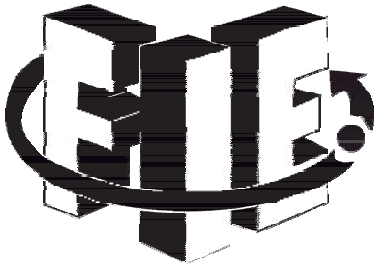


Universidad Michoacana de San Nicolás de Hidalgo

FACULTAD DE INGENIERÍA ELÉCTRICA

Simulación de Sistemas Digitales Usando el Software 20-SIM

Tesis

Que para obtener el título de

Ingeniero en Electrónica

Presenta

Jesús Villagomez Rivera

Asesor de tesis

Dr. Gilberto González Avalos

Morelia, Michoacán, Octubre de 2012

Agradecimientos

Quiero expresar mis más sinceros agradecimientos:

A mis padres Félix Villagomez Guerrero y Adela Rivera Centeno por todo el apoyo incondicional que me dieron a lo largo de todos mis estudios, por haber estado siempre ahí cuando los necesite.

A mi hermano Félix Villagomez Rivera por todas y cada una de las cosas que ha hecho por mí.

A mi asesor de tesis el Dr. Gilberto Gonzales Avalos por su apoyo, paciencia y dedicación durante este trabajo de tesis.

A mi gran amigo Noé Barrera Gallegos por toda la ayuda y su tiempo que me brindo durante este trabajo de tesis.

A la Universidad Michoacana de San Nicolás de Hidalgo por ser mi casa de estudios durante todos estos años.

A la Facultad de Ingeniería Eléctrica y a todos y a cada uno de los profesores que me instruyeron durante mi preparación profesional.

Y sobre todo a Dios por haberme dado la oportunidad de haber terminado una carera profesional.

Dedicatoria

Dedico este trabajo de tesis:

A mi esposa Tania Núñez Abrego porque siempre está a mi lado cuando la necesito, por ser tan comprensiva y paciente conmigo, por todo el apoyo incondicional que me brinda, porque sabe aconsejarme y guiarme por el buen camino, y sobre todo, por ser la mujer que AMO.

A mi hijo Jesse Villagomez Núñez por la alegría que me brinda a cada momento, por su linda sonrisa y por todos los bellos momentos que me hace pasar.

Ambos, por ser mi inspiración y mi razón de ser.

A mis familiares, a mis padres y a mis hermanos.

Resumen

En este trabajo de tesis se hizo una investigación del uso de las simulaciones a lo largo de la historia y en la actualidad. Se investigo y demostró la importancia que tienen el uso de sistemas digitales para el modelado y diseño de sistemas y su uso en el área de control.

Se menciona el uso del algebra booleana en el desarrollo de sistemas digitales, de la lógica secuencial y de su uso para el diseño de sistemas digitales. Se explica el uso de los Mapas de Karnaugh y se menciona la importancia que tiene durante el proceso de diseño para mejorar y facilitar la construcción de circuitos de diseño.

Se explica el uso del software 20-SIM para simular diferentes tipos de sistemas. Se mostrarán algunas de las herramientas de las que dispone este programa y algunas de sus librerías, además se explica el lenguaje de referencia que usa este software el cual servirá para mejorar las características de sus componentes. Se menciona las herramientas que se usarán durante este trabajo de tesis y las características que tienen y se explicara la forma de construir y simular los sistemas diseñados.

Se diseña y modela un sistema de alarma para un automóvil de 3 puertas, utiliza como herramientas de diseño los mapas de Karnaugh, y posteriormente, con ayuda del software 20-SIM, se construye y se simula este sistema.

Se diseña y modela un sistema de control para un sistema aljibe-tinaco, el cual controla el encendido y apagado de un motor de CD que sirve para accionar una bomba hidráulica mecánica la cual servirá para realizar el sistema de llenado del tinaco para el sistema.

Se menciona a detalle el funcionamiento de contadores tanto síncrono como asíncronos y se diseñan ambos contadores para posteriormente ser simulados en 20-SIM.

Se realizan diferentes simulaciones y se presentan pruebas para mostrar resultados que se tienen para los sistemas desarrollados. Por último se presentan las conclusiones y las recomendaciones a las que se llegó.

Contenido

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

1.1 Introducción a la Simulación de Sistemas

Según Robert Shannon la simulación es el diseño y desarrollo de un modelo computarizado de un sistema o proceso, y conducir experimentalmente con este modelo con el propósito de entender el comportamiento del sistema del mundo real o evaluar varias estrategias con los cuales puedan operar el sistema.

La simulación de sistemas intenta modelizar sistemas reales o hipotéticos por computadora de forma que su funcionamiento puede ser estudiado y podemos predecir su comportamiento.

La historia y la evolución de la simulación por computadora han ido paralelas a la evolución de la Informática. Sus orígenes los encontramos en la segunda Guerra Mundial cuando dos matemáticos, J. V. Neumann y S. Ulam, tenían el reto de resolver un problema complejo relacionado con el comportamiento de los neutrones. Los experimentos basados en prueba y error eran muy caros y el problema era demasiado complicado para abordarlo mediante técnicas analíticas. La aproximación aplicada se basa en la utilización de números aleatorios y distribuciones de probabilidad. El método desarrollado fue llamado "método de Montecarlo" por el paralelismo entre la generación de números aleatorios y el juego de la ruleta.

Durante la Guerra Fría se intensificó el uso de la simulación para resolver problemas de interés militar; trayectorias y dinámicas de satélites artificiales, guiar mísiles, etc. Muchos de estos problemas exigen la resolución de sistemas de ecuaciones diferenciales no lineales. Para abordar estos problemas se utilizaron computadoras analógicos que usaban elementos electrónicos para resolver las operaciones matemáticas: integración, suma, multiplicación, generación de funciones, etc.

A partir de la década de los 60 empiezan a aparecer en el mercado programas de simulación de sistemas de acontecimientos discretos que poco a poco se empezaron a utilizar para resolver problemas de ámbito civil. Los más destacables fueron el GPSS de IBM (General Purpose System Simulator) y el SIMSCRIPT. Los modelos de acontecimientos discretos son muy utilizados en la actualidad para estudiar problemas de fabricación de procesos, logística, transporte, comunicaciones y servicios. Estos problemas se caracterizan por centrar su interés en los cambios que hay en el sistema como

consecuencia de los acontecimientos y en su capacidad para modelar los aspectos aleatorios del sistema.

La revolución que se produjo en la informática a partir de los años 80, tiene un impacto importante en la simulación por computadora. El uso de simuladores se generaliza en prácticamente todos los ámbitos de la ciencia y la ingeniería hoy en la actualidad.

La simulación tiene como objetivo:

1. Descubrir el comportamiento de un sistema

2. Postular teorías o hipótesis que expliquen el comportamiento observado.

3. Usar esas teorías para predecir el comportamiento futuro del sistema, es decir mirar los efectos que se producirían en el sistema mediante los cambios dentro de él o en su método de operación.

1.2 Objetivo

El objetivo de este trabajo de tesis es diseñar e implementar la simulación de un sistema digital por medio del software 20-SIM.

1.3 Justificación

La realización de este trabajo de tesis es mostrar un enfoque de los sistemas digitales en el uso de sistemas de control y las ventajas de trabajar con ellos, ya que los sistemas digitales son más eficaces, rápidos, seguros y confiables, todo esto comparado con los sistemas analógicos.

Así mismo, se mostrará la importancia de la simulación, ya que por medio de esta se puede observar los comportamientos de los sistemas, de esta manera se pueden realizar mejorar o se puede encontrar fallas en algún sistema.

1.4 Metodología de la Investigación

Para la alcanzar el objetivo de esta tesis se realizo una investigación detallada sobre la simulación y modelados de sistemas digitales, se estudio detalladamente el Algebra de Boole y la elaboración de Mapas de Karnaugh, también se hizo un estudio detallado sobre lógica secuencial y el Diseño de Sistemas Digitales.

Posteriormente, se estudio a detalle el software 20-SIM y sus herramientas con las cuales se hará la simulación de nuestros sistemas digitales. Por último, se realizará la revisión y el análisis de los resultados obtenidos.

1.5 Contenido de la Tesis

En el capítulo 1 se mostrará una pequeña introducción de lo que es la simulación de sistemas.

En el capítulo 2 se describe lo que son los sistemas digitales, se explicará el uso del algebra booleana que es la herramienta con la cual se llevará a cabo este trabajo de tesis, también de describirá lo que son las compuertas lógicas y sus funciones. También se hablará de los Flip Flop y los contadores.

En el capítulo 3 se describirá el funcionamiento del programa de simulación 20-Sim, las herramientas y los componentes que se utilizarán en este trabajo de tesis. También se describirá los procesos para realizar las simulaciones.

En el capítulo 4 se llevará a cabo la simulación y el modelado de una alarma, se realizará la simulación y el modelado de un control de un sistema aljibe-tinaco, y se modelará y simulará un contador tanto síncrono como asíncrono.

En el capítulo 5 se mostraran las conclusiones a las que se llego y las recomendaciones en este trabajo de tesis.

Capítulo 2

Modelado de Sistemas Digitales

2.1 Introducción

Un **sistema digital** es un conjunto de dispositivos destinados a la generación, transmisión, procesamiento o almacenamiento de señales digitales. También un sistema digital es una combinación de dispositivos diseñados para manipular cantidades físicas o información que estén representadas en forma digital; es decir, que sólo puedan tomar valores discretos.

Para el análisis y la síntesis de sistemas digitales binarios se utiliza como herramienta el álgebra de Boole.

- **Sistemas Digitales Combinacionales**: Aquellos en los que sus salidas sólo depende del estado de sus entradas en un momento dado. Por lo tanto, no necesita módulos de memoria, ya que las salidas no dependen de los estados previos de las entradas.
- **Sistemas Digitales Secuenciales**: Aquellos en los que sus salidas dependen además del estado de sus entradas en un momento dado, de estados previos. Esta clase de sistemas necesitan elementos de memoria que recojan la información de la 'historia pasada' del sistema.

Para la implementación de los circuitos digitales, se utilizan puertas lógicas (AND, OR y NOT), construidas generalmente a partir de transistores. Estas puertas siguen el comportamiento de algunas funciones del Álgebra de Boole.

Según el propósito de los sistemas digitales se clasifican en: a) sistemas de propósitos especiales y b) sistemas de propósitos generales. Estos últimos permiten el cambio de su comportamiento mediante la programación de algoritmos de soluciones de problemas específicos.

2.2 Algebra de Boole

El algebra de Boole como cualquier otro sistema matemático deductivo puede ser definida por un conjunto de elementos, un conjunto de operadores, un número de axiomas o postulados. El Álgebra de Boole, fue presentada originalmente por el inglés George Boole,

en el año de 1854, sin embargo, las primeras aplicaciones a circuitos de conmutación fueron desarrolladas por Claude Shannon hasta 1938.

El Algebra Booleana es un sistema algebraico definido en un conjunto **S**, el cual contiene dos o más elementos y entre los cuales se definen dos operaciones denominadas suma u operación OR (+) ,y producto o multiplicación u operación AND (*), las cuales cumplen con los siguientes postulados:

Postulado 1. Conjunto Cerrado. Un conjunto **S** es cerrado con respecto a un operador binario, si para cada par de elementos de **S**, el operador binario especifica una regla para obtener un elemento único de **S**.

Postulado 2. Ley Asociativa. Se dice que un operador binario * en un conjunto S es asociativo si:

$$(x*y)*z = x*(y*z) \qquad \text{para toda } x, y, z \in S$$

Postulado 3. Ley Conmutativa. Se dice que un operador binario * en un conjunto S es conmutativo si:

$$x*y = y*x \qquad \text{para todo } x, y \in S$$

Postulado 4. Elemento de Identidad. Que dice que un conjunto s tiene un elemento de identidad con respecto a la operación binaria * en S si existe un elemento e $\in$ S, con la propiedad:

$$e*x = xe = x \qquad \text{para toda } x \in e$$

Postulado 5. Inverso. Se dice que un conjunto S, que tiene un elemento de identidad e con respecto a un operador binario *, tiene un inverso si para cada x $\in$ S existe un elemento y $\in$ S tal que:

$$x*y = e$$

Postulado 6. Ley Distributiva. Si * y · son dos operadores binarios en un conjunto S, se dice que * es distributivo con respecto a · si:

$$x*(y \cdot z) = (x*y) \cdot (x*z)$$

El álgebra booleana difiere de manera importante del álgebra ordinaria en que las constantes y variables booleanas sólo pueden tener 2 valores posibles, 0 y 1. Una variable booleana es una cantidad que puede, en diferentes ocasiones, ser igual a 0 o a 1. Las variables booleanas se emplean con frecuencia para representar el nivel de voltaje representado en un alambre o en las terminales de entrada y salida de un circuito.

Ya que sólo puede haber dos valores, el álgebra booleana es relativamente fácil de manejar en comparación con el algebra ordinaria. En el algebra Booleana no existen fracciones, decimales, números negativos, raíces cuadradas, raíces cubicas, logaritmos, números imaginarios, etc. De hecho en el algebra Booleana sólo existen 3 operaciones básicas: OR, AND y NOT.

Estas operaciones básicas se llaman operaciones lógicas. Es posible construir circuitos digitales llamados compuertas lógicas que con diodos, transistores y resistencias conectados de cierta manera hacen que en la salida del circuito sea el resultado de una operación lógica básica (AND, OR, NOT) sobre el circuito.

2.3 Compuertas Lógicas

Una compuerta lógica, es un dispositivo electrónico que es la expresión física de un operador booleano en la lógica de conmutación. Cada puerta lógica consiste en una red de dispositivos interruptores que cumple las condiciones booleanas para el operador particular.

Claude Elwood Shannon experimentaba con relés o interruptores electromagnéticos para conseguir las condiciones de cada compuerta lógica, por ejemplo, para la función booleana AND colocaba interruptores en circuito serie, ya que con uno solo de éstos que tuviera la condición «abierto», la salida de la compuerta Y sería = 0, mientras que para la implementación de una compuerta O (OR), la conexión de los interruptores tiene una configuración en circuito paralelo.

2.3.1 Operación OR

Suponga que A y B representan dos variables lógicas independientes. Cuando A y B se combinan con la operación OR, el resultado, x, se puede expresar como:

$$x = A + B$$

En esta expresión el signo + no representa la adición ordinaria; en su lugar denota la operación OR cuyas reglas se dan en la Tabla 2.1.

Tabla 2.1. Tabla de verdad que define
la operación OR.

A	B	x=A+B
0	0	0
0	1	1
1	0	1
1	1	1

Al observar la tabla de verdad se advertirá que, donde A = B = 1 la suma OR es 1 y no 2 como en la adición ordinaria. Esto resulta fácil de recordar si observamos que sólo 0 y 1 son valores posibles en el algebra Booleana, de modo que el valor mayor que se puede obtener es 1. Este mismo resultado se obtiene si tenemos x = A + B + C, en el caso en el que A = B = C = 1.

2.3.2 Compuerta OR

En un circuito digital la compuerta OR es un Circuito que tiene dos o más entradas y cuya salida es igual a la suma OR de las entradas. En la Figura 2.1 se muestra la tabla de verdad y el símbolo correspondiente a una compuerta OR de 2 entradas.

A	B	x = A + B
0	0	0
0	1	1
1	0	1
1	1	1

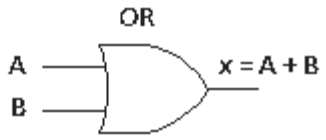


Figura 2.1. Tabla de verdad y símbolo correspondiente
de una compuerta OR de 2 entradas.

La entrada A y B son niveles de voltaje lógico cuyo valor es el resultado de la operación OR de A y B; esto es x = A + B. En otras palabras, La compuerta OR opera de las forma que su salida es ALTA (nivel lógico 1) si al entrada A, B o ambas están en un nivel lógico 1. La salida de la compuerta OR será BAJA (nivel lógico 0) si todas sus entradas están en el nivel lógico 0.

2.3.3 Operación AND

La operación AND es la segunda operación básica Booleana. Si dos variables lógicas A y B se combinan mediante la expresión AND, el resultado, x, se puede expresar como:

$$x = A \cdot B$$

En esta expresión el signo · representa la operación Booleana AND y no la multiplicación. Sin embargo, la operación AND en variables booleanas opera igual que la multiplicación común. La Tabla 2.2 muestra que sucede cuando 2 entradas lógicas A y B se combinan usando la operación AND para producir la salida x.

Tabla 2.2. Tabla de verdad de la compuerta AND.

A	B	X = A · B
0	0	0
0	1	0
1	0	0
1	1	1

En la tabla se muestra que x es 1 lógico sólo cuando A y B están en el nivel lógico 1. Para cualquier caso en que la entrada es 0, la salida es 0.

2.3.4 Compuerta AND

En la figura 2.2 se muestra la tabla de verdad y el símbolo lógico de una compuerta AND de 2 entradas. La salida de la compuerta AND es igual al producto AND de las entradas lógicas; es decir, x = AB. En otras palabras, la compuerta AND es un circuito que opera de tal forma que su salida es ALTA sólo cuando todas su entradas son ALTAS. Para los otros casis la salida de la compuerta AND es BAJA.

A	B	X = A · B
0	0	0
0	1	0
1	0	0
1	1	1

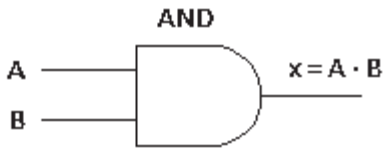


Figura 2.2. Tabla de verdad y símbolo correspondiente
de una compuerta AND de 2 entradas.

Esta misma operación es característica de las compuertas AND con más de 2 entradas. Por ejemplo la figura 2.3 se muestra una compuerta de 3 entradas y su respectiva tabla de verdad.

A	B	C	x = ABC
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

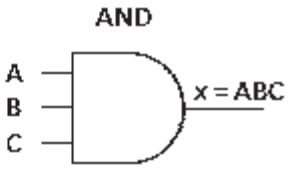


Figura 2.3. Tabla de verdad y símbolo correspondiente
para una compuerta AND de 3 entradas.

De nuevo observe que la salida de la compuerta es 1 sólo para el caso en donde A = B = C = 1. La expresión para la salida es x= ABC, en el caso de una compuerta AND de 4 entradas la salida es x = ABCD y así sucesivamente.

2.3.5 Operación NOT (INVERSOR)

La operación NOT (también llamada INVERSOR) difiere de las operaciones OR y AND en que se pueden realizar en una sola variable de entrada. Por ejemplo, si la variable A se somete a la operación NOT, el resultado x se puede expresar como:

$$x = \bar{A}$$

Donde la barra sobrepuesta representa la operación NOT. Eso indica que el valor lógico de $x = \bar{A}$ es igual al valor opuesto de A. La tabla 2.3 aclara esto para los dos casos A = 0 y A = 1.

Tabla 2.3. Tabla de verdad para una ccompuerta NOT

A	x=A
0	1
1	0

A la operación NOT también se le denomina inversión o complementación y es importante mencionar que otro indicador de inversión es el símbolo primo (‘).

2.3.6 Compuerta NOT

La compuerta NOT siempre tiene una sola entrada y su nivel lógico de salida invariablemente es opuesto al nivel lógico de esta entrada. En la figura 2.4 se muestra la tabla de verdad y el símbolo correspondiente de una compuerta NOT.

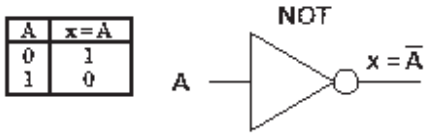


Figura 2.4. Tabla de verdad y símbolo correspondiente
Para una compuerta NOT.

En la tabla anterior podemos observar como el inversor afecta una señal de entrada. Invierte completamente la señal de entrada; así, cuando la entrada es igual = 0 la salida = 1, y viceversa.

2.4 Otro Tipo de Compuertas

En los circuitos digitales se utilizan ampliamente dos tipos más de compuertas lógicas: NOR y NAND. Estas compuertas en realidad combinan las operaciones básicas AND, OR y NOT, por lo que es relativamente simple escribir sus expresiones booleanas.

2.4.1 Compuerta NOR

La compuerta NOR opera como una compuerta OR seguida por un INVERSOR. En la figura 2.5(a) se muestra el símbolo de una compuerta NOR. Es igual al símbolo de la compuerta OR, excepto que tiene un círculo a la salida el cual representa la operación de inversión. De esta manera el circuito de la figura 2.5(a) y 2.5 (b) son equivalentes y la expresión de salida para la compuerta NOR es $X = \overline{A + B}$.

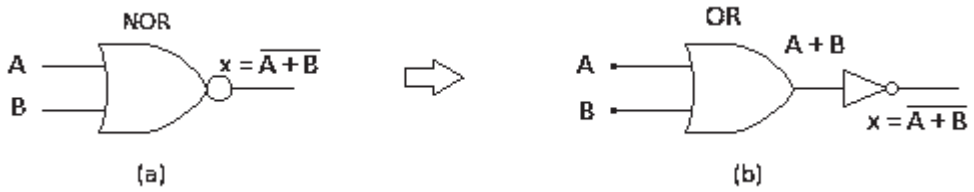


Figura 2.5. (a) Símbolo NOR,
(b) equivalente NOR.

La tabla 2.4 indica que la salida de la compuerta NOR es exactamente el inverso de la salida de la salida de la compuerta OR para todas las condiciones de entrada posible.

Tabla 2.4. Tabla de verdad para una compuerta NOR.

		OR	NOR
A	B	$x = A + B$	$x = \overline{A + B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

Una salida de compuerta OR para a ALTO cuando cualquier entrada es ALTA, la salida de la compuerta NOR para a BAJA cuando cualquier entrada es ALTA, esta misma operación se puede extender a compuertas NOR con más de dos entradas.

2.4.2 Compuerta NAND

La compuerta NAND opera como una compuerta AND seguida por un INVERSOR. En la figura 2.6(a) se muestra el símbolo de una compuerta NAND. Es igual al símbolo de la compuerta AND, excepto que tiene un círculo a la salida, de nuevo este pequeño círculo denota la operación de inversión. De esta manera el circuito de la figura 2.6(a) y 2.6 (b) son equivalentes y la expresión de salida para la compuerta NAND es $X = \overline{A \cdot B}$.

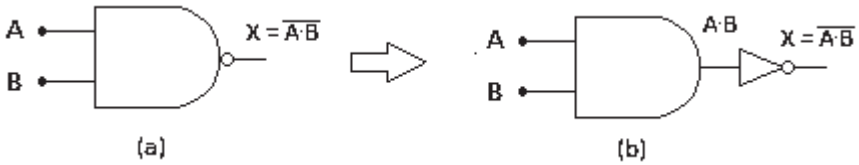


Figura 2.6. (a) Símbolo NAND
(b) equivalente NAND.

La tabla 2.5 muestra que la salida de la compuerta NAND es la inversa exacta de la compuerta AND en todas las posibles condiciones de entrada. La salida AND se vuelve ALTA solo cuando todas las entadas son ALTAS, en tanto que la salida NAND se vuelve BAJA solo cuando todas las entradas son ALTAS. Esta misma característica se aplica en las compuertas NAD qué tienen más de dos entradas.

		AND	NAND
A	B	AB	$\overline{AB}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Tabla 2.5. Tabla de verdad para
Una compuerta NAND

2.5 Diseño de Sistemas

Cualquier circuito sin importar que tan complejo sea, puede ser completamente descrito mediante el uso de las tres operaciones básicas Booleanas, ya que la compuerta OR, la compuerta AND y el circuito NOT son los bloques de construcción básica de los sistemas digitales. Por ejemplo considere el circuito de la Figura 2.7, este tiene 3 entradas A, B y C, y una sola salida, x. Utilizando la expresión booleana para cada compuerta, se puede expresar fácilmente la expresión para la salida.

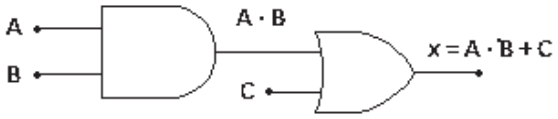


Figura 2.7. Circuito lógico con su expresión booleana.

La expresión para la salida de la compuerta AND se escribe A · B. Esta salida AND está conectada como una entrada a la compuerta OR junto con C, otra entrada. La compuerta OR opera sobre sus entradas de manera que su salida es la suma OR de las entradas. Así se puede expresar la salida OR como x = A · B + C (Esta expresión final también se podría escribir como x = C + A · B, puesto que no importa cual termino de la suma OR se escriba primero).

En ocasiones puede haber confusión con respecto a cual operación se realiza primero en una expresión. La expresión A · B + C se puede interpretar de dos formas (1) A B opera con C, o bien (2) A opera con AND con el termino B + C. Para evitar esta confusión, se entenderá si una expresión contiene ambas operación AND y OR, las operaciones AND se realizan primero a menos que existan paréntesis en la expresión, en cuyo caso la expresión dentro del paréntesis se llevará a cabo primero. Esta es la misma regla que se usa en el algebra común para determinar el orden de las operaciones.

Para ilustrar esto más ampliamente consideré el circuito de la Figura 2.8. La expresión para la salida de la compuerta OR es simplemente A + B. Esta salida sirve como una entrada para la compuerta AND junto con otra entrada, C. De esta manera, la salida de compuerta AND se expresa como x = (A + B) C. Observe el uso de paréntesis aquí para indicar que A y B operan primero con OR, antes que su suma OR realice la operación AND con C. Sin el paréntesis se interpretaría incorrectamente, puesto que A + B · C significa que A se opera con OR con el producto B · C.

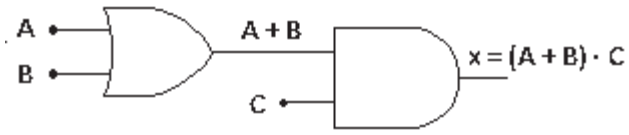


Figura 2.8. Circuito lógico cuya expresión requiere paréntesis.

Siempre que una expresión esté presente en un diagrama de un circuito lógico, su expresión de salida será simplemente igual a la expresión de entrada con una barra sobre ella. En la Figura 2.9 se muestra un ejemplo usando INVERSOR. Observamos que la entrada A se alimenta a través de un INVERSOR, cuya salida, por lo tanto, es $\bar{A}$. La salida del INVERSOR se alimenta a una compuerta OR, junto con B, de manera que la salida OR es igual a $\bar{A}$ + B. Observe que la barra solo está sobre A, lo que indica que A se invierte primero y luego se hace la operación de OR con B.

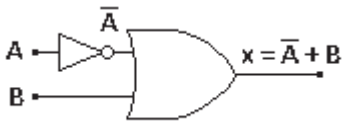


Figura 2.9. Circuito lógico que utiliza INVERSOR en una entrada.

En la Figura 2.10, la salida es igual a A + B y se alimenta a través de un INVERSOR. Por lo tanto la salida del INVERSOR es igual a $(\overline{A + B})$, puesto que invierte la expresión de entrada completa. Note que la barra cubre toda la expresión (A + B). Esto es importante porque, como se demostrará más adelante, las expresiones $(\overline{A + B})$ y $(\bar{A} + \bar{B})$ no son equivalentes. La expresión $(\overline{A + B})$ significa que opera con OR con B y luego se invierte su suma OR, en tanto que la expresión $(\bar{A} + \bar{B})$ indica que A se invierte y B se invierte, y luego ambos resultados se operan con OR.

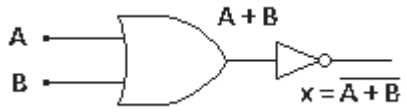


Figura 2.10. Circuito lógico que utiliza INVERSOR en la salida.

Cuando la operación de un circuito se define mediante una operación booleana se puede dibujar un diagrama de un circuito lógico de manera directa a partir de esa expresión. Por ejemplo, si necesitamos un circuito que fuese definido mediante $x = A \cdot B \cdot C$, de inmediato sabemos que requerimos una compuerta AND de 3 entradas. Si necesitáramos un circuito que estuviera definido por $x = A + \overline{B}$, usaríamos una compuerta OR de dos entradas con un inversor en una de las entradas. El mismo racionamiento que se usa en estos casos simples se puede aplicar a circuitos más complejos.

Suponga que deseamos construir un circuito cuya salida sea $y = AC + B\overline{C} + \overline{A}BC$. Esta expresión booleana contiene tres términos (AC, $B\overline{C}$, $\overline{A}BC$) los cuales están operados con OR. Lo anterior indica que se requiere una compuerta OR de tres entradas iguales a AC, BC y ABC. La Figura 2.11 ilustra este caso, que representa una compuerta OR de tres entradas etiquetadas como AC, $B\overline{C}$ y $\overline{A}BC$.

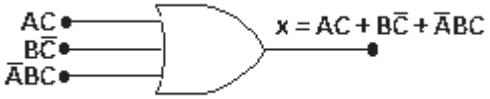


Figura 2.11. Compuerta OR de 3 entradas.

Cada compuerta OR es un término del producto AND, lo cual significa que se puede usar una entrada AND con entradas propias para generar cada uno de estos términos, lo anterior se ejemplifica en la Figura 2.12, en la que se muestra el diagrama final del circuito. Observe el número de INVERSORES para producir los términos $\overline{A}$ y $\overline{C}$ que se requieren en la expresión.

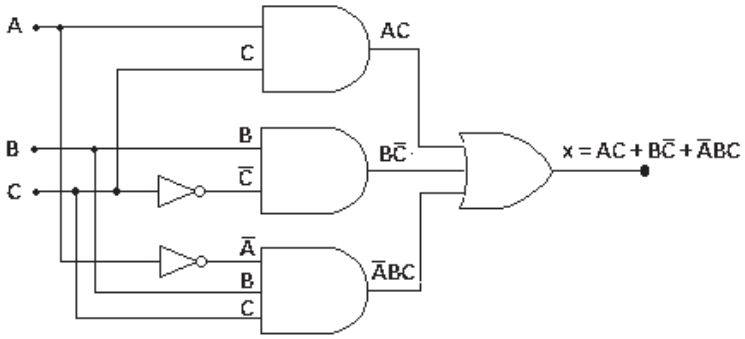


Figura 2.12. Construcción de un circuito lógico a partir de una expresión booleana.

Este mismo planteamiento general se puede seguir siempre, aunque como se verá existen algunas técnicas más ingeniosas y eficientes.

Por ejemplo, veamos el siguiente problema en el que se diseña un circuito para una maquina despachadora de refresco:

Una máquina despachadora de refrescos de lata que cuesta $4 pesos recibe monedas de $1 y de $2 pesos. La máquina recibe las monedas en ranuras distintas que se colocan en forma aleatoria. En cada casilla cabe sólo una moneda, hay 4 casillas para monedas de $1 peso y 2 casillas para monedas de $2 pesos. Diseñe el circuito que hace funcionar esta máquina.

Solución

Casos Posibles	Función
Caso 1. 4 monedas de $1	F = abcd
Caso 2. 1 moneda de $2 y 2 de $1	F = e{ab,ac,ad,bc,bd,cd} y F = f{ ab,ac,ad,bc,bd,cd }
Caso 3. 2 monedas de $2	F = ef

Así la función final queda:

F = abcd + eab + eac + ead + ebc + ebd + ecd + fab + fac + fad + fbc + fbd + fcd + ef

Podemos observar de la función anterior que al momento de realizar el circuito equivalente, tendremos un circuito demasiado grande ya que implica agregar bastantes compuertas AND y compuertas OR.

También podemos diseñar circuitos a partir de una tabla de verdad, por ejemplo, considere la Tabla 2.6 en la cual indica que la salida x será 1 en tres casos diferentes: A=0 B=1 C=0, A=0 B=1 C=1 y A=1 B=1 C=1. ¿Cómo se puede llevar a cabo esto? Sabemos que los termino AND A·B·C genera 1 sólo para la condición A=0 B=1 C=0, el termino AND A·B·C genera un 1 para la condición A=0 B=1 C=1 y el termino AND A·B·C genera un 1 para la condición A=1 B=1 C=1.

Tabla 2.6. Diseño de circuito lógico combinacional.

A	B	C	x	
0	0	0	0	
0	0	1	0	
0	1	0	1	→ $\bar{A}B\bar{C}$
0	1	1	1	→ $\bar{A}BC$
1	0	0	0	
1	0	1	0	
1	1	0	0	
1	1	1	1	→ ABC

Como x debe ser ALTA para cualquier condición, resulta claro que estos términos se deben operar con OR para producir la salida deseada: x. Note que en cada caso en que la variable sea 0, aparece invertida en el termino AND. La expresión de la suma de productos para x se obtiene operando con OR los tres términos AND.

$$x = \overline{A}B\overline{C} + \overline{A}BC + ABC$$

Una vez que la expresión de salida ha sido determinada a partir de la tabla de verdad en término de suma de productos, se puede implementar fácilmente usando compuertas AND, OR e INVERSORES. El circuito para esta función se muestra en la Figura 2.13 en donde podemos ver como se utiliza la compuerta AND en combinación con la compuerta OR y los INVERSORES.

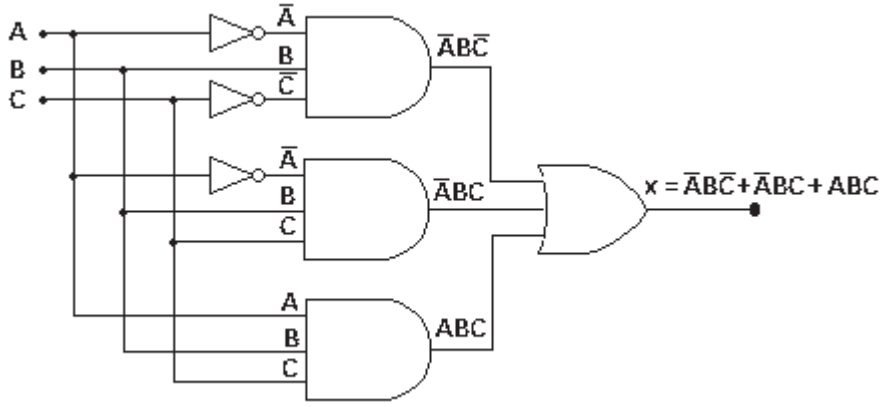


Figura 2.13. La salida de la compuerta AND se opera
Con OR para producir la salida final.

Sin embargo, por lo general la expresión de salida se puede simplificar y por ende resultar en un circuito más eficiente.

2.6 Mapas de Karnaugh

Una vez obtenida la expresión para un circuito lógico, podemos reducirla a una forma más simple que contenga menos términos, o menos variables en uno o más términos .La nueva expresión se puede usar para implementar un circuito que sea equivalente al circuito original, pero que tenga menos compuertas y conexiones.

Las funciones de Booleana pueden ser simplificadas por medio algebraico, sin embargo el procedimiento de minimización es un tanto raro ya que carece de reglas específicas para predecir cada paso sucesivo en el proceso de manipulación. El método de mapa presenta un

procedimiento simple y directo para minimizar las funciones Booleana. El método de mapas, propuesto primeramente por Veitch y modificado por Karnaugh se conoce como el diagrama de Veitch o el Mapa de Karnaugh.

El mapa es un diagrama hecho de cuadros. Cada cuadro determina un término mínimo. Como cualquier función de Booleana puede ser expresada como una suma de términos mínimos, se desprende que dicha función, se reconoce gráficamente en el mapa a partir del área encerrada por aquellos cuadros cuyos términos medios se incluyen en la función. De hecho el mapa presenta un diagrama visual de todas las formas vivibles en las que puede ser representada una función en la forma normalizada. Al reconocer varios patrones, el usuario puede derivar expresiones algebraicas alternas para la misma función de las cuales se puede escoger la más simple. Se asume que la expresión algebraica más simple es cualquiera en una suma de productos o producto de suma que tiene mismo número de literales.

Los Mapas de Karnaugh son una herramienta muy utilizada para la simplificación de circuitos lógicos. Cuando se tiene una función lógica con su tabla de verdad y se desea implementar esa función de la manera más económica posible se utiliza este método.

2.6.1 Formato Del Método De Karnaugh

El mapa de Karnaugh al igual que en una tabla de verdad, es un medio para mostrar la relación entre entradas lógicas y salida deseada. En la Figura 2.14 se muestran tres ejemplos de mapas de Karnaugh para dos, tres y cuatro variables junto con las tablas de verdad correspondientes. Mediante estos ejemplos se ilustran los siguientes puntos importantes:

1.- La tabla de verdad proporciona el valor de la salida X para cada combinación de valores de entrada. El mapa de Karnaugh proporciona la misma información en un formato diferente. Cada caso en la tabla de verdad corresponde a una celda en el mapa de Karnaugh. Por ejemplo, en la Figura 2.14(a) la condición A = 0, B = 0 en la tabla de verdad corresponde a la celda $\overline{A}\overline{B}$ en el mapa de Karnaugh. Como la tabla de verdad muestra que X = 1 para este caso, se coloca un 1 en la celda $\overline{A}\overline{B}$ del mapa de Karnaugh. De manera similar, la condición A = 1, B = 1 en la tabla de verdad correspondiente a la celda AB del mapa de Karnaugh. Debido a que X = 1 para este caso, se coloca un 1 en la celda AB. Las demás celdas se llenan con ceros. Esta misma idea se usa en los mapas de 3 y 4 variables.

2.- Las celdas del mapa de Karnaugh se marcan de tal manera que las celdas horizontales adyacentes difieren sólo en una variable. Por ejemplo, la celda superior izquierda en el mapa de cuatro variables es $\overline{A}\overline{B}\overline{C}\overline{D}$, en tanto que la celda inmediatamente a su derecha es $\overline{A}\overline{B}\overline{C}D$ (sólo la variable D es diferente). De manera similar, las celdas verticalmente

adyacentes sólo difieren en una sola variable. Por ejemplo, la celda superior izquierda es $\overline{A}\overline{B}\overline{C}\overline{D}$, en tanto que la celda directamente abajo es $A\overline{B}\overline{C}\overline{D}$ (sólo la variable B es distinta).

Note que cada celda en la fila superior se considera adyacente a una celda correspondiente en la celda inferior. Por ejemplo, la celda $\overline{A}\overline{B}CD$ en la fila superior adyacente a la celda $A\overline{B}CD$ en la fila inferior, puesto que solo difieren en la variable A. Usted podría considerar que la parte superior del mapa ha sido enrollada para que toque la parte inferior. De manera similar, las celdas en la columna de la extrema izquierda son adyacentes a la celda de la columna extrema derecha.

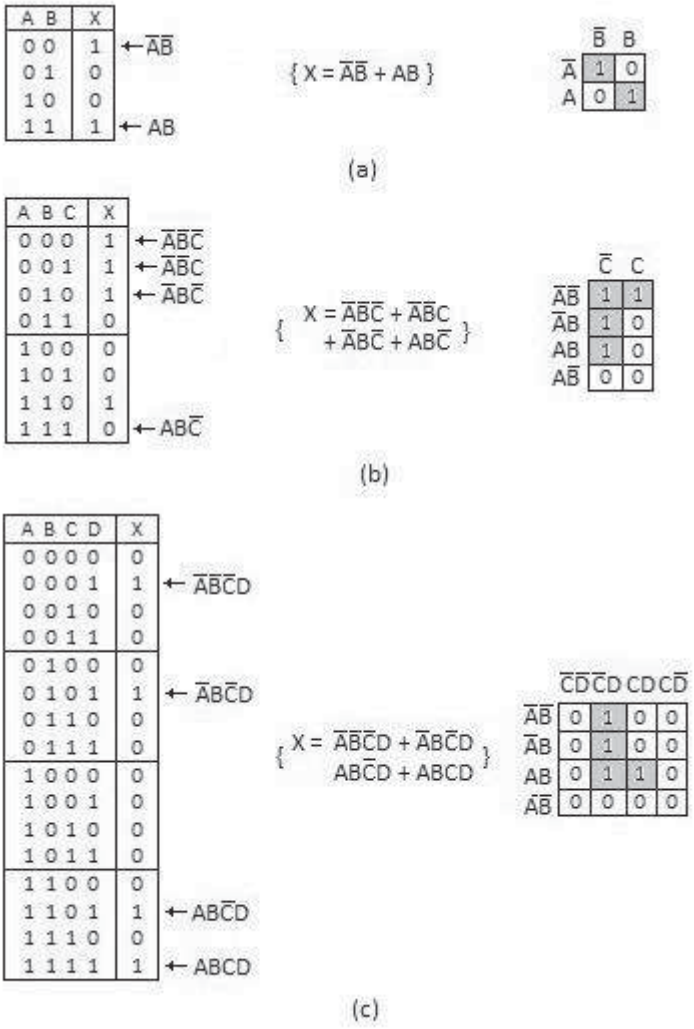


Figura 2.14. Mapa De Karnaugh y tabla de verdad para
(a) dos, (b) tres y (c) cuatro variables.

3.- A fin de que la celda vertical y horizontalmente adyacente difieran sólo en una variable, la marcación de arriba hacia abajo se debe de hacer en el orden que se muestra: $\overline{A}\overline{B}$, $\overline{A}B$, AB, $A\overline{B}$. Lo mismo es cierto para la marcación de izquierda a derecha: $\overline{C}\overline{D}$, $\overline{C}D$, CD, $C\overline{D}$.

4.- Una vez que el mapa de Karnaugh se ha llenado con unos y ceros se puede obtener la suma de productos para la salida X, operando con OR las celdas que contengan un 1. En el mapa de tres variables de la figura 2.14(b) las celdas $\overline{A}\overline{B}\overline{C}$, $\overline{A}B\overline{C}$, $\overline{A}B\overline{C}$ y $AB\overline{C}$ contienen un 1, de manera que $X = \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} + \overline{A}B\overline{C} + AB\overline{C}$.

2.6.2 Agrupamiento

La expresión para la salida X se puede simplificar cambiando adecuadamente en el mapa de Karnaugh estas celdas que contengan unos. El proceso para combinar estos unos se llama agrupamiento.

2.6.2.1 Agrupamiento de Grupos de Dos (Pares)

La Figura 2.15(a) es el mapa de Karnaugh para una tabla de tres variables. En este mapa contienen un par de unos que son verticalmente adyacentes entre sí; el primero representa $\overline{A}B\overline{C}$ y el segundo $AB\overline{C}$. Note que en estos dos términos la variable A aparece tanto en forma normal como complementada (invertida). En tanto que B y $\overline{C}$ aparecen sin cambio. Estos términos se pueden agrupar (combinar) para dar una resultante que elimina la variable A, puesto que aparece tanto en forma sin complementar como complementadas. Lo anterior se demuestra fácilmente así:

$$\begin{aligned} X &= \overline{A}B\overline{C} + AB\overline{C} \\ &= B\overline{C}\,(\overline{A} + A) \\ &= B\overline{C}\,(1) = B\overline{C} \end{aligned}$$

Este mismo principio es válido para cualquier par de unos vertical y horizontalmente adyacentes. En la figura 2.15(b) se muestra un ejemplo de dos unos horizontalmente adyacentes. Estos se pueden agrupar y eliminar la variable C, ya que aparecen tanto en formas sin complementar como complementadas para dar una resultante de $X = A\overline{B}$.

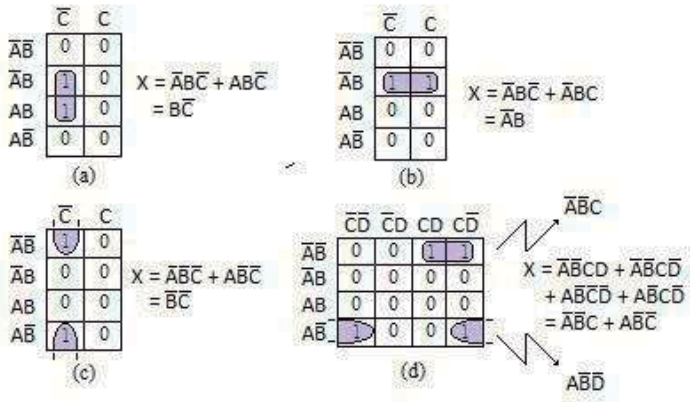


Figura 2.15. Ejemplo de agrupamiento de pares de unos adyacentes

La Figura 2.15(c) se muestra otro ejemplo. En un mapa de Karnaugh la fila de celda superior e inferior se consideran adyacentes. Así, los 2 unos en este mapa se pueden agrupar para proporcionar una resultante de $\overline{A}\overline{B}\overline{C} + A\overline{B}\overline{C} = \overline{B}\overline{C}$.

En la figura 2.15(d) se muestra un mapa de Karnaugh con dos pares de unos que se pueden agrupar. Los dos unos de la fila superior están horizontalmente adyacentes; los dos unos de la fila inferior también están adyacentes, puesto que en un mapa de Karnaugh la columna a la extrema izquierda y la columna a la extrema derecha de las celdas se consideran adyacentes. Cuando el par superior de uno se agrupa se elimina la variable D (puesto que aparece como D y $\overline{D}$), para dar el termino $\overline{A}\overline{B}C$. Agrupando el par inferior se elimina la variable C para dar el término $A\overline{B}\overline{D}$. Ambos términos se operan con OR y dan el resultado final para X.

En resumen: El agrupamiento da un par de unos adyacente en un mapa de Karnaugh, elimina la variable que aparece en forma complementada y sin complementar.

2.6.2.2 Agrupamiento de Grupos de Cuatro (Cuádruples)

Un mapa de Karnaugh puede contener un grupo de cuatro unos que sean adyacentes entre sí. Este grupo se llama cuádruple. En la Figura 2.16 se muestran varios ejemplos de cuádruples. En la parte (a) los cuatro unos son verticalmente adyacentes, y en la parte (b) son horizontalmente adyacentes. El mapa de Karnaugh en la Figura 2.16(c) contiene cuatro unos en una celda y se consideran adyacentes entre sí. Los cuatro unos en la figura 2.16(d) también son adyacentes como los de la figura 2.16(e) porque, como se indico antes, la fila superior e inferior se consideran adyacentes entre si, al igual que las columnas de la extrema izquierda y la extrema derecha.

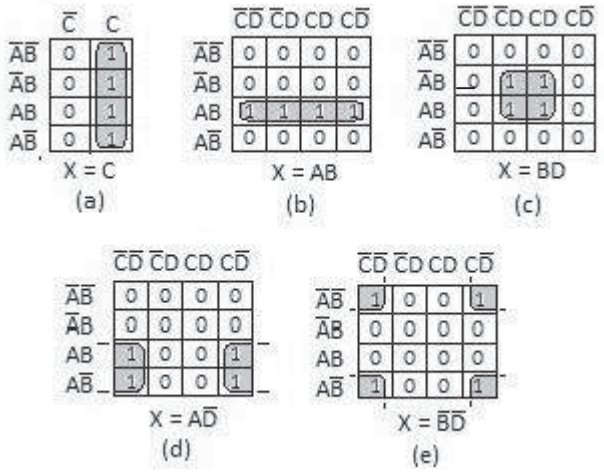


Figura 2.16. Ejemplos sobre agrupamientos de conjuntos de cuatro unos (cuádruples)

Cuando se agrupa un cuádruple el tiempo resultante tendrá sólo las variables que no cambian para toda las celdas en el cuádruplo. Por ejemplo, en la figura 4.16(a) las 4 celdas que contienen un 1 son $\overline{A}\overline{B}C$, $\overline{A}BC$, ABC y $A\overline{B}C$. El análisis de estos términos determina que sólo la variable C permanece sin cambios (tanto A como B aparecen en forma tanto complementada y sin complementar). La expresión que resulta para X es simplemente X = C. Esto se puede demostrar como sigue:

$$\begin{aligned} X &= \overline{A}\overline{B}C + \overline{A}BC + ABC + A\overline{B}C \\ &= \overline{A}C\,(B + \overline{B}) + AC(B + \overline{B}) \\ &= \overline{A}C + AC \\ &= C(\overline{A} + A) = C \end{aligned}$$

A manera de ejemplo, considere la figura 2.16(d) en la cual los cuatro cuadrados que contienen unos son $AB\overline{C}\overline{D}$, $A\overline{B}\overline{C}\overline{D}$, $AB\overline{C}D$ y $A\overline{B}\overline{C}D$. El análisis de estos términos determina que solo las variables A y $\overline{D}$ permanecen sin cambio. De manera que la expresión simplificada para X es:

$$X = A\overline{D}$$

En resumen: el agrupamiento de un cuádruplo de unos adyacentes elimina las dos variables que aparecen tanto en forma complementada como sin complementar.

2.6.2.3 Agrupamientos de Grupos de Ocho (Octetos)

Un grupo de ocho unos que son adyacentes entre si se llaman octetos. En la Figura 2.17 se muestran otros ejemplos de octetos. Cuando se agrupa un octeto en un grupo de cuatro variables se eliminan tres de estas porque solo una variable permanece sin cambio, por ejemplo, el análisis de las ocho celdas agrupadas en la Figura 2.17(a) muestra que sólo la variable B está en la misma forma para las ocho celdas: las otras variables aparecen en forma complementada y sin complementar. Así, para este mapa, X = B. En las figuras 2.17(b), 2.17(c) y 2.17(d) las variables $\overline{C}$, $\overline{B}$ y $\overline{D}$ están en la misma forma para las 8 celdas respectivamente.

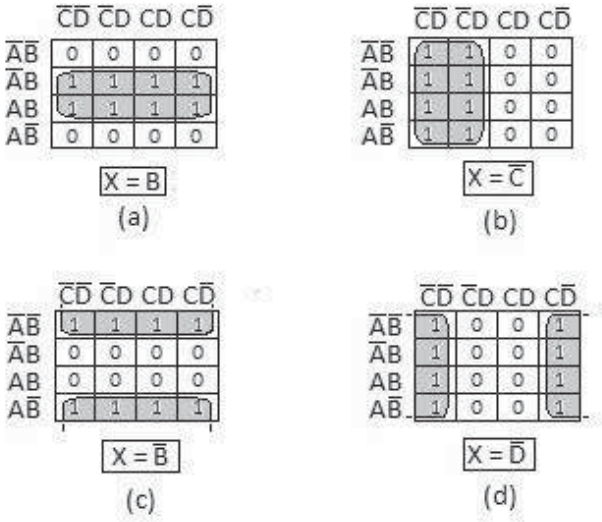


Figura 2.17. Ejemplo sobre agrupamiento de conjuntos de 8 unos (Octetos)

En resumen: el agrupamiento para un octeto de unos adyacentes elimina las tres variables que aparecen tanto en forma complementada como sin complementar.

2.6.3 Proceso Completo de Simplificación

Hemos visto como se puede usar el agrupamiento de pares, cuádruples y octetos en un mapa de Karnaugh para obtener una expresión simplificada. Podemos resumir la regla para agrupamientos de cualquier tamaño:

Cuando aparece una variable en forma complementada o sin complementar dentro de un agrupamiento, esa variable se elimina de la expresión. Las variables que son iguales en todas las celdas del agrupamiento deben aparecer en la expresión final.

Debe ser claro que un grupo mayor de unos elimina más variables. Para ser exacto, un agrupamiento de dos elimina una variable, un agrupamiento de cuatro elimina dos y uno de ocho elimina tres. Ahora se usara este principio para obtener una expresión lógica simplificada, a partir de un mapa de Karnaugh que contiene cualquier combinación de unos y ceros.

Los pasos que se indican a continuación son los que se siguen al usar el método del mapa de Karnaugh para simplificar una expresión Booleana:

1.- Se construye el mapa de Karnaugh y se colocan los unos en las celdas correspondientes a los unos en la tabla de verdad. Los ceros se colocan en las otras celdas.

2.- Se examina el mapa para ver si hay unos adyacentes y se agrupan los que no son adyacentes a cualquier otro uno, a estos se les llama unos aislados.

3.- Enseguida, se buscan los unos que sean adyacentes a sólo otro uno. Se agrupa cualquier par que contenga un uno en estas condiciones.

4.- Se agrupa cualquier octeto incluso si contienen algunos unos que hayan sido agrupados.

5.- Se agrupa cualquier cuádruple que contenga uno o más unos que no hayan sido agrupados, asegurando de usar el numero mínimo de agrupamientos.

6.- Se agrupan otros pares necesarios para incluir cualesquiera unos que no hayan sido ya agrupados, asegurándose de que se use el menor número de agrupamientos.

7.- La suma OR se forma de todos los términos que género cualquier agrupamiento.

Estos pasos se seguirán con exactitud. En cada caso, la expresión lógica resultante estará en su forma más simple de suma de productos.

2.7 Lógica Secuencial

Los circuitos lógicos que se han considerado hasta el momento son circuitos combinacionales o combinatorios, cuyos niveles de salida, en cualquier instante, dependen de los niveles presentes en las entradas en ese momento. En la figura 2.18 se muestra un diagrama general de un sistema combinacional. Cualquier condición anterior al nivel de entrada no afecta a las salidas, porque no poseen memoria.

Figura 2.18. Diagrama general de un sistema combinacional

Los sistemas combinacionales no permiten, por si solos, resolver el problema de almacenar el estado de las entradas en un instante dado y poder utilizarlo para tomar decisiones posteriormente, cuando cambie el estado de aquellas.

Sí al sistema combinacional se le agrega retroalimentación el sistema se convierte en secuencial, como lo sugiere la figura 2.19.

Figura 2.19. Diagrama general de un sistema secuencial

La retroalimentación puede estar constituida por:

a) El tiempo de propagación de las compuertas que forman la parte combinacional.
b) Celdas secuenciales básicas con una sola variable de salida.

Los sistemas secuenciales son capaces de memorizar el estado de las entradas y convertirlo en un estado interno1 del propio sistema. Así, el valor de la salida en un instante determinado no depende solamente del estado de las entradas en dicho instante, sino también del estado interno. Un sistema secuencial es un sistema automático que recibe el nombre de autómata finito debido a que posee un número finito de estados internos.

La celda secuencial básica es el Flip-Flop, que está formado por un ensamble de compuertas lógicas. Aunque una compuerta lógica, por sí misma, no tiene la capacidad de almacenamiento, pueden conectarse varias de ellas de manera que permitan almacenar información. Existen varias configuraciones de compuertas que se utilizan para producir estos Flip-Flops.

2.7.1 Flip Flops

Los Flip-Flop son circuitos electrónicos que están compuestos mediante compuertas lógicas y que sirven como elementos de memoria en circuitos combinacionales. La Figura 2.20 muestra un diagrama de bloques de un sistema digital general que combina compuertas lógicas combinacionales con un sistema de memoria.

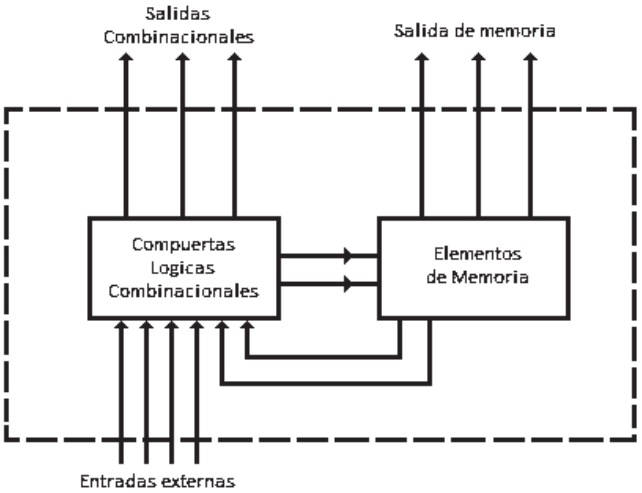


Figura 2.20. Diagrama general de un sistema digital.

Aunque una compuerta lógica, por si misma, no tiene la capacidad de almacenar, se pueden conectar varias a la vez de tal manera que permitan el almacenamiento de la información. Existen varias combinaciones de compuertas que se utilizan para producir estos circuitos llamados Flip-Flops.

La Figura 2.21(a) muestra el símbolo general empleado para un Flip-Flop. El símbolo indica que el Flip-Flop tiene dos salidas, marcadas como Q y $\overline{Q}$, que son inversas entre sí. La salida Q recibe el nombre de salida normal, mientras que $\overline{Q}$ es la salida negada o invertida del Flip-Flop. Claro está que la salida $\overline{Q}$ es el inverso de la entrada Q.

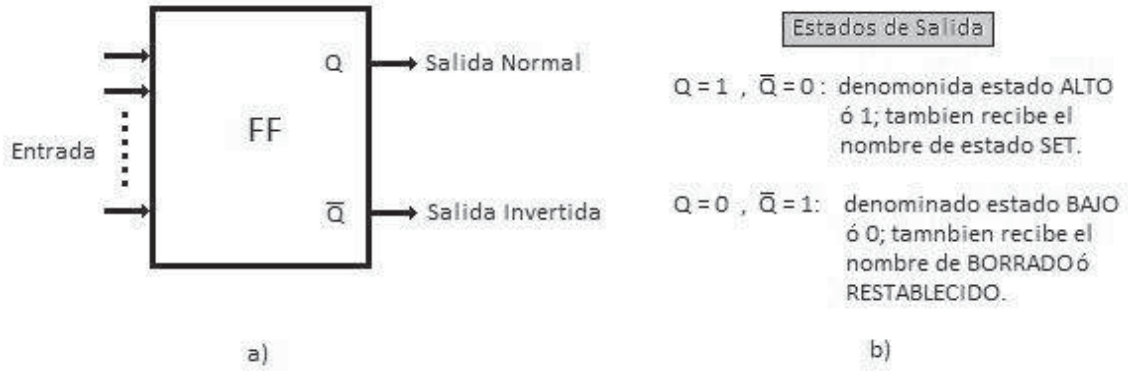
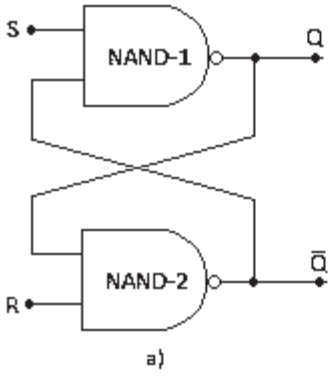


Figura 2.21. (a) Símbolo general para un Flip-Flop
(b) Estados de salida de un Flip-Flop

En la figura 2.21(b) se muestran los 2 estados de operación posibles para el Flip Flop. Observe que al estado ALTO o 1 (Q =1, Q = 0) también se le denomina estado ESTABLECIDO. Siempre que las entradas para un Flip-Flop hagan que este pase al estado Q=1, le llamaremos SET (establecer) el Flip Flop; el Flip Flop ha sido establecido. De manera similar, al estado BAJO ó 0 (Q=0, Q=1) también se le llama estado BORRADO ó RESTABLECIDO. Siempre que las entradas para un Flip Flop hagan que pase al estado Q=0, se le llama BORRADO ó RESTABLECIDO del Flip Flop; el Flip Flop se ha borrado (restablecido). Como se verá, muchos Flip Flop tendrán una entrada SET (establecer) ó una entrada BORRAR (CLEAR) que se usa para exitar el Flip Flop a un estado de salida especifico.

Podemos observar en la figura 2.21(a) que los Flip Flop puede tener una o más entradas, dichas entradas se usar para ocasionar que el Flip Flop cambie hacia atrás y hacia adelante, es decir, bascule ("Flip Flop") entre sus estados de salida posible. Descubriremos que la mayoría de las entradas de Flip Flop sólo necesitan estar activadas momentáneamente (pulsadas) a fin de causar un cambio en el estado de salida del Flip Flop y la salida permanecerá en ese mismo estado, incluso después que el pulso de entrada haya terminado, esta es la característica de memoria del Flip Flop.

Como se dijo anteriormente, existen diferentes arreglos para construir un Flip Flop, pero la combinación más elemental es a base de un arreglo de dos compuertas NAND también llamado LATCH. En la Figura 2.22(a) se muestra el arreglo de las compuertas NAND para formar un Flip-Flop.



SET	CLEAR	SALIDA
1	1	Sin cambio
0	1	Q = 1
1	0	Q = 0
0	0	*Invalida

* produce Q = $\overline{Q}$ = 1

b)

Figura 2.22. (a) Flip-Flop con compuertas NAND.
(b) Tabla de verdad

Las dos compuertas NAND están marcadas de tal forma que la salida de las NAND-1 sea la entrada de la NAND-2, y viceversa. Las salidas de las compuertas, marcadas como Q y $\overline{Q}$, respectivamente, son las salidas del registro básico. Bajo condiciones normales, estas salidas serán siempre inversas una de la otra. Hay dos registros básicos o cierres de entradas: la entrada SET que establece a Q al estado 1; la entrada RESET que es la que manda Q al estado 0.

Las entradas SET y CLEAR normalmente están en estado ALTO y una de ellas será pulsada a BAJO cuando se quiere cambiar el estado de salida del registro básico. Hay dos estados de salida igualmente probables cuando las entradas SET = CLEAR = 1. Una posibilidad es cuando se tiene Q = 0 y Q =1. Con Q = 0, las entradas de NAND-2 son 0 y 1, mismas que producen Q = 1. El uno de Q ocasiona que NAND-1 tenga un 1 en ambas entradas a fin de producir una salida 0 en Q. En efecto, lo que se tiene es el estado bajo en la salida NAND-1, que produce un nivel ALTO en la salida NAND-2, lo que a su vez conserva la salida NAND-1 en estado BAJO.

La segunda posibilidad es cuando Q = 1 y Q = 0. El estado ALTO de NAND-1 produce un estado BAJO en la salida NAND-2, que a su vez conserva la salida AND-1 en estado ALTO. Así, hay dos posibles estados de salida cuando SET = CLEAR = 1, pero este estado dependerá de lo que haya ocurrido anteriormente en la entrada.

En la Figura 2.22(b) muestra la tabla de verdad del comportamiento de un Flip-Flop y se resume de la siguiente manera:

1.- SET = CLEAR = 1. Esta condición es el estado normal de reposo y no tiene efecto en el estado de salida. Las salidas Q y Q permanecerán en el estado que tenían antes de esta condición de entrada.

2.- SET = 0, CLEAR = 1. Esto siempre causará que la salida pase al estado Q = 1, en donde permanecerá incluso después de que SET retorne a ALTO. A esto se le llama establecimiento del LATCH.

3.- SET = 1, CLEAR = 0. Esto siempre producirá el estado Q = 0, en el cual la salida permanecerá incluso después de que la entrada CLEAR retorne a ALTO. A esto se le llama establecimiento o restablecimiento del LATCH.

4.- SET = CLEAR = 0. Esta condición intenta establecer y borrar el LATCH y puede producir resultados ambiguos. No se puede emplear.

2.7.1.1 Flip-Flop Sincronizados Por Reloj

Los sistemas digitales pueden funcionar de forma asíncrona o síncrona. En los sistemas asíncronos, la salida de los circuitos lógicos pueden cambiar de estado en el momento en que varíen una o más de las entradas. En un sistema asíncrono por lo general es más difícil diseñar y detectar las fallas que en un sistema síncrono.

En los sistemas síncronos, los tiempos exactos en que cualquier salida puede cambiar de estado se determinan mediante una señal comúnmente llamada reloj (CLK). Esta señal de reloj por lo general es un tren de pulsos rectangulares o una onda cuadrada, como se muestra en la Figura 2.23. La señal de reloj se distribuye a todas las partes del sistema, y la mayoría de las salidas del sistema (si no es que todas) puede cambiar de estado sólo cuando el reloj hace una transición. Las transiciones también son llamada borde Cuando un reloj cambia de 0 a 1 se le llama Transición con Pendiente Positiva (TPP o ↑) y cuando cambia de un 1 a 0 se le llama Transición con Pendiente Negativo (TPN o ↓).

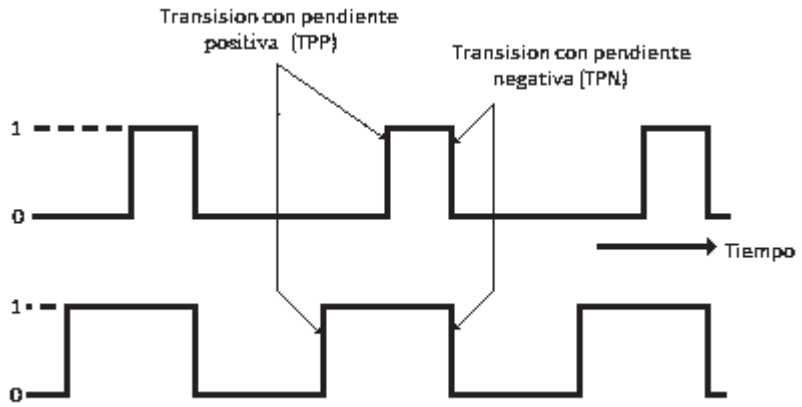
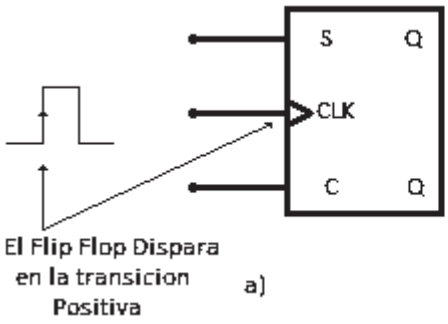


Figura 2.23. Señales de Reloj.

La sincronización de las señales de reloj se logra mediante el uso de Flip Flops sincronizados por reloj que se diseñan para cambiar estados en una u otra de las transiciones del reloj.

2.7.1.2 Flip-Flop SC

En la Figura 2.24(a) se muestra el símbolo lógico para un Flip-Flop SC sincronizado por reloj, el cuál se dispara por el borde de transición positiva de la señal de reloj. Esto significa que el Flip Flop puede cambiar estados solo cuando una señal aplicada a su entrada de reloj hace una transición de 0 a 1.



Entradas			Salida
S	C	CLK	Q
0	0	↑	Q_0 (Sin Cambio)
1	0	↑	1
0	1	↑	0
1	1	↑	Ambigua

Q_0 es el nivel de salida de ↑ (FTT) de CLK
↓ (TPN) de CLK no produce cambio en Q

b)

Figura 2.24. (a) Símbolo de un Flip-Flop SC.
(b) Tabla de verdad de un Flip-Flop SC.

La tabla de verdad de la figura 2.24(b) muestra como la salida del Flip-Flop responde ante la transición positiva de la señal en la entrada CLK para las diversas combinaciones de las entradas S y C.

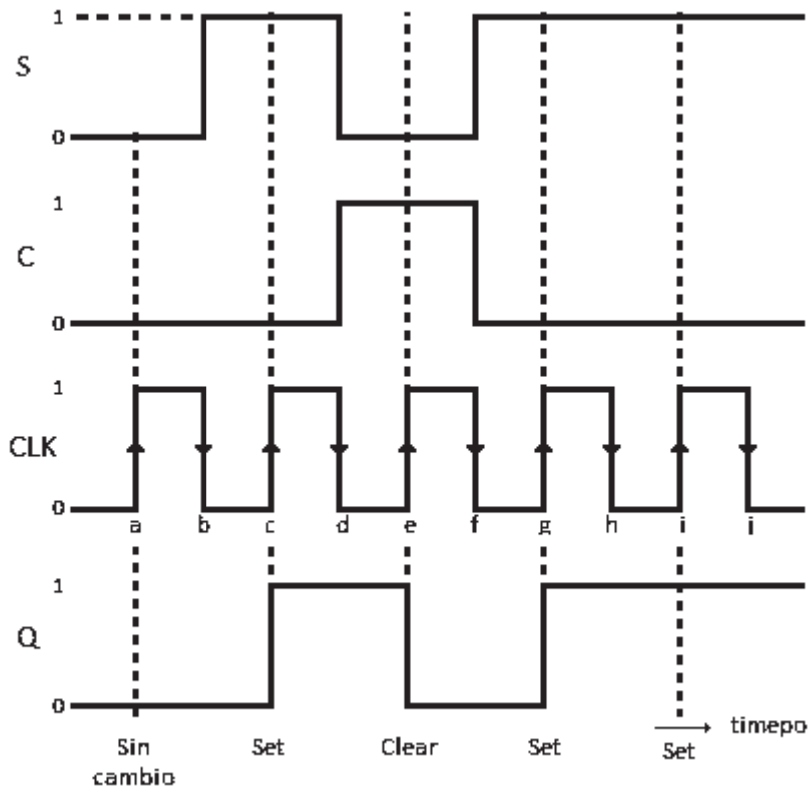


Figura 2.25. Forma de Onda de un Flip Flop SC

Las formas de onda de la figura 2.25 muestran la operación del Flip Flop SC sincronizado por reloj. Si se supone que los requisitos de establecimiento y retención se cumplen en todos los casos, estas formas de onda se pueden analizar como sigue:

1.- Inicialmente todas las entradas son 0 y se supone que la salida Q es 0, es decir, $Q_0 = 0$.

2.- Cuando ocurre la TTP del primer pulso de reloj (punto a), las entradas S y C son 0, por lo tanto el Flip Flop no se afecta y permanece en el estado Q = 0 (es decir $Q = Q_0$).

3.- Cuando ocurre la TTP del segundo pulso de reloj (punto c) la entrada S ahora es alta, con C aun baja. Así, el Flip Flop se fija al estado 1 en el borde ascendente (transición con pendiente positiva) del pulso del reloj.

4.- Cuando el tercer pulso de reloj hace su transición positiva (punto e) encuentra que S = 0 y C = 1, lo que ocasiona que el Flip Flop se borre al estado 0.

5.- El curto pulso establece el Flip Flop de nuevo al estado Q = 1 (punto p) porque S = 1 y C = 0 cuando ocurre la transición positiva.

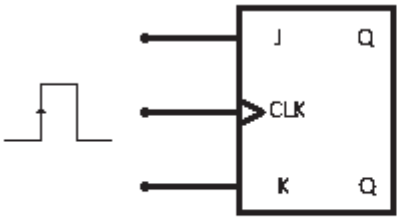
6.- El quinto pulso también encuentra que S = 1 y C = 0 cuando hace su transición de pendiente positiva. Sin embargo, Q ya es alta y por lo tanto permanece en ese estado.

7.- La condición S = C = 1 no se debe usar porque da como resultado una combinación ambigua.

Se debe notar a partir de estas formas de onda que el Flip Flop no resulta afectado por la transición de pendiente positiva de los pulsos de reloj. Observe que los niveles S y C no tienen efecto en el Flip Flop, excepto cuando ocurre una transición de pendiente positiva de la señal de reloj. Las entradas S y C son entradas síncronas de control, controlaran el estado al que pasara el Flip Flop cuando ocurra el pulso de reloj; la entrada CLK es la entrada de disparo que causa que el Flip Flop cambie de estado de acuerdo a como estén las entradas S y C cuando ocurra la transision activa del reloj.

2.7.1.3 Flip Flop JK

En la figura 2.26(a) se muestra un Flip Flop JK sincronizado por el reloj que se dispara por el borde de pendiente positiva de la señal del reloj. Las entradas J y K controlan el estado del flip flop de la misma manera que las entradas S y C lo hacen para el flip flop sincronizado por el reloj, excepto por una diferencia importante: la condicion J = K = 1 no resulta a una salida ambigua. Para esta condicion 1,1, el flip flop siempre pasará a su lado opuesto cuando se lleve a cabo la transición positiva de la señal del reloj. A esto se le llama operación modo de cambio de estado, esto lo podemos ver en la tabla de verdad de la figura 2.26(b).



J	K	CLK	Q
0	0	↑	Q_0 (Sin cambio)
1	0	↑	1
0	1	↑	0
1	1	↑	Q (Cambia estado)

a) b)

Figura 2.26. (a) Simbolo de un Flip Flop JK
(b) Tabla de verdad de un Flip Flop JK.

En este modo, si J y K se dejan en alto, el flip flop cambiará estados (conmutará) para cada trasición con pendiente positiva del reloj.

La tabla de verdad de la figura 2.25(b) resume como responde el flip flop JK a la transición con pendiente positiva para cada combinacion de cada JK. Observe que la tabla de verdad es la misma que para el flip flop SC sinbcronizado por el reloj, excepto para la condición de J = K = 1. Esta condición resulta en $Q = Q_{0,}$ lo cual significa que el nuevo

valor de Q sera el inverso del valor que tenía antes de la transición con pendiente positiva; está es la operación cambio de estado.

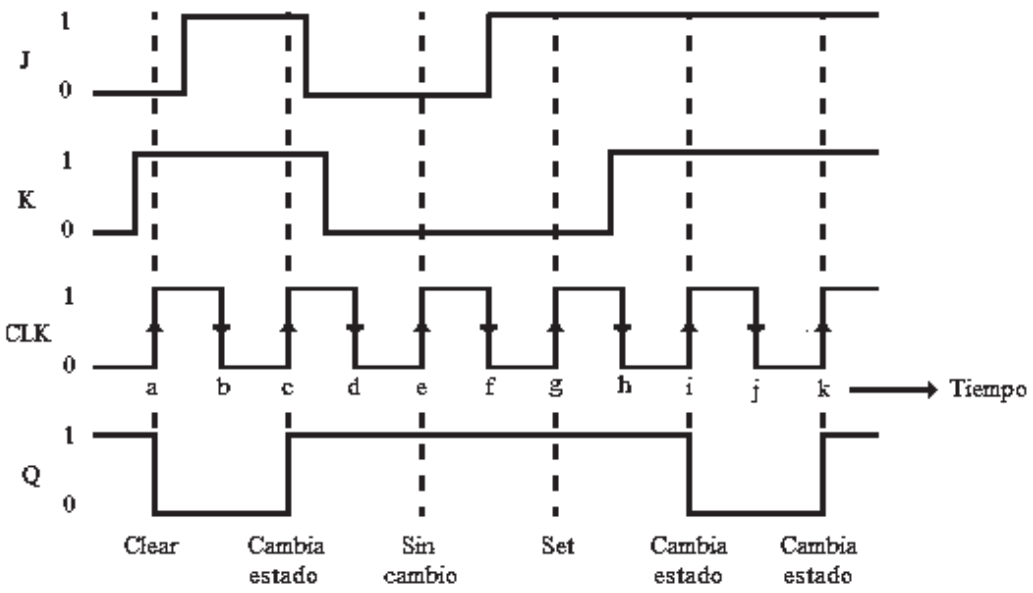


Figura 2.27. Formas de Onda del Flip Flop JK

La operación de este flip flop se ilustra mediante las formas de onda de la figura 2.27. Una vez mas se supone que se cumplen los requisitos de establecimiento y retencion:

1.- Inicialmente todas las entradas son cero y se supone que la salida Q es 1; es decir, Q_0=1.

2.- Cuando ocurre el borde con pendiete positiva del primer pulso de reloj (punta a), existe la condición J = 0, K = 1. Asi, el flip flop se limpiara al estado Q = 1.

3.- El segundo pulso de reloj encuentra que J = K = 1 cuando hace su transición positiva (punto c). Esto causa que el flip flop cambie a su estado opuesto, Q = 1.

4.- En el punto e en la forma de onda del reloj J y K son cero, de manera que el flip flop no cambia estados en esta transision.

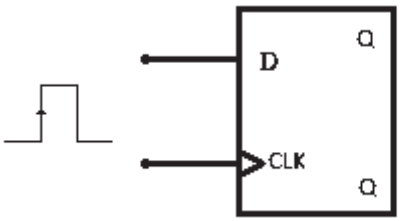
5.- En el punto g, j = 1 y k = 0, esta es la condición que establece Q al estado 1. Sin embargo Q ya es uno y por lo tanto permanecerá en ese estado.

6.- En el punto i, J = K = 1 y por lo tanto el flip flop cambia a su estado opuesto. Lo mismo ocurre en el punto k.

A partir de esta variación observe que el flip flop no se afecta por el borde de pendiente negativa de los pulsos de reloj. Asimismo, los niveles de entrada J y K no tienen efecto, excepto cuando ocurre una transición con pendiente negativa de la señal de reloj. Las entradas J y K por si mismas no pueden ocasionar que el flip flop cambie estado.

2.7.1.4 Flip Flop D

En la figura 2.28 se muestra el símbolo y la tabla de verdad para un flip flop D sincronizado por reloj que dispara en una transición con pendiente positiva. A diferencia de los flip flop SC y JK, este flip flop sólo tienen una entrada síncrona de control, D, que significa datos. La operación del flip flop es muy simple: Q pasa al mismo estado que este presente en la entrada D cuando ocurra una transision con pendiente positiva en CLK. En otras palabras, el nivel presente en D se almacenará en el flip flop en el instante en el que ocurre la transición con pendiente positiva. La formas de onda de la figura 2.29 ilustran esta operación.



J	CLK	Q
0	↑	0
1	↑	1

Figura 2.28. Simbolo y tabla de verdad de un Flip Flop D.

Suponga que inicialmente Q es alta. Cuando ocurre la primera trasnsición con pendiente positiva en el punto a, la entrada D esta en bajo; de esta manera, Q pasará al estado 0. Aunque el nivel en la entrada D cambie, cambia entre los puntos a y b, no tienen efecto en Q; Q almacena el nivel bajo que estaba en D en el punto a. Cuando ocurre la transición con pendiente positiva en b, Q pasa a alto puesto que d esta en alto en ese momento. Q almacena estado alto hasta que la transición con pendiente negativa en el punto c ocasiona que Q pase a baja puesto que D esta en bajo en ese momento. De manera similar, la salida Q adopta los niveles presentes en D cuando ocurre la transición con pendiente posit va en los pundos d, e, f y g. Observe que Q permanece en alto en el punto e, porque D aún está en alto.

De nuevo, es importante recordar que Q sólo puede cambiar cuando ocurre una transición con pendiente negativa. La entrada D no tienen efecto entre las transiciónes con pendiente positiva.

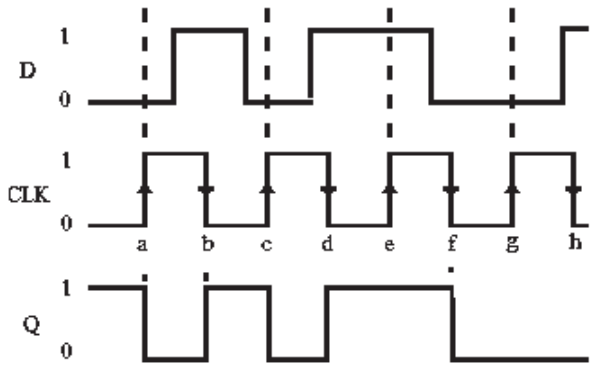


Figura 2.29. Forma de onda de un Flip Flop D.

2.7.2 Contadores

Un **contador** es un circuito secuencial construido a partir de Flip Flops y compuertas lógicas capaces de realizar el cómputo de los impulsos que recibe en la entrada destinada a tal efecto, almacenar datos o actuar como divisor de frecuencia. Habitualmente, el cómputo se realiza en un código binario, que con frecuencia será el binario natural o el BCD natural (contador de décadas).

Los contadores pueden ser clasificados en diferentes tipos, a continuación se muestra una pequeña clasificación de estos dispositivos:

1.- Según la forma en que conmutan los Flip Flop, podemos hablar de contadores síncronos (todos los Flip Flop conmutan a la vez con una señal de reloj común), o asíncronos (el reloj no es común y los Flip Flop conmutan uno tras otro).

2.- Según el sentido de la cuenta, se distinguen en ascendentes, descendentes y UP-DOWN (ascendentes o descendentes según la señal de control).

3.- Según la cantidad de números que pueden contar, se puede hablar de contadores binarios de n bits (cuentan todos los números posibles de n bits, desde 0 hasta $2^n - 1$), contadores BCD (cuentan del 0 al 9) y contadores Módulo N (cuentan desde el 0 hasta el N-cuarto).

El número máximo de estados por los que pasa un contador se denomina módulo del contador. Este número viene determinado por la expresión 2^n donde n indica el número de bits del contador. Ejemplo, un contador de módulo 4 pasa por 4 estados, y contaría del 0 al 3. Si necesitamos un contador con un módulo distinto de 2^n, lo que haremos es añadir un circuito combinacional.

En este capítulo solo mencionaremos la primera clasificación que es la base principal de los contadores.

2.7.2.1 Contadores Asíncronos

El termino asíncrono se refiere a los sucesos que no poseen una relación temporal fija entre ellos y que, generalmente, no ocurren al mismo tiempo. Un contador asíncrono es aquel en que los Flip Flops del contador no cambian de estado exactamente al mismo tiempo, dado que no comparten el mismo impulso de reloj. En la figura 2.30 se muestra un circuito contador asíncrono de 4 bits y su forma de onda.

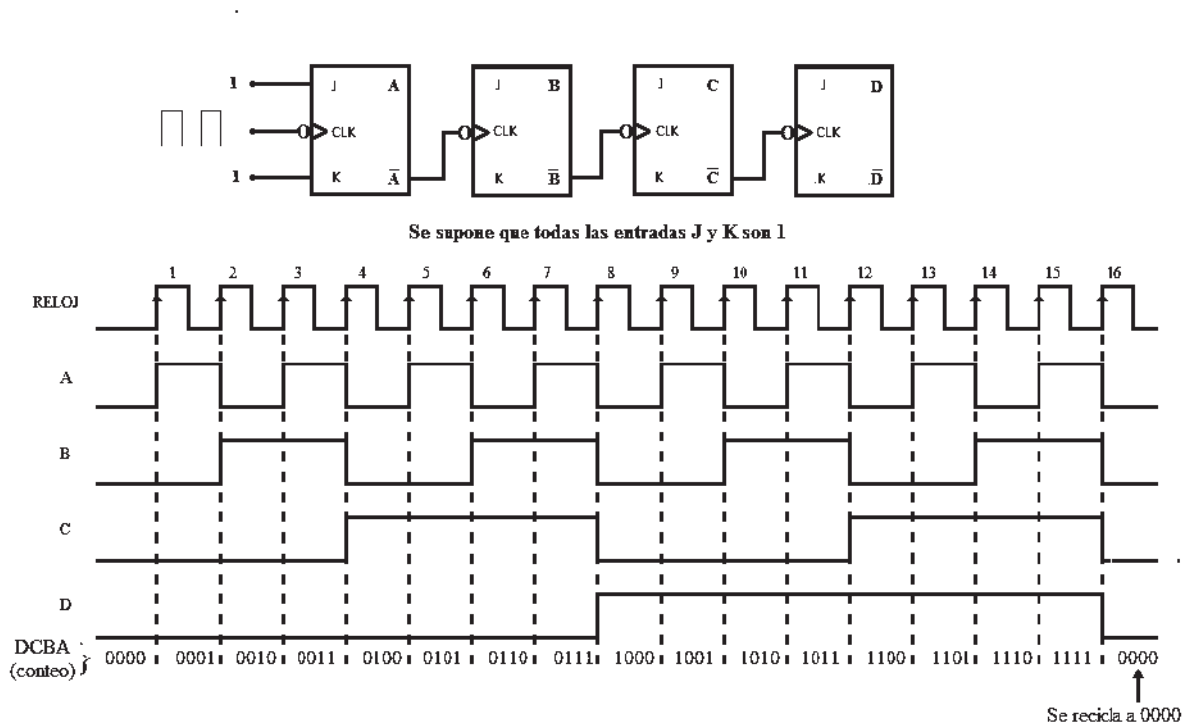


Figura 2.30. Contador asíncrono de 4 BITS y forma de onda.

A continuación se muestran los siguientes puntos con respecto a su operación:

1. Los pulsos de reloj se aplican sólo a la entrada CLK del Flip-Flop A. Así, el Flip-Flop A cambiará (a su estado opuesto) cada vez que los pulsos de reloj hagan una transición positiva (de BAJO a ALTO). Note que J=K=1 para todos los Flip Flops.

2.- La salida negada de Flip Flop A actúa como la entrada CLK para el Flip Flop B, y por lo tanto, el Flip Flop B cambiará de estado cada vez que la salida $\overline{A}$ pase de 0 a 1. De manera similar, el Flip Flop C cambiara de estado cuando $\overline{B}$ pase de 0 a 1, y el Flip Flop D cambiara de estado cuando $\overline{C}$ pase de 0 a 1.

3.- Las salidas A, B, C y D del Flip Flop representan un número binario de 4 bits, con D como el MSB. Supongamos que todos los Flip Flops se han borrado al estado 0 (las entradas CLEAR no se muestran). Las formas de onda de la figura 2.30 muestran que una secuencia de conteo 0000 a 1111 continúa a medida que se aplican continuamente los pulsos de reloj.

4.- Después que haya ocurrido la TPN del decimoquinto pulso de reloj, los Flip Flop del contador se encuentran en la condición 1111. En el decimosexto TPN, el Flip Flop A pasa de 1 a 0, lo que ocasiona que el Flip Flop B pase de uno a cero, ect., hasta que el contador este en el estado 0000. En otras palabras, el contador ha pasado por un ciclo completo (0000 a 1111) y se ha reciclado nuevamente a 0000, desde donde iniciará un nuevo ciclo de conteo a medida que se apliquen los siguientes pulsos de reloj.

En este contador cada salida negada del Flip Flop excita la entrada CLK del siguiente Flip Flop. Este tipo de configuración de contador se llama Contador Asíncrono porque los Flip Flops no cambian estado en sincronía exacta con los pulsos del reloj aplicados; solo el Flip Flop al responde a los pulsos del reloj. El Flip Flop B debe esperar a que el Flip Flop A cambie estados antes que él lo pueda hacer; el Flip Flop C debe esperar a que el Flip Flop B cambie, etc. De esta manera hay un retardo entre las respuestas sucesivas de los Flip Flop. Este retardo comúnmente es del orden de 5 a 20 ns por Flip Flop. Como se verá en algunos casos este retardo puede representar un problema. Con frecuencia a este tipo de contador también se le denomina contador de rizo, debido a la forma a que los Flip Flop responden uno después de otro a una especie de efecto ondulatorio.

En los esquemas de circuitos existe la convención de dibujar los circuitos (cuando esto es posible) de modo que el flujo de la señal corra de izquierda a derecha con entrada a la izquierda y salida a la derecha. Por ejemplo, en la Figura 2.30 las entradas CLK de cada Flip Flop se encuentran a la izquierda, las salidas de la derecha y las señales de entrada provienen de la izquierda. Emplearemos esta configuración en el que el Flip Flop A (el cual es el LSB) es el Flip Flop en el extremo izquierdo, y el Flip Flop D (el cual es el MSB) es el Flip Flop en el extremo derecho, esto para apegáramos a la convención de flujo de la señal de izquierda a derecha. También se podría utilizar la configuración en la que el Flip Flop A se pone a la derecha y el Flip Flop D a la izquierda, lo que indica que el orden de los Flip Flops es el mismo que el orden de los Bits.

2.7.2.2 Contadores Síncronos

Los problemas de los contadores asíncronos se debe a los problemas de retardo de propagación acumulados de los Flip Flops; dicho de otra manera, los Flip Flop no cambian de estado simultáneamente en sincronía con los pulsos de entrada. Esta limitación se puede superar con la implementación de contadores síncronos o en paralelo en los cuales todos los Flip Flops se disparan simultáneamente (en paralelo) mediante los pulsos de entrada de reloj. Como los pulsos de entrada se aplican a todos los Flip Flops, se debe emplear un medio para controlar cuando debe de cambiar de estado un Flip Flop y cuando debe permanecer sin cambio ante un pulso de reloj. Esto también se puede lograr empleando las entradas J y K de los Flip Flops.

El termino síncrono se refiere a los eventos que tienen una relación temporal fija entre sí. Con respecto al funcionamiento del contador, síncrono significa que todos los Flip Flops del contador reciben en el mismo instante la señal del reloj. En la figura 2.31 se muestra un contador síncrono de 4 BITS.

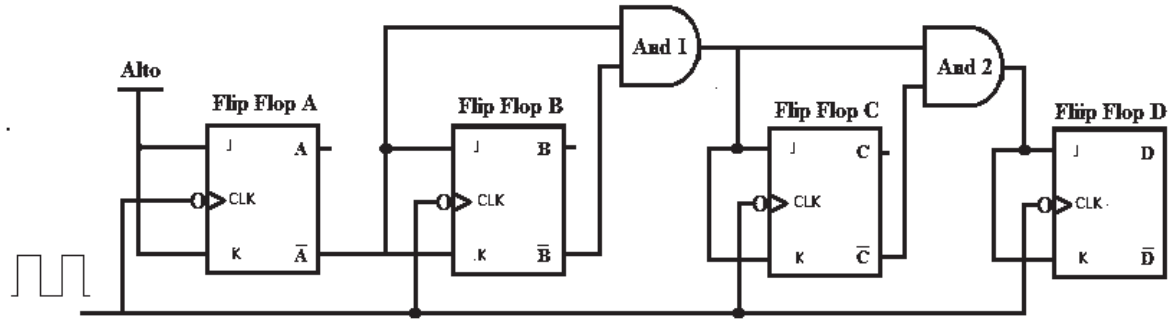


Figura 2.31. Contador síncrono de 4 BITS.

En la figura 2.32 se muestra las formas de onda de un contador síncrono de 4 BITS.

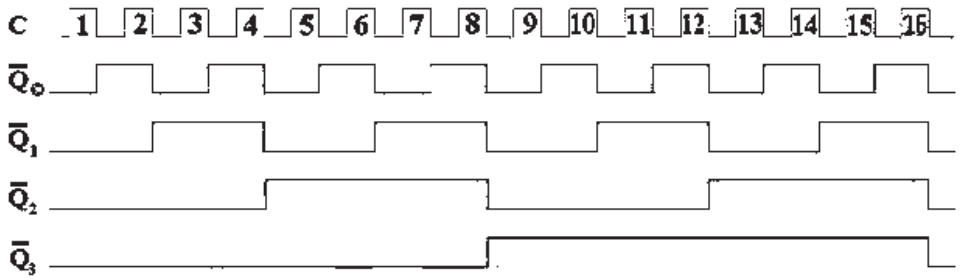


Figura 2.32 Formas de onda de un
contador síncrono de 4 BITS

Si comparamos la configuración del circuito para este contador síncrono con su contraparte asíncrona de la figura 2.30, podemos observar las siguientes diferencias notables:

1.- La entrada CLK de todos los Flip Flops están conectadas entre sí, de modo que la señal de entrada de reloj se aplica a cada Flip Flop de manera simultánea.

2.- Solo en Flip Flop A, el LBS, tienen su entrada J y K tienen su entrada permanente en el nivel ALTO. Las entradas J y K de los otros Flip Flop se excitan por alguna combinación de las salidas de los Flip Flop.

3.- El contador síncrono requiere más circuitería que el asíncrono.

Para que este circuito realice el conteo adecuadamente en una Transición con Pendiente Positiva (TPP) especifica del reloj, únicamente los Flip Flop que se supone cambiaran estados en esa Transición con Pendiente Negativa deberán tener J = K = 1 cuando ocurra

esa Transición con Pendiente Positiva. Analicemos la secuencia de conteo de la Tabla 2.7 para ver que significa esto con respecto a cada Flip Flop:

En la secuencia de conteo se muestra que el Flip Flop A debe de cambiar estados en cada Transición con Pendiente Positiva. Por esta razón sus entradas J y K están permanentemente en ALTO, de manera que cambiara de estado en cada Transición con Pendiente Positiva de la entrada de reloj.

Tabla 2.7. Secuencia de conteo de un contador síncrono.

Conteo	$\overline{D}$	$\overline{C}$	$\overline{B}$	$\overline{A}$
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1
0	0	0	0	0
.	.	.	.	.
.	.	.	.	.
.	.	etc	.	.

Como la salida $\overline{A}$ del Flip Flop A esta conectada a las entradas del Flip Flop B, este cambiara de estado cada vez que la salida $\overline{A} = 1$, y exista un cambio en la entrada de reloj. Esto se produce en los instantes de reloj 2, 4, 6, 8, 10, 12, 14 y 16.

Observe que para que la salida $\overline{C}$ cambie de estado, es necesario que tanto $\overline{A}$ como $\overline{B}$ estén a nivel ALTO. Esta condición se detecta mediante la compuerta And 1, cuyas entradas se aplican a las entradas J y K del Flip Flop C. Siempre que $\overline{A}$ y $\overline{B}$ están a nivel ALTO, la salida de la puerta And 1 hace que las entradas J y K del Flip Flop C se pongan a nivel ALTO, y el Flip Flop C basculara en el siguiente pulso de reloj. Esto ocurrirá en los pulsos de reloj 4, 8, 12 y 16. El resto de la veces, las entradas J y K del Flip Flop C estarán en BAJO, al igual que la salida de la compuerta And 1, y el Flip Flop C no cambiara de estado.

La salida $\overline{D}$ del Flip Flop D cambiara de esta solo en dos ocasiones, justo cuando A, B y C estén en ALTO. Esta condición se codifica con la compuerta And 2 de modo que, cuando ocurra un pulso de reloj, el Flip Flop D cambiara de estado. En los demás estados, las entradas J y K del Flip Flop D se mantendrán en estado BAJO, lo que provoca la condición de no cambio.

Capítulo 3

Simulación de Sistemas en 20-SIM

3.1. Introducción

20-SIM es un programa de simulación y modelado que corre bajo Microsoft Windows. Con 20-SIM tú puedes simular el comportamiento de sistemas dinámicos, tales como sistemas eléctricos, mecánicos e hidráulicos o cualquier combinación de ellos.

20-SIM tiene soporte completo para modelado gráfico, permite analizar y diseñar sistemas dinámicos en una forma intuitiva y sencilla, sin comprometer energía. 20-SIM soporta el uso de componentes. Esto te permitirá entrar en los modelos como en la ingeniería del dibujo: escogiendo componentes de una librería y conectándolos, tu esquema de ingeniería es en realidad reconstruido, sin utilizar una sola línea de matemáticas.

3.2. Iniciando en 20-SIM

20-SIM consta de dos ventanas principales una que es de editor y otra de simulación, y muchas herramientas. El editor se abre cuando iniciamos 20-SIM y en el podemos crear nuestros sistemas. En la figura 3.1 se muestra la ventana del editor de 20-SIM.

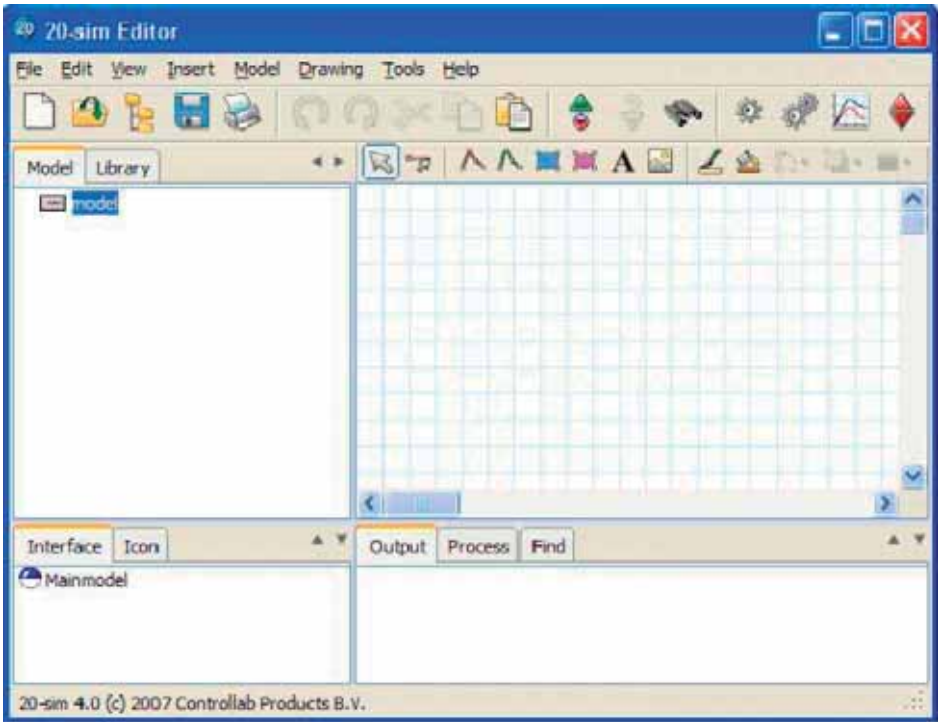


Figura 3.1. Editor del 20-SIM.

El editor consta de cuatro partes:

1.- Lengüeta de Modelos / Lengüeta de Librerías: Esta es la parte de la mitad a la izquierda. La lengüeta de modelos muestra todos los elementos que son creados en el editor. La lengüeta de librerías muestra la librería de 20-SIM.

2.- Editor Gráfico / Editor de Ecuación: Este es el grande espacio en la parte derecha. En este editor se puede crear modelos gráfico e introducir ecuaciones.

3.- Lengüeta de Salida / Lengüeta de Procesos / Lengüeta de Búsqueda: Esta es la parte inferior a la derecha. La lengüeta de Salida muestra los archivos que son abiertos y almacenados. La lengüeta de procesos muestra los mensajes compilados. Lengüeta de búsqueda muestra los resultados de la búsqueda.

4.- Lengüeta de Interfaz / Lengüeta de Icono: Esta es la parte inferior a la izquierda. La lengüeta de interfaz muestra la interfaz de un modelo seleccionado. La lengüeta de Icono muestra el icono de un modelo seleccionado. Doble click abrirá el editor de icono.

3.3 Librerías

20-SIM al igual que muchos otros programas, cuenta con muchas librerías. Al darle clic en la Lengüeta de Librerías abrirá el navegador de librerías esto mostrará todas las librerías de 20-SIM. En la figura 3.2 se puede observar en la librería de componentes eléctricos de 20-SIM.

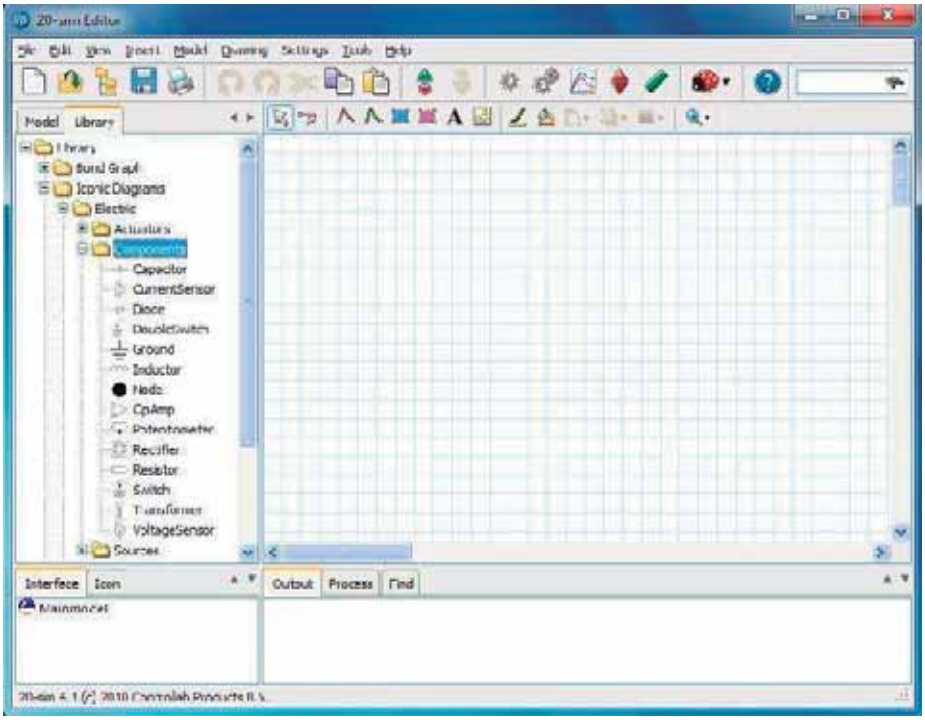


Figura 3.2. Componentes eléctricos en librería de 20-SIM.

Todos los modelos de 20-SIM son almacenados con extensión .emx. Los modelos de librerías pueden ser encontrados cuando 20-SIM fue instalado. Todos los archivos creados podrán ser almacenados en la carpeta que el usuario desee.

3.4. Arrastrar y Soltar

20-SIM funciona de una manera simple y fácil para armar nuestros modelos, basta con buscar un elemento en la librería de 20-SIM y arrastrarlo hasta el editor gráfico (área grande blanca).

Por ejemplo, arrastraremos un modelo de un rectificador de onda de la librería de 20-SIM que se encuentra en la librería de ejemplos, solo basta con llegar hasta la librería y arrastrarlo hasta el editor de gráficos. En la figura 3.3 se muestra cómo quedará nuestro editor.

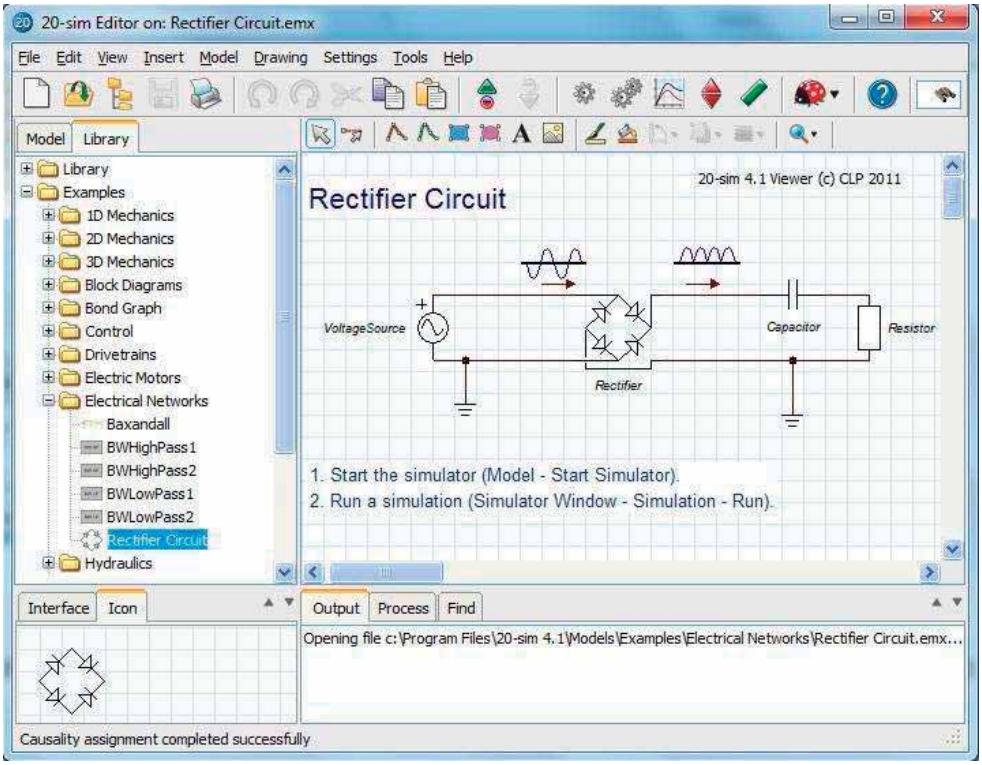


Figura 3.3. Rectificador de Onda.

Se puede inspeccionar el modelo ampliando la ventana del editor o usando el botón de zoom.

Una vez que tenemos nuestro modelo seleccionamos en el menú de modelo **Iniciar Simulador**, este botón se encuentra en la parte superior derecha como se muestra en la figura 3.4.

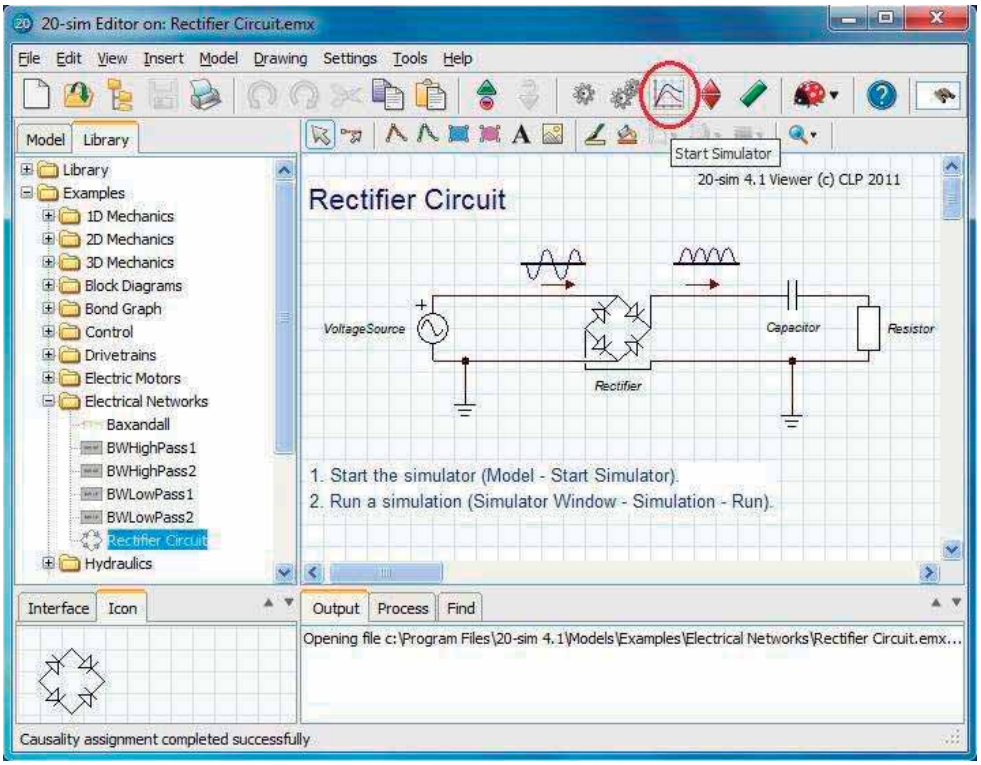


Figura 3.4. Botón Iniciar Simulador.

El simulador se abrirá en una nueva ventana y se verá como se muestra en la figura 3.5.

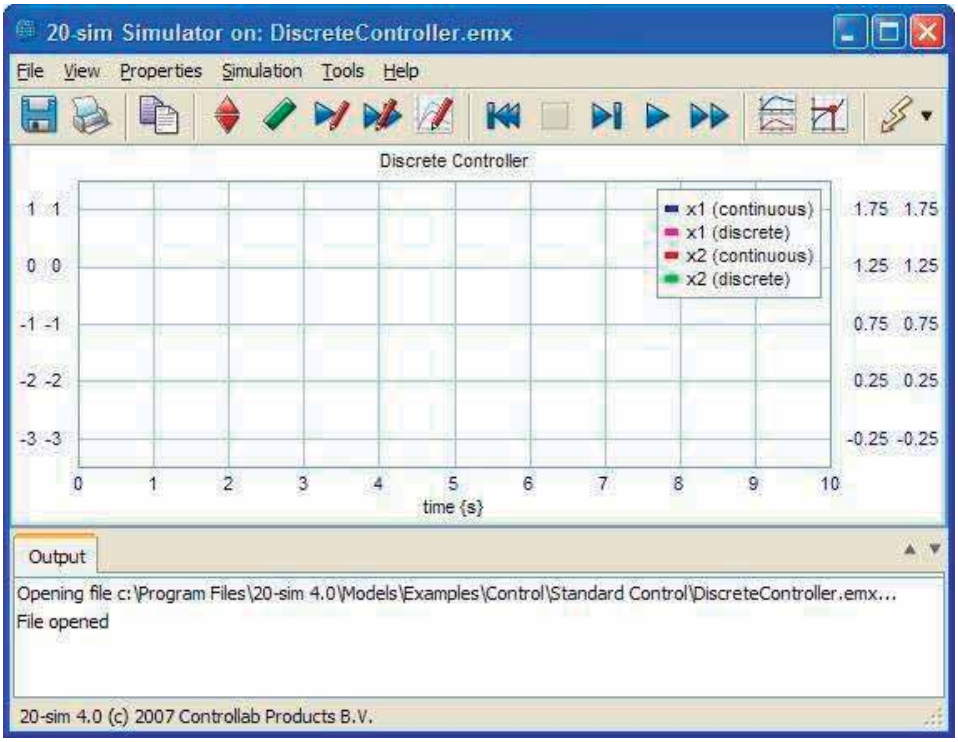


Figura 3.5. Simulador 20-SIM.

Una vez abierto el simulador, es necesario agregar los componentes de los cuales queremos ver los resultados que queramos mostrar en las graficas, para esto es necesario dar click en el botón de **Gráfico** ubicada en el menú del simulador. Esta abrirá una ventana como se muestra en la figura 3.6.

Figura 3.6. Ventana de opciones del botón **Gráfico**.

Una vez abierto la ventana de opciones del botón **Gráfico,** daremos click en el botón **Agregar Curva,** esto abrirá otra ventana en donde podremos seleccionar las variables que queremos que se muestren en las gráficas. Esta ventana de selección de variables la podemos observar en la figura 3.7.

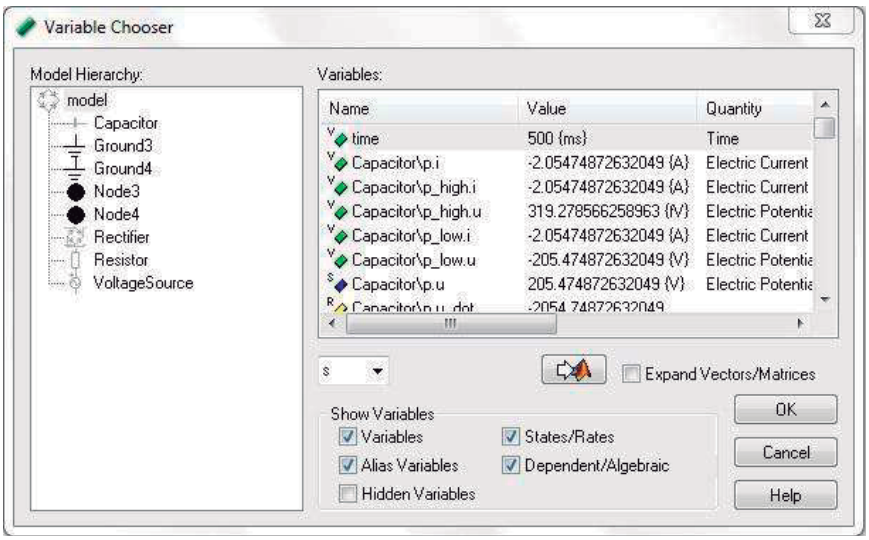


Figura 3.7. Ventana de selección de Variables.

Como podemos observar en la figura 3.7 aparece un menú en donde podemos elegir las variables que se quieran agregar a la gráfica, sólo basta con seleccionar el componente y dar click en el botón **OK.** En la figura 3.8 se muestra la ventana de opciones del botón de grafico con 3 variables ya seleccionadas las cuales son: Voltaje de Entrada, Salida del Capacitor y Salida del Resistor.

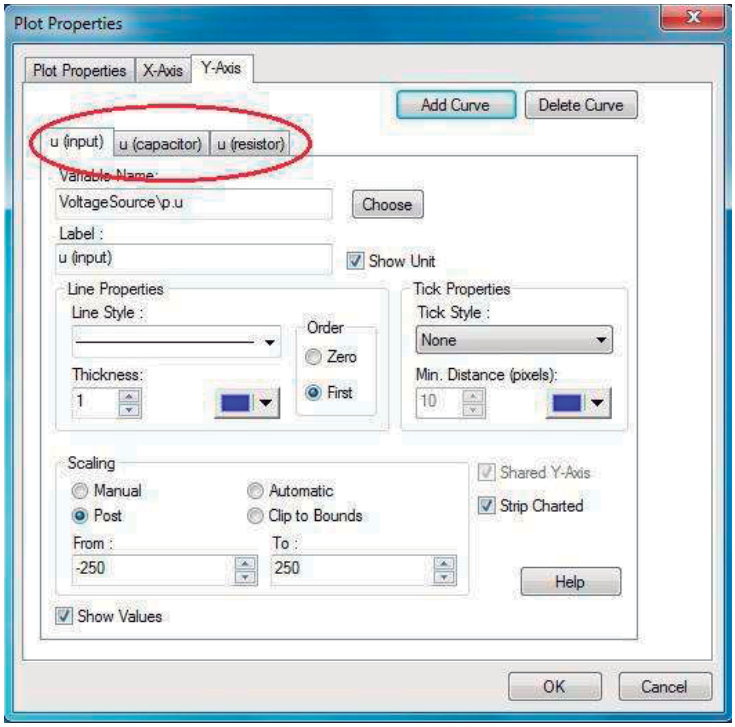


Figura 3.8. Variables ya seleccionadas.

Una vez seleccionadas nuestras variables, sólo basta dar click en el botón **OK** y estaremos listos para iniciar la corrida de nuestra simulación. En el menú del simulador selecciona **correr** ahora la simulación será realizada, esto podemos verlo en la figura 3.9.

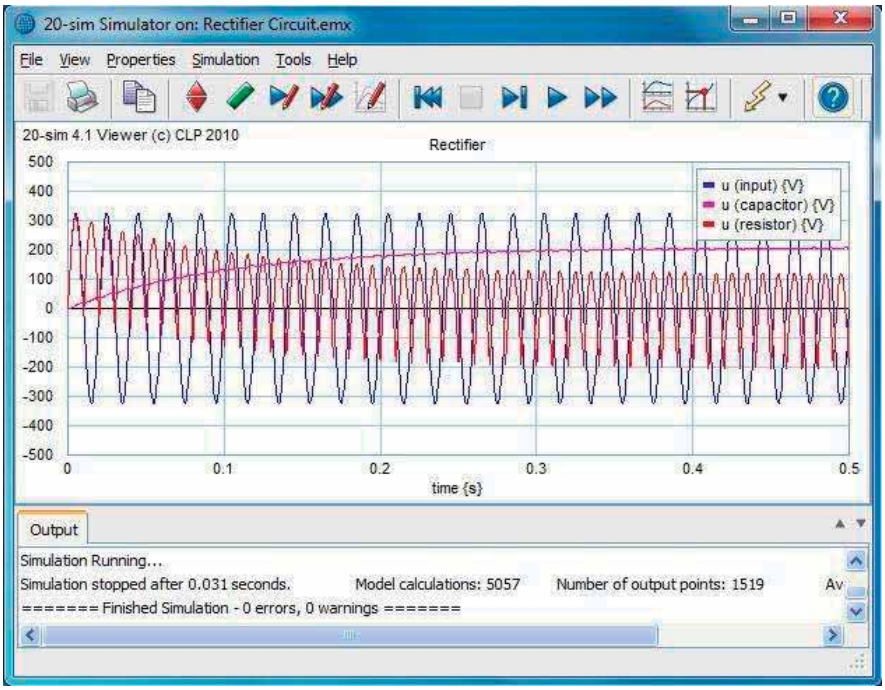


Figura 3.9. Simulación de nuestro sistema.

Podemos observar que las gráficas de nuestro sistema se grafican en una sola imagen, para poder ver las gráficas en forma independiente damos click en el botón **Distribuir Curvas,** con esto podremos ver las curvas en forma independiente como se muestra en la figura 3.10.

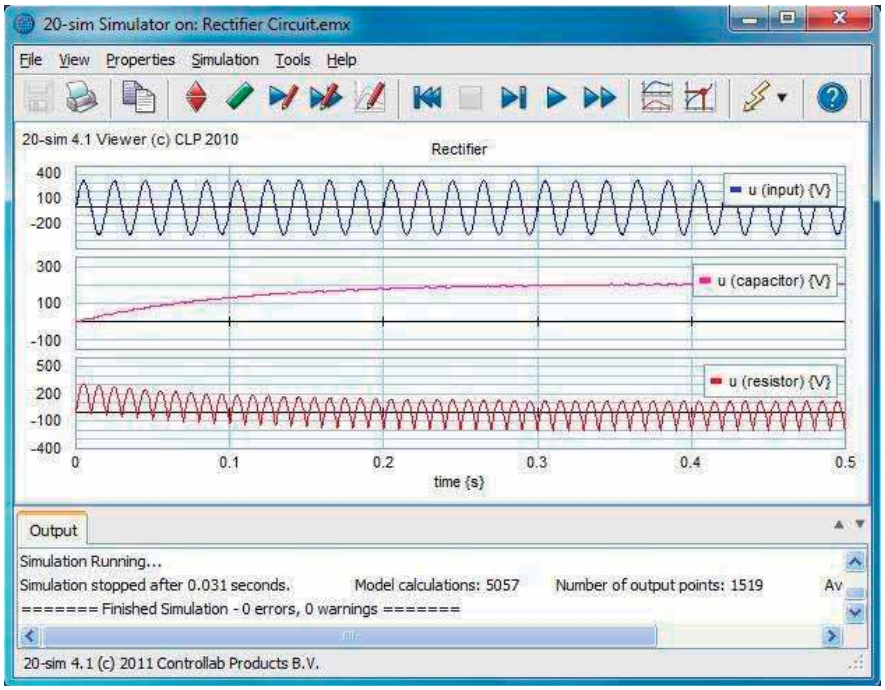


Figura 3.10. Gráficas individuales.

Ahora sabemos cómo crear y correr un modelo. 20-SIM consta de una variedad de ejemplos de modelos que podemos encontrarlo en la pestaña de librerías.

Uno puede crear sus propios modelos ya sean eléctricos, mecánicos o hidráulicos, sólo es necesario navegar por las libarías de 20-SIM, arrastrar los elementos hasta el editor, hacer las conexiones correspondientes y correr la simulación.

20-SIM también cuenta con compuertas lógicas que se encuentran en la librería de componentes lógicos.

3.5 Lenguaje de Referencia

Cada submodelo en 20-SIM tiene una implementación. Esta puede ser una composición de submodelos de bajo nivel, los cuales a su vez están compuestos por submodelos de bajo nivel, etc. Todos los submodelos de 20-SIM están representados por diagramas de bloques los cuales a su vez representaran algún dispositivo ya sea eléctrico, mecánico ó hidráulico. Los submodelos consisten de un conjunto de ecuaciones matemáticas y son conocidas como **Ecuaciones del Submodelo**. En la figura 3.11 se muestra un diagrama de bloques general de un submodelo en 20-SIM.

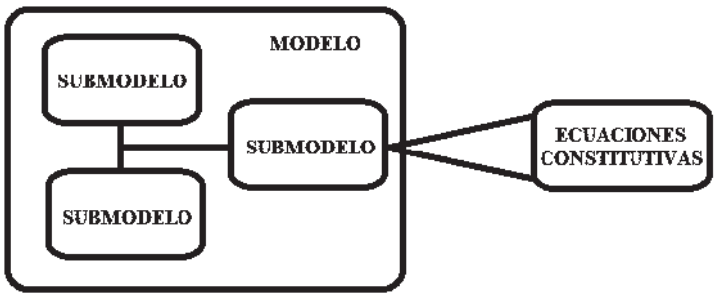


Figura 3.11. Diagrama de Bloques de un submodelo en 20-SIM.

Todas las ecuaciones usadas en 20-SIM son descritas en lenguaje SIDOPS+. La mayoría de las ecuaciones en SIDOPS+ son iguales a las notaciones matemáticas estándar. Independiente del modelado, se puede construir nuestras propias ecuaciones del sistema o modificar las ecuaciones ya existentes de algún submodelo de 20-SIM. El software 20-SIM cuanta con cerca de 80 funciones que sirven para reconstruir los submodelos, esto con la finalidad de facilitar o hacer mejor el trabajo a realizar.

Par poder ver las ecuaciones que definen a un submodelo, sólo basta con seleccionar el dispositivo que se encuentra en el editor de 20-SIM y dar doble click sobre él, ó seleccionar el dispositivo y utilizar la tecla **Ir Abajo.** En la figura 3.12 se muestra un ejemplo de cómo se utiliza este botón **Ir Abajo**. Como ejemplo, se tomó una compuerta AND previamente agregada al editor de 20-SIM.

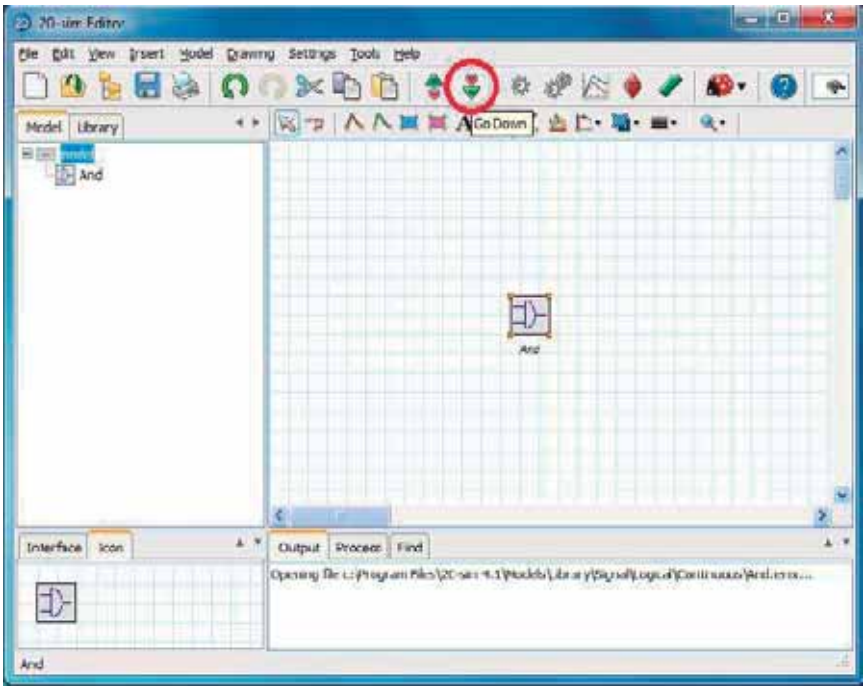


Figura 3.12. Tecla **Ir Abajo**.

Una vez presionado el botón de **Ir Abajo**, aparecerá una ventana como se muestra en la figura 3.13. En esta nueva ventana podemos observar las ecuaciones simples que definen a esta compuerta AND.

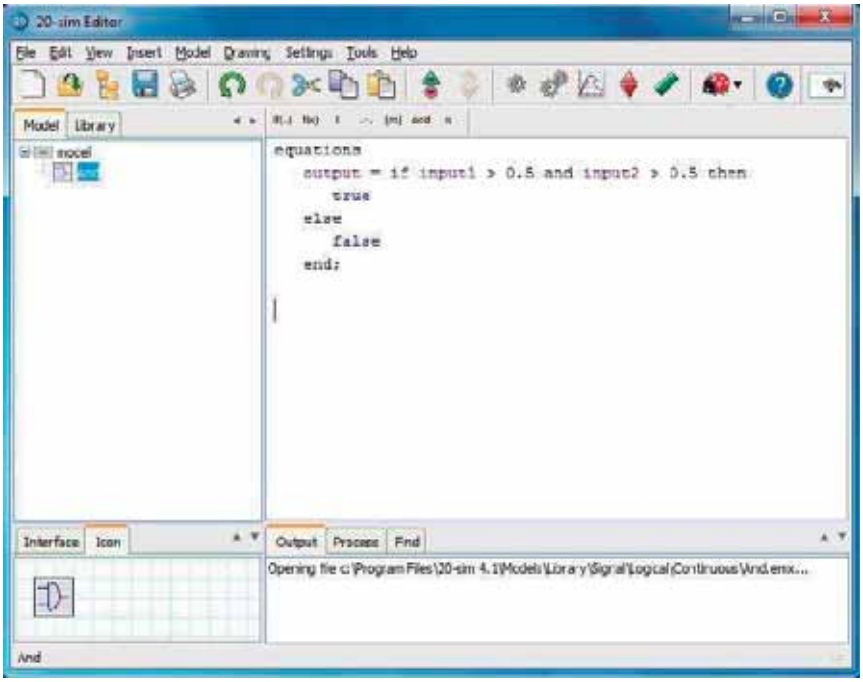


Figura 3.13. Ecuaciones que definen a la compuerta AND.

Una vez que podemos ver las ecuaciones que definen al submodelo, estas pueden ser modificadas ya sea para mejorar el submodelo ó para agregar algún puerto. Este puerto puede ser de entrada ó puede ser de salida, esto dependerá de lo que se necesite.

Como podemos observar de la figura 3.13, las ecuaciones que definen a la compuerta AND solo tienen 2 puertos de entrada llamadas *input1* e *input2*, y un puerto de salida el cual es llamado *output*.

Por ejemplo, en la figura 3.14 muestra como se ha modificado las ecuaciones de la compuerta AND para agregar un puerto de entrada llamado *imput3*. Con esto la compuerta AND tendrá la posibilidad de funcionar con 3 entradas en lugar de las únicas 2 entradas asignadas por el programa.

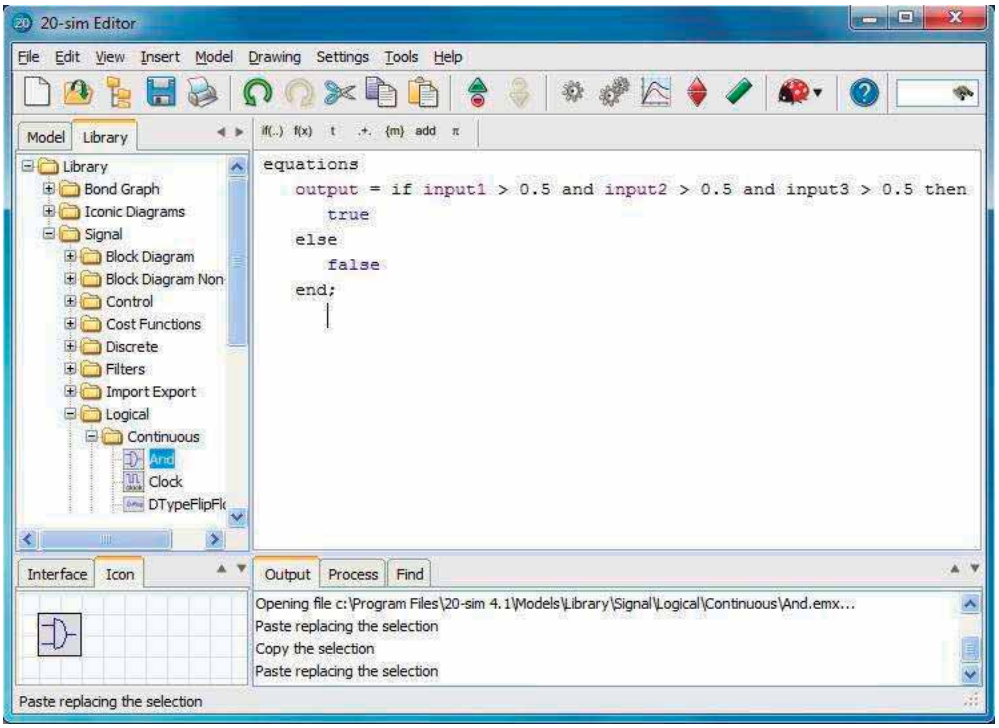


Figura 3.14. Ecuaciones de compuerta AND
para 3 puertos de entrada.

Una vez modificado las ecuaciones de la compuerta AND, faltara también agregar el puerto de entrada en el diagrama de bloques.

Para agregar el puerto de entrada en el diagrama de bloques, es necesario ir al editor de 20-SIM y hacer click secundario sobre el diagrama de bloques de la compuerta AND y seleccionar la opción **Editar Interfaz**. Esto lo podemos ver en la figura 3.15.

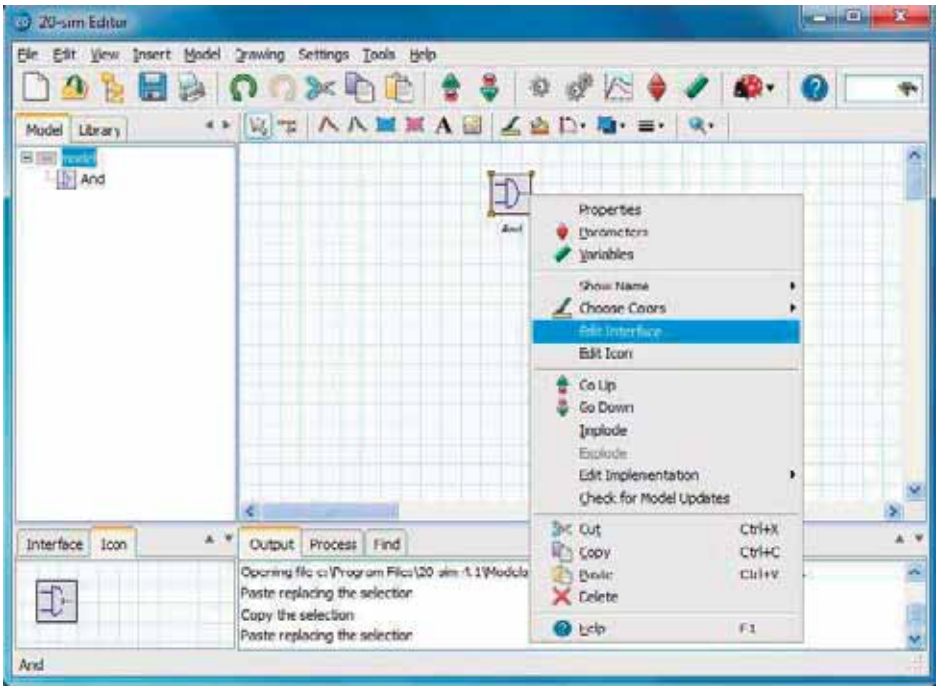


Figura 3.15 Opción para editar interfaz.

Una vez seleccionado la opción del **Editar Interfaz**, nos aparecerá una nueva ventana como se muestra en la figura 3.16, en la cual se agregan los puertos al diagrama de bloque.

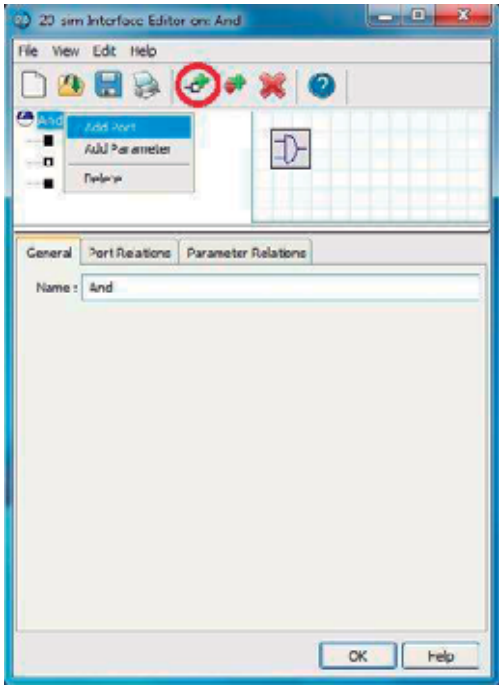


Figura 3.16 Botón Agregar Puerto

Existen 2 formas de agregar los puertos, se puede usar el botón de **Agregar Puertos** el cual se encuentra en la parte superior ó dando click secundario sobre la compuerta AND y seleccionando **Agregar Puerto**.

Una vez presionado el botón de **Agregar Puerto** se abre una nueva ventana como la que aparece en la figura 3.17. En esta ventana se seleccionará las características que tendrá este nuevo puerto agregado, tales como: el nombre del puerto, de que tipo es y la orientación de la señal, es decir, si es una señal de entrada ó es una señal de salida.

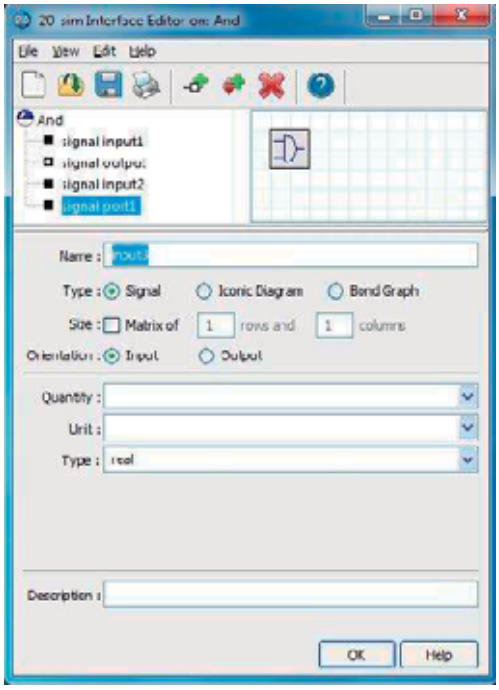


Figura 3.17 Opciones del nuevo puerto.

Como podemos observar, en la figura 3.17 se mostro las diferentes opciones de nuestro nuevo puerto, para este caso se eligió el nombre de input3 ya que en las ecuaciones fue asignado ese nombre para el puerto de entrada, se eligió que fuera una entrada de tipo señal ya que es una entrada de una compuerta lógica y por último se eligió como una señal de entrada, ya que es la entrada agregada a la compuerta AND.

Una vez seleccionados todos los valores de nuestro puerto, solo basta con hacer click en el botón O**K** y con esto ya se habrá agregado nuestro puerto de entrada. Una vez hecho esto, ya podrá ser usada la compuerta AND con el nuevo puerto de entrada.

Cabe señalar que si no se elige el mismo nombre del puerto en el diagrama de bloques que en las ecuaciones del sistema, este puerto de entrada no funcionará, y el programa marcara errores.

Capítulo 4

Circuitos Combinacionales y Secuenciales en 20-SIM

4.1 Introducción.

En este capítulo se realiza el modelado y simulación de una alarma, un modelado y simulación de un motor y el modelado y simulación de un contador tanto síncrono como asíncrono usando lógica combinacional con el software 20 SIM.

4.2 Modelado y Simulación de una Alarma.

En esta parte se modela y simula el sistema de alarma para un automóvil de 3 puertas cuyo diagrama de bloques se muestra en la figura 4.1. Como se puede observar en la figura 4.1, el sistema consta de un interruptor de encendido el cual activará los sensores los cuales se encuentran en cada una de las puertas del automóvil. Los sensores del automóvil estarán conectados al sistema de control el cual se encargará de avisar a la alarma cuando una de las puertas siempre que la alarma se mantenga activada.

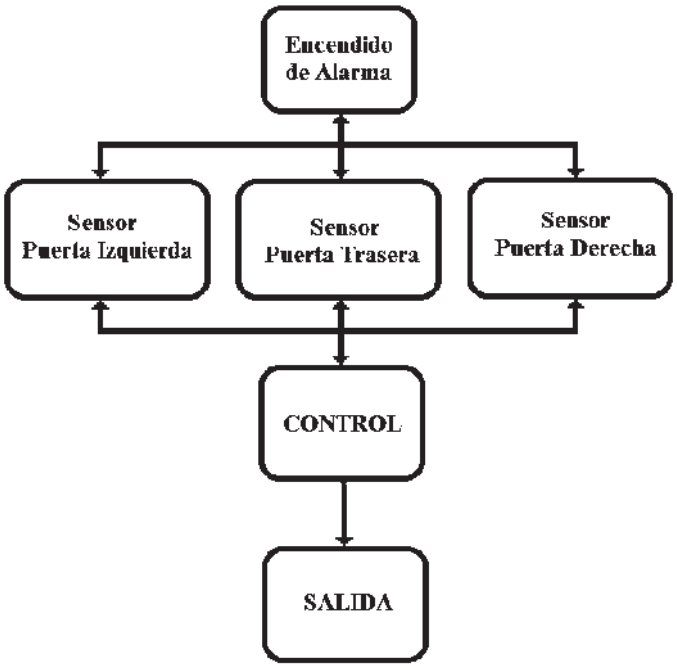


Figura 4.1. Diagrama de bloques del sistema de alarma

Para diseñar el sistema de control se considero la tabla 4.1 la cual nos indica los posibles estados en la que el sistema de alarma estará encendido. Como podemos observar en la tabla 4.1, solo tendremos en estado alto la alarma (literal x) cuando el estado de encendido y apagado (literal S) esté en estado alto y este por lo menos uno o varios de los sensores de las puertas en estado alto (literales P1, P2 y P3), de lo contrario, nuestra alarma permanecerá en estado bajo.

Tabla 4.1. Tabla de vedad para el sistema de la alarma

Encendido Apagado (S)	Puerta Izquierda (P1)	Puerta Derecha (P2)	Puerta Trasera (P3)		Alarma (x)
0	0	0	0		0
0	0	0	1		0
0	0	1	0		0
0	0	1	1		0
0	1	0	0		0
0	1	0	1		0
0	1	1	0		0
0	1	1	1		0
1	0	0	0		0
1	0	0	1		1
1	0	1	0		1
1	0	1	1		1
1	1	0	0		1
1	1	0	1		1
1	1	1	0		1
1	1	1	1		1

Una vez que tenemos la tabla de verdad para el sistema, procedemos a obtener el Mapa de Karnaugh para simplificar la función, logrando así el circuito más sencillo para este sistema.

Como el método lo indica, el Mapa de Karnaugh es un medio para mostrar la relación entre entradas lógicas y salida deseada, y se procede a colocar los estados deseados en las casillas correspondientes. En la figura 4.2 se muestra el mapa de Karnaugh para este sistema.

	$\overline{C}\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\overline{B}$	0	0	0	0
$\overline{A}B$	0	0	0	0
AB	1	1	1	1
$A\overline{B}$	0	1	1	1

Figura 4.2. Mapa de Karnaugh para sistema de alarma

Una vez colocado los estados en las casillas correspondientes, se puede observar que se agruparán 2 grupos de 4 y 1 grupo de 2 para formar la salida correspondiente. Este agrupamiento lo podemos observar en la figura 4.3.

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	0	0	0
$\bar{A}B$	0	0	0	0
AB	1	1	1	1
$A\bar{B}$	0	1	1	1

Figura 4.3 Agrupamiento de Variables

Como podemos ver se pudo agrupar 3 grupos de cuatro unos los cuales son adyacentes entre sí. Este grupo se llama cuádruple. Podemos ver que en la tercera columna hay cuatro unos que son verticalmente adyacentes, en la parte inferior derecha hay otro grupo de cuatro unos que son adyacentes entre sí y por último un grupo en la parte inferior al centro que son también adyacentes. De acuerdo al agrupamiento la función queda como sigue:

$$X = AB + AC + AD$$

De esta función podemos observar que necesitaremos de 3 compuertas AND y una compuerta OR para realizar la función deseada. La figura 4.4 nos muestra el Circuito de nuestro sistema.

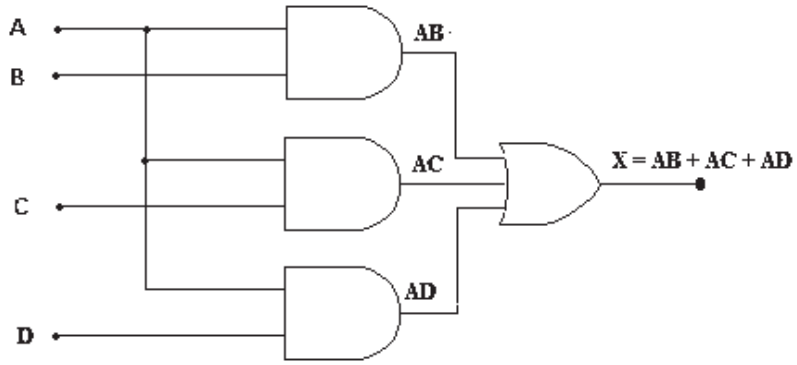


Figura 4.4 Circuito del Sistema

Ya que se tiene el circuito equivalente se procede a realizar la simulación. El software 20-SIM en sus librerías cuenta con compuertas lógicas las cuales se usaran para armar y simular el circuito.

Una vez estando en el menú principal de 20-SIM navegamos en la Lengüeta de Librerías del programa y nos dirigimos a la ruta **Librería / Señal / Lógico / Continuo,** en este menú encontraremos todos los componentes lógicos con los que cuenta el software 20-SIM. Esto se muestra en la figura 4.5.

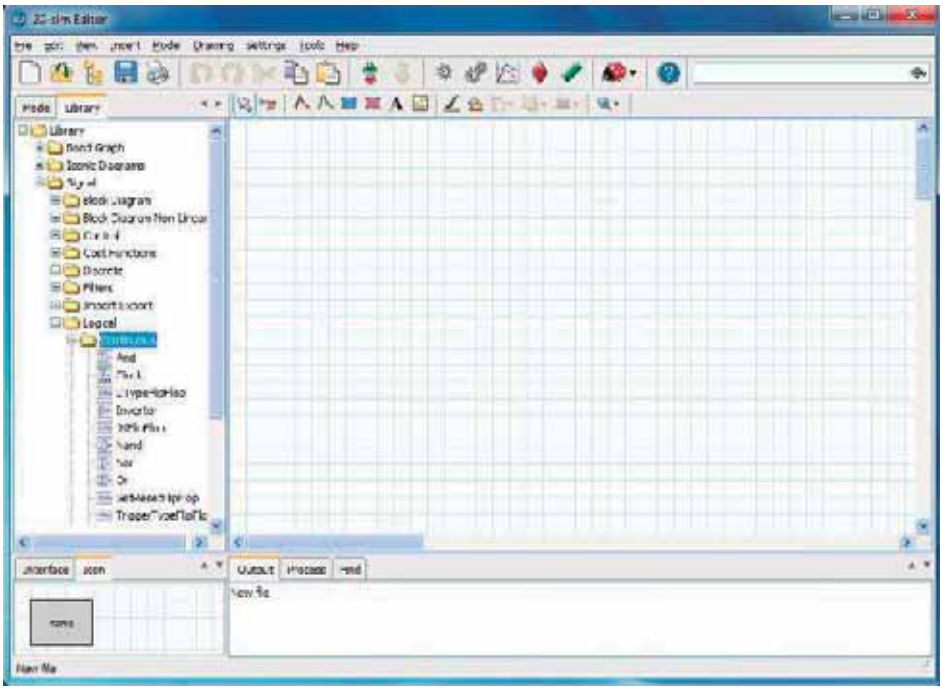


Figura 4.5 Librería de Componentes Lógicos 20-SIM

Como podemos observar, el programa cuenta con los componentes principales utilizados en la lógica combinacional, de este menú sólo usaremos tres compuertas AND y una compuerta OR las cuales arrastraremos hasta el editor el cual está en la parte derecha como se muestra en la figura 4.6.

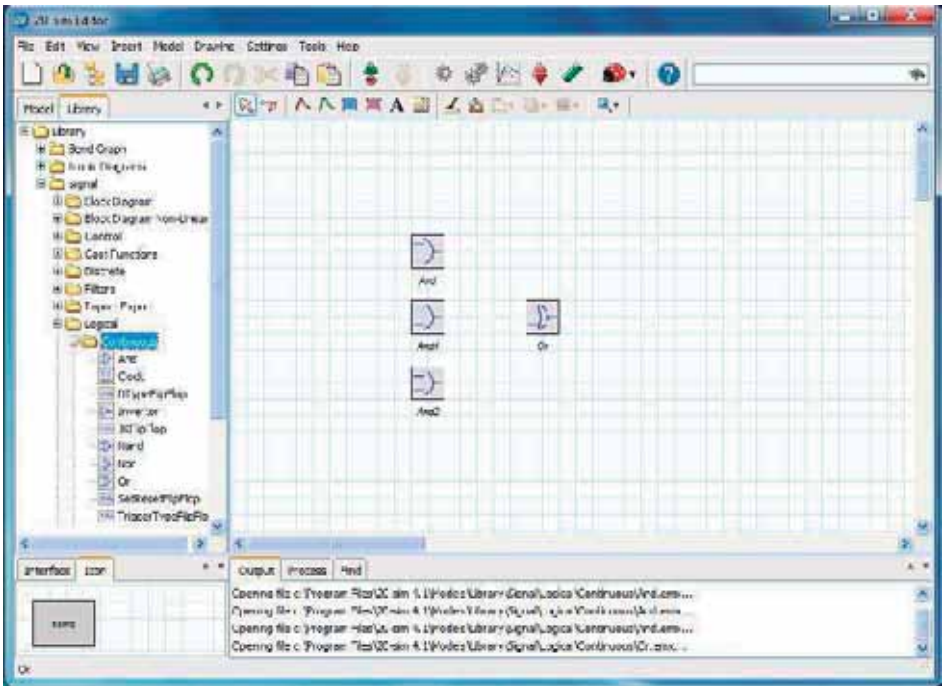


Figura 4.6 Componentes lógicos en el editor 20-SIM

La compuerta OR de 20-SIM sólo cuenta con 2 entradas lógicas; como se mencionó en el Capítulo 3, 20-SIM se basa en un lenguaje de referencia, cada modelo de 20-SIM tiene una implementación la cual puede ser de submodelos de bajo nivel llamados Ecuaciones del Modelo.

Todas las ecuaciones usadas en 20-SIM están descritas en lenguaje SIDOPS+ las cuales podemos modificar para mejorar los modelos e incluso crear nuestros propios modelos.

Para ver las ecuaciones de la compuerta OR basta con seleccionar el componente y dar click en botón de **Ir Abajo,** esto mostrará las ecuaciones de nuestro modelo. En la figura 4.7 podemos ver en la parte superior el botón que nos manda al Lenguaje de Referencia.

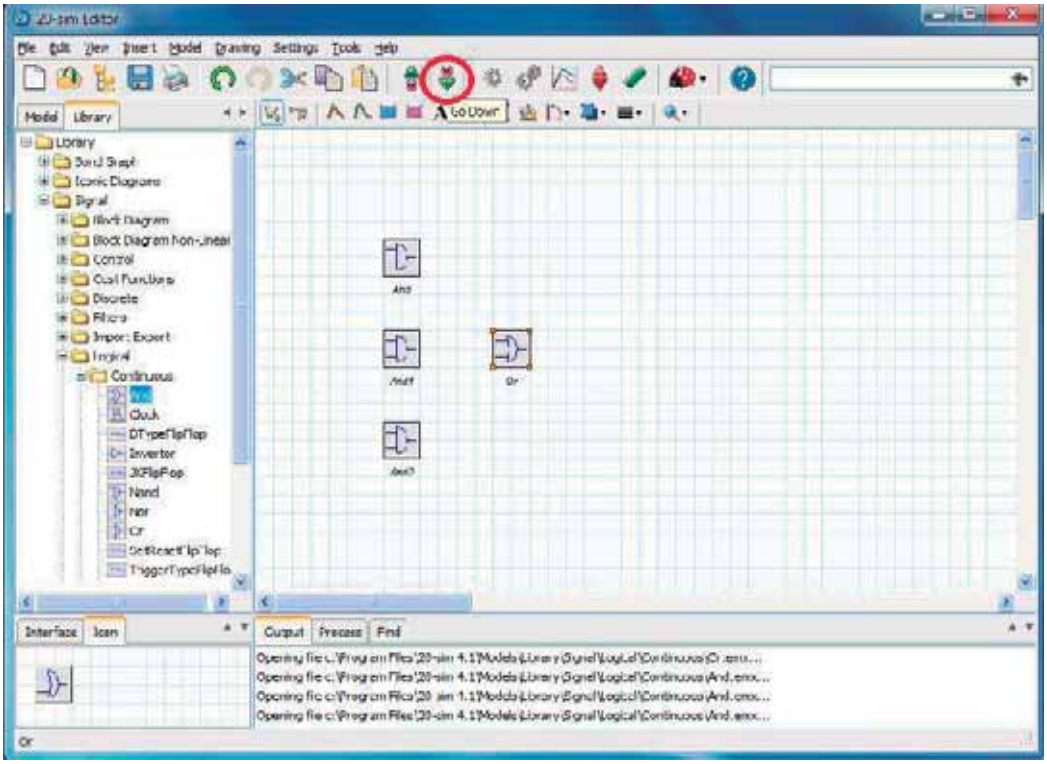


Figura 4.7 Botón para Lenguaje de Referencia

Una vez hecho esto podemos observar como el programa muestra las ecuaciones del modelo como se muestra en la figura 4.8.

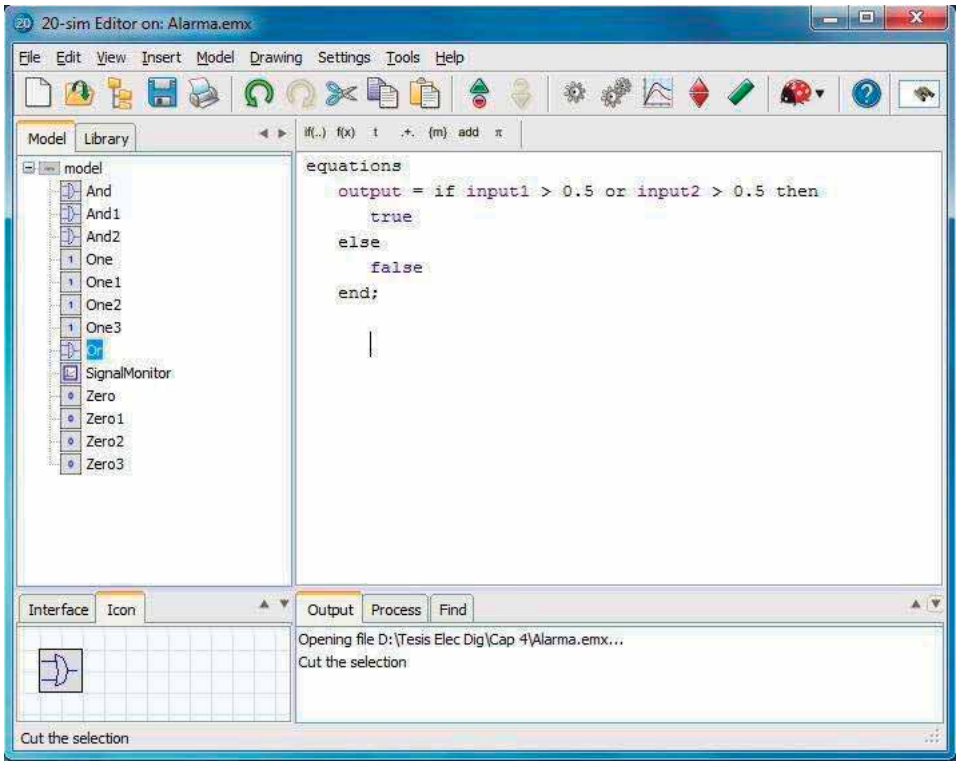


Figura 4.8 Ecuaciones de una compuerta OR

Para esta compuerta OR las ecuaciones que la definen se muestran a continuación:

```
equations
   output = if input1 > 0.5 or input2 > 0.5 then
      true
   else
      false
   end;
```

Como podemos observar en la programación esta compuerta OR sólo cuenta con 2 entradas, lo que se hará es modificar la programacion para agregar una tercera entrada, esto lo podemos ver a continuacion:

```
equations
   output = if input1 > 0.5 or input2 > 0.5 or input3 > 0.5 then
      true
   else
      false
   end;
```

Ya que hemos agregado la entrada 3 en la compuerta OR sólo basta habilitarla para poder usarla en el modelo, para esto damos click derecho sobre la compuerta OR y en el menú seleccionamos **Editar Interfaz** como se muestra en la figura 4.9.

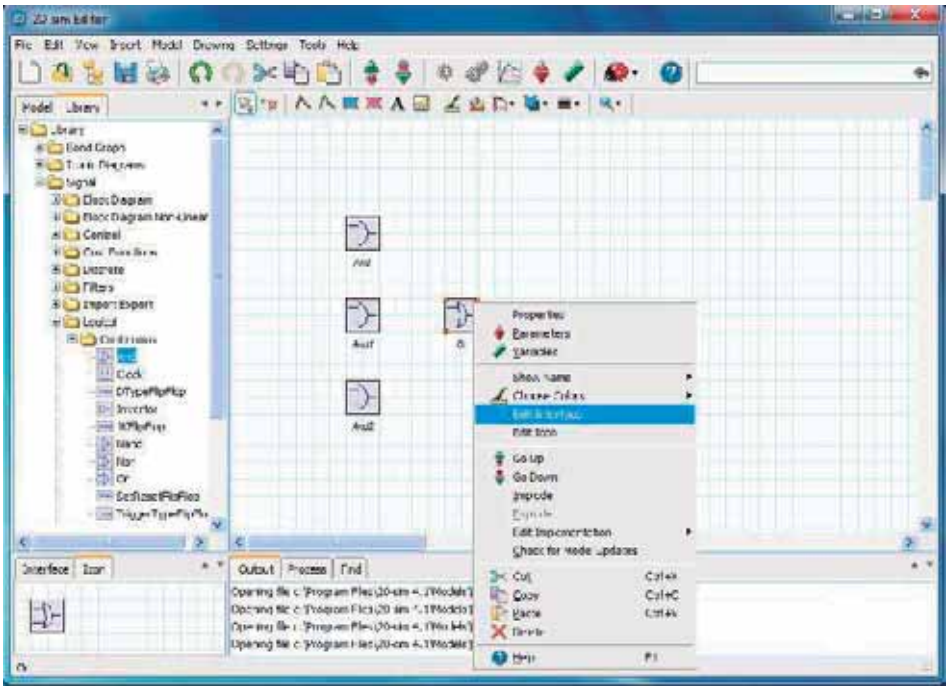


Figura 4.9 Botón para entrar a interfaz.

Una vez entrado al editor de interfaz del modelo saldrá una ventana como se muestra en la figura 4.10, en esta ventana podemos modificar la interfaz, los puertos de entrada, los puertos de salida y cambiar las propiedades de cada uno de los puertos.

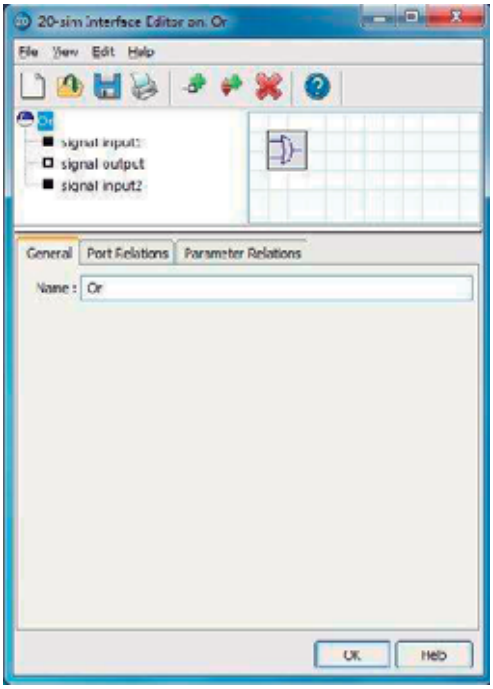


Figura 4.10 Editor de Interfaz

Una vez estando en el editor de interfaz podremos habilitar el puerto de entrada, para esto podemos dar click en el botón **Agregar Puerto** que se encuentra en la parte superior de la ventana o podemos hacer click derecho en el menú OR y seleccionamos **Agregar Puerto**, esto podemos observarlo en la figura 4.11.

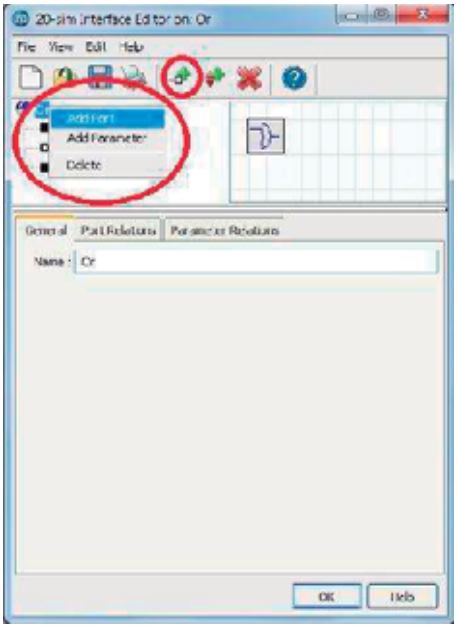


Figura 4.11 Botón para agregar puertos

Una vez que agregamos el puerto saldrá un recuadro como se muestra en la figura 4.12 en el cual daremos como nombre input3, escogeremos que sea de tipo señal y la escogeremos como señal de entrada y damos click en el botón **OK**.

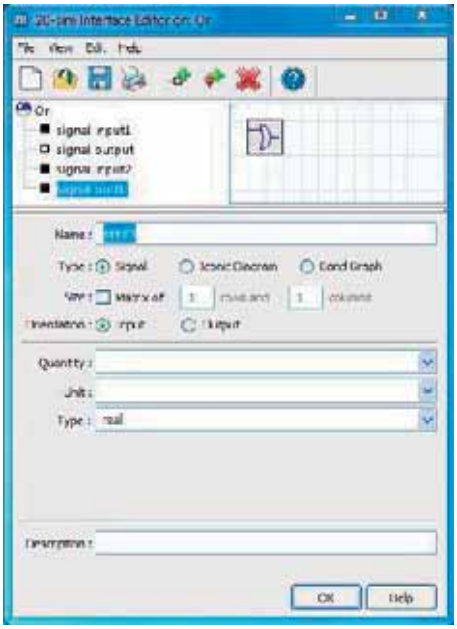


Figura 4.12 Selección de opciones del puerto.

Una vez que agregamos el puerto sólo hace falta conectar las compuertas para formar nuestro sistema, agregaremos también 8 bloques de los cuales, 4 darán un uno lógico a la salida (One, One1, One2 y One3) y 4 darán un cero lógico a la salida (Zero, Zero1, Zero2 y Zero3), estos bloques simularán los sensores de cada una de las puertas; esto quiere decir que los bloques One y Zero corresponde al encendido y apagado de la alarma, los bloques One1 y Zero1 corresponden al sensor de la puerta izquierda, los bloques One2 y Zero2 corresponden al sensor de la puerta derecha y los bloques One3 y Zero3 corresponden al sensor de la puerta trasera.

Agregaremos también un bloque llamado **Monitor de Señal** en el cual veremos los resultados de nuestra simulación, la salida de nuestro circuito será conectado al Monitor de Señal el cual graficara la salida deseada.

El circuito completo se muestra en la figura 4.13 en donde podemos observar que la entrada Zero está conectado al encendido de la alarma (S), lo que indica que la alarma estará apagada, también podemos observar que la entrada One1 (Sensor 1), One2 (Sensor 2) y One3 (Sensor 3) están conectadas a la compuerta And, And1 y And3 respectivamente, lo que indica que los sensores estarán encendidos (estado ALTO).

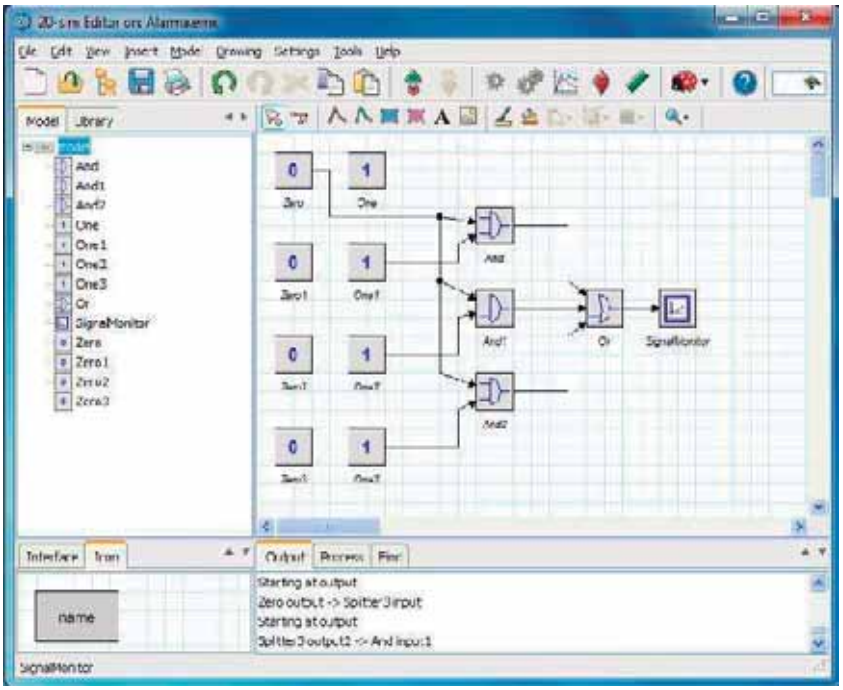


Figura 4.13. Conexión del sistema con alarma apagada.

Una vez que tenemos el sistema conectado vamos al simulador y agregamos las variables que vamos a graficar, para esta simulación agregaremos cada uno de los sensores de entrada, los cuales son: **Botón de Encendido, Puerta Izquierda, Puerta Derecha, Puerta Trasera y Señal de Salida (Alarma).** Estas variables podemos observarlas en la figura 4.14.

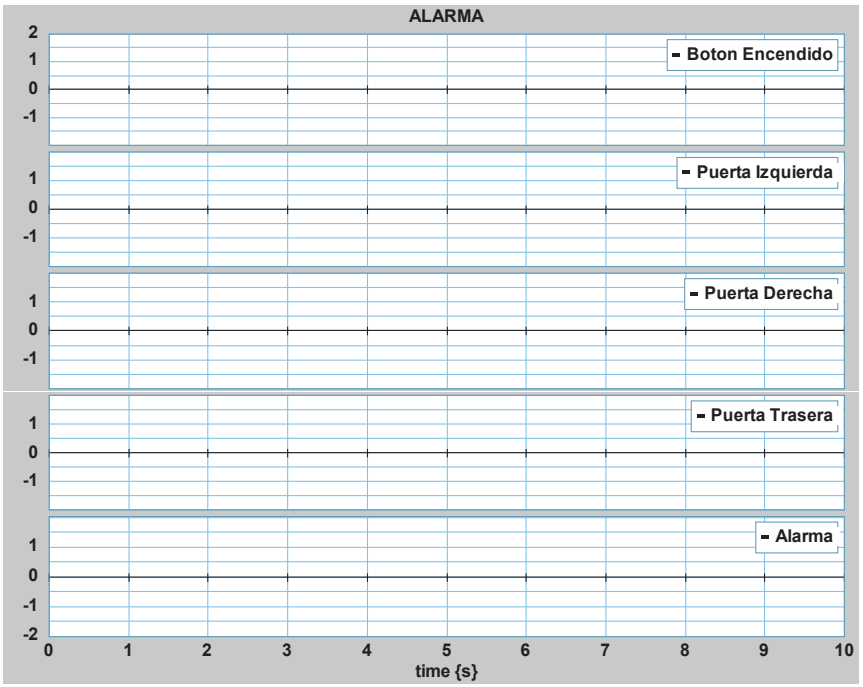


Figura 4.14. Señales de Salida

Una vez que hemos agregado las variables al simulador procedemos a ejecutar la simulación, la graficas de salida se pueden ver en la figura 4.15.

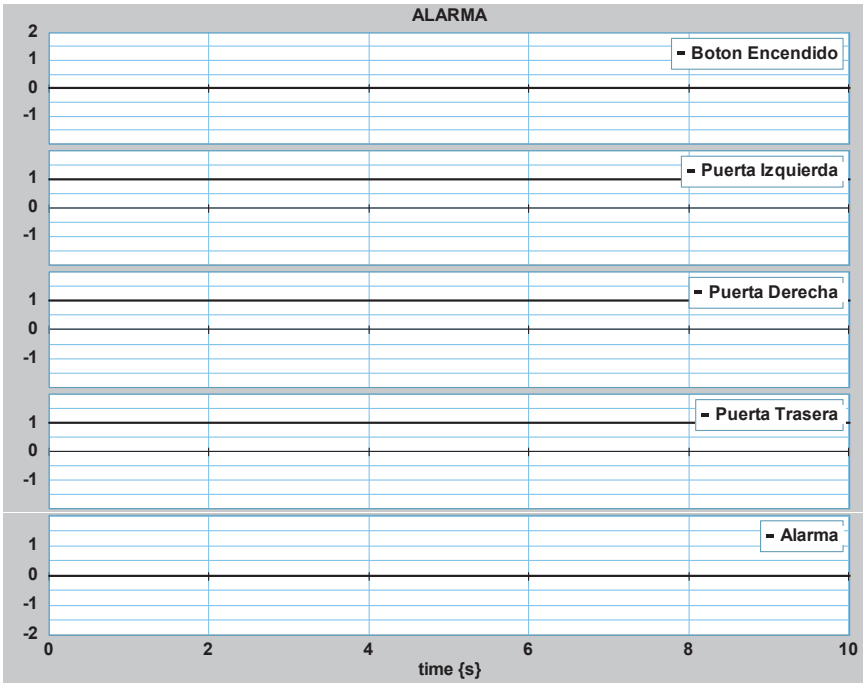


Figura 4.15. Simulación con Alarma apagada

Como podemos observar en la figura 4.15, los sensores de las 3 puertas se mantienen en alto, pero el botón de encendido de la alarma se mantiene en bajo por lo que la señal del monitor permanece en bajo.

Lo que haremos a continuación será variar las entradas de los sensores pero con el botón de encendido en alto para ver el comportamiento de nuestro sistema con los diferentes sensores de las puertas en alto.

En primer lugar pondremos tanto el botón de encendido como el sensor de la puerta izquierda en alto y el sensor de la puerta derecha y el sensor de la puerta trasera en bajo.

En la figura 4.16 se puede observar cómo se conectará el circuito para esta configuración, observe que el bloque One (botón de Encendido) está conectado a todas las compuertas, lo que indica que la alarma estará encendida, el bloque One1 está conectada a la compuerta And lo que indica que el sensor de la puerta izquierda está en estado ALTO y los bloques Zero2 y Zero3 están conectados a las compuertas And1 y And2 lo que indica que los sensores de la puerta derecha y la puerta trasera estarán en estado BAJO.

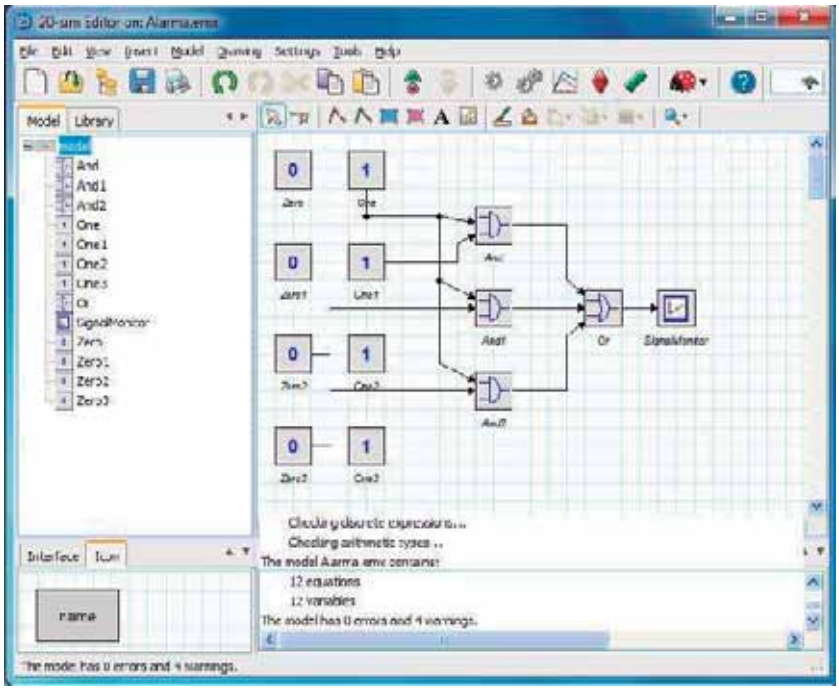


Figura 4.16 Conexión de sistema con alarma encendida
y sensor de puerta izquierda en ALTO.

La simulación de nuestro sistema con la alarma encendida y el sensor de la puerta izquierda en ALTO se muestran en la figura 4.17.

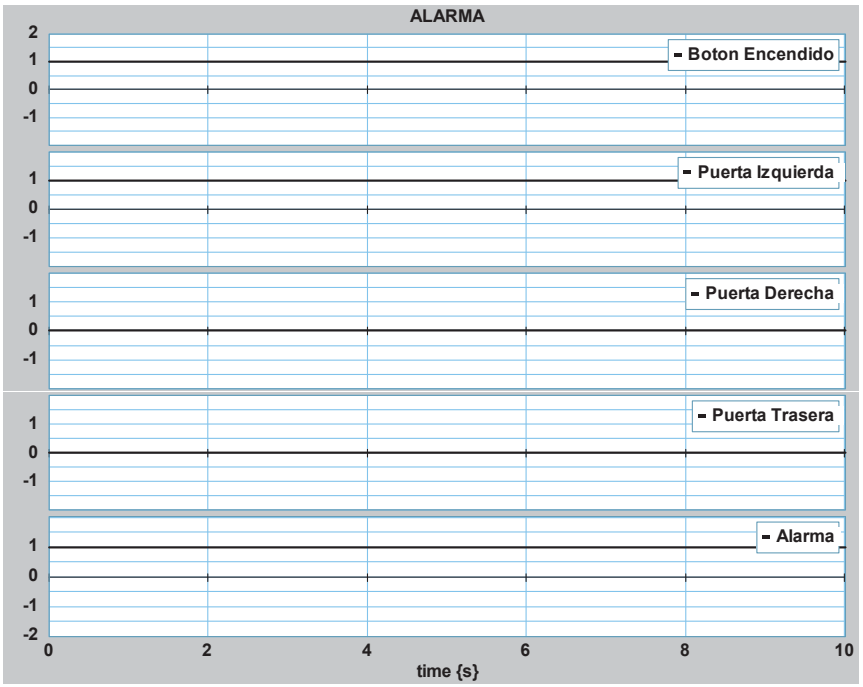


Figura 4.17 Simulación de sistema con alarma encendida y sensor de puerta izquierda en ALTO.

Podemos observar en las gráficas que el botón de encendido permanece en ALTO y que el sensor de la puerta izquierda también está en ALTO, esto quiere decir que la alarma está encendida y que el sensor de la puerta izquierda está activado por lo que a la salida veremos la alarma en estado ALTO en el monitor de señal.

Ahora pondremos el sensor de la puerta derecha en alto y el sensor de la puerta izquierda y el sensor de la puerta trasera en bajo. En la figura 4.18 podemos observar el circuito para esta configuración.

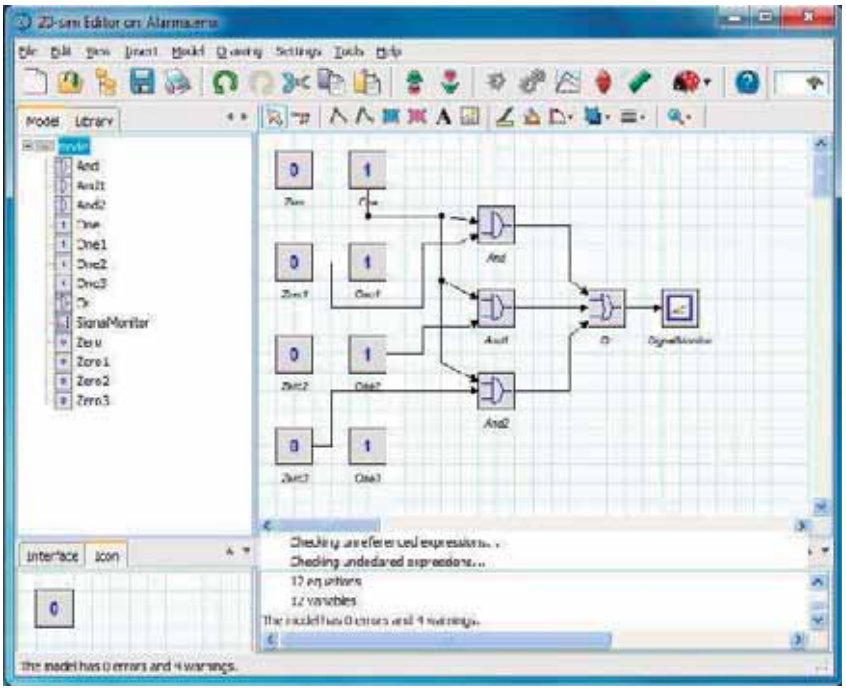


Figura 4.18 Conexión de sistema con alarma encendida
y sensor de puerta derecha en ALTO

A diferencia de la conexión anterior, One2 indica que el sensor de la puerta derecha está en estado ALTO, los demás sensores se mantendrán en estado BAJO. Las gráficas de la figura 4.19 muestran la simulación de este circuito.

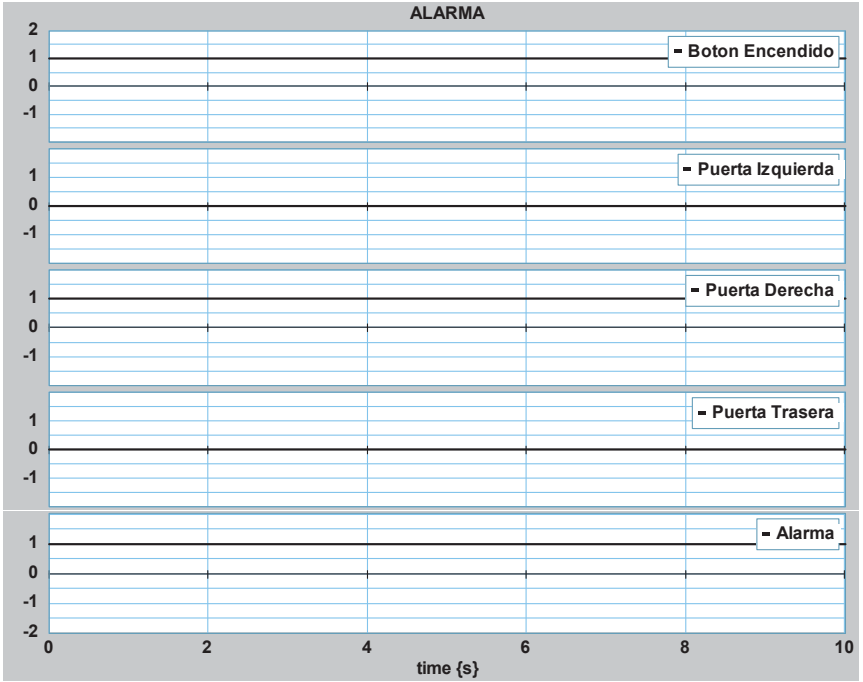


Figura 4.19. Simulación de sistema con alarma encendida
y sensor de puerta derecha en ALTO

En la figura 4.19 podemos observar que el botón de encendido y el sensor de la puerta derecha se encuentran en estado ALTO, y que el sensor de la puerta izquierda y el sensor de la puerta trasera se encuentra en estado BAJO, por lo tanto la salida en el monitor de señal estará en estado ALTO ya que indica que la alarma esta activada ya que se encuentra abierta una puerta mientras esta activa la alarma.

Ahora se pondrá el sensor de la puerta trasera en estado ALTO y los sensores de las puertas izquierda y derecha en estado BAJO. En la figura 4.20 podemos ver la conexión de esta configuración.

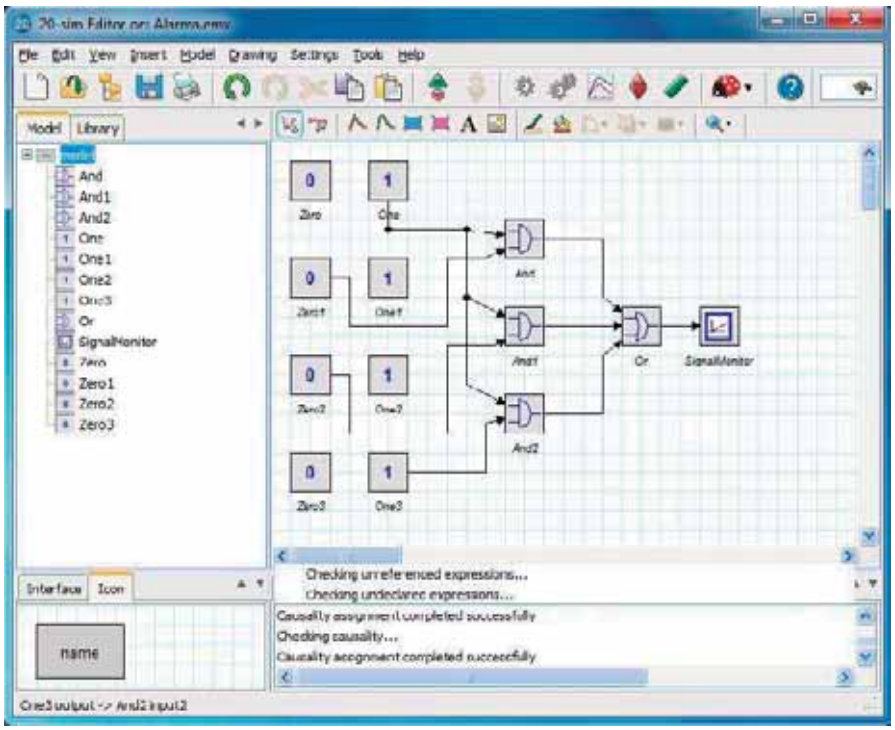


Figura 4.20 Conexión de sistema con alarma encendida
y sensor de puerta trasera en ALTO

La simulación de esta configuración se muestra en la figura 4.21 en la cual podemos ver que el sensor de la puerta trasera y el botón de encendido están en estado ALTO y los sensores de la puerta izquierda y la puerta derecha están en estado BAJO, por lo que se tendra el Monitor de señal en estado ALTO, lo cual indica qua la alarma estará encendida.

De las simulaciones anteriores podemos concluir que sólo basta que una de los sensores esté en estado ALTO para que entre en funcionamiento la alarma, y que el sistema de alarma funciona a la perfección.

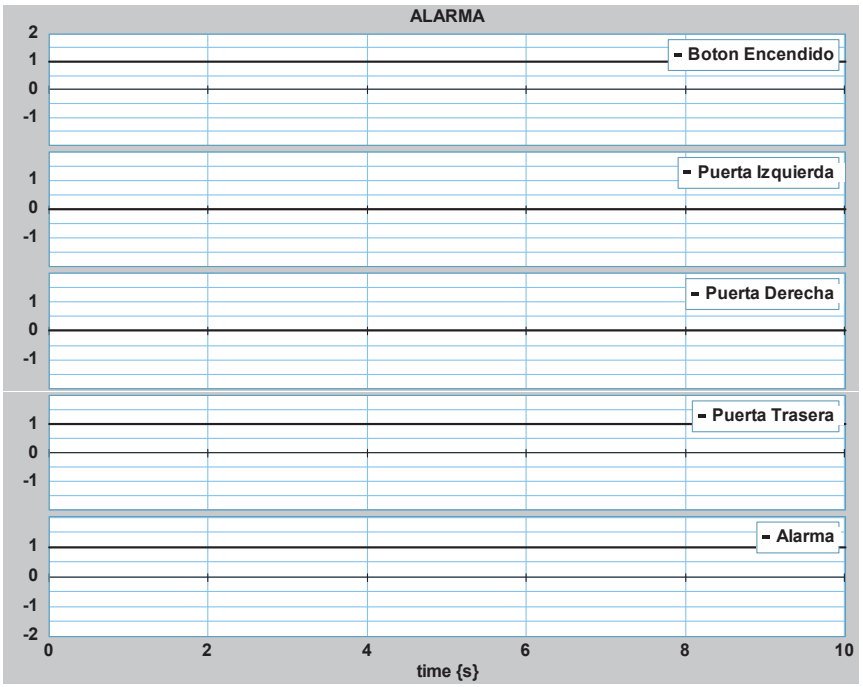


Figura 4.21. Simulación de sistema con alarma encendida
y sensor de puerta trasera en ALTO

4.3 Modelado y Simulación de un Circuito Combinacional de un Motor.

En esta sección se modela y simula el control de un motor que servirá para activar una bomba hidráulica mecánica la cual sirve para controlar un sistema Aljibe-Tinaco cuyo diagrama de bloques se muestra en la figura 4.22.

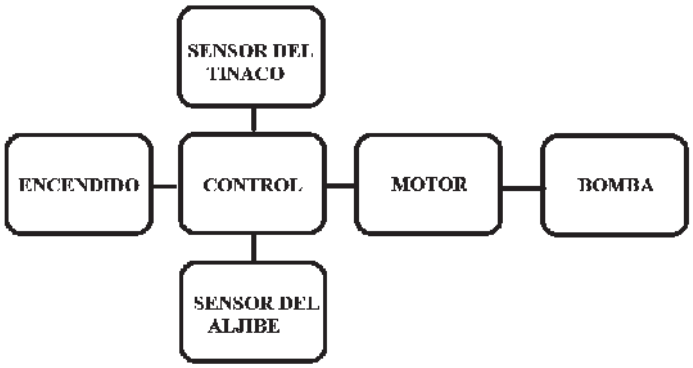


Figura 4.22 Diagrama de bloques del sistema

El sistema cuenta con un interruptor de encendido o apagado, habrá 2 sensores indicadores de nivel que estarán en el tinaco y 1 sensor que estará en el aljibe, estos sensores se conectarán a la parte de control, el cuales se encargarán de indicar en qué momento debe apagar o encender el motor para poner en funcionamiento la bomba hidráulica mecánica. En la figura 4.23 se muestra un dibujo general del sistema.

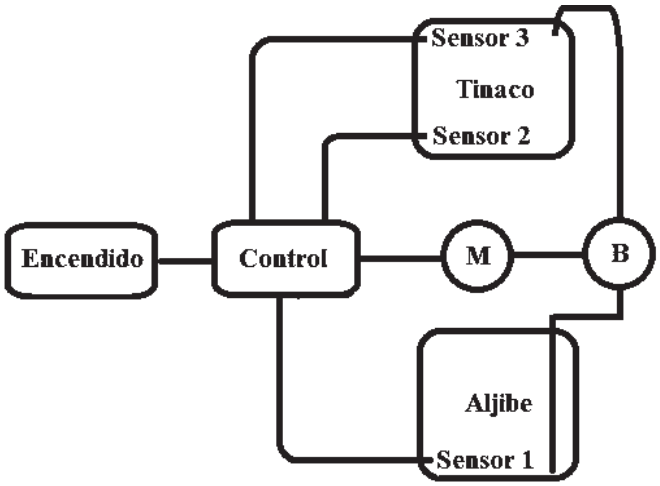


Figura 4.23 Esquema general.

El sistema funciona de la siguiente manera: al momento activar el botón de encendido los sensores se encargarán de indicar a la parte de control los niveles en los que se encuentra tanto el tinaco como el aljibe, el motor encenderá para activar la bomba sólo si el sensor 1 está en ALTO lo cual indica que hay agua en el aljibe y si el sensor 2 está en nivel BAJO, lo que indica que el tinaco no tiene suficiente agua, el motor se mantendrá encendido hasta que el Sensor 3 se ponga en nivel ALTO, lo cual indica que el tinaco está lleno.

Par diseñar el sistema de control se considero la tabla 4.2 en la cual podemos ver los estados en los que el motor estará encendido.

Tabla 4.2 Tabla de verdad para el sistema

Encendido Apagado (S)	Sensor 1 (S1)	Sensor 2 (S2)	Sensor 3 (S3)		Motor (x)
0	0	0	0		0
0	0	0	1		0
0	0	1	0		0
0	0	1	1		0
0	1	0	0		0
0	1	0	1		0
0	1	1	0		0
0	1	1	1		0
1	0	0	0		0
1	0	0	1		0
1	0	1	0		0
1	0	1	1		0
1	1	0	0		1
1	1	0	1		0
1	1	1	0		1
1	1	1	1		0

Un vez que tenemos la tabla de verdad y los estados que interesan para la salida, procedemos a realizar el mapa de Karnaugh para reducir el sistema, en la figura 4.24 se muestra el agrupamiento de la variable en el mapa de Karnaugh.

	$\overline{C}\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\overline{B}$	0	0	0	0
$\overline{A}B$	0	0	0	0
AB	1	0	0	1
$A\overline{B}$	0	0	0	0

Figura 4.24. Agrupamiento de variables.

Podemos observar del mapa de Karnaugh que sólo podemos agrupar un par, el cual dará como resultado la siguiente expresión:

$$X = AB\overline{D}$$

De acuerdo a la expresión de salida nuestro circuito equivalente estará conformado por 2 compuertas AND y una compuerta IVERSORA, este circuito se muestra en la figura 4.25.

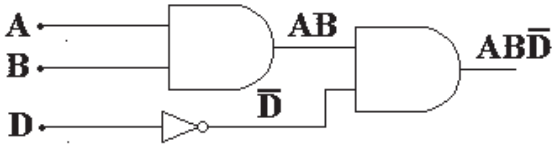


Figura 4.25. Circuito equivalente.

Una vez que tenemos el circuito equivalente procedemos a armar nuestro sistema en el software 20-SIM. Para simular este sistema necesitaremos 2 compuertas logicas AND, un INVERSOR, una fuente de CD, un Motor de CD, un sensor de velocidad y utilizaremos tambien un switch de nivel.

Cabe destacar que el switch que se utilizará representa un switch casi ideal. El switch se cerrará cuando el valor del umbral de entrada (input) sea mayor al valor de entrada de switcheo (vt) y se cerrará cuando este valor sea menor que el valor de entrada de switcheo.

El valor de entrada de switcheo es de 0.5 por defaul y sera fácil de manipularlo con compuertas lógicas. El circuito interno del switch se muestra en la figura 4.26.

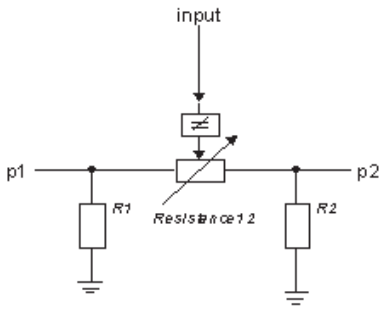


Figura 4.26. Circuito interno del switch.

Como se mencionó anteriormente, este switch en casi ideal y al momento de cerrar el circuito existirá una resistencia muy pequeña entre p1 y p2 la cual es de 0.00001 ohms, al momento de abrir el circuito la resistencia entre p1 y p2 será de 100000 ohms. Estos valores de resistencia pueden ser modificados para tener una mejor precisión aunque no ideal.

En la figura 4.27 se muestra todos los componentes que se usarán en esta simulación. Todos los componentes los podemos encontrar tanto en las librerías de componentes eléctricos como en las librerías de componentes lógicos, sólo bastará con arrastrar los elementos de las librerías hasta el editor de 20-SIM.

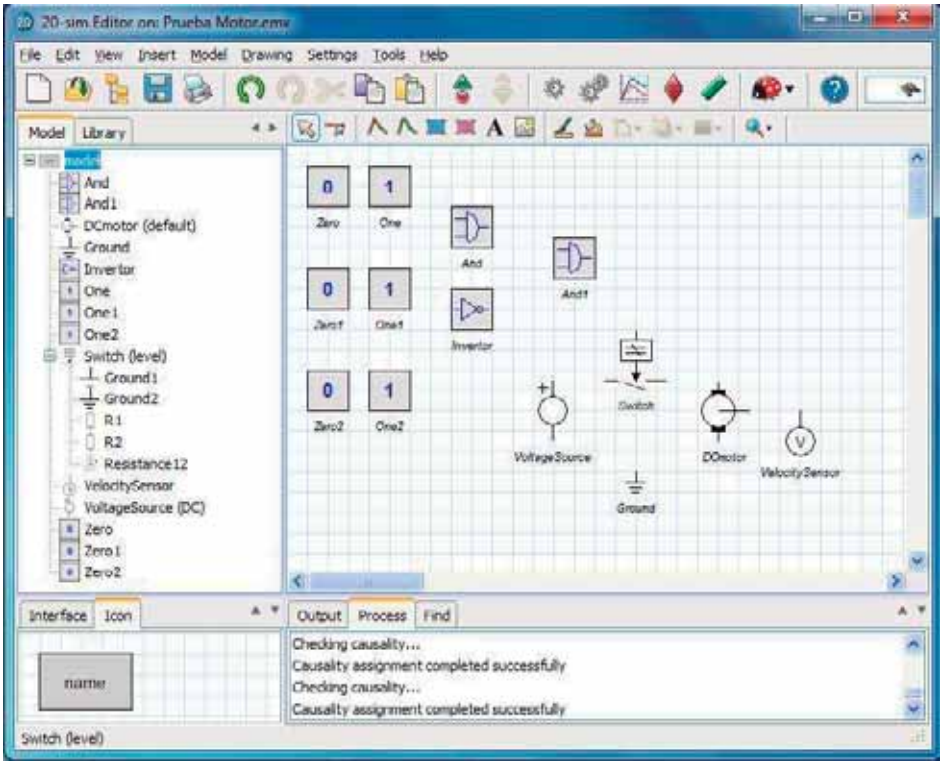


Figura 4.27. Componentes del sistema

El circuito se conectará de tal forma que los bloques Zero y One representen el botón de encendido del sistema, los bloque Zero1 y One1 representaran el sensor1 y los bloques Zero2 y One2 representarán el sensor 3, note que para este caso no es necesario agregar las entradas del sensor 2 ya que mediante mapas de Karnaugh no fue necesario usarla.

La salida del sistema estará representada por la compuerta AND1 la cual se conectará a la entrada del switch de nivel el cual conectará la bomba con la fuente de voltaje. Si el switch está en estado ALTO se creará un corto circuito y la fuente alimentará el motor, por lo tanto, el rotor del motor girará y permitirá ver la señal en el sensor de velocidad.

Si el switch se mantiene abierto no fluirá corriente lo que provocara que el motor se mantenga apagado y no veamos señal en el sensor de velocidad.

En la figura 4.28 se muestra la conexión completa del circuito para este sistema.

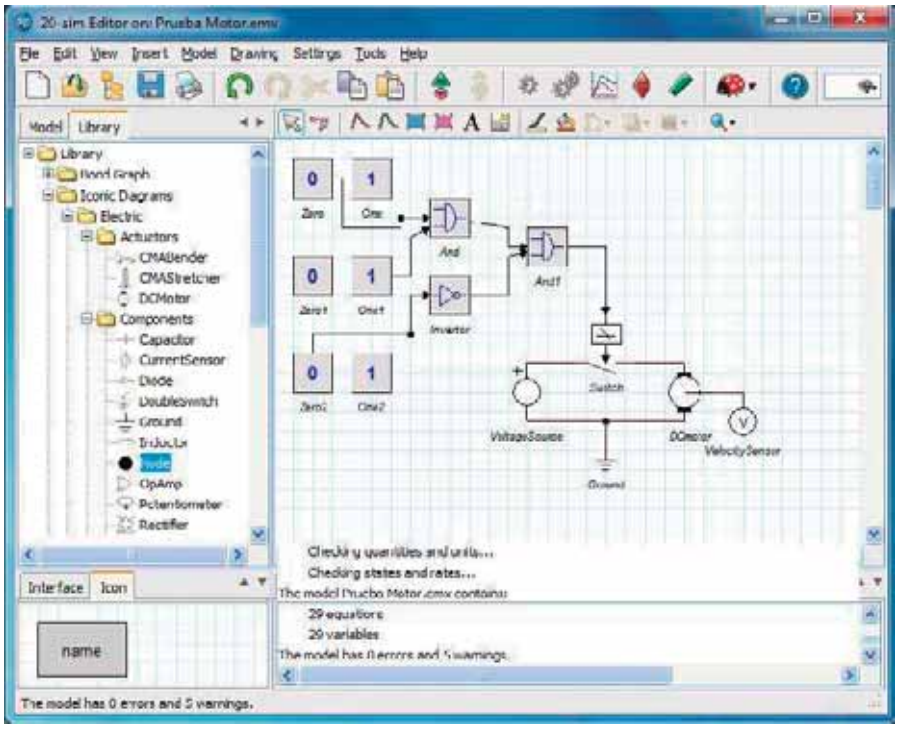


Figura 4.28. Conexión del sistema.

Una vez que se haya terminado de conectar el circuito se procede a simularlo. La simulación con estas condiciones se puede ver en la figura 4.29.

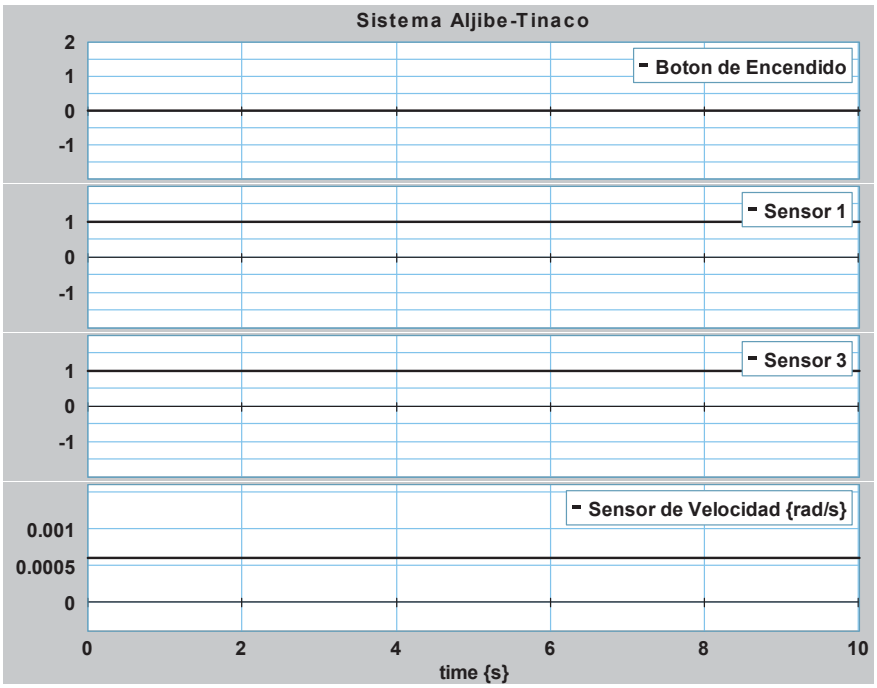


Figura 4.29. Simulación cuando el botón
de encendido esta en nivel BAJO.

De acuerdo a las gráficas anteriores podemos notar que en el circuito está conectado el bloque Zero, lo cual indica que el botón de encendido se mantiene en estado BAJO, los bloques One1 está conectado al circuito, esto quiere decir que el sensor 1 está en ALTO lo que indica que el aljibe cuenta con agua, el bloque Zero2 indica que el sensor 2 está en BAJO lo que indica que el tinaco no está lleno.

Con esta configuración el motor se mantendrá apagado ya que el sensor de velocidad no detecta velocidad, y en la gráfica veremos que la velocidad es casi cero, esto por motivos de que el switch no es ideal, por lo tanto tendremos una pequeña tensión que alimentará al motor la cual se considera despreciable.

Ahora simularemos el sistema considerando que ya se ha encendido el sistema, esto quiere decir, que el botón de encendido estará en estado ALTO. En la figura 4.30 se muestra la conexión del circuito para esta configuración.

En la figura 4.30 podemos observar que el sistema se encuentra encendido, el sistema de control detectara que en el sensor 1 se mantiene en estado ALTO, lo que significa que en el aljibe dispone de agua y el sensor 3 se mantendrá en BAJO lo que indica que el tinaco le hace falta agua, esto provocara que en la salida de control cierre el switch y este activara el motor el cual pondrá en funcionamiento la bomba hidráulica mecánica.

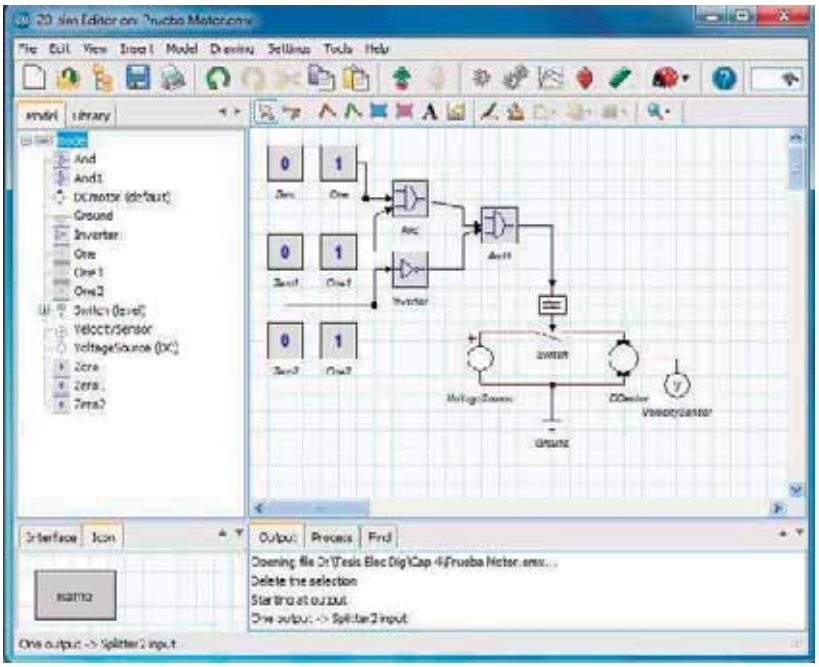


Figura 4.30 Conexión para sensor 1 en ALTO
y sensor 3 en BAJO.

Una vez que se ha terminado de conectar el sistema, se procede a simular. Las graficas para esta simulación se muestran en la figura 4.31.

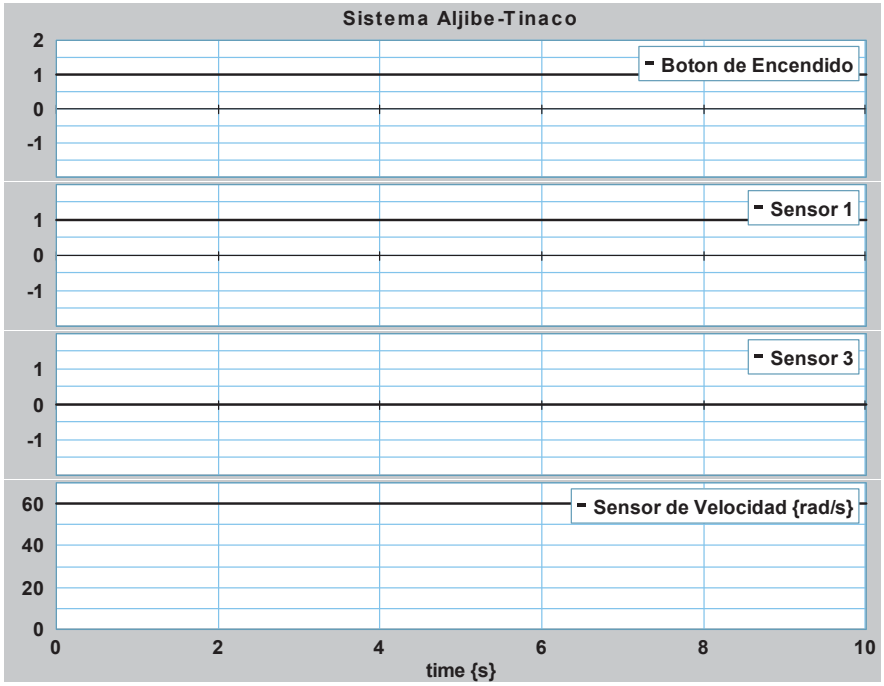


Figura 4.31 Simulación para sensor 1 en ALTO
y sensor 3 en BAJO.

Como se muestra en la figura 4.31 podemos ver que el botón de encendido está en estado ALTO, el sensor 1 está en estado ALTO, el sensor 3 está en estado BAJO, por lo que lo que el sensor de velocidad se mantendrá en estado ALTO. La velocidad del motor es directamente proporcional al voltaje de entrada, podemos observar que la velocidad que entrega el sensor de velocidad es de 60 rad/s lo que indica que el voltaje de entrada es de 60 volts.

Ahora simularemos el sistema considerando que el sensor 1 y el sensor 3 se encuentran en estado ALTO, esto por consecuencia apagará el motor.

En la figura 4.32 se muestra la conexión para esta configuración, podemos observar que el bloque One está conectado al circuito lo que indica que el sistema esta encendido, el bloque One1 indica que el sensor 1 está en estado ALTO y el bloque One2 indica que el sensor 3 está en estado ALTO, esto indica que el tinaco ya ha sido llenado. La salida del sistema se pondrá en estado BAJO lo que provocara que el switch se mantenga abierto, esto a su vez impedirá que la fuente voltaje alimente el motor, por lo tanto este se mantendrá apagado.

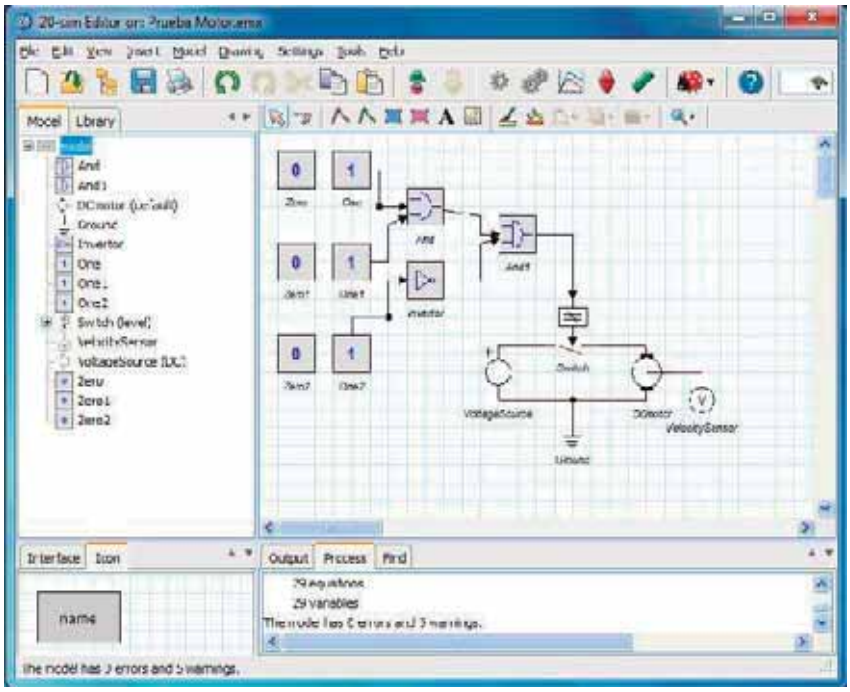


Figura 4.32 Conexión para sensor 1 en ALTO
y sensor 3 en ALTO.

Ahora se simulara el sistema considerando que el sensor 3 ya ha sido puesto en estado ALTO, esto indica que el tinaco ya ha sido llenado. Esta simulación se puede ver en la figura 4.33.

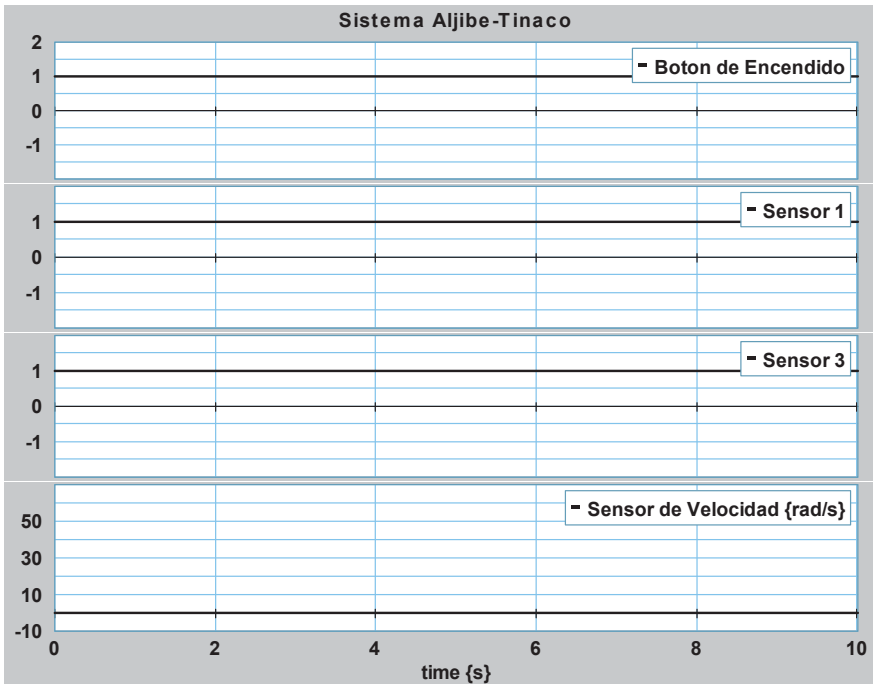


Figura 4.33 Simulación con sensor 1 en ALTO
y sensor 3 en ALTO

Como podemos ver en la figura 4.33, el botón de encendido está en estado ALTO, el sensor 1 y el sensor 3 se encuentran en estado ALTO lo que indica que el aljibe cuenta con suficiente agua para ser bombeada y que el tinaco ya ha sido llenado, esto por consecuencia apagara el motor y lo que indica que el sensor de velocidad se mantiene en estado bajo.

De acuerdo a las simulaciones hechas hasta el momento, podemos observar que nuestro sistema funciona a la perfección y que el motor encenderá en las condiciones que se requiera.

4.4 Modelado y Simulación de un Contador Asíncrono en 20-SIM.

En esta parte se modelará y simulará un contador asíncrono de 4 bits utilizando el software 20-SIM. Como se vio en el capítulo 2, el término asíncrono se refiere a los sucesos que no poseen una relación temporal fija entre ellos, y que, generalmente, no ocurre al mismo tiempo. Un contador asíncrono es aquel en que los Flip Flops del contador no cambian exactamente al mismo tiempo, zdado que no comparten el mismo impulso de reloj.

En la figura 4.34 se muestra un contador de 4 bits para que funcione de forma asíncrona el cual está construido con Flip Flops JK.

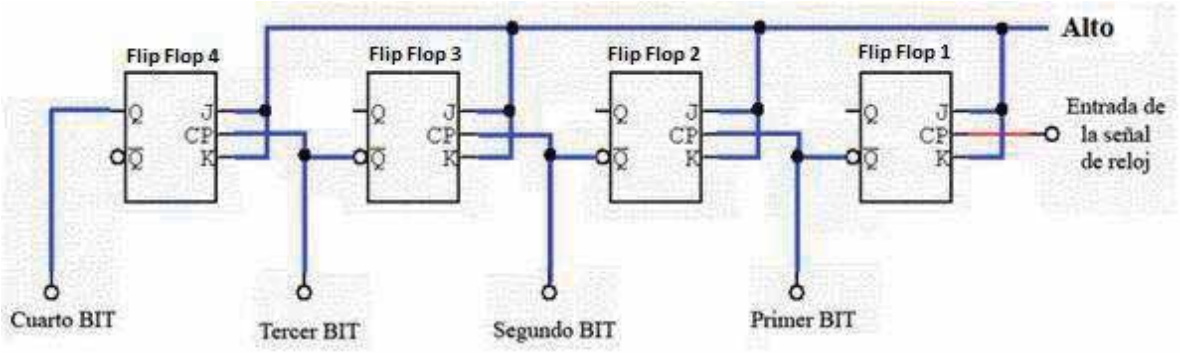


Figura 4.34. Contador asíncrono de 4 bits

Este contador se implementa con Flip Flops disparados por flanco negativo. Observe que el reloj está conectado únicamente a la entrada de reloj del primer Flip Flop, el segundo Flip Flop se disparará mediante la salida $\overline{Q}$ del primer Flip Flop, el tercer Flip Flop se dispara mediante la salida $\overline{Q}$ del segundo Flip Flop y el cuarto Flip Flop se dispara mediante la salida $\overline{Q}$ del tercer Flip Flop. El primer Flip Flop cambiará de estado durante el flanco positivo de cada impulso de reloj, el segundo, el tercero y el cuarto Flip Flops sólo cambiarán cuando es disparado por una transición positiva de de la salida $\overline{Q}$ del Flip Flop que lo antecede. Debido al retardo de propagación inherente al paso de las señales a través de los Flip Flops, la transición de los impulsos de entrada del reloj, no pueden ocurrir nunca al mismo tiempo en cada uno de los Flip Flops. Los Flip Flops nunca se dispararán de forma simultánea, por lo que el modo de funcionamiento de este contador es asíncrono.

En la figura 4.35 se muestra el diagrama de tiempo para un contador asíncrono de 4 bits.

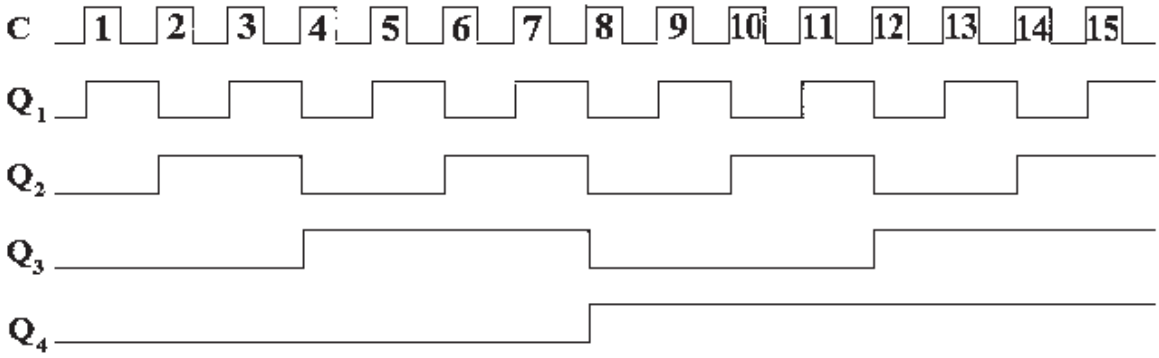


Figura 4.35 Diagrama de estados de un
contador asíncrono de 4 BITS.

El pulso de reloj hace que la salida Q del primer Flip Flop pase a nivel ALTO. Al mismo tiempo la salida $\overline{Q}$ del Flip Flop uno pasa a estado BAJO pero esta no afecta a la salida del Flip Flop 2, ya que tiene que ser un flanco positivo la que le dispare. Esta transición

tampoco activará al Flip Flop tres y el Flip Flop cuatro, ya que estos se activarán hasta que haya una transición positiva de una salida $\overline{Q}$ del Flip Flop que lo antecede.

Como podemos observar en el diagrama de la figura 4.34 Q del primer Flip Flop basculara en cada pulso positivo de reloj, las salida Q del segundo Flip Flop basculará en cada pulso positivo de la salida $\overline{Q}$ del primer Flip Flop, el tercer Flip Flop basculará en cada pulso positivo de la salida $\overline{Q}$ del Flip Flop 2 y el cuarto Flip Flop basculará en cada pulso positivo de la salida $\overline{Q}$ del Flip Flop 3.

20-SIM cuenta con 3 tipos diferentes de Flip Flops: el Flip Flop RS, el Flip Flop T y el Flip Flop D. Para simular este circuito usaremos el Flip Flop T el cual funciona de la misma forma que un Flip Flop JK, la única diferencia es que estas 2 entradas forman una sola entrada cortocircuitada llamada T.

Los Flip Flops T se encuentran en la librería de componentes lógicos de 20-SIM, pero con la característica que este Flip Flop T sólo cuenta con una entrada de control que será la entrada del reloj y una salida Q. Como se utiliza la salida $\overline{Q}$ de cada uno de los Flip Flops T para disparar los Flip Flops, será necesario agregar este puerto de salida a cada uno de los Flip Flop T que usaremos para este contador asíncrono de 4 BITS.

La programación que definen a este Flip Flop T se muestra a continuación:

```
parameters
   boolean initial = true;       // initial output
variables
   real oldcontrol;
   real oldoutput;
equations
   oldcontrol = dly (control,0);
   oldoutput = dly (output, initial);
   output = if oldcontrol <= 0.5 and control > 0.5 then
      if oldoutput <= 0.5 then
         true
       else
         false
      end
   else
      oldoutput
   end;
```

Como se menciono anteriormente, es necesario agregar el puerto de salida $\overline{Q}$ para poder usar este Flip Flop T, por lo que la programación quedara como se ve a continuación:

```
parameters
   boolean initial = false;          // initial output
variables
   real oldcontrol;
   real oldoutput;
equations
   oldcontrol = dly (control,0);
   oldoutput = dly (output, initial);
   output = if oldcontrol <= 0.5 and control > 0.5 then
      if oldoutput <= 0.5 then
         true
       else
         false
      end
   else
      oldoutput
   end;
Q_neg = if output <= 0.5 then
      true
   else
      false
   end;
```

Observemos que sólo se ha agregado la salida Q_neg, la cual dará la salida negada del Flip Flop. Sólo bastará agregar los puertos en el bloque del Flip Flop.

Una vez modificado nuestro Flip Flop T, procederemos a simularlo en 20-SIM, para esto necesitaremos cuatro bloques de Flip Flops tipo T y un bloque que genere una señal de reloj el cual se encuentra en las librerías de componentes lógicos. Para este caso no será necesario un monitor de señal ya que se agregarán las señales de salida en el simulador.

Una vez agregados todos los componentes, lo conectaremos como se mostro en la figura 4.35 para crear nuestro contador. Como se menciono anteriormente este Flip Flop sólo contaba con 2 puertos una entrada de reloj y la salida Q del Flip Flop, pero, ¿qué sucede con la entrada T? 20-SIM pone por default la entrada T en estado alto, por tal motivo no es necesario activarla con una entrada. El circuito completo lo podemos ver en la figura 4.36.

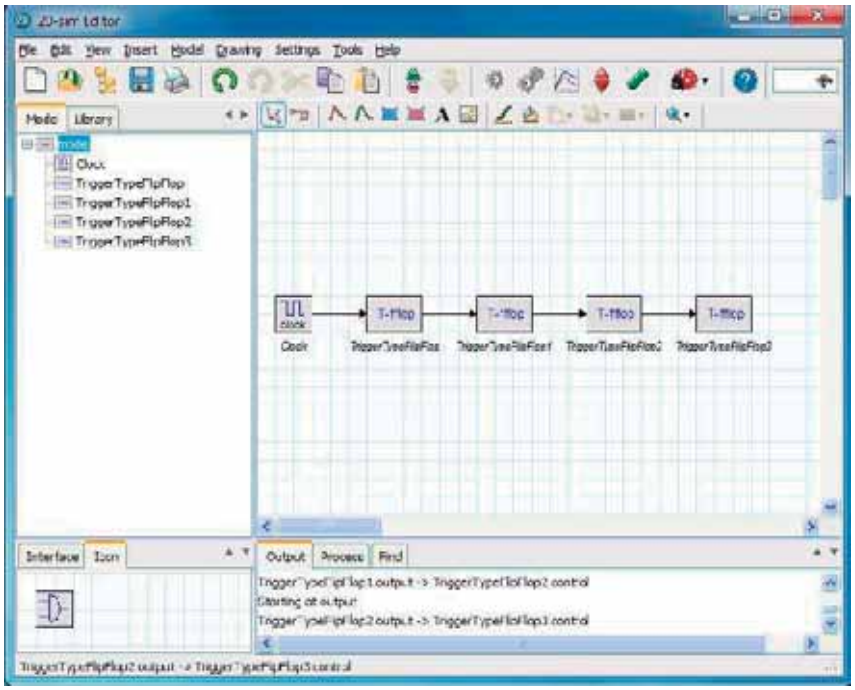


Figura 4.36. Contador asíncrono en 20-SIM

Como podemos observar sólo el primer Flip Flop tiene la señal de reloj, los demos Flip Flops tienen conectada a su entrada la salida Q_neg del Flip Flop que los antecede, esta entrada Q_neg es la salida creada anteriormente para cada uno de los Flip Flops.

Una vez que tenemos el circuito conectado, sólo basta con correr la simulación, en el monitor de salida agregaremos la salida Q de cada uno de los Flip Flops, esto dará el conteo en cada una de las salidas. En la figura 3.37 se muestra la simulación del contador asíncrono.

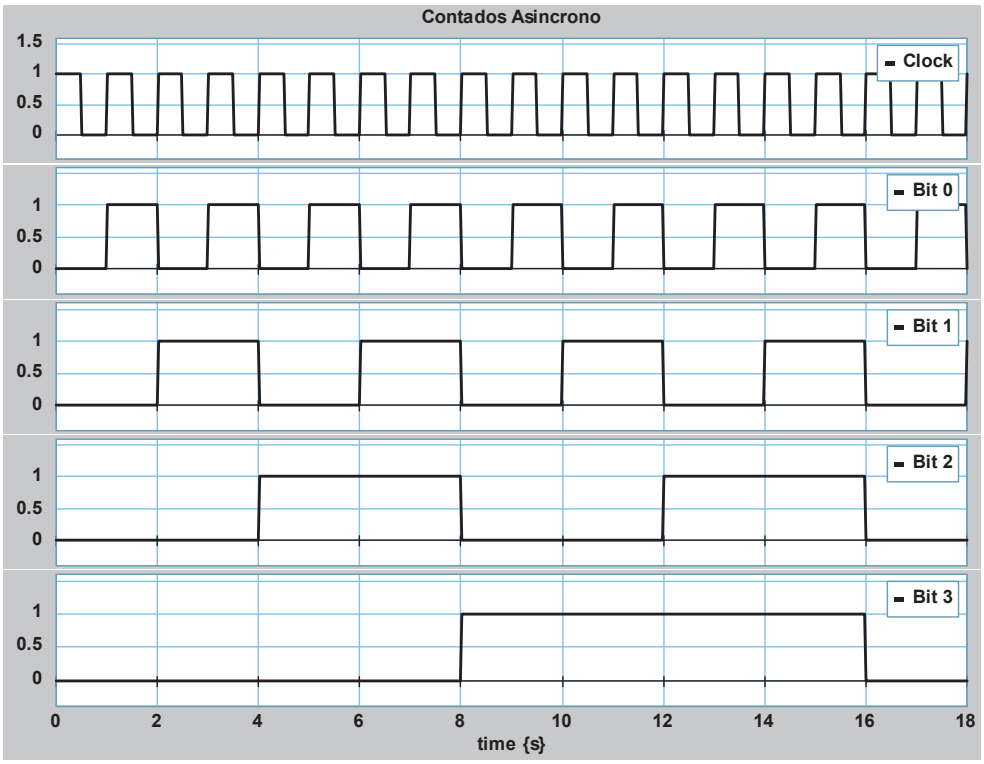


Figura 4.37 Simulación del contador asíncrono.

Como podemos observar en la simulación, el Flip Flop 1 se dispara en cada pulso del reloj. El Flip Flop 2, el Flip Flop 3 y el Flip Flop 4, se disparan en la transición Q negativa del Flip Flop que lo antecede, lo que indicara que será dispara durante la transición positiva de la salida Q.

Como podemos observar, el contador asíncrono funciona a la perfección, la salidas $\overline{Q}$ de todos los Flip Flop indicará el conteo por cada disparo del reloj.

4.5 Modelado y Simulación de un Contador Síncrono 20-SIM.

En esta sección se modelara y simulara un contador asíncrono de 4 bits utilizando el software 20-SIM. Como se mencionó en el capítulo 2, el termino síncrono se refiere a los eventos que tienen una relación temporal fija entre sí. Con respecto al funcionamiento del contador, síncrono significa que todos los Flip Flops del contador reciben la señal de reloj.

En la figura 4.38 se muestra un contador de 4 bits para que funcione de forma síncrona el cual está construido con Flip Flops JK.

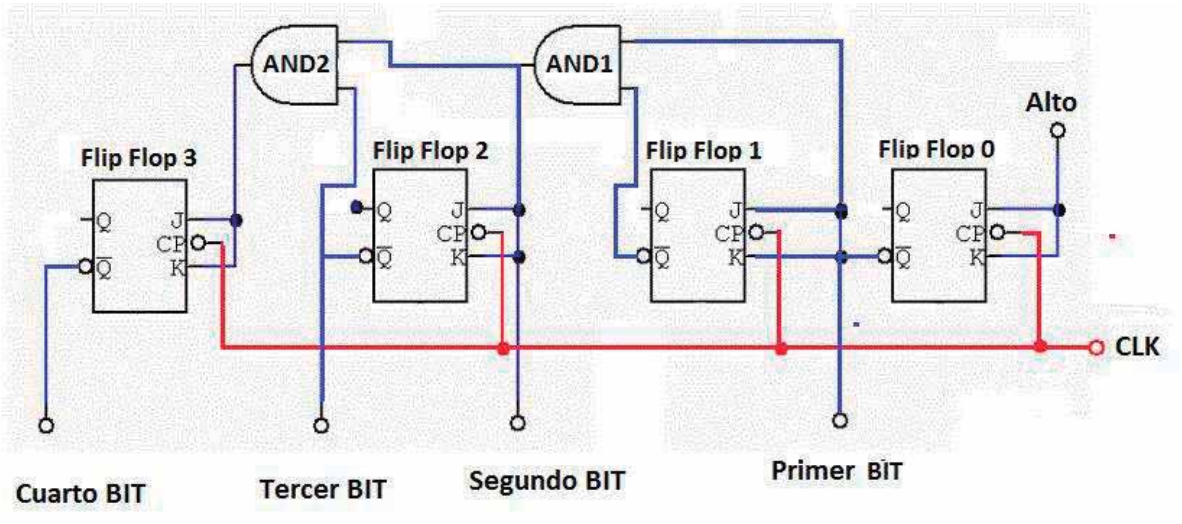


Figura 4.38 Contador síncrono de 4 BITS.

Este contador también se implementa con Flip Flops disparados por flanco negativo. En la figura 4.39 se muestra el diagrama de tiempo para el contador síncrono de 4 BITS.

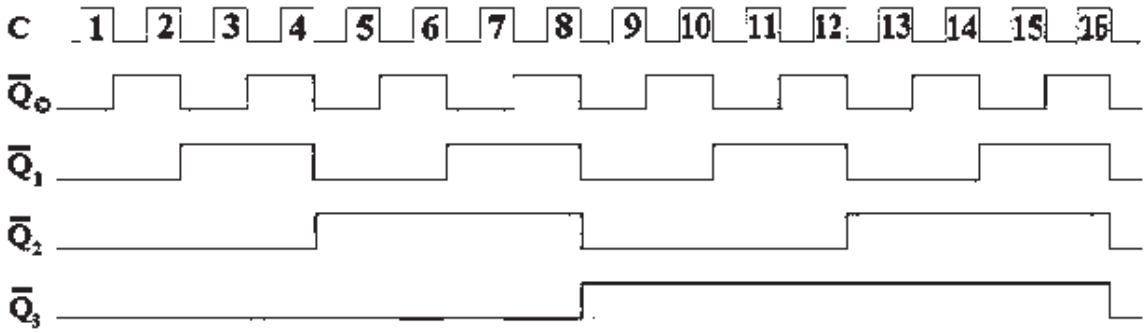


Figura 4.39 Diagrama de tiempo de
un contador síncrono de 4 BITS.

En primer lugar vamos a fijarnos en $\overline{Q}_0$, observe que $\overline{Q}_0$ cambia en cada impulso de reloj a medida que el contador avanza de su estado original hasta su estado final, para nuevamente iniciar un ciclo en su estado original. Para conseguir este funcionamiento, el Flip Flop tienen que mantenerse en estado de basculación, aplicando constantemente niveles ALTOS en su entrada J y K. Tenga en cuenta que $\overline{Q}1$ pasa al estado contrario cada vez que $\overline{Q}_0$ está a 1, $\overline{Q}_2$ pasa al estado contrario cada vez que $\overline{Q}_1$ está a 1 y $\overline{Q}_3$ pasa al estado contrario cada vez que $\overline{Q}_2$ está a 1. El impulso CLK16 hace que el contador inicie un nuevo ciclo. Para conseguir este modo de operación, se conecta $\overline{Q}_0$ del Flip Flop 0 a las entradas J y K del Flip Flop 1. Cuando $\overline{Q}_0$ esta a 1 y se produce un impulso de reloj. El Flip Flop 1 se encuentra en modo de basculación y, por tanto, cambia de estado. El resto de las veces, cuando $\overline{Q}_0$ es 0, el Flip Flop 1 está en modo de no cambio quedando en su estado actual.

Observe que las veces que $\overline{Q}_2$ cambia de estado, debe cumplirse la única condición de que tanto $\overline{Q}_0$, como $\overline{Q}_1$ estén a nivel ALTO. Esta condición se detecta mediante la puerta ADN, cuya salida se agrega a las entradas del Flip Flop 2. Siempre que $\overline{Q}_0$ y $\overline{Q}_1$ estén a nivel ALTO, la salida de la puerta AND1 hace que las entradas J y K del Flip Flop 2 se pongan a nivel ALTO, y el Flip Flop 2 bascule en el siguiente impulso de reloj. El resto de las veces las entradas J y K del Flip Flop 2 se mantienen en estado BAJO, al igual que la salida de la puerta AND1 y el Flip Flop no cambia de estado.

La cuarta etapa, el Flip Flop 3, varía solo 2 veces en la secuencia. Observe que estas dos secuencias ocurren justo cuando $\overline{Q}_0$, $\overline{Q}_1$ y $\overline{Q}_2$ están a nivel ALTO, esta condición se decodifica mediante la compuerta AND 2, de modo que, cuando se produce un impulso de reloj, el Flip Flop 3 cambia de estado. En los demás casos, las entradas J y K del Flip Flop 3 están a nivel bajo y se produce la condición de no cambio.

Como se menciono anteriormente 20-SIM cuenta con 3 diferentes tipos de Flip Flops: Flip Flop RS, Flip Flops D y Flip Flops T. Para esta simulación es necesario de Flip Flop JK, pero 20-SIM no cuenta con estos tipos de Flip Flops. Para obtener un Flip Flop del tipo JK se modificara la programación del Flip Flop RS para implementar un Flip Flop JK. La programación del Flip Flop RS se muestra a continuación:

```
parameters
   boolean initial = true;       // initial output
variables
   real oldoutput;
equations
   oldoutput = dly (output, initial);
   output = if input1 <= 0.5 then
      if input2 <= 0.5 then
         oldoutput
      else
         0.0
      end
   else
      if input2 <= 0.5 then
         1.0
      else
         oldoutput
      end
   end;
```

Como podemos observar, este Flip Flop cuenta con 2 entradas y una sola salida. Lo que haremos será modificarlo para que funcione como un Flip Flop y tenga las entradas de reloj y la salida negada. La programación de este Flip Flop se muestra a continuación:

```
parameters
   boolean initial = true;       // initial output
variables
   real oldcontrol, oldoutput;
equations
   oldcontrol = dly (control, 0.0);
   oldoutput = dly (output, initial);
   output = if oldcontrol <= 0.5 and control >0.5 then
               if input1 <= 0.5 then
                  if input2 <= 0.5 then
                     oldoutput
                  else
                     0.0
                  end
               else
                  if input2 <= 0.5 then
                     1.0
                  else
                     if oldoutput <= 0.5 then
                        1.0
                     else
                        0.0
                     end
                  end
               end
            else
               oldoutput
            end;

   output_neg = if oldoutput <= 0.5 then
      true
   else
      false
   end;
```

Como podemos observar, se ha modificado el Flip Flop RS para que funcione como Flip Flop JK, se ha agregado la entrada de reloj y también se ha agregado la salida negada del Flip Flop.

Una vez modificado nuestro Flip Flop RS, procederemos a simular nuestro contador síncrono en 20-SIM, para esto necesitaremos cuatro bloques de Flip Flops RS, un bloque que nos genere una señal de reloj, un bloque de uno lógico el cual servirá para mantener las entradas J y K del Flip Flop 1 en estado ALTO. Para este caso tampoco será necesario un monitor de señal ya que se agregaran las señales de salida en el simulador.

Una vez agregados todos los componentes, lo conectaremos como se mostro en la figura 4.38 para crear nuestro contador síncrono. El circuito completo lo podemos ver en la figura 4.40.

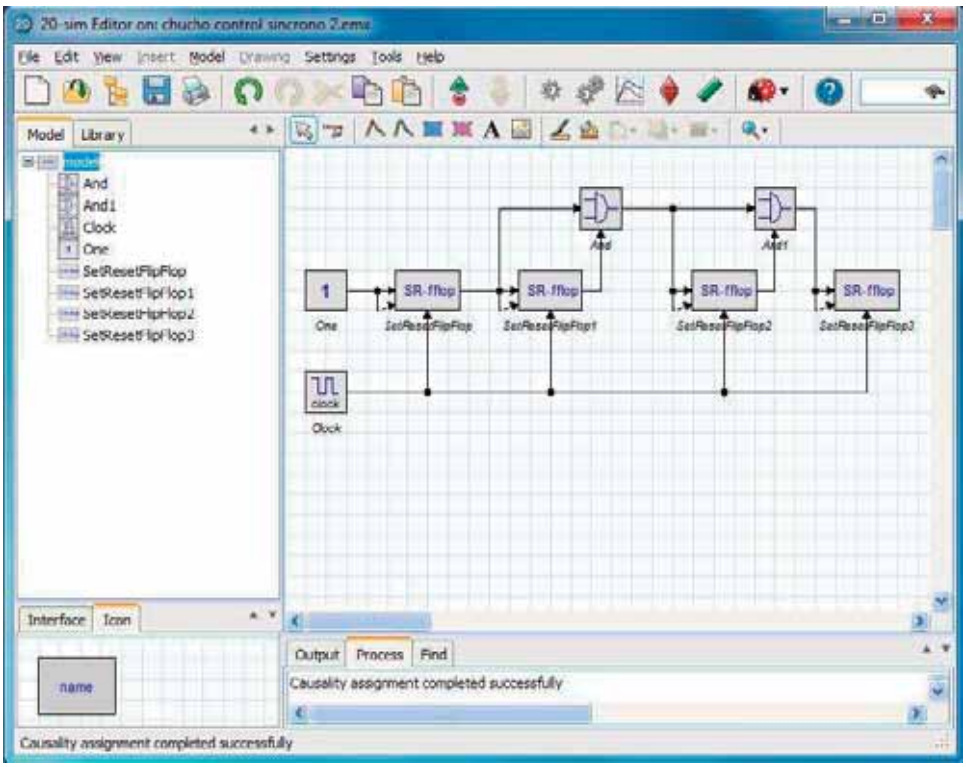


Figura 4.40. Contador asíncrono en 20-SIM

Como podemos observar la señal de reloj va conectada a todas las entradas de control de los Flip Flops, la salida negada del Flip Flop 0 (SetResetFlipFlop) se conecta a la entrada J y K del Flip flop 1 (SetResetFlipFlop) y también a la entrada de la compuerta AND1 (And), la salida negada del Flip Flop 1 (SetResetFlipFlop1) se conecta también a la entrada de la compuerta And (AND1). La salida de la compuerta AND1 (And) se conecta a las entradas J y K del Flip Flop 2 (SetResetFlipFlop2) y también a la entrada de la compuerta AND2 (And1). La salida del Flip Flop 2 se conecta también a la entrada de la compuerta AND2 (And1) y por último la salida de la compuerta AND2 (And1) se conecta a las entradas J y K de el Flip Flop 3 (SetResetFlipFlop3).

Una vez que tenemos el circuito conectado, solo basta con correr la simulación, en el monitor del simulador agregaremos la salida Q negada de cada uno de los Flip Flops, esto nos dará el conteo en cada una de las salidas. En la figura 4.41 se muestra la simulación del contador síncrono.

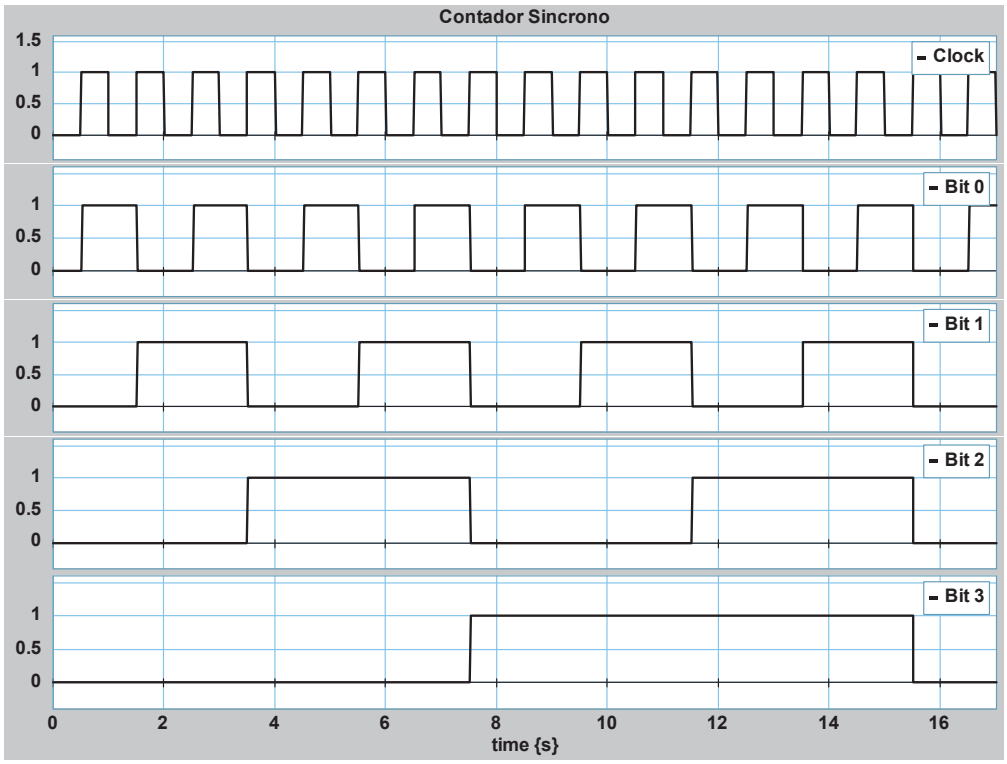


Figura 4.41 Simulación del contador síncrono.

Como podemos observar en la simulación, el Flip Flop 1 se dispara en cada pulso del reloj. El Flip Flop 2, el Flip Flop 3 y el Flip Flop 4, se disparan en la transición negativa del Flip Flop que lo antecede.

Como podemos observar, el contador asíncrono funciona a la perfección, la salidas Q negada de todos los Flip Flop indicarán el conteo por cada disparo del reloj.

Capítulo 5

Conclusiones y Recomendaciones

En este capítulo se presentaran las conclusiones y recomendaciones a las que se llego durante este trabajo de tesis.

5.1 Conclusiones

Al momento de hacer el marco teórico para este trabajo de tesis se puede concluir que los sistemas digitales son muy útiles y prácticos a la hora de diseñar y construir sistemas de control, debido a que los sistemas digitales son muy eficientes, seguros, y confiables.

También se puede concluir que los métodos de mapas de Karnaugh son una herramienta muy útil en el diseño de sistemas digitales, los cuales nos permiten reducir las funciones del sistema diseñado, a un circuito más simple, esto con la finalidad de reducir el costo de construcción.

20-SIM es un moderno software de modelado y simulación de sistemas que es capaz de adaptarse con facilidad y rapidez a diversas funciones, es capaz de simular el comportamiento de sistemas dinámicos tales como eléctricos, mecanismos e hidráulicos ó alguna combinación de cualquiera de ellos. 20-SIM es una herramienta fácil de usar ya que su interfaz es muy sencilla y permite trabajar con una gran cantidad de herramientas con las cuales se puede modelar y simular cualquier tipo de sistema. Los resultados dados por el simulador son entregados en gráficas que son de fácil entendimiento. El simulador cuenta con bastantes herramientas con las cuales podemos cambiar las propiedades de la simulación, tales como el tiempo de simulación, los parámetros de salida, el método de simulación y el color de las graficas, entre otras.

Se puede concluir que el sistema de alarma es un sistema muy sencillo y eficaz, ya que con ayuda de mapas de Karnaugh se pueden reducir las funciones a una forma más simple. 20-SIM permitió simular el sistema y con esto podemos concluir que nuestro sistema de alarma funciona a la perfección con respecto a las diferentes condiciones con las que fue creado el sistema.

Durante el diseño de un circuito combinacional que controla a un motor para un sistema aljibe-tinaco, se pudo observar que el uso de sistemas digitales es muy eficaz para realizar el control del motor.

Al realizar las simulaciones de los contadores tanto síncronos como asíncronos, podemos concluir que 20-SIM es una muy buena herramienta para observar el comportamiento de estos tipos de contadores, y que los resultados obtenidos fueros los esperados en el momento de su diseño.

Podemos concluir que 20-SIM es un software muy completo y fácil de usar, esto comparado con otros programas de simulación, ya que 20-SIM es más ligero y tiene bastantes herramientas con las cuales se puede interactuar fácilmente, además que el lenguaje de referencia permite modificar nuestros componentes para mejorar el diseño y simulación de nuestros sistemas.

Una de las limitaciones que tiene 20-SIM, es que no es un software libre y para utilizar el programa se tiene que comprar una licencia, de lo contrario no se puede trabajar en el.

5.2 Recomendaciones

Este trabajo de tesis se enfoca exclusivamente a sistemas digitales, en algunas partes a sistema de control. A continuación se muestran algunas propuestas ó recomendaciones para trabajos futuros:

1.- Mejorar el sistema de alarma aplicándolo a algún sistema hidráulico ó mecánico combinando alguno de los diferentes sistemas en 20-SIM.

2.- Agregar una bomba mecánica, un tinaco y sensores de flujo o presión en el sistema para controlar el motor, haciendo un sistema más complejo.

3.- Algunas de las otras aplicaciones que se les puede dar a los contador, es que con ellos se pueden construir contadores de décadas, se puede también implementar un reloj digital o se pueden usar para implementar un sistema de control de un estacionamiento, esto también se podría simular en 20-SIM.

Bibliografía

1. Lógica Digital y Diseño de Computadoras
   M. Morris Mano
   Editorial Prentice Hall

2. Sistemas digitales. Principios y aplicaciones
   8va Edición
   Tocci, Widmer
   Editorial Prentice Hall

3. Sistemas Digitales y Tecnologia de Computadores
   2da Edicion
   Javier García, Ignacio Angulo, José María Angulo
   Editorial Paraninfo

4. Fundamentos de Sistemas Digitales
   7ma Edición
   Thomas L. Floyd
   Editorial Prentice Hall

5. Diseño Digital
   3ra Edición
   M. Morris Mano
   Editorial Pearson Educación

6. Documentación 20-SIM
   http://www.20sim.com/downloads/manuals.html

7. Historia de las simulaciones
   http://es.wikipedia.org/wiki/Simulaci%C3%B3n_por_computadora