3.2.7

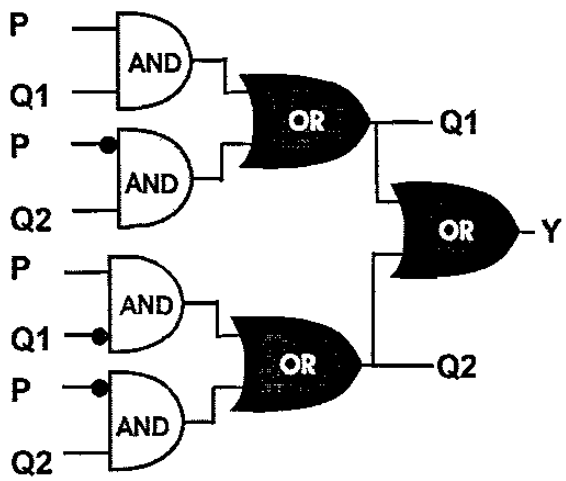


Diagrama Esquemático

Figura 3.3.2.8

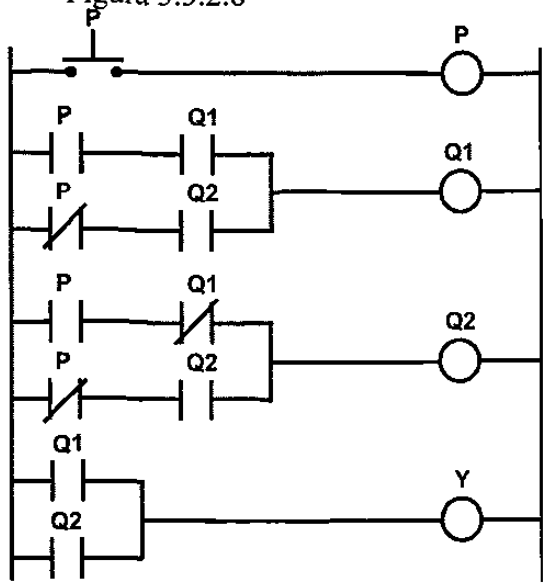
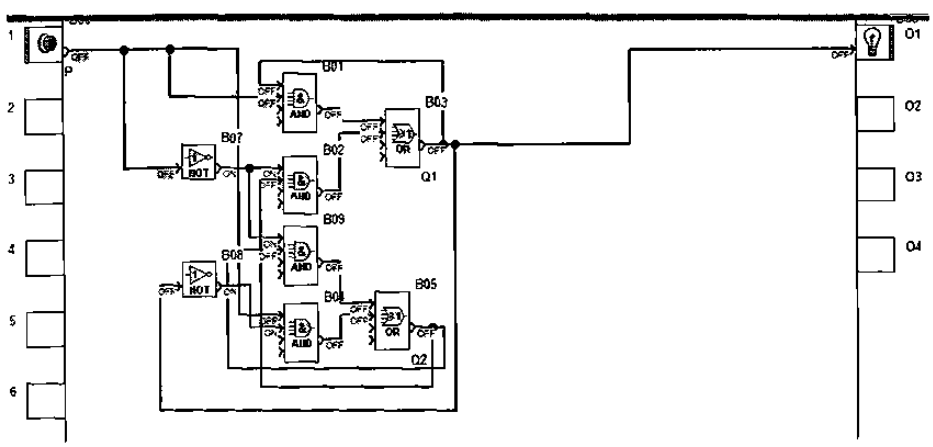


Diagrama Escalera

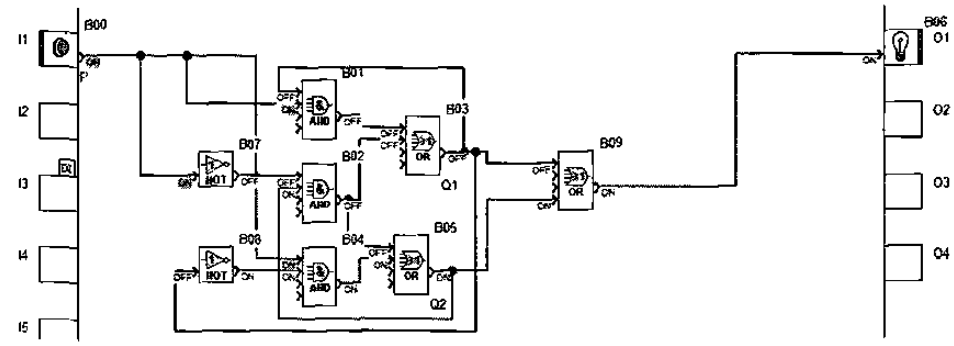
14.- Implementación.

Implementación y simulación en un PLD Marca Crouzet por medio del diagrama esquemático.

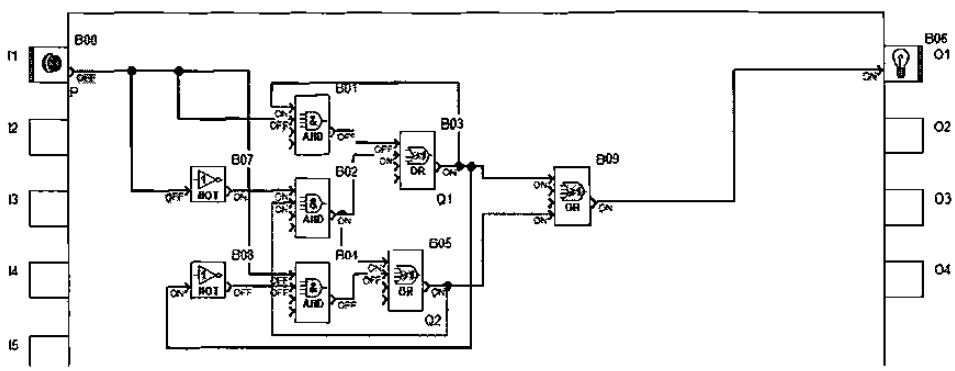
Evento 1
P=0, Y=0



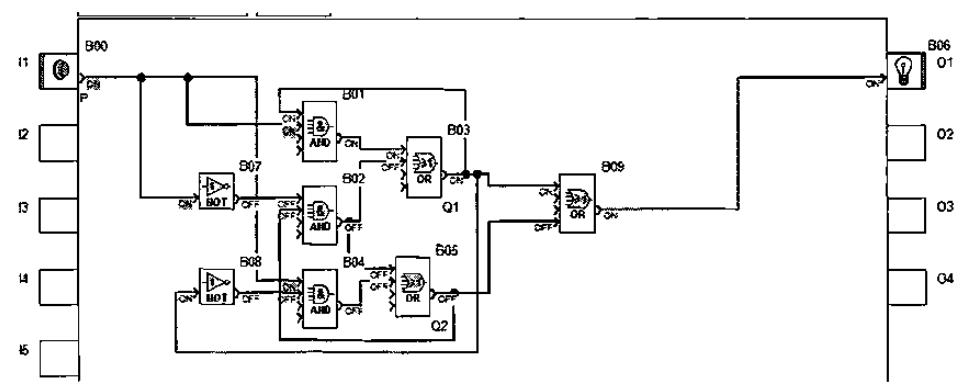
Evento 2
P=1, Y=1



Evento 3
P=0, Y=1



Evento 4
P=1, Y=1



Evento 5
P=0, Y=0
Regresa a condiciones iniciales

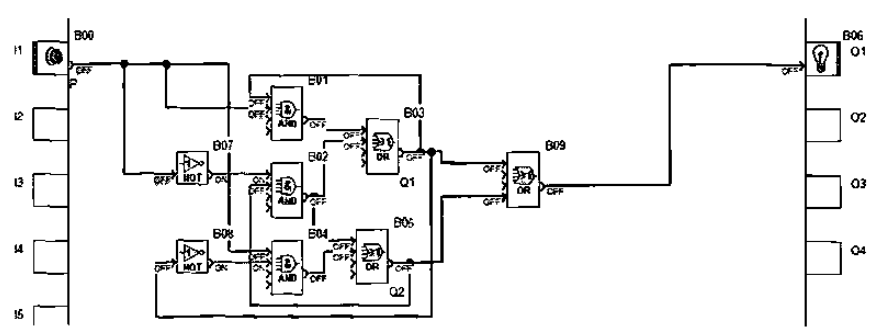


Figura 3.3.2.9.- Implementación de Eventos en un PLD.

3.3.3.- Caso 3

Diseñe un sistema secuencial asíncrono que contenga una entrada **P** de modo que al oprimir el botón por primera vez la salida Y deberá de permanecer en un valor de cero y al soltar el botón la salida deberá cambiar a uno, al oprimirlo por segunda vez la señal de salida deberá de continuar en uno y al soltarlo cambiar a cero como lo indica la siguiente grafica:

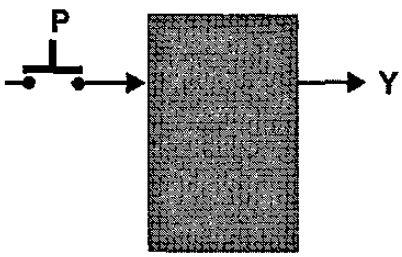


Figura 3.3.3.1.- Diagrama de Bloques.

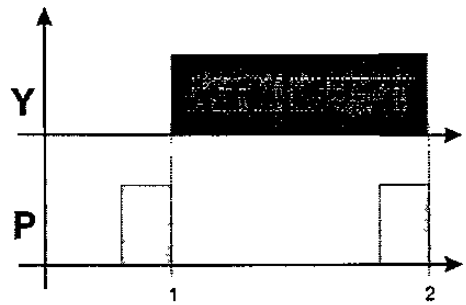


Figura 3.3.3.2.- Diagrama de Tiempos.

1.-Especificar el sistema

Diagrama de transición

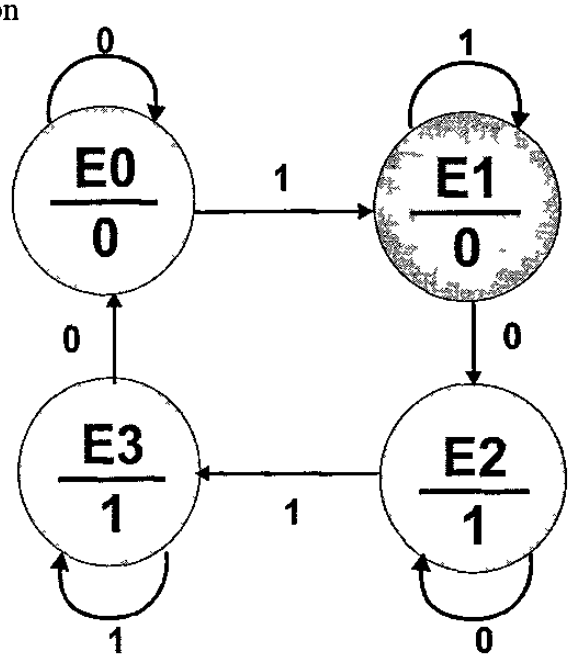


Figura 3.3.3.3.- Diagrama de Transición.

2.- Tabla de flujo primitiva

	P		Y
	0	1	
a	**E0**	E1	0
b	E2	**E1**	0
c	**E2**	E3	1
d	E0	**E3**	1

Tabla 3.3.3.1.- Tabla de Flujo Primitiva

Podemos observar en la tabla anterior que con respecto al ejemplo anterior el único cambio es la salida Y de modo que no es necesario seguir todo el procedimiento propuesto:

10.- Obtención de las ecuaciones por medio de minimización.

La salida Y esta en función de Q1 y Q2 de modo que de la tabla anterior tenemos el resultado que se muestra en la figura 3.3.3.4.

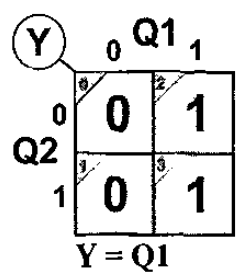


Figura 3.3.3.4.- Mapa de Karnaugh.

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;
```

```
test_vectors
(P->Y)
0->.x.;
1->.x.;
0->.x.;
1->.x.;
0->.x.;
1->.x.;
0->.x.;
1->.x.;
0->.x.;
END
```

12.- Simulación

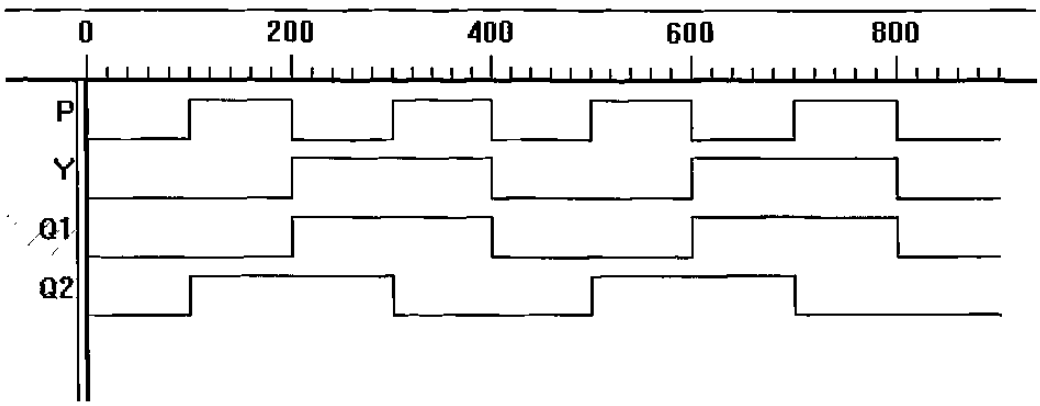


Figura 3.3.3.5.- Simulación.

13.- Representación.

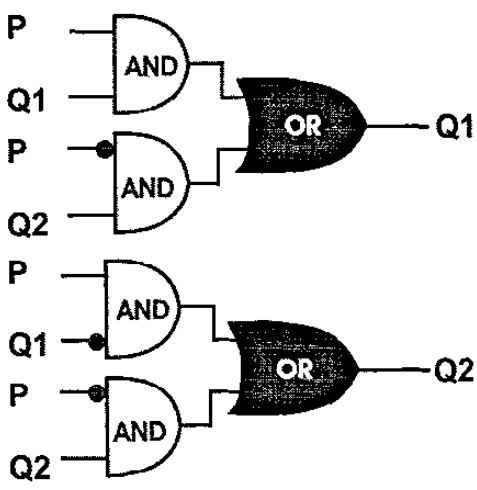


Figura 3.3.3.6.- Diagrama esquemático

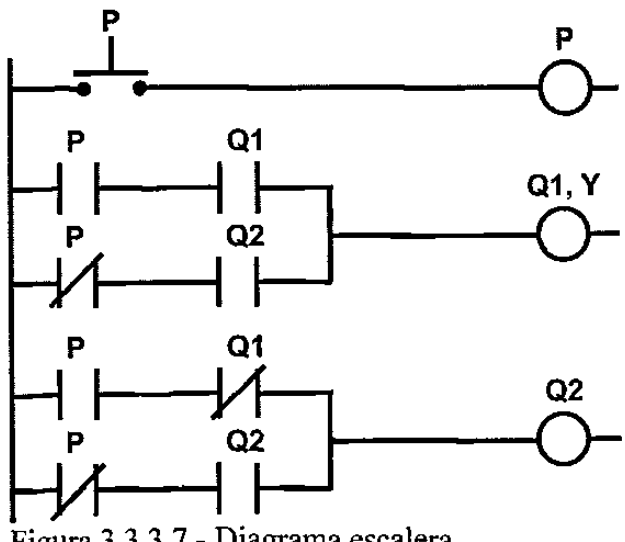
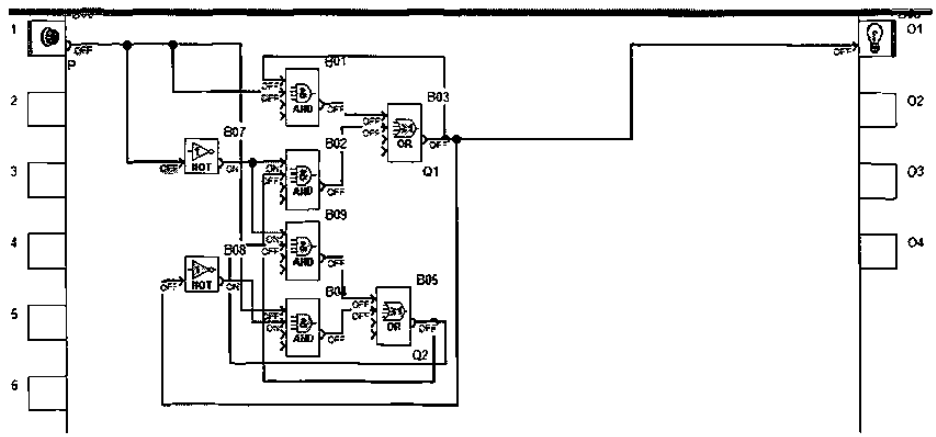


Figura 3.3.3.7.- Diagrama escalera

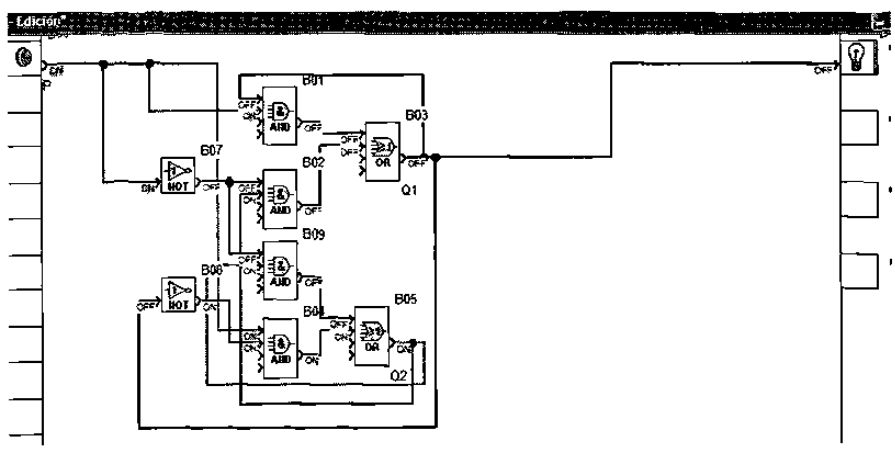
14.- Implementación.

Implementación y simulación en un PLD Marca Crouzet por medio del diagrama esquemático, se puede observar en la implementación que fue necesario no efectuar la simplificación de las AND común de P' Q2 como en los casos anteriores y agregar una AND mas.

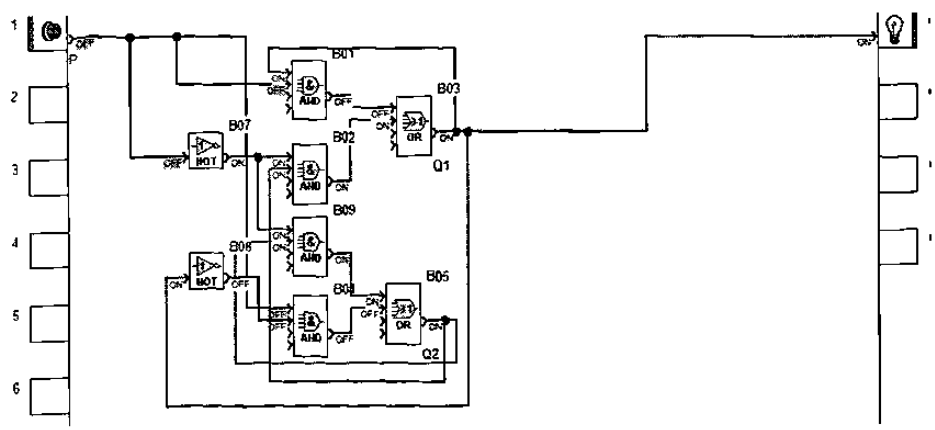
Evento 1
P=0, Y=0



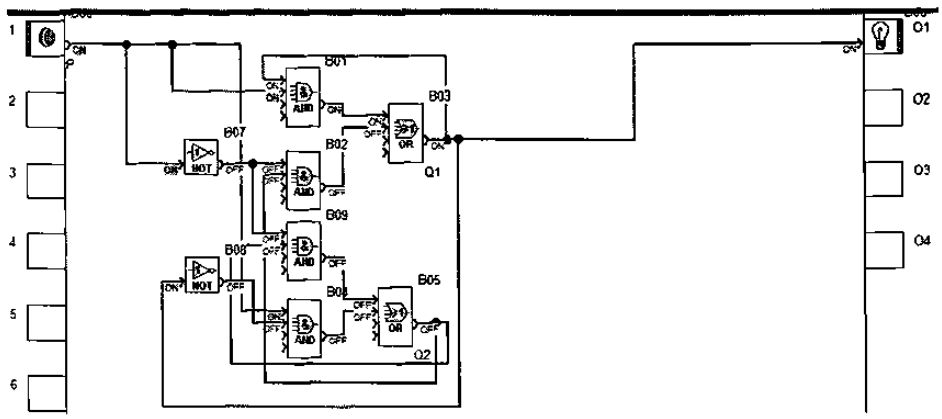
Evento 2
P=1, Y=0



Evento 3
P=0, Y=1



Evento 4
P=1, Y=1



Evento 5
P=0, Y=0
Regresa a condiciones iniciales

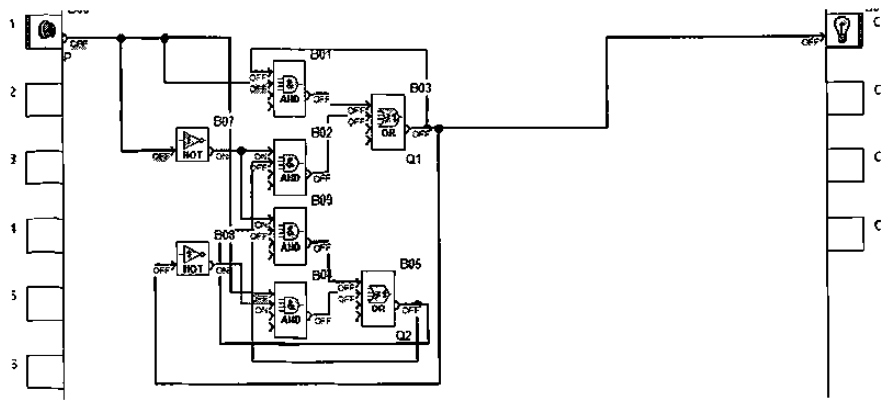


Figura 3.3.3.8.- Implementación de Eventos en un PLD.

3.3.4.- Caso 4

Diseñe un sistema secuencial asíncrono que contenga una entrada **P** de modo que al oprimir el botón por primera vez la salida Y deberá de permanecer en un valor de cero y al soltar el botón la salida deberá cambiar a uno, al oprimirlo por segunda vez la señal de salida deberá de cambiar a cero y al soltarlo permanecer con el mismo valor como lo indica la siguiente grafica:

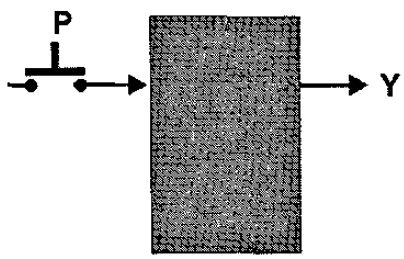


Figura 3.3.4.1.- Diagrama de bloques

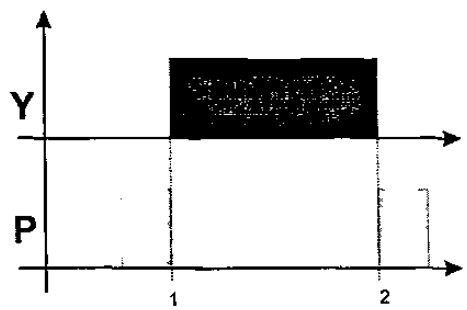


Figura 3.3.4.2.- Diagrama de tiempos

1.- Especificar el Sistema

Diagrama de transición

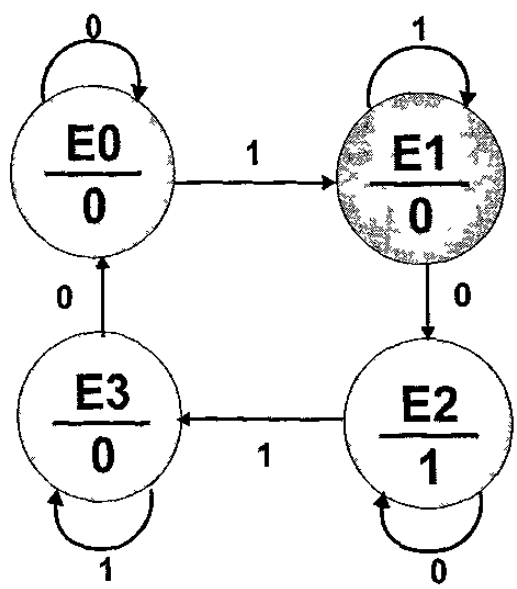


Figura 3.3.4.3.- Diagrama de Transición.

2.- Tabla de flujo primitiva

	P		Y
	0	1	
a	E0	E1	0
b	E2	E1	0
c	E2	E3	0
d	E0	E3	1

Tabla 3.3.4.1.- Tabla de Flujo Primitiva

Podemos observar en la tabla anterior que con respecto al ejemplo anterior el único cambio es la salida Y de modo que no es necesario seguir todo el procedimiento propuesto:

10.- Obtención de las ecuaciones por medio de minimización.

La salida Y esta en función de Q1 y Q2 de modo que de la tabla anterior tenemos el resultado que se muestra en la figura 3.3.4.4.

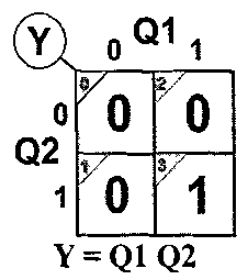


Figura 3.3.4.4.- Mapa de Karnaugh

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
```

12.- Simulación

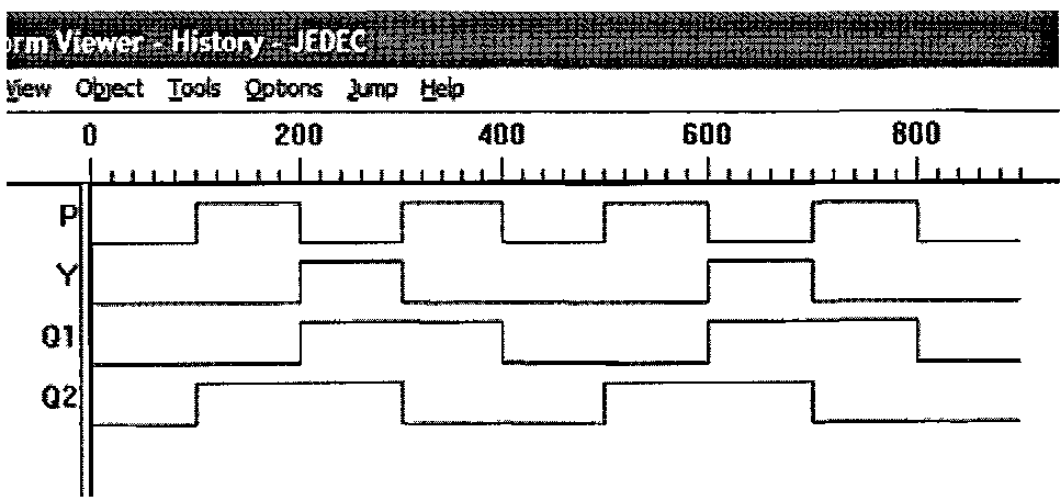


Figura 3.3.4.5.- Simulación.

13.- Representación Grafica

Figura 3.3.4.6

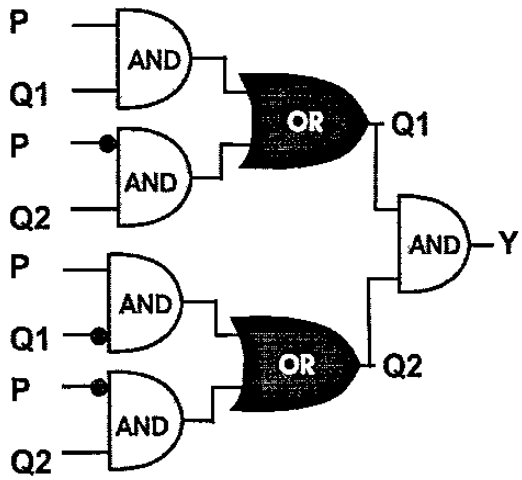


Diagrama Esquemático

Figura 3.3.4.7

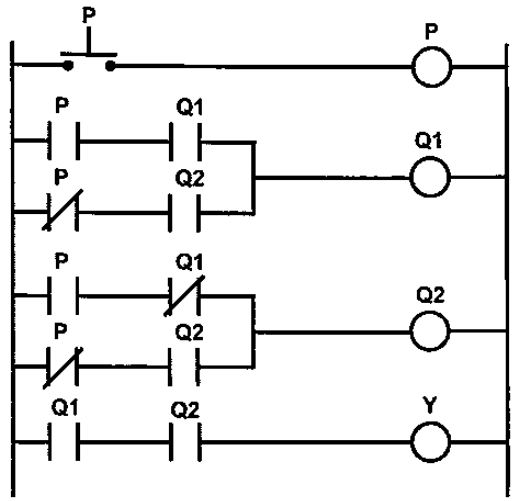


Diagrama Escalera

3.3.5.- Arranque Paro

Diseñar un sistema secuencial asíncrono para los eventos arranque paro en un sistema. Este sistema deberá de contar con dos botones uno de arranque normalmente abierto NO y otro para paro normalmente cerrado (NC) respectivamente.

El sistema opera de forma muy sencilla, en un inicio el sistema esta detenido, por lo que para activarlo es necesario presionar el botón de encendido, y para detenerlo simplemente se utiliza el botón de paro con lo que el sistema se detiene.

1.- Especificación del sistema

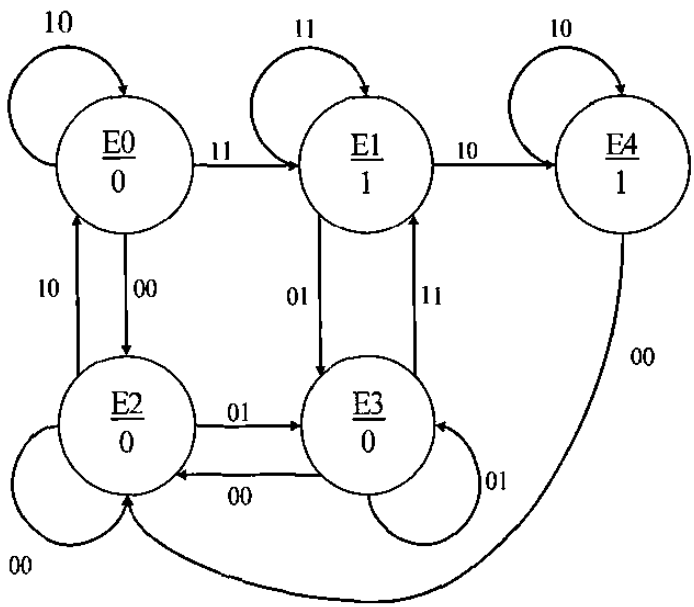


Figura 3.3.5.1.- Diagrama de Transición.

En un inicio el botón de paro NC esta activo y el de arranque NO (P=1 y A=0) que es el estado E0. En el siguiente estado E1 el botón de arranque es presionado y el de paro sigue presionado (A=1 y P=1) por lo que es sistema enciende, si después se activa el botón de paro (P=0) se pasa a un estado intermedio que es E3 en el cual el sistema se apagaría.

En el siguiente estado E4 cuando el botón de arranque se suelta (A=0 y P=1), por el diseño del sistema, este se queda encendido, por lo que para ser apagado necesita de que el botón de paro sea oprimido que sucede en el estado E2 cuando P y A son iguales a 0. A partir del estado E2 el estado siguiente posible es E0 cuando el sistema esta listo para encender (P= y A=0), en el caso de que el botón de paro siga abierto (P=0) el sistema permanecerá en este estado

- No se puede parar de condición 11 a 00 o de 01→ 10, solamente se permite un cambio a la vez.

2.- Tabla de Flujo Primitiva

	00	01	10	11	S
0	E2	X	E0	E1	0
1	X	E3	E4	E1	1
2	E2	E3	E0	X	0
3	E2	E3	X	E1	0
4	E2	X	E4	E1	1

Tabla 3.3.5.1.- Tabla de Flujo Primitiva

3.- Reducción de Estados.

Estables en la misma columna.

00) E2

01) E3

10) E0, E4

11) E1

2.- Que tengan la misma salida

E0 = 0

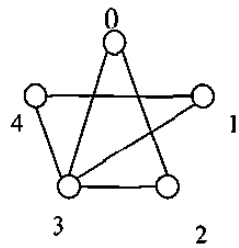
E4 = 1 E0 ≠ E4

3.- Que es el estado siguiente sea equivalente.

No son equivalentes.

No hay reducción de estados.

4.- Mezcla de Filas



(Sin tomar en cuenta las salidas)

0 con 1 → No 0 con 2 → Si 0 con 3 → Si 0 con 4 → No	1 con 2 → No 1 con 3 → Si 1 con 4 → Si	2 con 3 → Si 2 con 4 → No	3 con 4 → Si

Se mezcla la 0,2 y 3 por tener la misma salida.

	00	01	10	11	SS
0,2,3	E2	E3	E0	E1	0
1,4	E2	E3	E4	E1	1

6.- Tabla de Estados Internos

Substituyendo

a → E0,E3,E2 por los estados transitorios de la fila 0

b → E1,E4 por los estados transitorios de la fila 1

	00	01	10	11	S
a	a	a	a	b	0
b	a	a	b	b	1

Tabla 3.3.5.2.-Tabla de estados internos.

Transiciones

00) a → b

01) a → b

10) No hay

11) b →

7.- Asignación de Valores a los Estados.

Se sustituye a = 0 y b = 1 y S = Q.

Q	00	01	10	11	S
0	0	0	0	1	01
1	0	0	1	1	1

Tabla 3.3.5.3.-Tabla de asignación de valores

La salida no se considera para la tabla de verdad.

Acomodo de la Tabla de Verdad

	Q	P	A	Q+
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	0
6	1	1	0	1
7	1	1	1	1

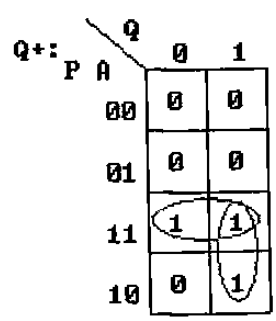


Figura 3.3.5.2.- Tabla de verdad

10.- Obtención de las Ecuaciones por medio le la minimización

$Q = QP+PA$

$Q = P\,(Q+A)$

13.- Diagrama Esquemático

a) Diagrama Escalera

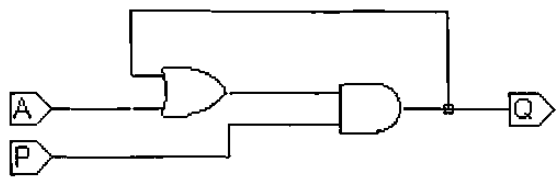


Figura 3.3.5.3.- Diagrama Esquemático.

b) Diagrama Escalera

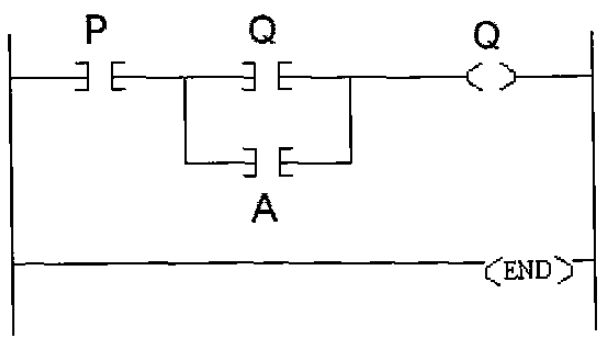


Figura 3.3.5.4.- Diagrama Escalera

3.3.6.- Detector de 3 Niveles

Diseñe un sistema secuencial asíncrono para la detección del nivel de un tanque que por medio de tres sensores llamados B bajo, M medio, A alto se obtengan las salidas S1 y S2 con los valores presentados en la siguiente grafica:

1.- Especificar el sistema

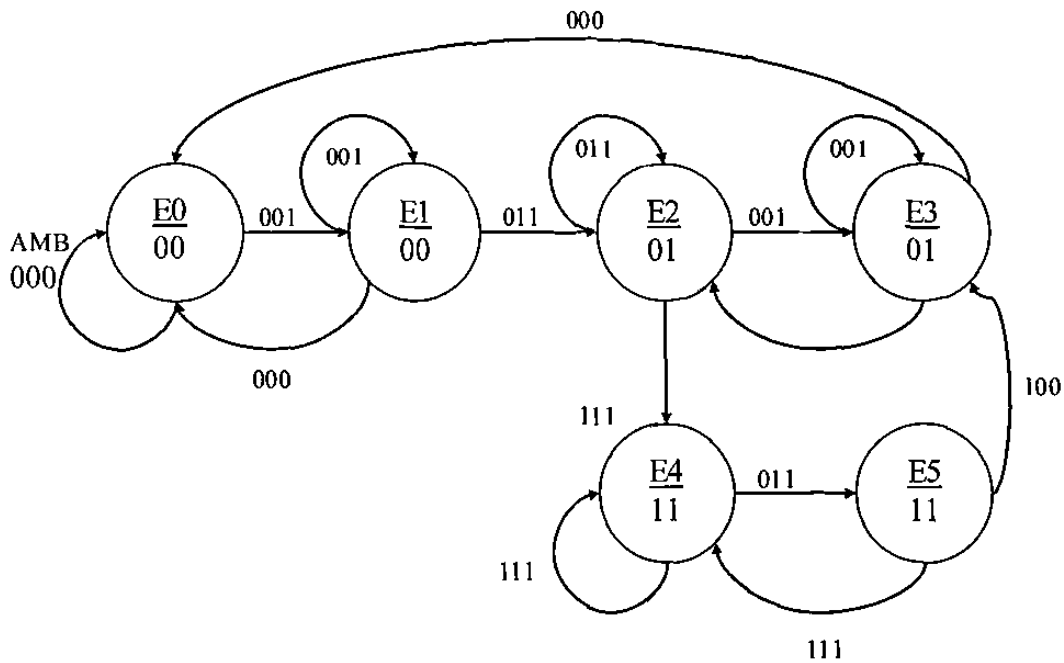


Figura 3.3.6.1.- Diagrama de Transición.

En el primer estado E0 todos los sensores están en 0 por lo que se asume que el tanque esta vació. Cuando el sensor de nivel bajo se activa, el sistema pasa al siguiente estado E1 que indica el nivel bajo, de este estado pueden seguir 2 opciones antes de avanzar al siguiente estado E2, una es que el nivel bajo vuelva a ser 0 por lo que regresaría al estado E0 y otra que es seguir en el mismo estado. En todos estos casos las salidas S1 y s2 son iguales a 0.

Partiendo de que el sensor de nivel bajo sigue activo (001) la siguiente opción cuando el sensor de nivel medio se activa (011 y S1=0, S2=1) el sistema pasa al estado E2, en este estado si el sensor de nivel medio vuelve a 0, no se regresa al estado anterior si no que avanza al estado E3 (001 y S1=0,S2=1) esto se hace para evitar un efecto de oleaje

ya que es un sistema de detección de nivel donde estos casos pueden ocurrir. En caso de que el sensor de nivel alto se active, se avanza al siguiente estado de nivel lleno (111 y S1=1, S2=1).

En el estado E3 nivel bajo puede ocurrir que el sensor de nivel bajo valga 0 (000 y S2=0, S1=0) por lo que el sistema regresaría al estado inicial E0 o puede ocurrir que se active el sensor de nivel medio por lo que regresaría al estado E2 (011 y S2=1, S2= 0).

Finalmente partiendo del estado E2 (011 y S1=0, S2=1), al activarse el sensor de nivel alto (111) el sistema pasa al ultimo estado E4 (S2=1 y S1=1) donde ambas salidas se activan, después de este estado solo puede ocurrir dos casos, uno es permanecer en el mismo estado y otro donde llegara a nivel medio que seria al estado E5 donde ambas salidas siguen activas y solo en caso de que se censara un nivel bajo cambiaria el estado de las salidas al regresar el sistema al estado E3 (001,S2=1S1=0).

2.- Tabla de Flujo Primitiva

	000	001	010	011	100	101	110	111	S2	S1
0	E0	E1	X	--	X	X	X	--	0	0
1	E0	E1	X	E2	X	X	X	--	0	0
2	--	E3	X	E2	X	X	X	E4	0	1
3	E0	E3	X	E2	X	X	X	--	0	1
4	--	--	X	E5	X	X	X	E4	1	1
5	--	E3	X	E5	X	X	X	E4	1	1

Tabla 3.3.6.1.-Tabla de flujo primitiva.

3.- Reducción de Estados

000) E0

001) E1, E3

011) E2, E5

111) E4

No hay reducción de estados debido a que las salidas de los estados son diferentes.

4.- Mezcla de filas

Se escoge porque las salidas son comunes, mezclamos 0 y1, 2 y 3, 5, 4.

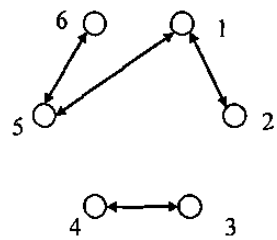


Figura 3.3.6.3.- Mezcla de filas.

	000	001	010	011	100	101	110	111	S2	S1
0	E0	E1	X	E2	X	X	X	X	0	0
1	E0	E3	X	E2	X	X	X	E4	0	1
2	X	E2	X	E5	X	X	X	E4	1	1

Tabla 3.3.6.1.2-Tabla de flujo primitiva.

6.- Tabla de Estados Internos

a = E0, E1

b = E3, E2

c = E5, E4

Sustituyendo los estados estables y transitorios por a, b o c.

	000	001	010	011	100	101	110	111	S2	S1
a	a	a	x	b	x	x	x	x	0	0
b	a	b	x	b	x	x	x	c	0	1
c	x	b	x	c	x	x	x	c	1	1

Tabla 3.3.6.2-Tabla de estados internos.

Transiciones

000) a → b

001) b → c

011) b → a

111) c → b

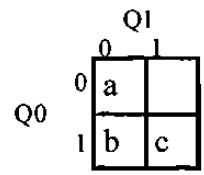


Figura 3.3.6.4.- Transiciones

7.- Asignación de Valores a los Estados.

	Q1	Q0
a	0	0
b	0	2
c	1	1

Q1 Q0	000	001	010	011	100	101	110	111	**S2**	**S1**
00	00	01	X	01	X	X	X	XX	**0**	**0**
01	00	01	X	01	X	X	X	11	**0**	**1**
10	X	X	X	X	X	X	X	X	**X**	**X**
11	X	01	X	11	X	X	X	11	**1**	**1**

Tabla 3.3.6.3-Tabla de asignación de valores a los estados.

8.- Tabla de Estados Totales

Input Variable Names: Q1 Q0 A M B Output Function Names: Q1+ Q0+	
00000	00
00001	00
00010	--
00011	01
00100	--
00101	--
00110	--
00111	--
01000	00
01001	01
01010	--
01011	01
01100	--
01101	--
01110	--
01111	11
10000	--
10001	--
10010	--
10011	--
10100	--
10101	--
10110	--
10111	--
11000	--
11001	01
11010	--
11011	11
11100	--
11101	--
11110	--
11111	11

Tabla 3.3.6.4-Tabla de estados totales.

10.- Obtención de las ecuaciones por medio de minimización.

Q1+ = A + Q1M

Q0+ = M + Q0B

11.- Elaboración del archivo en formato ABEL – HDL.

```
MODULE detecniv

"Entradas
A,M,B PIN 1,2,3;

"SALIDAS
S2,S1 PIN 19,18 ISTYPE'COM'.

EQUATIONS
S2=A#S2&M;
S1=M#S1&B;

TEST_VECTORS
([A,M,B]->[S2,S1])
[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.];
[0,0,1]->[.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.];
[0,1,1]->[.X.,.X.];
[0,0,1]->[.X.,.X.];
[0,0,0]->[.X.,.X.];

END
```

12.- Simulación en IspStarter de Lattice semiconductor.

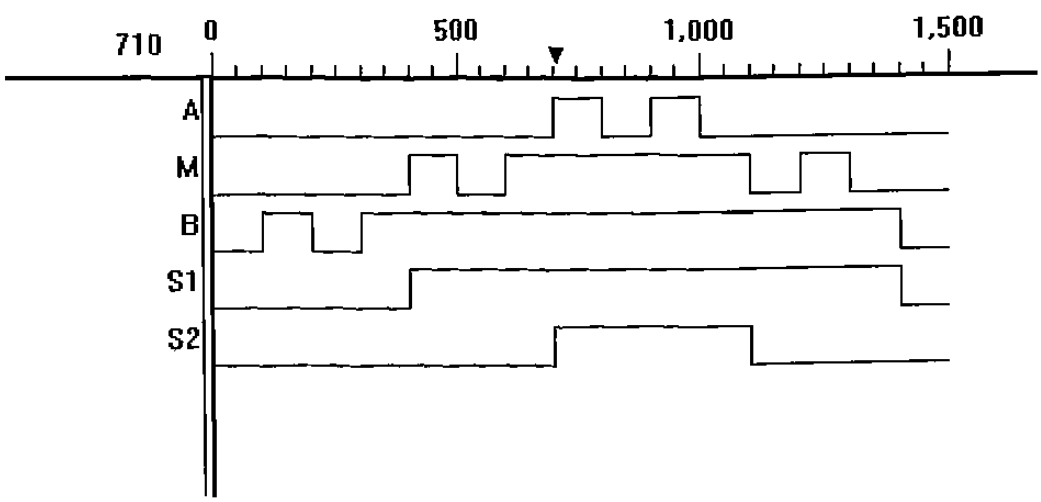


Figura 3.3.6.5.- Simulación.

13.- Diagramas.

a) Diagrama Esquemático

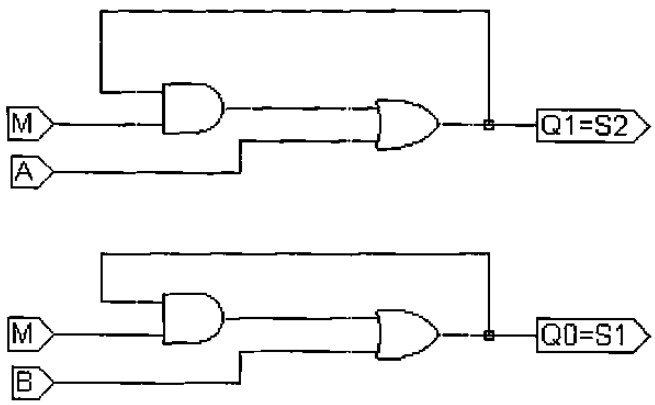


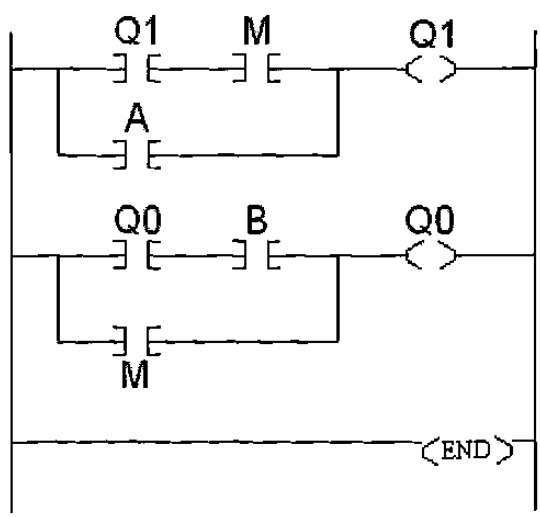
Figura 3.3.6.6.- Diagrama Esquemático.

b) Diagrama Escalera

Q0+ = M + Q0B

Q1+ = A + Q1M

Figura 3.3.6.7.- Diagrama Escalera



3.3.7.- Dos Bombas Simultaneas Caso 1

El tanque de la figura se alimenta por medio de 2 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 3 sensores de entrada nivel A (Alto), M (Medio) y B (Bajo), y dos salidas S2 y 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 y S1.

1.- Partiendo de que el tanque se encuentra vacío (S2=0 Y S1=0), se inicia el llenado funcionando ambas bombas A y b, hasta llegar al nivel medio (S2=0 y S1=1) y de ahí solo trabajar la bomba A, si se vacía de nuevo (S2=0 y S1=0) encenderán de nuevo ambas bombas y al llegar al nivel medio (S2=0 Y S1=1) trabajara solo la bomba B.

2.- Cada vez que se vacié el tanque (S2=0 y S1=0), y de ahí pase al nivel medio (S2=1 Y S1= 0), deberá de alternarse el funcionamiento de la bomba 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 de ahí pase al nivel medio (S2=1 y S1 = 0), deberá de alternarse el funcionamiento de la bomba A y B.

1.- Especificar el sistema

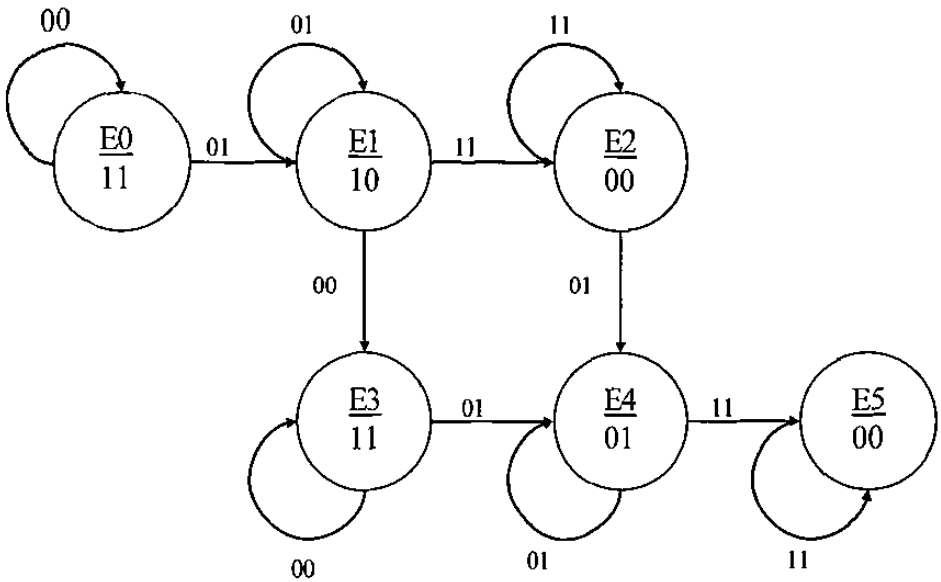


Figura 3.3.7.1.- Diagrama de Transición.

Iniciando con el estado E0 cuando el tanque esta vacío las salidas están en 0 (S_2=0 y S_1=0), y las bombas están encendidas (BA=1 y BB=1); si el tanque sigue vacío se mantiene en ese mismo estado, y si se empieza a llenar hasta llegar a un estado medio, por lo que pasa al siguiente estado E1. En este estado (S_2=0 y S_1=1) se mantiene solamente la bomba A encendidas hasta que se termine de llenar el tanque, por lo que el sistema avanza al siguiente estado E2 (S_2=1 y S_1=1). En caso de que se vacíe el tanque pasa al estado E3 donde nuevamente se activan ambas bombas hasta llegar al nivel medio.

Partiendo de el estado E2 nivel alto, donde ambas bombas están apagadas, si el tanque disminuyera a nivel medio trabajaría la bomba B por lo que el sistema pasaría al siguiente estado E4, en este estado al terminar de llenarse el tanque el siguiente estado seria E5 en el cual al terminar de llenarse el tanque se apagaría la bomba B (S_2=1 y S_1=1).

2.- Tabla de Flujo Primitiva

	00	01	10	11	A	B
0	**E0**	E1	X	--	1	1
1	E3	**E1**	X	E2	1	0
2	--	E4	X	**E2**	0	0
3	E0	**E4**	X	E5	0	1
4	--	E1	X	**E5**	0	0
5	**E3**	E4	X	X	1	1

Tabla 3.3.7.1.-Tabla de flujo primitiva.

4.- Mezcla de Filas

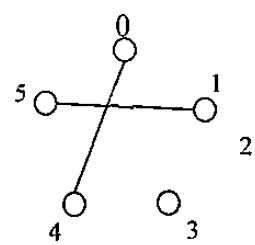


Figura 3.3.7.2.- Mezcla de Filas.

5.- Expandir Salidas

Método de Mealy para expandir salidas.

	00	01	10	11	00	01	10	11
0,4	E0	E1	X	E5	11	10	XX	00
2,5	E3	E4	X	E2	11	01	XX	00
1	E3	E1	X	E2	1X	10	XX	X0
3	E0	E4	X	E5	X1	01	XX	0X

Tabla 3.3.7.2.-Expansión de salidas.

6.- Tabla de Estados Internos

a = E0, E5

b = E3, E2

c = E1

d = E4

	00	01	10	11
a	a	c	X	a
b	b	d	X	b
c	b	c	X	b
d	d	d	X	a

Tabla 3.3.7.3.-Tabla de estados internos..

00) b →c, a→ d

01)a →c, b→ d

11) c →b, a→ d

7.- Asignación de Valores a los Estados.

Q1 Q2

a = 0 0
b = 1 1
c = 1 0
d = 0 1

Q1 \ Q2	0	1
0	a	c
1	d	b

Q2 Q1	00	01	10	11	00	01	10	11
00	00	10	XX	00	11	10	XX	00
01	00	01	XX	00	X1	01	XX	0X
10	11	10	XX	11	1X	10	XX	X0
11	11	01	XX	11	11	01	XX	00

Tabla 3.3.7.4.-Tabla de asignación de valores a los estados.

8.- Tabla de Estados Totales

```
 Input Variable Names:  S1  S2  Q2  Q1
 Output Function Names:  Q2+  Q1+  A  B
```

0000	0011
0001	00-1
0010	111-
0011	1111
0100	1010
0101	0101
0110	1010
0111	0101
1000	----
1001	----
1010	----
1011	----
1100	0000
1101	000-
1110	11-0
1111	1100

Tabla 3.3.7.5.-Tabla de estados totales.

10.- Obtención de las ecuaciones por medio de minimización.

Q2+ = S2'Q2 + S1'S2 Q1' + S1 Q2

Q1+ = S2'Q2 + S1'S2 Q1 + S1 Q2

A = S2' + S1'Q1'

B = S2' + S1'Q1

11.- Elaboración del archivo en formato ABEL – HDL

```
MODULE cuno

"ENTRADA
S1,S2 PIN 1,2;

"SALIDAS
Q2,Q1,A,B PIN 19,18,17,16 ISTYPE'COM';

EQUATIONS
Q2=!S2&Q2#!S1&S2&!Q1#S1&Q2;
Q1=!S2&Q2#!S1&S2&Q1#S1&Q2;
A=!S2#!S1&!Q1;
B=!S2#!S1&Q1;

TEST_VECTORS
([S1,S2]->[A,B,Q2,Q1])
[0,0]->[.X.,.X.,.X.,.X.];
[0,1]->[.X.,.X.,.X.,.X.];
[1,1]->[.X.,.X.,.X.,.X.];
[0,1]->[.X.,.X.,.X.,.X.];
[1,1]->[.X.,.X.,.X.,.X.];
[0,1]->[.X.,.X.,.X.,.X.];
[0,0]->[.X.,.X.,.X.,.X.];
[0,1]->[.X.,.X.,.X.,.X.];
[0,0]->[.X.,.X.,.X.,.X.];
[0,1]->[.X.,.X.,.X.,.X.];
[1,1]->[.X.,.X.,.X.,.X.];

END
```

10.- Simulación en IspStarter de Lattice semiconductor.

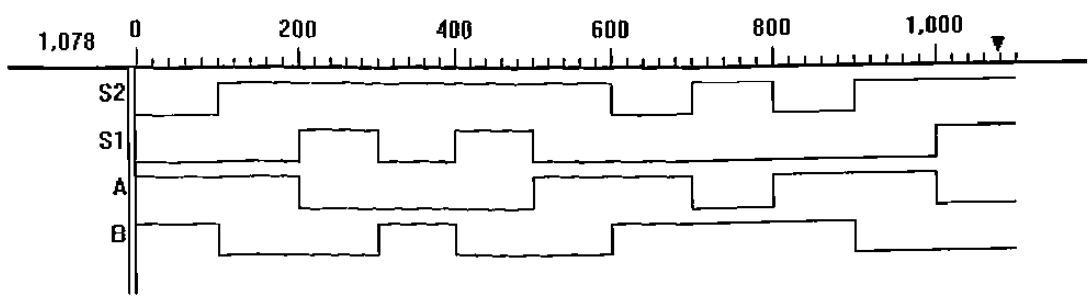


Figura 3.3.7.3.- Simulación.

13.- Diagramas

a) Diagrama Esquemático

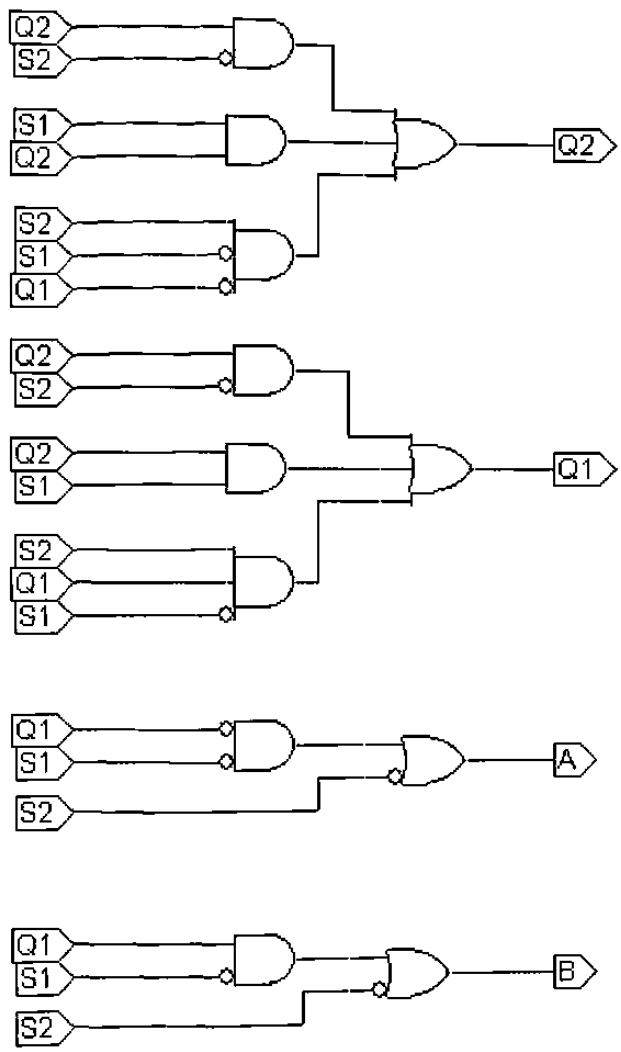


Figura 3.3.7.4.- Diagrama Esquemático.

b) Diagrama Escalera

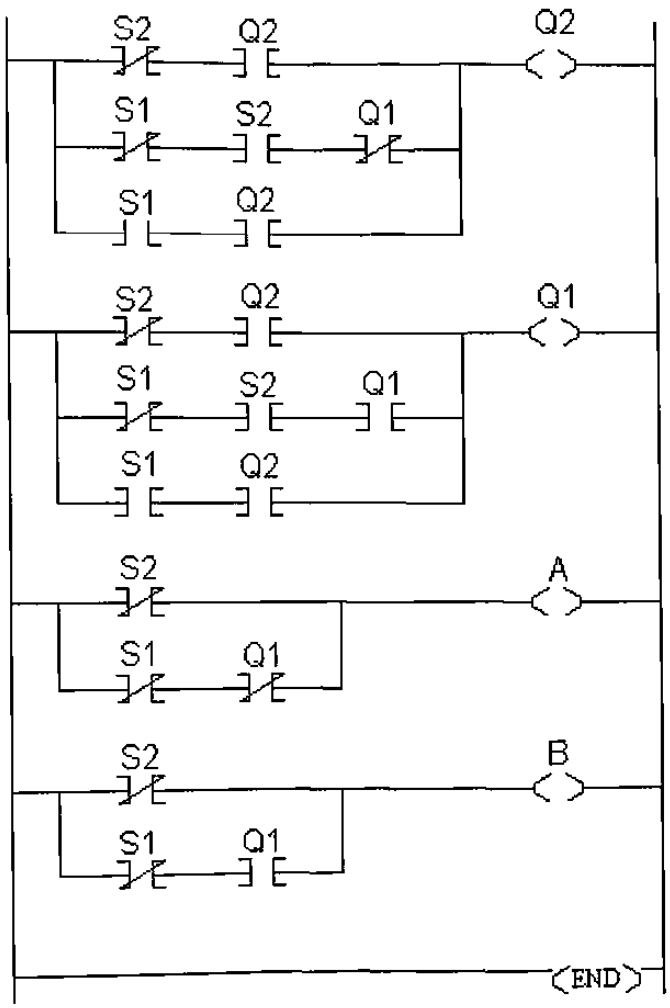


Figura 3.3.7.5.- Diagrama Escalera.

3.3.8 Dos Bombas Simultaneas Caso 2

El tanque de la figura se alimenta por medio de 2 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 3 sensores de entrada nivel A (Alto), M (Medio) y B (Bajo), y dos salidas S2 y 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 y S1.

1.- Partiendo de que el tanque se encuentra vació (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) y de ahí solo trabajar la bomba A. Sí se vacía de nuevo encenderán ambas bombas y al llegar al nivel medio (S2=0 y S1=1) trabajara de nuevo la bomba A, cuando el tanque se llene (S2= 1 y S1 =1) se deberán apagar ambas bombas, al llegar de nuevo al nivel medio (S2=0 y S1 = 1) deberá trabajar ahora solo la bomba B y si no es suficiente de modo que se vacié el tanque deberán de trabajar ambas bombas hasta llegar de nuevo al nivel medio (S2=0 y S1=1) y de ahí trabajar de nuevo solo la bomba B hasta llenarlo (S2=1 y S1=1).
2.- 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 de trabajar una sola bomba alternándose en su funcionamiento.
3.- Las bombas no se alternaran en su funcionamiento cuando el nivel pase de bajo a medio.

1.- Especificar el sistema

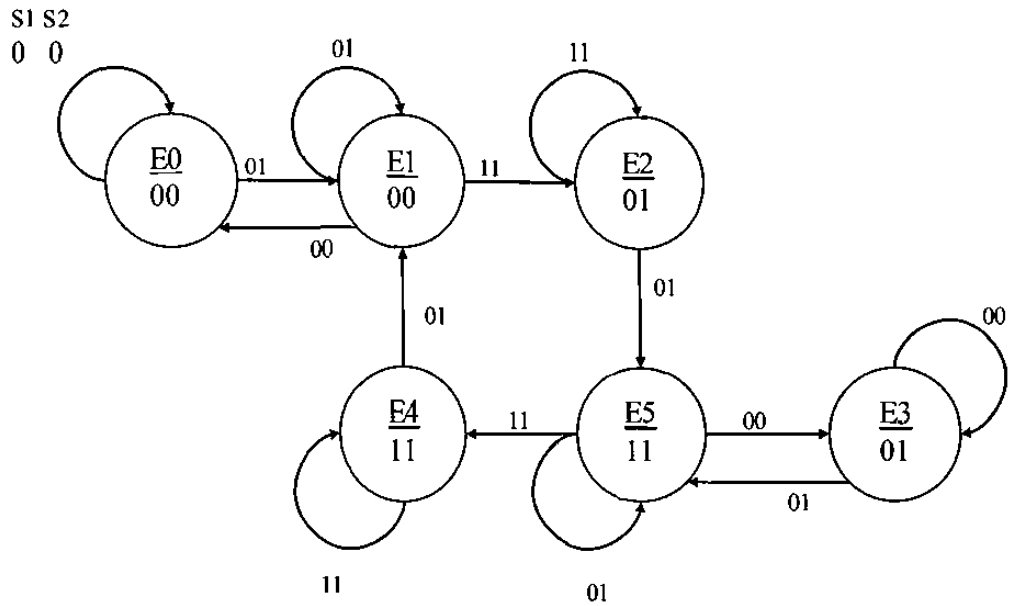


Figura 3.3.8.1.- Diagrama de Transición.

2.- Tabla de Flujo Primitiva

	00	01	10	11	A	B
0	E0	E1	X	X	1	1
1	E0	E1	X	E2	1	0
2	X	E3	X	E2	0	0
3	E4	E3	X	E5	0	1
4	E4	E3	X	X	1	1
5	X	E1	X	E5	0	0

Tabla 3.3.8.1.- Tabla de Flujo Primitiva

- La condición 10 no puede suceder, ya que no existe en el detector de nivel que se usa de referencia de los niveles del tanque.

3.- Reducción de Estados.

a) Estados estables en la misma columna.

00) E0, E4

01) E1, E3

11) E2, E5

b) Misma salida

E0 = E4

E1 ≠ E3

E2 = E5

c) Sus estados siguientes son equivalentes?

No son equivalentes porque E1≠E3

No hay reducción de estados.

4.- Mezcla de filas

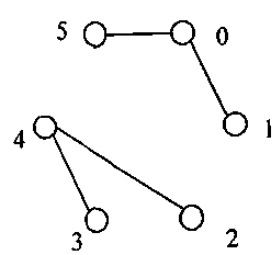


Figura 3.3.8.2.- Mezcla de Filas.

Se mezcla 0 con 5
Se mezcla 4 con 2

5.- Expandir tabla de salidas.

Ya que las salidas son diferentes se usa el metido de Mealy para expandir salidas.

	00	01	10	11	00	01	10	11
0,5	**E0**	E1	X	**E5**	11	10	X	00
1	E0	**E1**	X	E2	**1X**	**10**	X	**X0**
2,4	E4	E3	X	E2	11	**01**	X	00
3	E4	E3	X	E5	**X1**	01	X	**0X**

Tabla 3.3.8.2.- Tabla de Flujo Primitiva

6.- Tabla de Estados Internos

a = E0, E5

b = E1

c = E2,E4

d = E3

Sin consideradas salidas

	00	01	10	11
a	**a**	b	x	**a**
b	a	b	x	c
c	**c**	d	x	**c**
d	c	d	x	a

Tabla 3.3.8.3.- Tabla de estados internos.

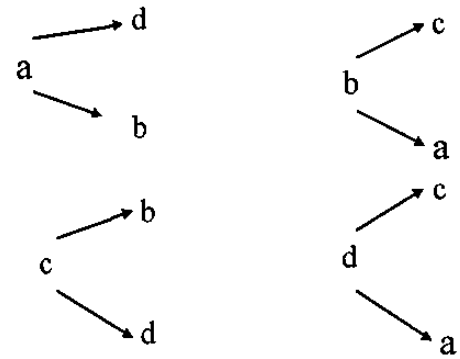
Transiciones

De estables a transitorios

00) a → b, c →d

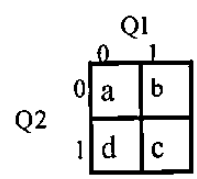
01)b→a, d →c

11) c→b, a →d



7.- Asignación de Valores a los Estados

	Q2	Q1
a	0	0
b	0	1
c	1	0
d	1	1



A partir de los valores obtenidos para a, b, c y d se obtiene la siguiente tabla.

	S2 S1					SALIDAS			
	Q2 Q1	*00*	*01*	*10*	*11*	*00*	*01*	*10*	*11*
0	0 0	00	01	X	00	11	**10**	X	00
1	0 1	00	11	X	11	**1X**	01	X	**X0**
2	1 0	11	10	X	00	**X1**	01	X	**0X**
3	1 1	11	10	X	11	11	**01**	X	00

Tabla 3.3.8.4.- Tabla de asignación de valores.

a = 00
b = 01
c = 11
d = 10

8.- Tabla de Estados Totales

```
 Input Variable Names:  S2  S1  Q2  Q1
 Output Function Names:  Q2+  Q1+  A  B
```

0000	0011
0001	001-
0010	11-1
0011	1111
0100	0110
0101	0110
0110	1001
0111	1001
1000	----
1001	----
1010	----
1011	----
1100	0000
1101	11-0
1110	000-
1111	1100

Tabla 3.3.8.5.- Tabla de estados totales.

10.- Obtención de las ecuaciones por medio de minimización.

Q2+ = S2'Q2 + S2 Q1

Q1+ = S1'Q2 + S2'S1 Q2' + S2 Q1

A = S1' + S2'Q2'

B = S1' + S2'Q2

11.- Elaboración del archivo en formato ABEL – HDL

```
MODULE cdos
"Entrada

S2,S1 pin 1,2;

"Salidas
A,B,Q2,Q1 PIN ISTYPE'COM';

EQUATIONS
Q2=!S2&Q2#S2&Q1;
Q1=!S1&Q2#!S2&S1&!Q2#S2&Q1;
A=!S1#!S2&!Q2;
B=!S1#!S2&Q2;

TEST_VECTORS
([S2,S1]->[Q2,Q1,A,B])
[0,0]->[.X.,.X.,.X.,.X.];
[0,1]->[.X.,.X.,.X.,.X.];
[1,1]->[.X.,.X.,.X.,.X.];
[0,1]->[.X.,.X.,.X.,.X.];
[0,0]->[.X.,.X.,.X.,.X.];
[0,1]->[.X.,.X.,.X.,.X.];
[1,1]->[.X.,.X.,.X.,.X.];
[0,1]->[.X.,.X.,.X.,.X.];
[0,0]->[.X.,.X.,.X.,.X.];
[0,1]->[.X.,.X.,.X.,.X.];
[1,1]->[.X.,.X.,.X.,.X.];
[0,1]->[.X.,.X.,.X.,.X.];

END
```

12.- Simulación en IspStarter de Lattice semiconductor.

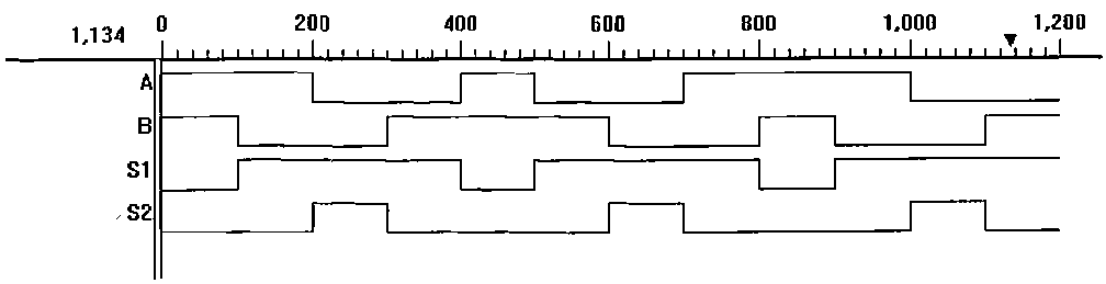


Figura 3.3.8.3.- Simulación..

13.- Diagramas

a) Diagrama Esquemático

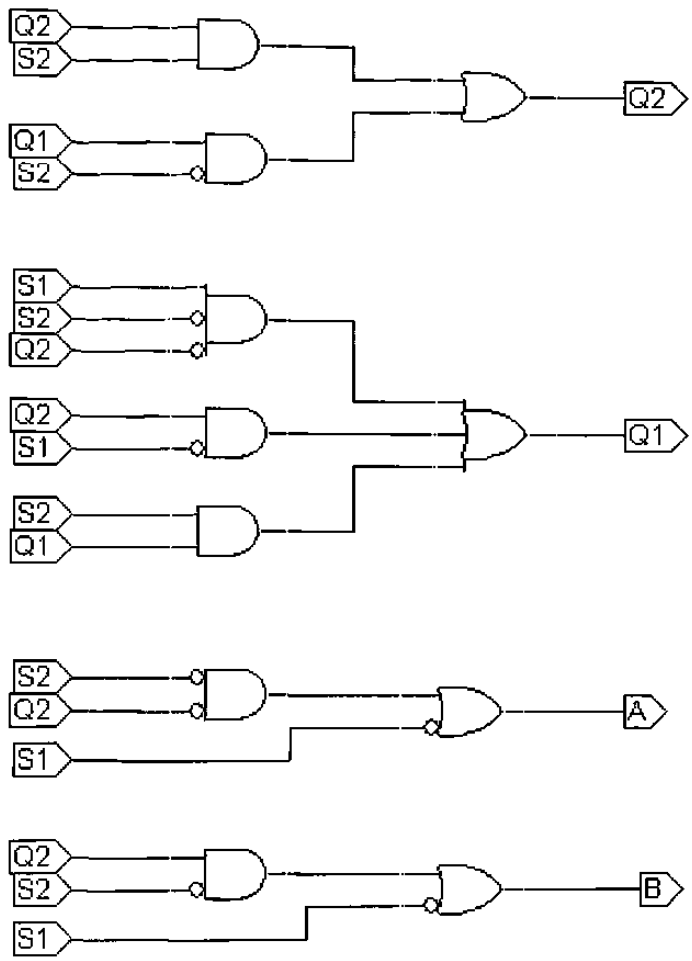


Figura 3.3.8.4.- Diagrama Esquemático.

b) Diagrama Escalera

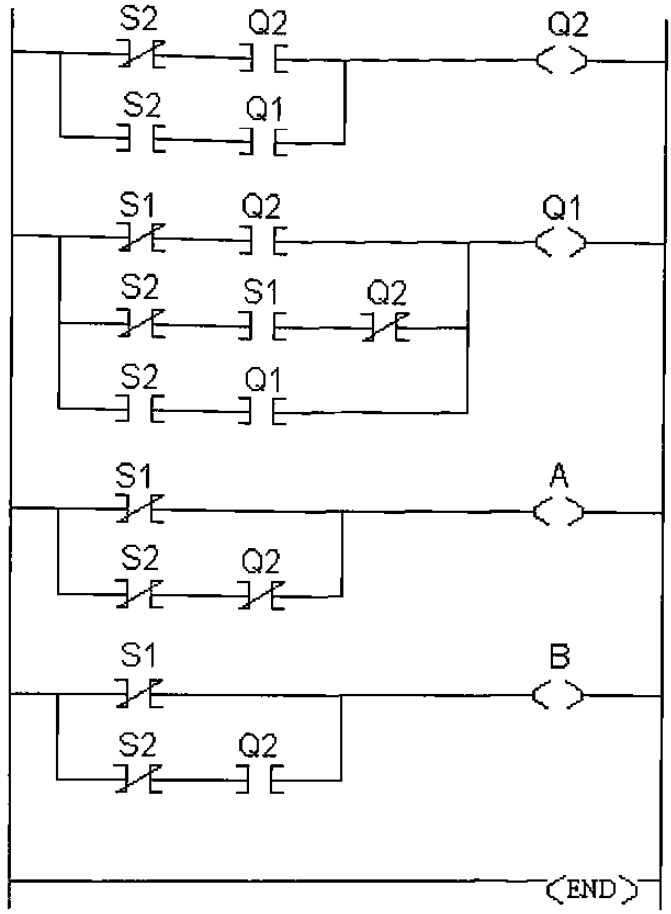
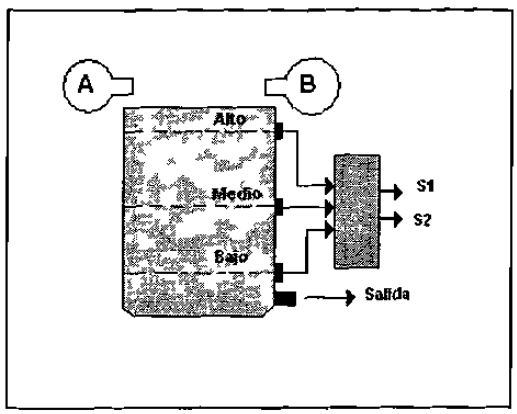


Figura 3.3.8.5.- Diagrama Escalera.

3.3.9.- Dos Bombas Simultaneas Caso 3

El tanque tiene un sistema detector de niveles que consta de 3 sensores de entrada nivel A (Alto), M (Medio) y B (Bajo), y dos salidas S_2, S_1 que indican lo siguiente:



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 S_2, S_1.

1. Partiendo de que el tanque se encuentra vacío (S_2=0 y S_1=0), se inicia el llenado con ambas bombas A y B, hasta llenar el tanque (S_2=1 y S_1=1) y entonces desconectarlas.
2. Una vez lleno si el nivel llega a medio (S_2=0 y S_1=1), deberá solo trabajar la bomba A, si el nivel sigue bajando hasta el nivel bajo (S_2=0 y S_1=0), deberán de trabajar de nuevo las bombas A y B, hasta llenar el tanque (S_2=1 y S_1=1) y entonces desconectarlas.
3. Una vez lleno si el nivel llega de nuevo a medio (S_2=0 y S_1=1), deberá solo trabajar la bomba B, si el nivel sigue bajando hasta el nivel bajo (S_2=0 y S_1=0), deberán de trabajar de nuevo ambas bombas A y B, hasta llenar el tanque (S_2=1 y S_1=1) y entonces desconectarlas.
4. Cada vez que el nivel pase de lleno a medio, deberá de trabajar una sola bomba alternándose en su funcionamiento.
5. Cada vez que se vacíe trabajaran ambas bombas hasta llenar el tanque.

Pasos para la resolución del problema planteado.

El primer paso a seguir al resolver este problema es la especificación del sistema, de esta manera se puede observar mejor el comportamiento de este.

1.- Especificación del sistema.

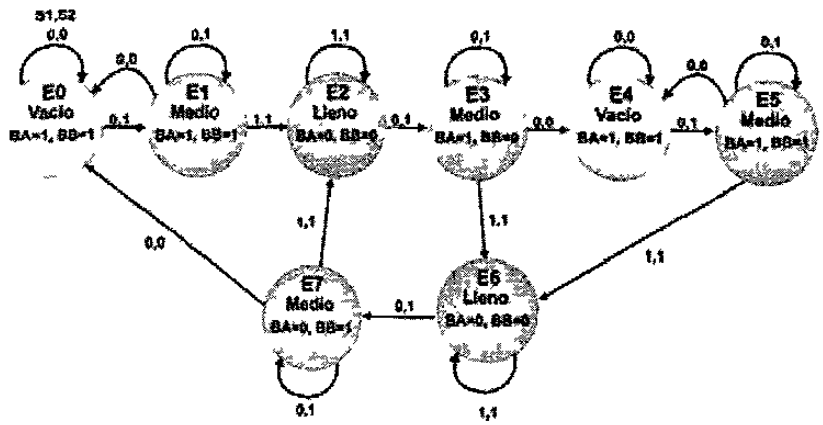


Figura 3.3.9.1.- Diagrama de Transición.

Empezando con el estado E0 que se considera cuando esta vacío el tanque (S_2=0 y S_1=0), y las bombas están encendidas (BA=1 y BB=1); si el tanque sigue vacío se mantiene en ese mismo estado, y si se empieza a llenar hasta llegar a un estado medio, pasa al siguiente estado E1. En este estado (S_2=0 y S_1=1) se mantienen las dos bombas encendidas hasta que se termine de llenar el tanque, pasando de esta manera a un estado E2 (S_2=1 y S_1=1).

En el siguiente estado se desconectan las bombas (BA=0, BB=0); si en algún momento en el estado E2 se llegará a vaciar llegando hasta el nivel medio, se encendería solamente la bomba BA, pasando a un nuevo estado E3 (S_2=0 y S_1=1); en este punto nosotros podemos tener dos desviaciones, una sería si estando en el nivel medio se llegará a vaciar, que entonces pasaría a un estado E4 (S_2=0 y S_1=0) y se activarían de nuevo las bombas BA y BB, ahora, si en el estado E3 se llegará a llenar, que seria la otra desviación, pasaría a un nuevo estado E6 (S_2=1 y S_1=1) donde se apagará la bomba BA.

Ahora regresando al estado E4 donde el tanque esta vacío y ambas bombas están activas hasta que llegue a el nivel medio, pasará entonces a un estado E5 (S_2=0 y S_1=1),

manteniendo ambas bombas encendidas hasta llenarse el tanque, cuando esto ocurra, cambiaremos de estado del E5 al E6, en donde el tanque se encuentra lleno (S_2=1 y S_1=1), y se desconectan ambas bombas.

Una vez que empiece a vaciarse el tanque hasta llegar al nivel medio, pasaremos ahora a otro nuevo estado E7 (S_2=0 y S_1=1), en donde ahora solamente trabajara la bomba BB. En este estado se presentan también dos desviaciones; una es si estando en el nivel medio se vaciara el tanque, pasaría al estado inicial E0, donde se activarían ambas bombas, y la otra desviación sería si estando en el estado E7 trabajando con la bomba BB, se llenará el tanque, entonces pasaría al estado E2, donde se apagarán ambas bomba.

De esta manera se completa el ciclo, alternándose ambas bombas después de que se llene el tanque y baje al nivel medio.

2.- Tabla de Flujo Primitiva

	00	01	10	11	BA	BB
0	**E0**	E1	X	X	1	1
1	E0	**E1**	X	E2	1	1
2	X	E3	X	**E2**	0	0
3	E4	**E3**	X	E6	1	0
4	**E4**	E5	X	X	1	1
5	E4	**E5**	X	E6	1	1
6	X	E7	X	**E6**	0	0
7	E0	**E7**	X	E2	0	1

Tabla 3.3.9.1.-Tabla de flujo primitiva.

Podremos notar que el nivel 10 no puede presentarse, ya que no puede estar lleno el nivel alto y el nivel bajo vació; con esto nosotros completamos todos los estados que puedan presentarse.

3.- Reducción de Estados.

a) Estados estables en la misma columna.

00) E0, E4
01) E1, E3, E5, E7
11) E2, E6

b) Misma salida

00) E0 → E4

01) E1 → E5

11) E2 → E6

c) Sus estados siguientes son equivalentes

02) E1 ≠ E5

03) E2 ≠ E6

NO HAY REDUCCION DE ESTADOS

4.- Mezcla de filas.

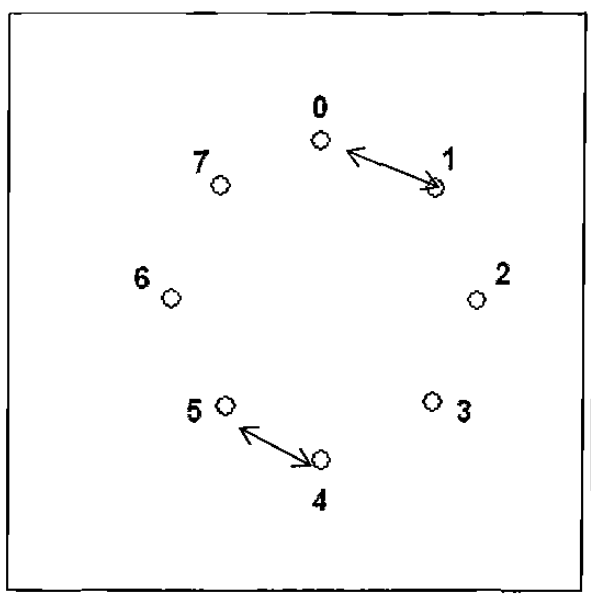


Figura 3.3.9.2.- Mezcla de Filas.

Mezclando las filas 0 con 1 y 4 con 5, obtenemos:

	00	01	10	11	BA	BB
0-1	E0	E1	----	E2	1	1
2	----	E3	----	E2	0	0
3	E4	E3	----	E6	1	0
4-5	E4	E5	----	E6	1	1
6	----	E7	----	E6	0	0
7	E0	E7	----	E2	0	1

Tabla 3.3.9.2.-Tabla de reducción de estados.

6.- Tabla de estados internos.

Sustituyendo:

a por E0 y E1
b por E2
c por E3
d por E4 y E5
e por E6
f por E7

Obtenemos:

	00	01	10	11
a	**a**	**a**	----	b
b	----	c	----	**b**
c	d	**c**	----	e
d	**d**	**d**	----	e
f	----	f	----	**e**
g	a	**f**	----	b

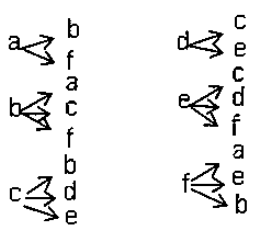
Tabla 3.3.9.3.-Tabla de estados internos.

Donde tenemos que en las filas, los estados transitorios pasa a estados estables, analizando esto obtenemos lo siguiente:

00) d → c, a → f

01) c → b, f → e

11) b → a, e → c, e → d, b → f



7.- Asignación de valores a los estados.

Como es imposible que todas se relacionen, suponemos unos estados transitorios, acomodándolos a nuestra conveniencia, buscando relacionar a todos los estados para que cumpla con los requisitos, en este caso se nombro a β y α como estados transitorios, como se ve en la figura.

Q1 Q0 \ Q3 Q2	00	01	11	10
00	β			
01	e	f	α	
11	c	b		
10	d			

	00	01	10	11	BA	BB
a	a	a	----	b	**1**	**1**
b	----	c	----	b	**0**	**0**
c	d	c	----	e	**1**	**0**
d	d	d	----	β	**1**	**1**
e	----	f	----	e	**0**	**0**
f	α	f	----	b	**0**	**1**
α	a	----	----	----	----	**1**
β	----	----	----	e	—	----

De la siguiente manera se asignan los valores a los estados:

	Q3	Q2	Q1	Q0
a	**1**	**1**	**1**	**1**
b	**0**	**1**	**1**	**1**
c	**0**	**0**	**1**	**1**
d	**0**	**0**	**1**	**0**
e	**0**	**0**	**0**	**1**
f	**0**	**1**	**0**	**1**
α	**1**	**1**	**0**	**1**
β	**0**	**0**	**0**	**0**

Tabla 3.3.9.4.-Tabla de asignación de valores a los estados.

	Q3	*Q2*	*Q1*	*Q0*	*00*	*01*	*10*	*11*	*BA*	*BB*
a	1	1	1	1	1111	1111	———	0111	1	1
b	0	1	1	1	———	0011	———	0111	0	0
c	0	0	1	1	0010	0011	———	0001	1	0
d	0	0	1	0	0010	0010	———	0000	1	1
e	0	0	0	1	———	0101	———	0001	0	0
f	0	1	0	1	1101	0101	———	0111	0	1
α	1	1	0	1	1111	———	———	———	—	1
β	0	0	0	0	———	———	———	0001	—	—

8.- Tabla de Estados Totales.

```
Input Variable Names:  Q3  Q2  Q1  Q0  S1  S2
Output Function Names:  Q3+  Q2+  Q1+  Q0+  BA  BB

000000  ------
000001  ------
000010  ------
000011  0001--
000100  ------
000101  010100
000110  ------
000111  000100
001000  001011
001001  001011
001010  ------
001011  000011
001100  001010
001101  001110
001110  ------
001111  000110
010000  ------
010001  ------
010010  ------
010011  ------
010100  110101
010101  010101
010110  ------
010111  011101
011000  ------
011001  ------
011010  ------
011011  ------
011100  ------
011101  001100
011110  ------
011111  011100
100000  ------
100001  ------
100010  ------
100011  ------
100100  ------
100101  ------
100110  ------
100111  ------
101000  ------
101001  ------
101010  ------
101011  ------
101100  ------
101101  ------
101110  ------
101111  ------
110000  ------
110001  ------
110010  ------
110011  ------
110100  1111-1
110101  ------
110110  ------
110111  ------
111000  ------
111001  ------
111010  ------
111011  ------
111100  111111
111101  111111
111110  ------
111111  011111
```

Tabla 3.3.9.5.-Tabla de estados totales.

10.- Obtención de las ecuaciones por medio de minimización.

```
Q3+ = Q3 S1' + Q1'S2'

Q2+ = Q1'S1' + Q2 S1 + Q3

Q1+ = Q1 S1' + Q2 S1 + Q3

Q0+ = Q1' + Q0 S2 + Q2

BA  = Q2'Q1 + Q3

BB  = Q0' + Q2 Q1' + Q3
```

La minimización se obtuvo por medio del programa Logic Aid.

11.- Elaborar el Archivo en formato ABEL-HDL.

```
MODULE caso3

"Entradas

S1,S2 PIN 1,2;

"SALIDAS

Q3,Q2,Q1,Q0,BA,BB PIN ISTYPE'COM';

EQUATIONS

Q3=Q3&!S1#!Q1&!S2;
Q2=!Q1&!S1#Q2&S1#Q3;
Q1=Q1&!S1#Q2&S1#Q3;
Q0=!Q1#Q0&S2#Q2;
BA=!Q2&Q1#Q3;
BB=!Q0#Q2&!Q1#Q3;

TEST_VECTORS
([S1,S2]->[BA,BB,Q3,Q2,Q1,Q0])
[0,0]->[.X...X...X...X...X...X.];
[0,1]->[.X...X...X...X...X...X.];
[0,0]->[.X...X...X...X...X...X.];
[0,1]->[.X...X...X...X...X...X.];
[1,1]->[.X...X...X...X...X...X.];
[0,1]->[.X...X...X...X...X...X.];
[1,1]->[.X...X...X...X...X...X.];
[0,1]->[.X...X...X...X...X...X.];
[1,1]->[.X...X...X...X...X...X.];
[0,1]->[.X...X...X...X...X...X.];
[0,0]->[.X...X...X...X...X...X.];
[0,1]->[.X...X...X...X...X...X.];
[0,0]->[.X...X...X...X...X...X.];
[0,1]->[.X...X...X...X...X...X.];
[0,0]->[.X...X...X...X...X...X.];
[0,1]->[.X...X...X...X...X...X.];
[0,0]->[.X...X...X...X...X...X.];
[0,1]->[.X...X...X...X...X...X.];
[1,1]->[.X...X...X...X...X...X.];
[0,1]->[.X...X...X...X...X...X.];
[0,0]->[.X...X...X...X...X...X.];
[0,1]->[.X...X...X...X...X...X.];
[1,1]->[.X...X...X...X...X...X.];
[0,1]->[.X...X...X...X...X...X.];
[0,0]->[.X...X...X...X...X...X.];
[0,1]->[.X...X...X...X...X...X.];
[0,0]->[.X...X...X...X...X...X.];
[0,1]->[.X...X...X...X...X...X.];

END
```

12.- Simulación en IspStarter de Lattice semiconductor.

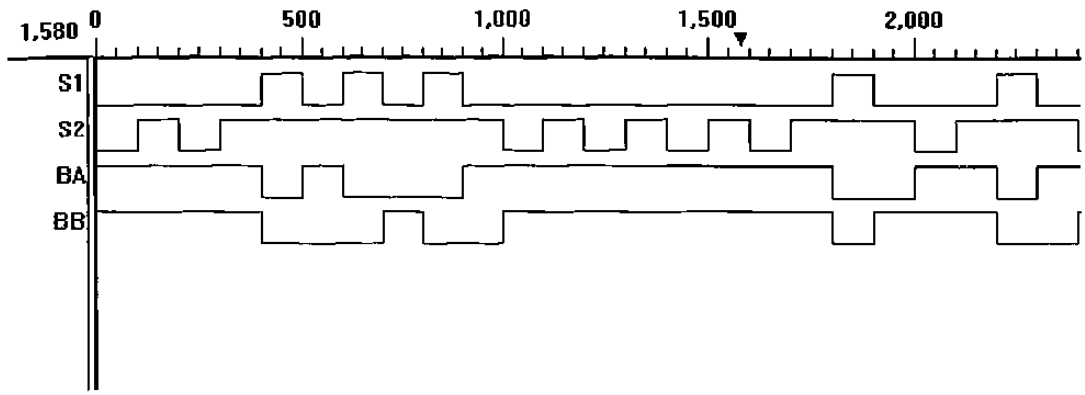


Figura 3.3.9.3.- Simulación..

13.- Diagramas

a) Diagrama Esquemático

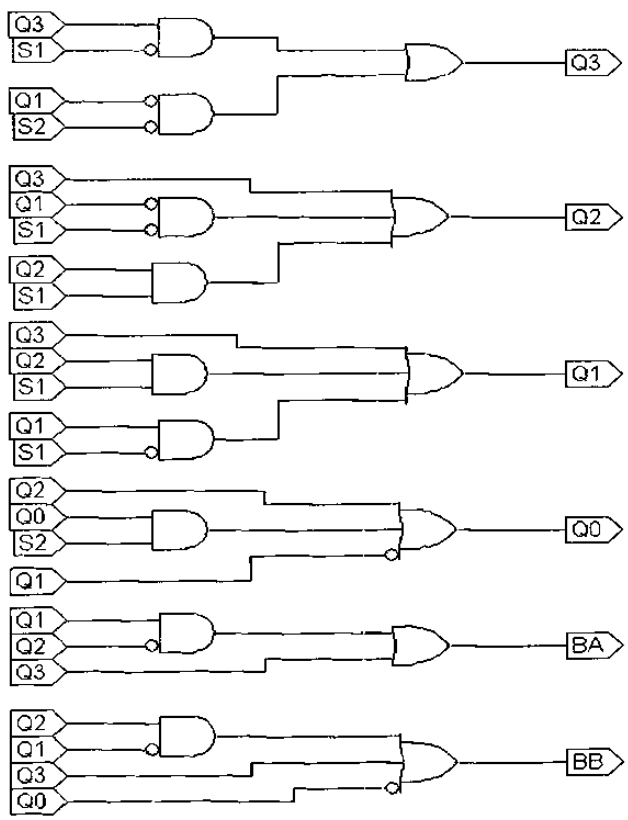


Figura 3.3.9.4.- Diagrama Esquematico.

b) Diagrama Escalera

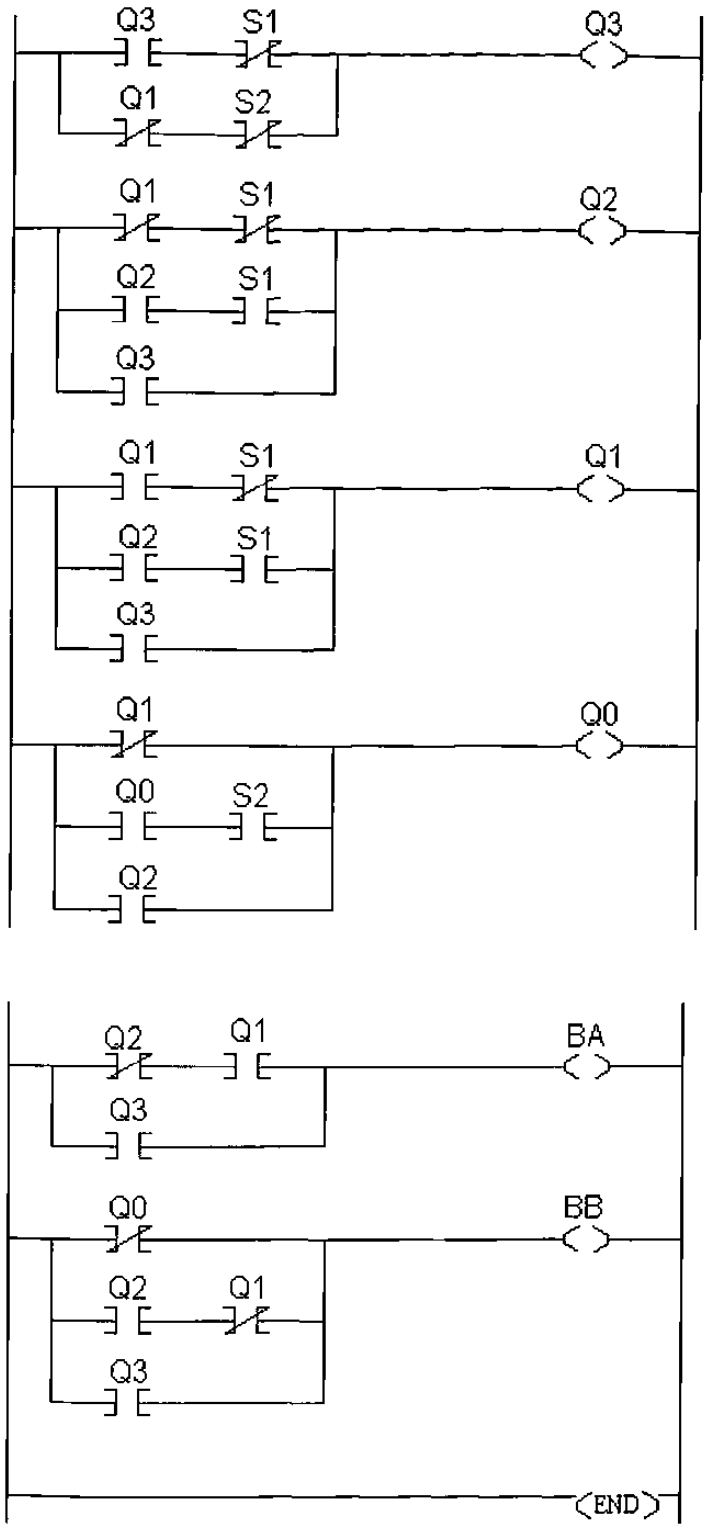


Figura 3.3.9.5.- Diagrama Escalera.

3.3.10.- Puerta de Garage

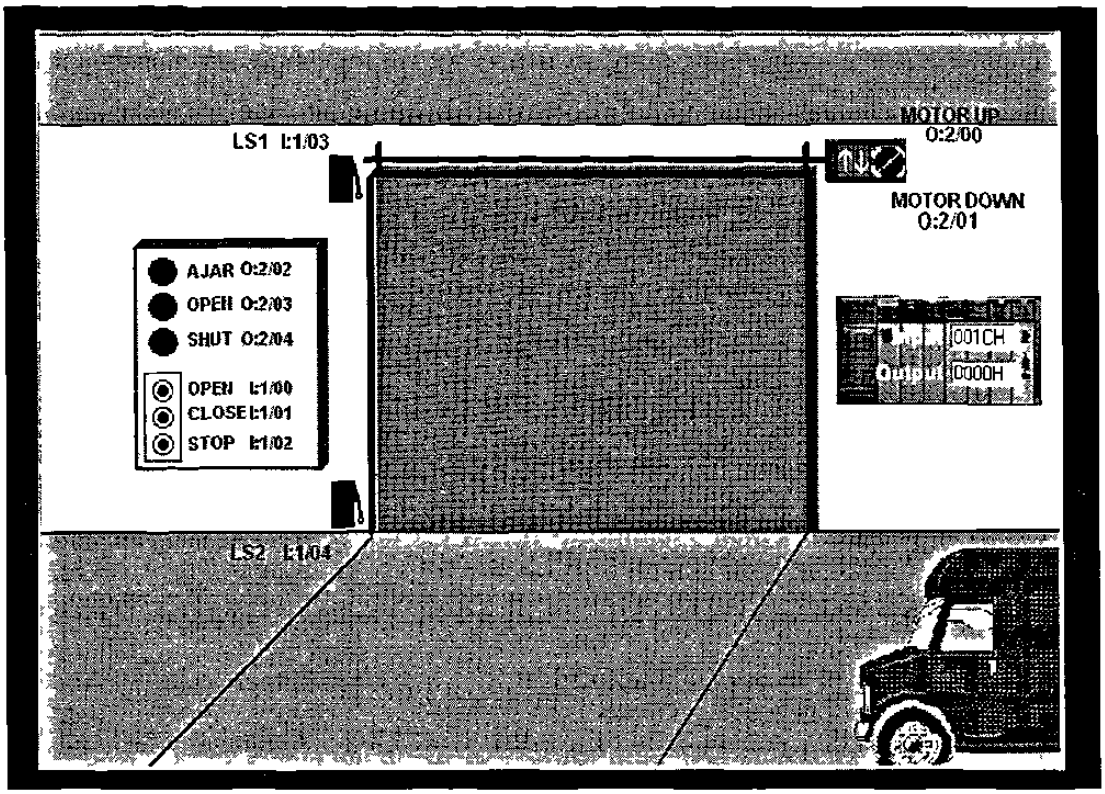


Figura 3.3.10. - Puerta de Garage

Se diseñará el control de una puerta de Garaje, el cuál debe de cumplir con las siguientes características de funcionamiento:

El movimiento de la puerta se detendrá inmediatamente cuando se presione el switch Stop y permanecerá detenida cuando se deje de presionar el switch.

Al presionar el switch Open, la puerta se abrirá siempre y cuando la puerta no esté completamente abierta. Si el switch es dejado de presionar la puerta continuará abriéndose hasta abrirse por completo.

Al presionar el switch Close hará que la puerta se cierre a menos que esté completamente cerrada. El cierre de la puerta se mantendrá hasta completarse aun cuando deje de presionar el switch de cierre.

Si la puerta está completamente abierta, el presionar el switch de Apertura de Puerta no energizará el motor.

Si la puerta está completamente cerrada, el presionar el switch de Cierre de Puerta no debe energizar el motor.

Bajo ninguna circunstancia los dos embobinados (motor up y motor down) del motor deben energizarse simultáneamente.

La luz Ajar deberá iluminarse si la puerta no está completamente cerrada o completamente abierta.

La luz Open se encenderá cuando la puerta esté completamente abierta.

La luz Shut se encenderá cuando la puerta esté completamente cerrada.

Este problema de divide en tres partes para su resolución, la primera consta de un sistema secuencial asíncrono, en la utilización de los botones para activar la puerta del garaje, la segunda parte, es también un sistema secuencial asíncrono, donde se plantean los sensores además de la activación del motor de subir o bajar; y la ultima parte se trata de un sistema combinacional, que será las luces indicadoras de salidas que nos informan si esta abierta o cerrada o intermedia la puerta del garaje.

Se analizara la primera parte, en el cual los botones tienen memoria.

Pasos para la resolución del problema planteado.

1.- Especificación del sistema

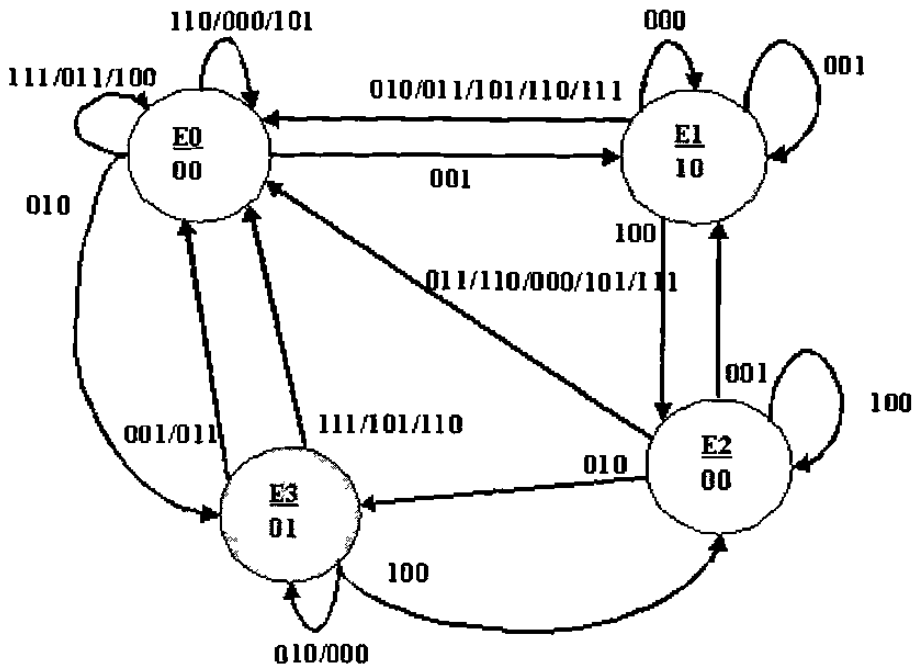


Figura 3.3.10.2.- Diagrama de Transición del Sistema

Con los tres botones de inicio tendríamos 3 entradas, pero se puede reducir a solamente tener 2 variables de entradas de la siguiente manera (Ver figura 14):

C igual a cerrar
O igual a abrir
S igual a parar

Donde X1 y X2 son las salidas

	X1	X2
C	1	0
O	0	1
S	0	0

Tabla 3.3.10.1.-Variables del Sistema

1. Tabla de flujo primitiva

	SOC								SALIDAS	
	000	*001*	*010*	*011*	*100*	*101*	*110*	*111*	X1	X2
0	**E0**	E1	E3	**E0**	**E0**	**E0**	**E0**	**E0**	**0**	**0**
1	**E1**	**E1**	E0	E0	E2	E0	E0	E0	**1**	**0**
2	E0	E1	E3	E0	**E2**	E0	E0	E0	**0**	**0**
3	**E3**	E0	**E3**	E0	E2	E0	E0	E0	**0**	**1**

Tabla 3.3.10.2.-Tabla de flujo primitiva

2. Reducción de Estados
   NO HAY REDUCCION DE ESTADOS
3. Mezcla de filas
   NO HAY MEZCLA DE FILAS

4. Tabla de estados internos

Sustituyendo:

a por E0
b por E1
c por E2
d por E3

	000	001	010	011	100	101	110	111	X1	X2
a	**a**	b	d	**a**	**a**	**a**	**a**	**a**	**0**	**0**
b	**b**	**b**	a	a	c	a	a	a	**1**	**0**
c	a	b	d	a	**c**	a	a	a	**0**	**0**
d	**d**	a	**d**	a	c	a	a	a	**0**	**1**

Tabla 3.3.10.3.-Estados Internos

En donde observamos que:

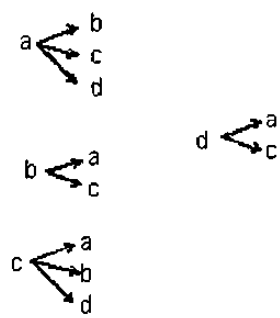


Figura 3.3.10.3.- Reducción de Filas

5. Asignación de valores a los estados

Se nombran β y α como estados transitorios, para la relacionar las variables con su correspondientes

	Q1Q0	00	01	11	10
Q2	0	b	a	d	
	1	α	c	β	

Figura 3.3.10.4.- Asignación de valores

	000	001	010	011	100	101	110	111	X1	X2
a	a	b	d	a	a	a	a	a	0	0
b	b	b	a	a	α	a	a	a	1	0
c	a	b	β	a	c	a	a	a	0	0
d	d	a	d	a	c	a	a	a	0	1
α	X	X	X	X	c	X	X	X	X	0
β	X	X	d	X	X	X	X	X	0	X

Tabla 3.3.10.4.-Tabla de asignación de valores.

De la siguiente manera se asignan los valores a los estados:

	Q2	Q1	Q0	000	001	010	011	100	101	110	111	X1	X2
a	0	0	1	001	000	011	001	001	001	001	001	0	0
b	0	0	0	000	000	001	001	100	001	001	001	1	0
c	1	0	1	001	000	111	001	101	001	001	001	0	0
d	0	1	1	011	001	011	001	101	001	001	001	0	1
α	1	0	0	X	X	X	X	101	X	X	X	X	0
β	1	1	1	X	X	011	X	X	X	X	X	0	X

Tabla 3.3.10.5.-Asignación de valores

6. Tabla de estados totales

Input Variable Names: Q4 Q3 Q2 S 0 C Output Function Names: Q4+ Q3+ Q2+ X1 X2	
000000	00010
000001	00010
000010	00110
000011	00110
000100	10010
000101	00110
000110	00110
000111	00110
001000	00100
001001	00000
001010	01100
001011	00100
001100	00100
001101	00100
001110	00100
001111	00100
010000	-----
010001	-----
010010	-----
010011	-----
010100	-----
010101	-----
010110	-----
010111	-----
011000	01101
011001	00101
011010	01101
011011	00101
011100	10101
011101	00101
011110	00101
011111	00101
100000	----0
100001	----0
100010	----0
100011	----0
100100	101-0
100101	----0
100110	----0
100111	----0
101000	00100
101001	00000
101010	11100
101011	00100
101100	10100
101101	00100
101110	00100
101111	00100
110000	-----
110001	-----
110010	-----
110011	-----
110100	-----
110101	-----
110110	-----
110111	-----
111000	---0-
111001	---0-
111010	0110-
111011	---0-
111100	---0-
111101	---0-
111110	---0-
111111	---0-

Tabla 3.3.10.6.-Tabla de estados totales

7. Obtención de las ecuaciones por medio de minimización.

Q4+ = Q2'S O'C' + Q3 S O'C' + Q4 Q3'S'O C' + Q4 S O'C'

Q3+ = Q2 S'O C' + Q3 S'C'

Q2+ = O + S C + Q2 C' + Q3 + Q4 Q2'

X1 = Q2'

X2 = Q3

Se analizara ahora la segunda parte, donde en esta abarca el sensor alto y el sensor bajo, y la acción del motor de subir y bajar.

1. Especificar el sistema.

Ahora como variables de entrada tenemos:

X1 y X2, salidas de la primera parte, el uso de los botones S, O y C.

SH, sensor alto.
SL, sensor bajo.

En donde:

SH	*SL*	*Condición de la Puerta del Garaje*
0	**0**	**Abierta**
0	**1**	**----------**
1	**0**	**Intermedia**
1	**1**	**Cerrada**

Tabla 3.3.10.7.-Especificación del sistema

Con estas condiciones obtenemos este diagrama de transición.

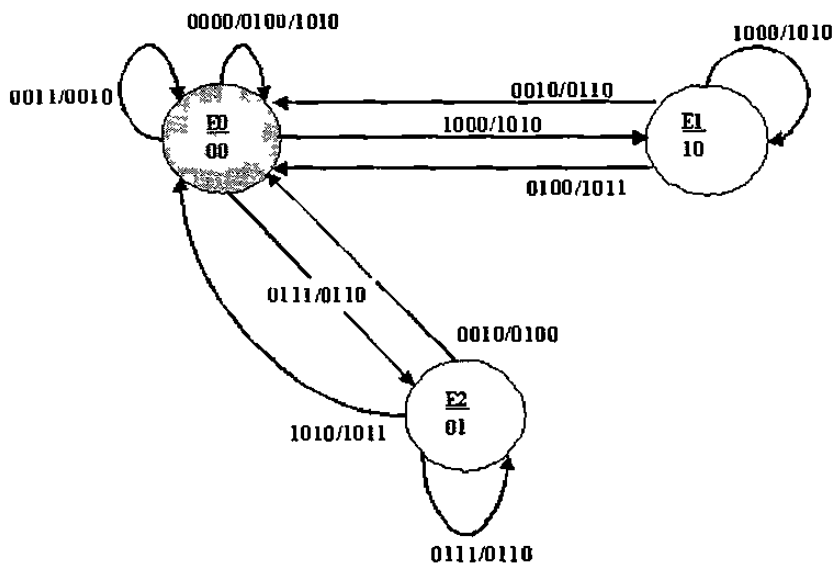


Figura 3.3.10.5.- Diagrama de Transición

2. Tabla de Flujo Primitiva.

	X1 X2 SH SL																Salidas	
	0000	*0001*	*0010*	*0011*	*0100*	*0101*	*0110*	*0111*	*1000*	*1001*	*1010*	*1011*	*1100*	*1101*	*1110*	*1111*	*MU*	*MD*
0	**E0**	X	**E0**	**E0**	**E0**	X	E2	E2	E1	X	E1	**E0**	X	X	X	X	**0**	**0**
1	X	X	E0	X	E0	X	E0	X	**E1**	X	**E1**	E0	X	X	X	X	**0**	**1**
2	X	X	E0	X	E0	X	**E2**	**E2**	X	X	E0	E0	X	X	X	X	**1**	**0**

Tabla 3.3.10.8.-Tabla de flujo primitiva

3. Reducción de Estados

NO HAY REDUCCION DE ESTADOS

4. Mezcla de filas.

NO HAY MEZCLA DE FILAS

5. Tabla de Estados Internos.

Se sustituye:

a por E0
b por E1
c por E2

Se obtiene la siguiente tabla:

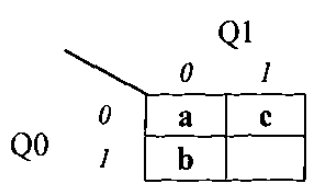
	X1 X2 SH SL																Salidas	
	0000	*0001*	*0010*	*0011*	*0100*	*0101*	*0110*	*0111*	*1000*	*1001*	*1010*	*1011*	*1100*	*1101*	*1110*	*1111*	*MU*	*MD*
a	**a**	x	**a**	**a**	**a**	x	c	c	b	x	b	**a**	x	x	x	x	**0**	**0**
b	x	x	a	x	a	x	a	x	**b**	x	**b**	a	x	x	x	x	**0**	**1**
c	x	x	a	x	a	x	**c**	**c**	x	x	a	a	x	x	x	x	**1**	**0**

Tabla 3.3.10.9.-Estados Internos

Donde tenemos que en las filas, los estados transitorios pasa a estados estables.

0010) De **a** a **b** y de **a** a **c**
0100) De **a** a **b** y de **a** a **c**
0110) De **c** a **a** y de **a** a **b**
0111) De **c** a **a**
1000) De **b** a **a**
1010) De **b** a **a** y de **a** a **c**
1011) De **a** a **b** y de **a** a **c**

6. Asignación de valores a los Estados.



	Q1	Q0
a	0	0
b	0	1
c	1	1

Figura 3.3.10.6.- Asignación de Valores a los estados

Sustituyendo los valores asignados a las variables en la tabla.

QI	Q0	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	MU	MD
0	0	00	X	00	00	00	X	11	11	01	X	01	00	X	X	X	X	0	0
0	1	X	X	00	X	00	X	00	X	01	X	01	00	X	X	X	X	0	1
1	1	X	X	00	X	00	X	11	11	X	X	00	00	X	X	X	X	1	0

Tabla 3.3.10.10.-Asignación de Valores

7. Tabla de Estados Totales.

```
Input Variable Names:  Q1  Q0  X1  X2  SH  SL
Output Function Names:  Q1+  Q0+  MU  MD

000000  0000                    100011  ----
000001  --00                    100100  ----
000010  0000                    100101  ----
000011  0000                    100110  ----
000100  0000                    100111  ----
000101  --00                    101000  ----
000110  1100                    101001  ----
000111  1100                    101010  ----
001000  0100                    101011  ----
001001  ----                    101100  ----
001010  0100                    101101  ----
001011  0000                    101110  ----
001100  ----                    101111  ----
001101  ----                    110000  --10
001110  ----                    110001  --10
001111  ----                    110010  0010
010000  --01                    110011  --10
010001  --01                    110100  0010
010010  0001                    110101  --10
010011  --01                    110110  1110
010100  0001                    110111  1110
010101  --01                    111000  --10
010110  0001                    111001  --10
010111  --01                    111010  0010
011000  0101                    111011  0010
011001  --01                    111100  ----
011010  0101                    111101  ----
011011  0001                    111110  ----
011100  ----                    111111  ----
011101  ----
011110  ----
011111  ----
100000  ----
100001  ----
100010  ----
```

Tabla 3.3.10.11.-Asignación de estados totales

8. Obtención de las ecuaciones por medio de minimización.

```
Q1+ = Q0'X2 SH + Q1 X2 SH

Q0+ = Q0'X2 SH + Q1'X1 SL' + Q1 X2 SH

MU = Q1

MD = Q1'Q0
```

9. Elaboración de el archivo en formato ABEL – HDL

Se hizo la elaboración de el archivo ABEL – HDL, adjuntando las ecuaciones de la primera parte del problema, los sistemas secuénciales asíncronos.

```
MODULE ejem12

"Entradas

S,O,C,SH,SL PIN 1,2,3,4,5;

"SALIDAS

Q4,Q3,Q2,Q1,Q0,X1,X2,MU,MD PIN 23..15 ISTYPE'COM';

EQUATIONS

Q4=!Q2&S&!O&!C#Q3&S&!O&!C#Q4&!Q3&!S&O&!C#Q4&S&!O&!C;
Q3=Q2&!S&O&!C#Q3&!S&!C;
Q2=O#S&C#Q2&!C#Q3#Q4&!Q2;
Q1=!Q0&X2&SH#Q1&X2&SH;
Q0=!Q0&X2&SH#!Q1&X1&!SL#Q1&X2&SH;
X1=!Q2;
X2=Q3;
MD=!Q1&Q0;
MU=Q1;


TEST_VECTORS
([S,O,C,SH,SL]->[Q4,Q3,Q2,Q1,Q0,X1,X2,MU,MD])

"DE CERRADO A ABIERTO Y DE ABIERTO A CERRADO
[0,0,0,1,1]->[.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.];
[0,1,0,1,1]->[.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.];
[0,0,0,1,1]->[.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.];
[0,0,0,1,0]->[.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.];
[0,0,0,1,0]->[.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.];
[0,0,0,0,0]->[.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.];
[0,0,0,0,0]->[.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.];
[0,0,1,0,0]->[.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.];
[0,0,0,0,0]->[.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.];
[0,0,0,1,0]->[.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.];
[0,0,0,1,0]->[.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.];
[0,0,0,1,1]->[.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.];
[0,0,0,1,1]->[.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.];
```

```
"DE CERRADO A MEDIO Y DE MEDIO A CERRADO
[0,1,0,1,1]->[.X..X..X..X..X..X..X..X..X.];
[0,0,0,1,0]->[.X..X..X..X..X..X..X..X..X.];
[0,0,0,1,0]->[.X..X..X..X..X..X..X..X..X.];
[1,0,0,1,0]->[.X..X..X..X..X..X..X..X..X.];
[0,0,0,1,0]->[.X..X..X..X..X..X..X..X..X.];
[0,0,0,1,0]->[.X..X..X..X..X..X..X..X..X.];
[0,0,1,1,0]->[.X..X..X..X..X..X..X..X..X.];
[0,0,0,1,0]->[.X..X..X..X..X..X..X..X..X.];
[0,0,0,1,1]->[.X..X..X..X..X..X..X..X..X.];
[0,0,0,1,1]->[.X..X..X..X..X..X..X..X..X.];

"DE CERRADO A MEDIO Y DE MEDIO A ABIERTO
[0,1,0,1,1]->[.X..X..X..X..X..X..X..X..X.];
[0,0,0,1,0]->[.X..X..X..X..X..X..X..X..X.];
[0,0,0,1,0]->[.X..X..X..X..X..X..X..X..X.];
[1,0,0,1,0]->[.X..X..X..X..X..X..X..X..X.];
[0,0,0,1,0]->[.X..X..X..X..X..X..X..X..X.];
[0,0,0,1,0]->[.X..X..X..X..X..X..X..X..X.];
[0,1,0,1,0]->[.X..X..X..X..X..X..X..X..X.];
[0,0,0,1,0]->[.X..X..X..X..X..X..X..X..X.];
[0,0,0,0,0]->[.X..X..X..X..X..X..X..X..X.];
[0,0,0,0,0]->[.X..X..X..X..X..X..X..X..X.];

"DE ABIERTO A MEDIO Y DE MEDIO A ABIERTO
[0,0,1,0,0]->[.X..X..X..X..X..X..X..X..X.];
[0,0,0,0,0]->[.X..X..X..X..X..X..X..X..X.];
[0,0,0,1,0]->[.X..X..X..X..X..X..X..X..X.];
[0,0,0,1,0]->[.X..X..X..X..X..X..X..X..X.];
[1,0,0,1,0]->[.X..X..X..X..X..X..X..X..X.];
[0,0,0,1,0]->[.X..X..X..X..X..X..X..X..X.];
[0,0,0,1,0]->[.X..X..X..X..X..X..X..X..X.];
[0,1,0,1,0]->[.X..X..X..X..X..X..X..X..X.];
[0,0,0,1,0]->[.X..X..X..X..X..X..X..X..X.];
[0,0,0,1,0]->[.X..X..X..X..X..X..X..X..X.];
[0,0,0,0,0]->[.X..X..X..X..X..X..X..X..X.];
[0,0,0,0,0]->[.X..X..X..X..X..X..X..X..X.];

"DE ABIERTO A MEDIO Y DE MEDIO A CERRADO
[0,0,1,0,0]->[.X..X..X..X..X..X..X..X..X.];
[0,0,0,0,0]->[.X..X..X..X..X..X..X..X..X.];
[0,0,0,1,0]->[.X..X..X..X..X..X..X..X..X.];
[0,0,0,1,0]->[.X..X..X..X..X..X..X..X..X.];
[1,0,0,1,0]->[.X..X..X..X..X..X..X..X..X.];
[0,0,0,1,0]->[.X..X..X..X..X..X..X..X..X.];
[0,0,0,1,0]->[.X..X..X..X..X..X..X..X..X.];
[0,0,1,1,0]->[.X..X..X..X..X..X..X..X..X.];
[0,0,0,1,0]->[.X..X..X..X..X..X..X..X..X.];
[0,0,0,1,0]->[.X..X..X..X..X..X..X..X..X.];
[0,0,0,1,1]->[.X..X..X..X..X..X..X..X..X.];
[0,0,0,1,1]->[.X..X..X..X..X..X..X..X..X.];

END
```

Figura 3.3.10.6.- Ejemplo de Programación en Abel HDL

Simulación en IspStarter de Lattice semiconductor.

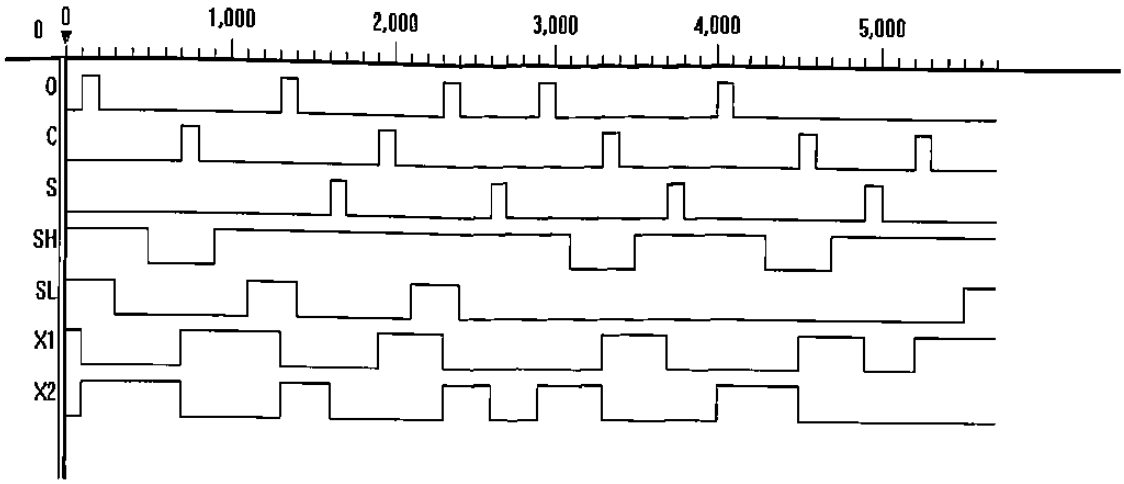
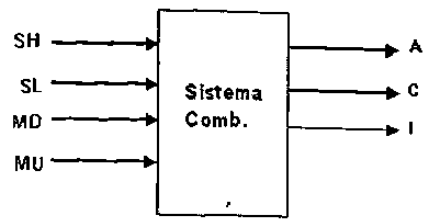


Figura 3.3.10.7.- Simulación

La tercera parte del problema es un sistema combinacional.

Especificación de entradas y salidas.



Elaboración del archivo en formato ABEL – HDL

```
MODULE LEDS

"ENTRADAS

SH,SL,MD,MU PIN 1,2,3,4;

"SALIDAS

A,C,I PIN 17..15 ISTYPE'COM';

EQUATIONS

A=!SH&!SL&!MD&!MU;
C=SH&SL&!MD&!MU;
I=SH&!SL&!MD&!MU;

TEST_VECTORS
([SH,SL,MD,MU]->[A,C,I])
[0,0,0,0]->[.X..X..X.];
[0,0,0,0]->[.X..X..X.];
[1,0,0,0]->[.X..X..X.];
[1,0,0,0]->[.X..X..X.];
[1,1,0,0]->[.X..X..X.];
[1,1,0,0]->[.X..X..X.];

END
```

Figura 3.3.10.8.- Programación en Abel HDL

Simulación en IspStarter de Lattice semiconductor.

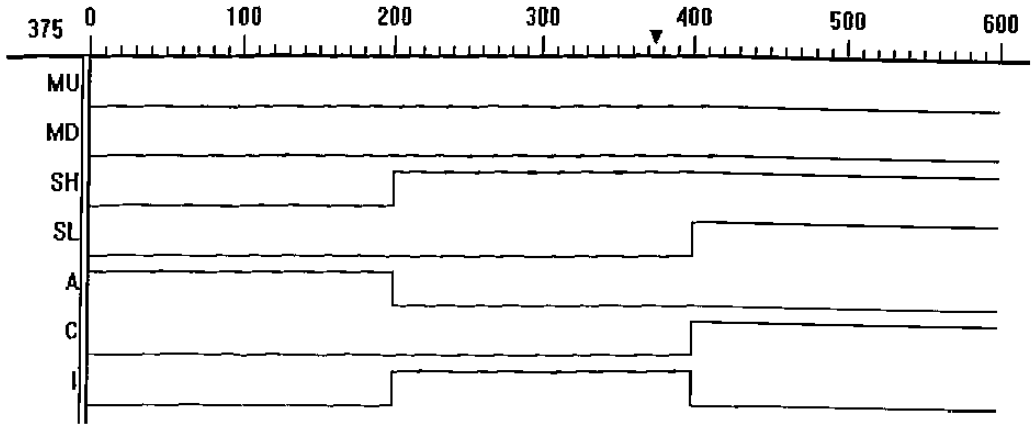


Figura 3.3.10.9.- Simulación

Diagrama Esquemático del Sistema

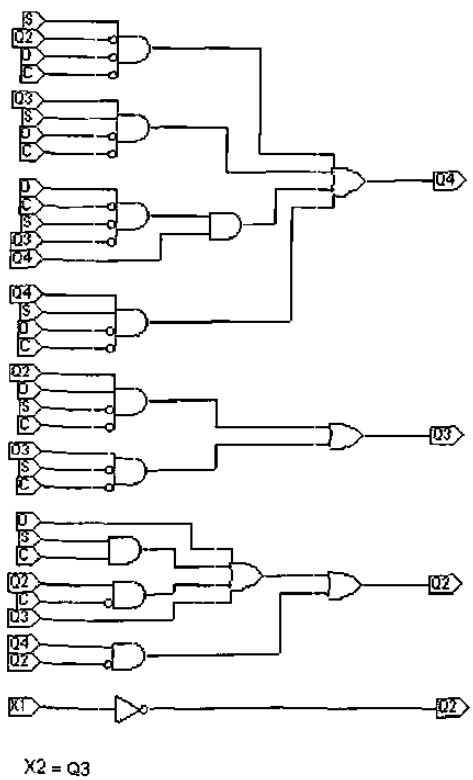


Figura 3.3.10.10.- Diagrama Esquemático.

Diagrama Escalera

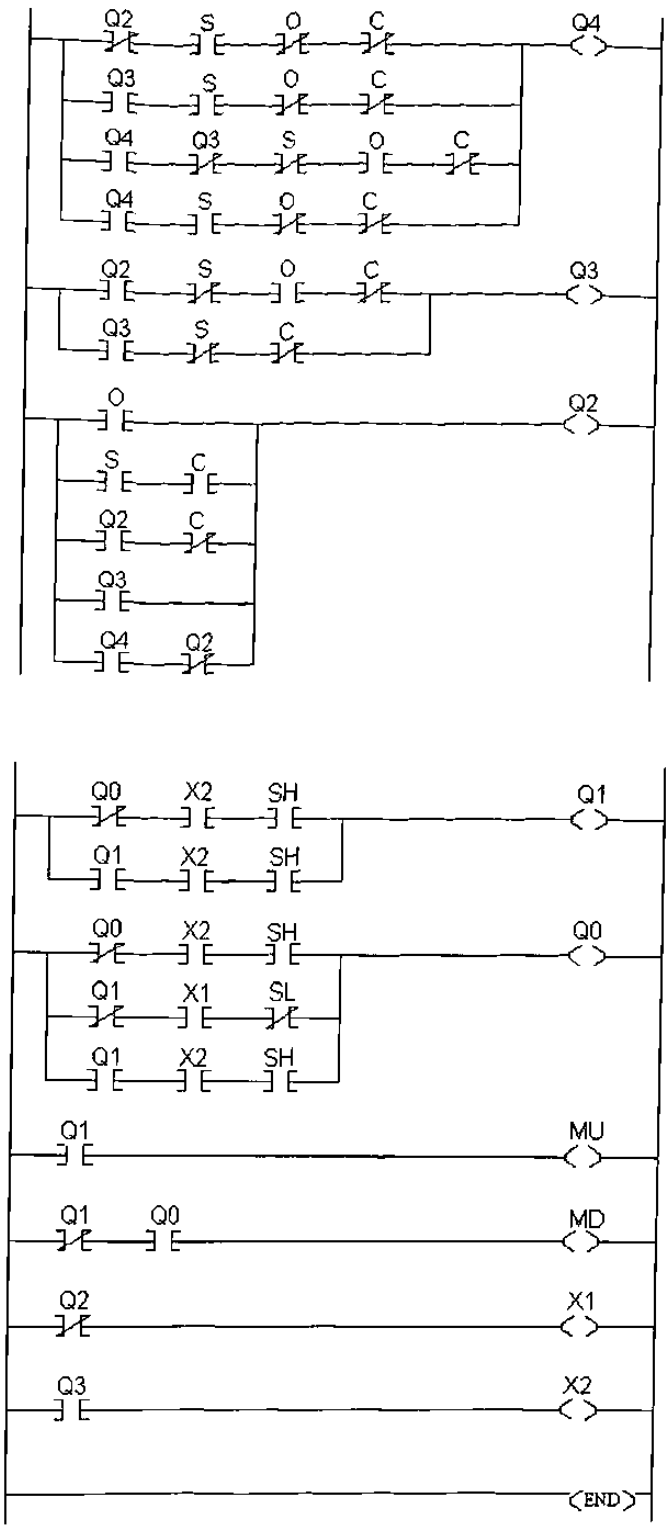


Figura 3.3.10.11.- Diagrama Escalera.

CAPITULO 4

CONCLUSIONES Y RECOMENDACIONES.

4.1.- Conclusiones

Una vez que se han seguido los pasos establecidos por la metodología propuesta se ha conseguido resolver cada una de las diferentes aplicaciones basadas en diferentes tipos de sistemas secuenciales asíncronos, de la forma más rápida y eficaz posible, se hace especial énfasis en la obtención de un sistema simplificado y optimo a la vez, lo que origina que nuestro sistema secuencial tenga una mayor respuesta al cambio de eventos sin perder tiempo en pasos innecesarios como se origina normalmente en los sistemas secuenciales diseñados de diferente forma.

Aun si existiesen casos de sistemas secuenciales de un grado de complejidad mayor, el seguir esta metodología conlleva a la obtención de la respuesta para nuestro problema. Respuesta que no ha de ser posible lograr de una forma más eficaz y simplificada si nos basáramos en los sistemas de resolución tradicionalmente utilizados.

Las herramientas computacionales empleadas (Software) sugeridas en esta tesis requieren de ciertos conocimientos o habilidades sencillas para su manejo, por lo que básicamente el tiempo que se emplee para lograr un mayor manejo del tema repercutirá

en el manejo de cada paso de nuestra metodología, al no utilizar herramientas demasiado complejas que entorpezcan el enfoque verdadero que esta tesis pretende brindar.

4.2.- Recomendaciones

Para lograr un mayor aprovechamiento de la metodología propuesta, es necesario tratar de ver la mayoría de los casos posibles que estén inmersos dentro de un problema secuencial asíncrono en el grado que su complejidad nos lo permita. Lo que nos llevara consecuentemente a lograr una mayor experiencia en el manejo de la metodología, que a la vez repercutirá en una mayor capacidad de análisis y dominio del tema.

En el caso de problemas complejos sería recomendable si esto fuera posible, seccionar o generar subsistemas del sistema principal y diseñarlos separadamente.

Al tratarse de un tema cuya resolución consiste en su implementación física, es recomendable trabajar en un área que nos brinde todas las herramientas necesarias para lograr dicha implementación. Herramientas como una computadora, un programador y lo que básicamente provee un laboratorio para las materias de electrónica lógica.

Con lo anterior el proceso de enseñanza-aprendizaje se realizara en un menor tiempo ya que se podrá corroborar el resultado teórico en el correcto desempeño de nuestro circuito secuencial como resultado de un buen seguimiento de la metodología propuesta.

BIBLIOGRAFÍA

Sistemas Digitales Principios y Aplicaciones
Tocci - Widmer
PRENTICE HALL

Fundamentos de Sistemas Digitales
T: L. Floyd
PRENTICE HALL

Teoría de Conmutación y Diseño Lógico
Frederick J. Hill y Gerald R. Peterson
LIMUSA

Fundamentos de Diseño Digital
Ing. Cesar A. Leal Chapa
FIME UANL

Fundamentals of Logia Design
Charles H. Roth, Jr.
WEST

Lógica Digital y Diseño de Computadoras
M. Morris Mano
PRENTICE HALL

Principios Digitales
Roger L. Tokheim Serie Shaum
MC GRAW HILL

Electrónica Digital
C.E. Strangio
INTERNATIONAL

Software

LogicAid
www.logicaid.com

ispEXPERT System Starter
www.latticesemi.com

LISTA DE FIGURAS

LISTA DE TABLAS