Sistemas Digitales y el Entrenador Lógico CE300.

Marco Antonio Pérez Cisneros[*] y Mark Readman[+]

[*]División de Electrónica y Computación, CUCEI, Universidad de Guadalajara, México.
[+]Consultor "Control Systems Principles"

SINOPSIS: Este es uno de una serie de articulos sobre modelos de sistemas, análisis y control, preparados por Mark Readman, control-systems-principles.co.uk para dar una idea de los importantes principios y procesos en control. En sistemas de control hay un número de métodos y sistemas genérico, los cuales son encontrados en todas las áreas de la industria y la tecnología. Estos libros pretenden explicar esos importantes métodos y sistemas en términos sencillos. Los libros describen lo que hace un tipo particular de método/sistema, cómo trabaja y después demostrar cómo controlarlo. Las demostraciones de control están hechas usando modelos de sistemas reales designados por nuestro fundador y señor padre Peter Wellstead, y han sido desarrollados para la manufactura por TQ Education and Training Ltd en su CE rango de equipamiento. Este libro usa la computadora basada en las herramientas de control y simulación del CE300 para demostrar.

1. Introducción.

En su educación técnica y académica, es usual para ingenieros de control estar entrenados en control de tiempo continuo y sistemas discretos de datos. En adición, usualmente se imparten cursos sobre Control Lógico Programable (PLC), pero raramente, los sistemas de control son explicados en forma que de toda la importancia a la lógica digital en la cual se basa el control. Es desafortunado, porque la mayoría de los sistemas de control más ampliamente usados son completamente utilizados en lógica digital, y casi todos los sistemas de control contienen un subsistema vital de lógica digital. Por ejemplo, un codificador óptico [2] como el usado con el CE300, es el análogo para el convertidor digital. El lazo cerrado de fase podría contener un detector de fase digital o ser completamente digital. Máquinas domésticas son controladas usando un sistema digital lógico secuencial. El Internet es un sistema completamente digital donde el tráfico es codificado, descodificado y controlado. Hay muchos mas ejemplos, pero con sólo basarse en éste, puede verse que es importante conocer las bases de los sistemas lógicos en orden para trabajar efectivamente como un especialista en sistemas de control.

En este manual, nosotros describiremos algunos experimentos de lógica digital usando el circuito entrenador lógico CE300. Este es una herramienta educacional para enseñar y aprender a cerca de la lógica digital y de los sistemas lógicos. Este tiene una interfaz la cual le permite a los sistemas lógicos ser designados, simulados y controlados. Primero nosotros vemos algunos sistemas lógicos combinacionales comunes. Luego, nosotros vemos algunos sistemas lógicos secuenciales y damos algunos ejemplos comunes en sistemas sincrónicos y asincrónicos. Este manual no está diseñado como una introducción a los sistemas lógicos, por lo que asumiremos que el lector tiene conocimientos básicos de álgebra Booleana, compuertas lógicas, tablas de verdad, mapas de Karnaugh y sistemas de números [3],[4]. La siguiente notación es usada:

$$a \quad \text{AND} \quad b \quad = \quad a\,b$$
$$a \quad \text{OR} \quad b \quad = \quad a + b$$
$$\text{NOT} \quad a \quad = \quad \overline{a}$$

En el sistema binario hay sólo dos números, 0 y 1. Con TTL, los circuitos lógicos binarios son representados por 0 Volts y 5 Volts respectivamente aunque otras formas convencionales son también

usadas como 3 Volts lógico. El TTL lógico tiene garantizado los límites. Con el CE300 es imposible fijar los límites y los tiempos de propagación para simular las especificaciones de las compuertas lógicas actuales.

2. Sistemas Lógicos Combinacionales.

Un sistema lógico combinacional tiene entradas y salidas (figura 1). Las salidas dependen solamente de las entradas. Para analizar circuitos lógicos combinacionales simples, usamos herramientas como las tablas de verdad y los mapas de Karnaugh [4]. Para sistemas más complejos nos auxiliamos de las herramientas proporcionadas por las computadoras.

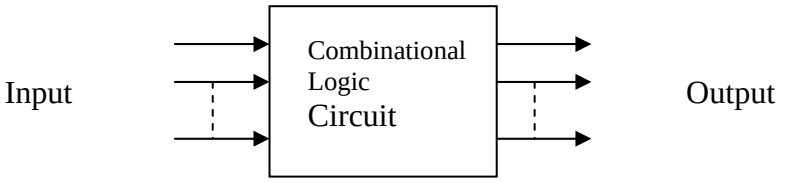


Figura 1. Las salidas "Outputs" son funciones lógicas de las entradas "Inputs".

2.1 Compuertas XOR.

Un simple pero versátil circuito lógico es la OR exclusiva o XOR. Las compuertas XOR son usadas extensivamente para construir codificadores y decodificadores [4] para codificar datos y como detector de fases en lazos cerrados de fase [1]. En una casa automatizada, la funcion XOR es programada dentro de la computadora para controlar la luz de dos o mas interruptores. Con dos entradas x_0, x_1 y salida *y* la expresión lógica para el XOR es:

$$y = \overline{x}_0\, x_1 + x_0\, \overline{x}_1 \qquad (2)$$

la cual es escrita también como $y = x_0 \oplus x_1$ (3)

x_0	x_1	y
0	0	0
0	1	1
1	0	1
1	1	0

Tabla 1. Compuerta XOR: tabla de verdad.

La Tabla 1 es la tabla de verdad para un XOR de dos entradas. Las dos filas sombreadas en la Tabla 1 indican cuándo la salida es alta y cuándo la ecuación 2 es verdadera. El bloque de la biblioteca CE300 para la compuerta OR puede ser configurado como una XOR. Es útil examinar un par de formas para implementar una compuerta XOR. En la figura 2 la implementación sobre la izquierda es una implementación funcional directa de la ecuación (2), mientras que la implementación de la derecha usa

solamente compuertas NAND. La equivalencia de estas dos implementaciones puede ser mostrada usando leyes de De Morgan, [3],[4].

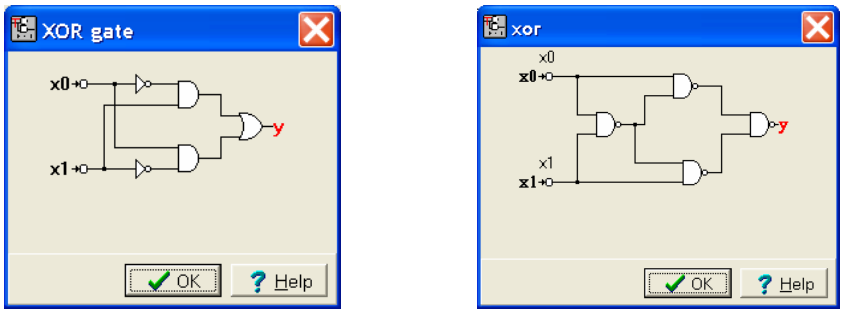


Figura 2. Dos implementaciones de compuertas XOR.

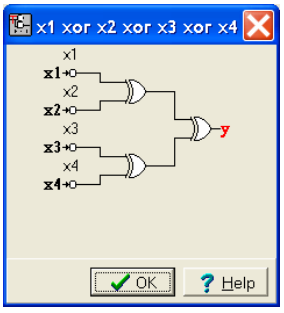
Una variación útil de la compuerta XOR es la compuerta XNOR la cual es el complemento de la compuerta XOR (4).

$$y = \overline{x_0 \oplus x_1} \qquad (4)$$

2.2 Compuerta XOR en cascada.

Varias variables pueden estar juntas en un arreglo de compuertas XOR, por lo que con cuatro entradas x_1, x_2, x_3, x_4 tendremos:

$$y = x_1 \oplus x_2 \oplus x_3 \oplus x_4 \qquad (1)$$



x_4	x_3	x_2	x_1	y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Figura 3. Realización y tabla de verdad para $y = x_1 \oplus x_2 \oplus x_3 \oplus x_4$

las cuales pueden ser implementadas con el CE300 como se muestra en la figura 3. La tabla de verdad para la ecuación (5) se muestra también en la figura 3.

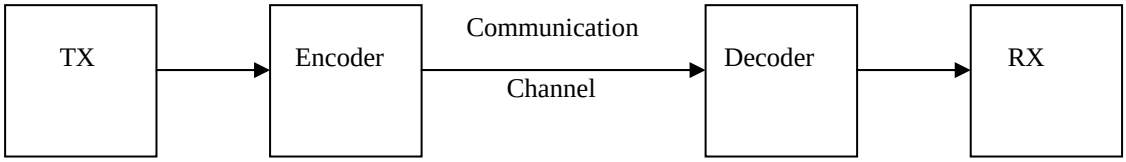


Figura 4. Canales de comunicaciones.

2.3 Compuerta XOR en prácticas.

Cuando la comunicación se realiza entre dos sistemas, el mensaje es codificado en el transmisor (TX) por el transmisor. En el receptor (RX), el mensaje se descodifica. Esto se hace para minimizar los efectos de ruido del canal y para la seguridad en los datos. Un ejemplo en el que se usan las compuertas XOR es un verificador de paridad. Un generador de parida cuenta el número de unos en cada byte (1 byte son 8 bits). Con paridad impar, el byte es aumentado en el MSB para asegurarse que el total de unos es impar. Mientras que con paridad par, el MSB es aumentado para mantener el número par de unos. De esta forma, cuando un byte es transmitido de un transmisor a un receptor, el receptor puede analizar el número correcto de unos en cada byte. Obviamente que el transmisor y el receptor tienen que estar de acuerdo para usar el mismo examinador de paridad. Generar los bits de paridad es fácil usando compuertas XOR. El generador de paridad de cuatro bits disponible en el CE300 tiene entradas A, B, C y D. De la tabla de verdad se puede ver que los bits de paridad par p_e y la impar p_o pueden ser generados usando tres compuertas XOR (6).

$$p_e = A \oplus B \oplus C \oplus D$$
$$p_o = \overline{A \oplus B \oplus C \oplus D} \qquad (6)$$

La implementación CE300 XOR del identificador de paridad es mostrada en la Figura 5. Una compuerta XOR adicional permite seleccionar el tipo de paridad.

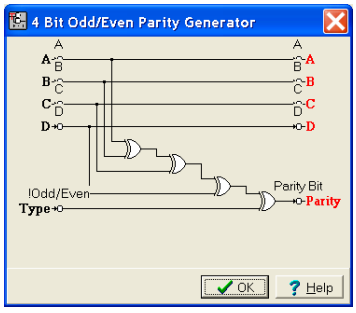
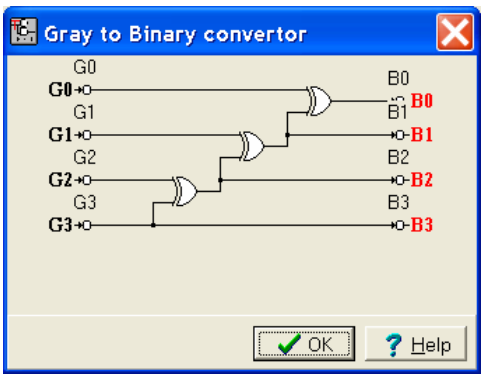


Figura 5. Checador de Paridad.

Otra aplicación que usa compuertas XOR es en las conversiones de código Gray a binario y viceversa como el usado en el manual sobre servosistemas de control digital [5]. Un codificador de código Gray es usado para codificar la posición de un eje. Un descodificador es usado para convertir el código Gray

a binario. Suponiendo un eje codificador de código Gray (Figura 6b) como en el descodificador lógico CE300, las ecuaciones son:



Binary			Gray		
b_2	b1	b0	g2	g1	g0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

Figura 6. a) Convertidor Gray-Binario, b) Codificador de Gray, c) Tabla de verdad

$$B_0 = G_3 \oplus G_2 \oplus G_1 \oplus G_0$$
$$B_1 = G_3 \oplus G_2 \oplus G_1$$
$$B_2 = G_3 \oplus G_2$$
$$B_1 = G_3$$

donde G_0-G_3 es el cuarto bit del código Gray del eje codificador y B_0-B_3 es el cuarto bit descodificado en código binario. El descodificador lógico es implementado usando compuertas XOR tomadas de la biblioteca del CE300 conectadas como se muestra en la figura 6ª. Esta idea puede ser extendida a algún número de bits y frecuentemente 12 o 16 bits del eje codificador son usados en prácticas. Una buena razón para usar código Gray es que hay sólo un cambio de bit entre dos posiciones adyacentes del codificador como se ve en la tabla de verdad de la Figura 6c. Esto reduce la posibilidad de que un error sea introducido cuando el codificador está rotando. Con un codificador binario todos los bits pueden cambiar simultáneamente entre dos posiciones adyacentes. Esto introduce la posibilidad de que se introduzca un error debido al ruido y a la sincronización. Claro que los errores por ruido o sincronización están presentes cuando se usa código Gray, sin embargo, debido a que sólo un bit cambia en un tiempo, la posibilidad de que ocurra un error se reduce significativamente.

2.4 Aritmética.

Una función básica de un circuito lógico combinacional es calcular sumas, restas, multiplicación y división en binario. Hay muchas formas de implementar la multiplicación de dos números binarios. Una forma es la lógica combinacional y full adders. La multiplicación de dos números puede ser implementada también mediante un ROM como el de la figura. EL CE300 tiene 256 bytes de ROM la cual tiene una dirección de 8 bits y una salida de igual número. Multiplicando 2 números binarios de 4 bits resulta en un producto de 8 bits. El cuarto bit multiplicando y el cuarto multiplicador se combinan para formar la dirección de 8 bits. Luego, simplemente calculamos la tabla de verdad, la cual tendrá 256 entradas. Para cada dirección hay un término producto el cual es acomodado en su posición correspondiente de la memoria para esa dirección de la ROM. Por lo que la ROM mantiene todos los productos que resultan de multiplicar 2 números de 4 bits. Un programa de CE300 para éste se muestra

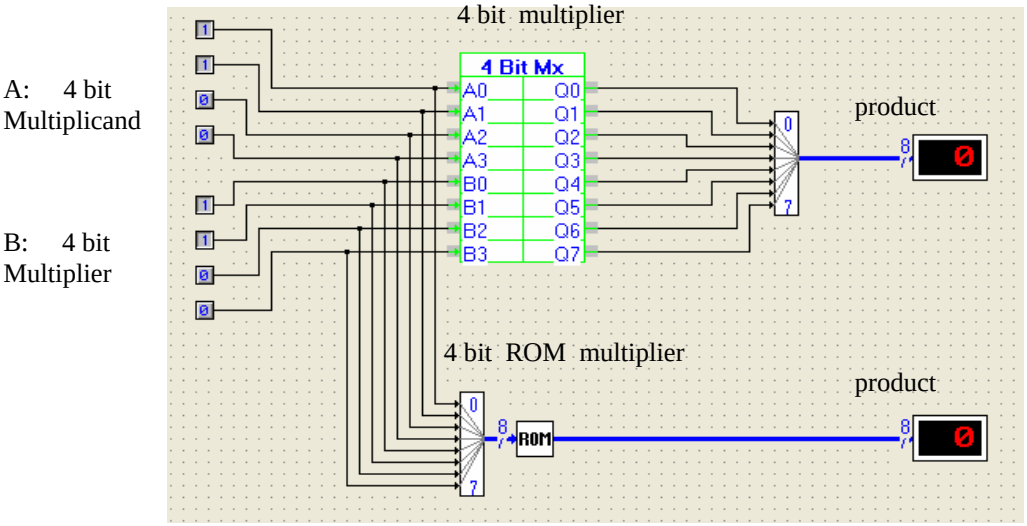


Figura 7: Implementación ROM de la multiplicación binaria: FourBitMul.ic3

en la Figura 7. Por comparación, el multiplicador de 4 bits disponible en el CE300 calcula el mismo producto, por lo que las dos respuestas pueden ser comparadas.

La tabla de verdad es fácil de generar usando los siguientes comandos de Matlab:

$A=[0:15]; Zh=dec2hex(A^{T}A)$

Ahora Zh es un vector columna en Hex de todos los posibles productos de los números de 4 bits y puede ser introducido directamente a la ROM comenzando en la dirección 00H.

Observe que hemos usado una ROM para implementar una tabla de verdad. Obviamente una ROM tiene memoria por lo que no es un circuito lógico combinacional. Sin embargo es útil saber que una tabla de verdad puede ser implementada en una ROM.

3. Sistemas Lógicos Combinacionales y Máquinas de Estado Finito.

Una diferencia importante entre un circuito lógico combinacional y uno secuencial es la presencia de memoria. La memoria requiere retroalimentación. Un sistema lógico secuencial usa flip-flops para acomodar el estado recurrente de la máquina. Por tanto, en un sistema lógico secuencial la salida dependerá del estado actual y de la entrada. La Figura 8 representa un sistema lógico secuencial o máquina de estado finito. Con una máquina de Moore, la salida es solamente una combinación lógica del estado actual. Si la salida depende del estado actual y de la entrada, entonces se refiere a una máquina de Mealy. Un ejemplo simple de un sistema secuencial es el flip-flop S-R como el que se muestra en la figura 9. Este es una máquina de Moore porque la salida es sólo el estado Q. En el estado set, Q=1 y en el estado reset Q=0. El diagrama del estado de transición para el flip-flop S-R se muestra en la figura 10.

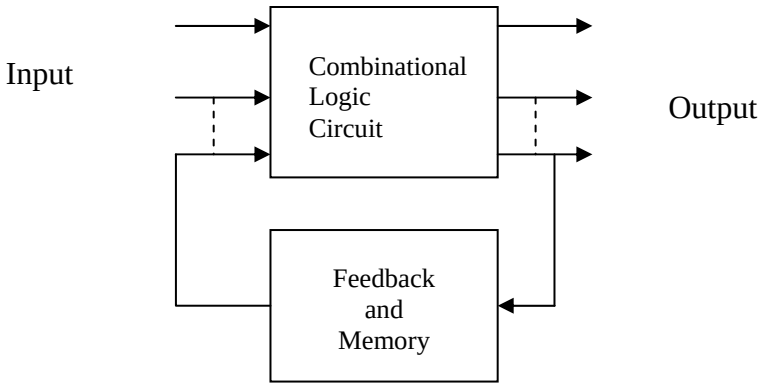


Figura 8. Máquina de estados finitos.

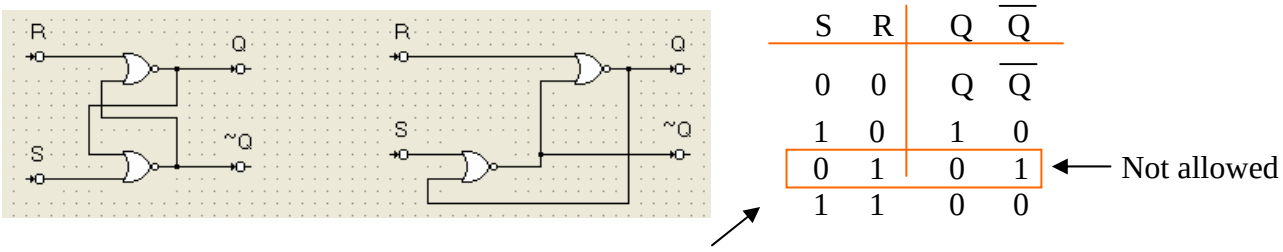


Figura 9. Flip Flop SR y tabla de verdad.

Flip Flops como el JK flip flop son elementales en la construcción de máquinas secuenciales.

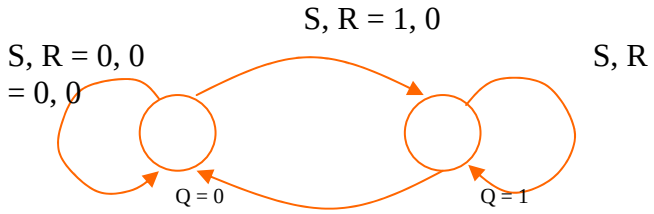


Figura 10. Diagrama de transición de estados SR.

3.1 Registros de Cambio.

Un registro de cambio es un sistema secuencial usado para acomodar un cambio de dato. Un registro de cambio de 4 bits consta de cuatro flip-flops J-K conectados como se muestra en la Figura 11.

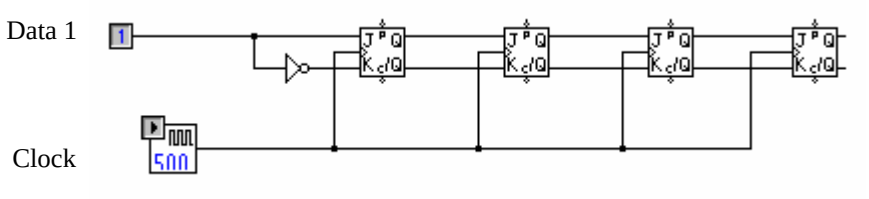


Figura 11: Registro de corrimiento de 4 bits: JKFFReg.ic3

Cada flip-flop acomoda un bit de dato. Datos nuevos llegan a la entrada y con cada pulso de reloj el dato es cambiado a la derecha. La idea es ser capaz de acomodar y mover datos dentro y fuera del registrador de cambios con respuesta a señales de reloj. El CE300 contiene registros de cambios de 4 bits. Registros de cambios pueden ser sincrónicos o asincrónicos. Una aplicación común es en conversiones serie-paralelo, donde en el siguiente ejemplo, cuatro bits de datos son cargados dentro del registro y luego de procesarlo puede ser leído en paralelo.

3.2 Retroalimentación Lineal del Registrador de Cambios.

Para usar la retroalimentación con un registrador podemos construir sistemas lógicos dinámicos que se comportan en interesantes y útiles formas. Cuando las compuertas lógicas XOR son usadas para retroalimentar las combinaciones lógicas de las salidas del registrador de cambios se le conocen como Registrador de Cambios con Retroalimentación Lineal (LSFR). LSFR's son usados para generar

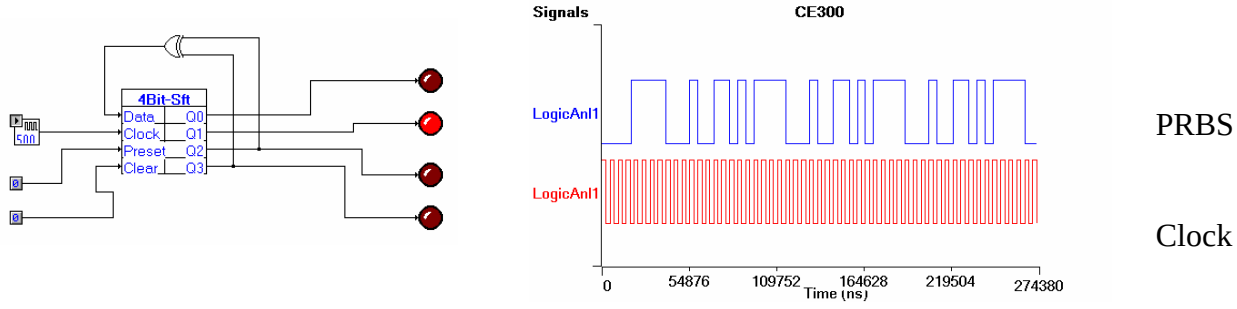


Figura 12: Generador 4 Bit PRBS: PRBS.ic3

señales periódicas con ruido de banda blanca limitada como propiedades de estática llamada Pseudo Random Binary Signals (PRBS) [3]. Si el periodo es suficientemente grande entonces sobre un tiempo menor a un periodo el signo aparecerá aleatoriamente. Estos signos pueden ser usados para codificar datos como un banco de datos de Internet o identificar las dinámicas de un sistema desconocido [2]. En el sistema CE300 mostrado en la figura 12 un LSFR de 4 bits es usado para generar una señal PRBS. La señal de reloj (roja) y la salida (azul) de una de las salidas son también mostradas en la figura 12. El registrador puede ser en cascada. Por ejemplo, dos registradores de 4 bits pueden ser colocados en cascada para hacer uno de ocho bits y un LFSR más grande. Con un cantador de pulsos, el MSB es

retroalimentado a la entrada del registrador por lo que los datos circulan alrededor del registrador de cambios. Con un doble contador de pulsos el complemento de la salida es retroalimentada para el dato de entrada. En la figura 13 hay un doble contador de pulsos que es usado para generar una salida de tres fases Q0, Q1 y Q2.

Esto es para generar tres señales periódicas con fase de 120° unas de otras. Observe que solamente la

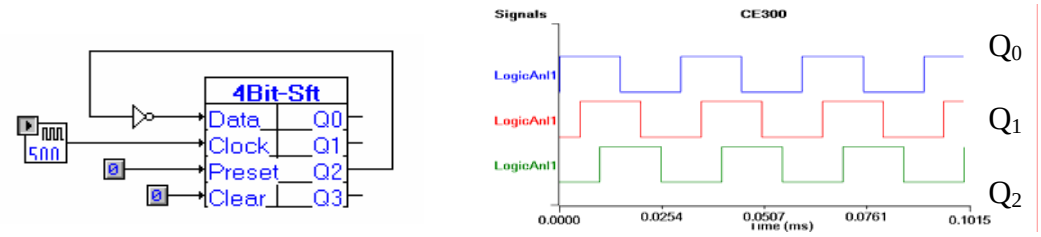


Figura 13: Contador de anillos entrelazado: twistedR.ic3

mitad de los estados son visitados con este contador y el MSB no es usado. Esto podría causar problemas si el estado inicial no es uno de los estados deseados.

3.3 Contadores Sincrónicos y Asincrónicos

Los contadores son una clase importante de los circuitos lógicos secuenciales. Un contador está construido usando flip-flops conectados por lo que la entrada de cada flip-flop es una función lógica de los estados. En un contador sincrónico cada flip-flop es manejado con la misma señal de reloj por lo que el estado de cada flip-flop cambia simultáneamente. Con N flip-flops la cuenta máxima es 2^N y el contador puede ser arrancado para contar hacia delante o atrás en binario. Para designar un contador, comenzamos con una tabla de estado que enlista todos los posibles estados del contador y el siguiente estado. Estados no usados son llamados “no importa”. De la tabla de estado podemos decidir la entrada a los flip-flops y usar un mapa de Karnaugh para minimizar el número de compuertas lógicas.

Por ejemplo, con tres flip-flops hay ocho estados y el conteo máximo es 2^3-1=7=111b. Usando tres flip-flops J-K conectados como se muestra en la figura 14 el contador contará hacia delante hasta siete en binario y luego regresará a cero. Por eso le llamamos un contador de módulo 8. Debido a que hay estados no utilizados no debemos preocuparnos por condiciones iniciales. Para cualquier condición inicial el contador continuará con la secuencia correcta. Sin embargo hay muchas aplicaciones donde un contador tiene que contar en alguna otra base numérica o código como código Gray.

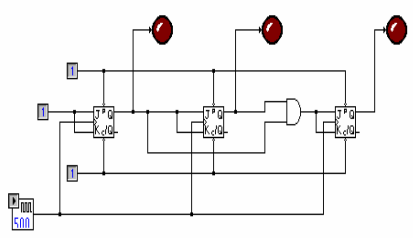


Figura 14: Contador de Modulo 8: syncount.ic3

El siguiente ejemplo muestra tres flip-flops JK conectados como un contador de módulo 5 (Figura 15). Por lo tanto la secuencia será 0,1,2,3,4,0,1… etc. Cuando el contador alcanza 4=100b entonces en el

siguiente pulso de reloj el contador tiene que ser forzado dentro del estado 0. Hay ocho posibles estados, tres de los cuales no son usados. Nos gustaría estar seguros que al comenzar de alguna condición inicial el contador terminará en el correcto estado. Idealmente si la condición inicial es un estado no deseado, nos gustaría forzar al contador a comenzar de cero [4].

De una tabla de estados podemos elaborar las entradas a los flip-flops en cada pulso de reloj. Estados no usados son regresados a cero de donde la cuenta comienza normalmente. Una aplicación común de contadores es para la división de frecuencia.

Sistemas asincrónicos son acontecimientos llevados por una entrada externa. Cada flip-flop genera una señal de reloj para el siguiente. Por ejemplo, el contador sincrónico de la sección previa puede ser implementado como uno asincrónico (Figura16). La compuerta NAND detecta cuándo el contador alcanza 101 y resetea el flip-flop JK a cero. Este flip-flop es un match usado para sintonía clara de los flip-flops. Esto previene problemas de sincronía debidas a ligeras diferencias de tiempos de reset para cada fip-flop. Observe la punta sobre el trazo más bajo de la figura 16 ocurre cuando los flip-flops JK están en reset para comenzar una nueva secuencia de conteo.

3.4 Control de retroalimentación de un contador up/down módulo 4.

Son muy comunes los sistemas dinámicos donde la retroalimentación es usada para controlar un sistema lógico. En la figura 18 se muestra cómo la retroalimentación puede ser usada para controlar un contador up/down. La referencia en este ejemplo es un generador de señal alimentado para un convertidor de digital a análogo. Esto proporciona una cuenta de referencia de cuatro bits en binario. Una señal de error es generada para comparar la salida del contador a la referencia de entrada usando un sumador de 4 bits. La salida del sumador es alimentada por compuertas OR las cuales controlan un reloj usando una compuerta AND y permitiendo al contador contar hacia delante o hacia atrás. Los bloques convertidores DA son usados para dar una indicación visual de la cuenta y la referencia. El contador cuenta la señal de referencia. Nota: Todos los bloques DA y AD están colocados para 4 bits. Reemplazando el contador por el servomotor CE300 y el codificador a código Gray

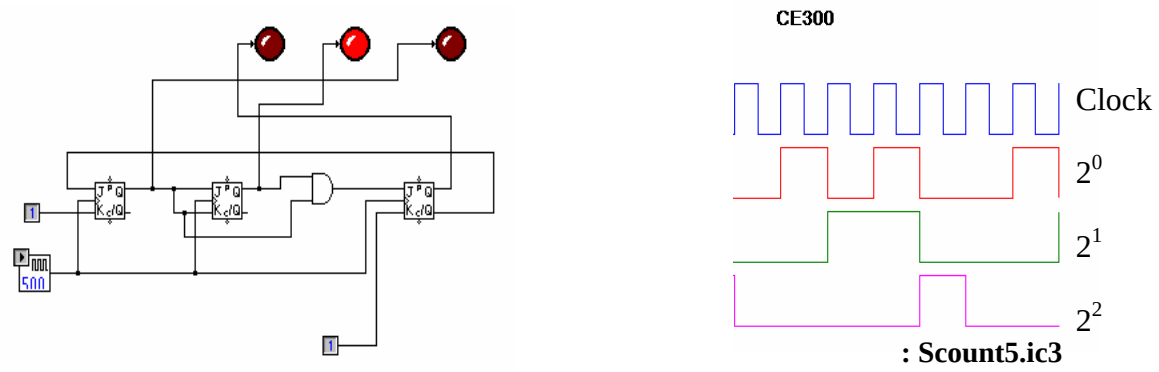


Figure 15. Modulo 5 synchronous up counter

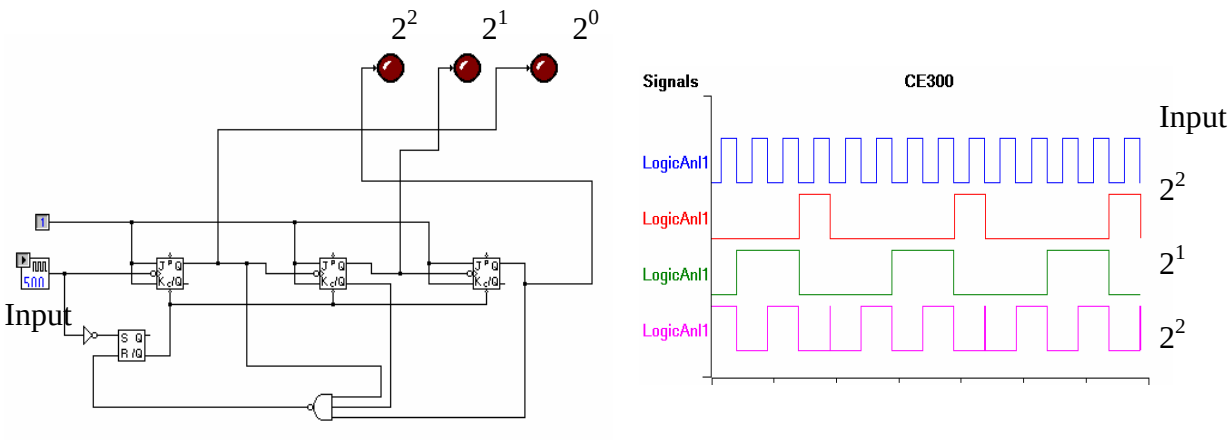


Figure 16: Modulo 5 asynchronous up counter: Acount5a.ic3

4. Toda la fase digital del lazo cerrado.

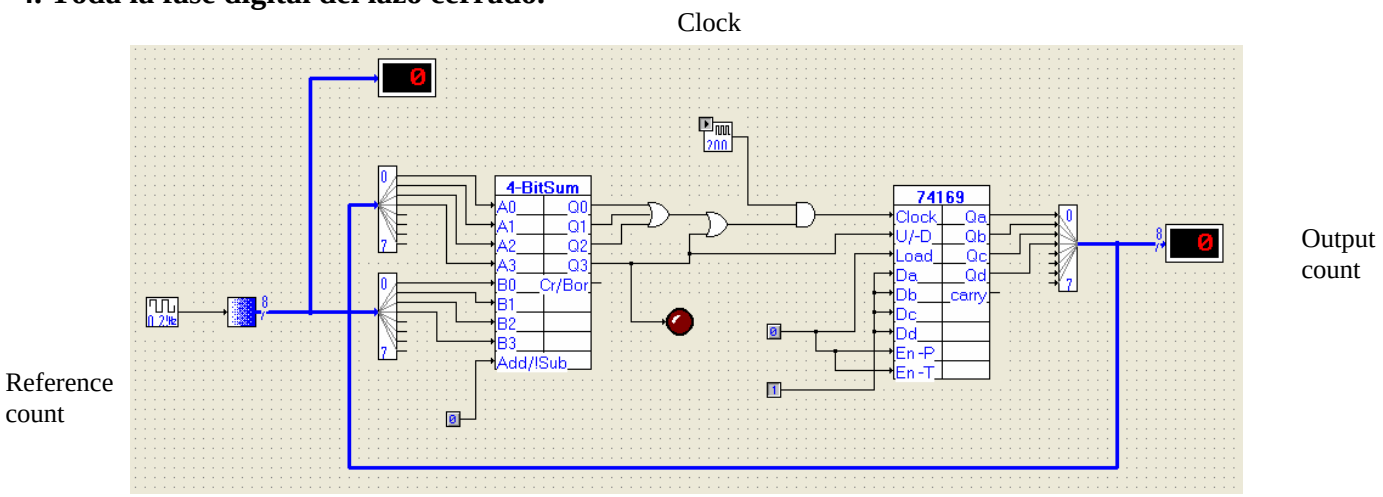


Figura 18: Control retroalimentado de contador: CountCon.ic3

Un lazo de fase cerrada (PLL) es un sistema retroalimentado donde un oscilador controlado de voltaje (VCO) es llevado por un filtrador de señal de error por lo que este lleva la cuenta de la señal de referencia [1]. Para más detalles y análisis de un PLL análogo ver el manual de PLL [8]. Un desarrollo más lejano del tradicional PLL analógico totalmente digital PLL o DPLL. El DPLL no tiene componentes analógicos. Los componentes básicos de un DPLL son los detectores de fase (DPD), los filtros de lazo digital (DLF) y un oscilador controlado digitalmente (DCO). Una implementación CE300 de un DPLL es mostrada en seguida. La referencia de entrada es una señal de reloj con una frecuencia de referencia deseada. La detección de la fase y el filtro es completada usando dos contadores y un sumador de 4 bits. Los contadores se comportan como filtros pasa bajas [1]. La salida del sumador lleva el DCO. En el CE300 es completada usando un convertidor de 4 bits AD, el cual lleva un VCO. Cuando el sistema es cerrado, la salida del sumador mantiene el VCO en fase con la frecuencia de referencia. Si la frecuencia de referencia se incrementa causará la cuenta de red de los dos contadores para incrementar hasta que el error de fase sea cero otra vez. La figura 22 muestra que la salida (azul) del DCO puede cerrarse sobre la referencia de entrada (rojo).

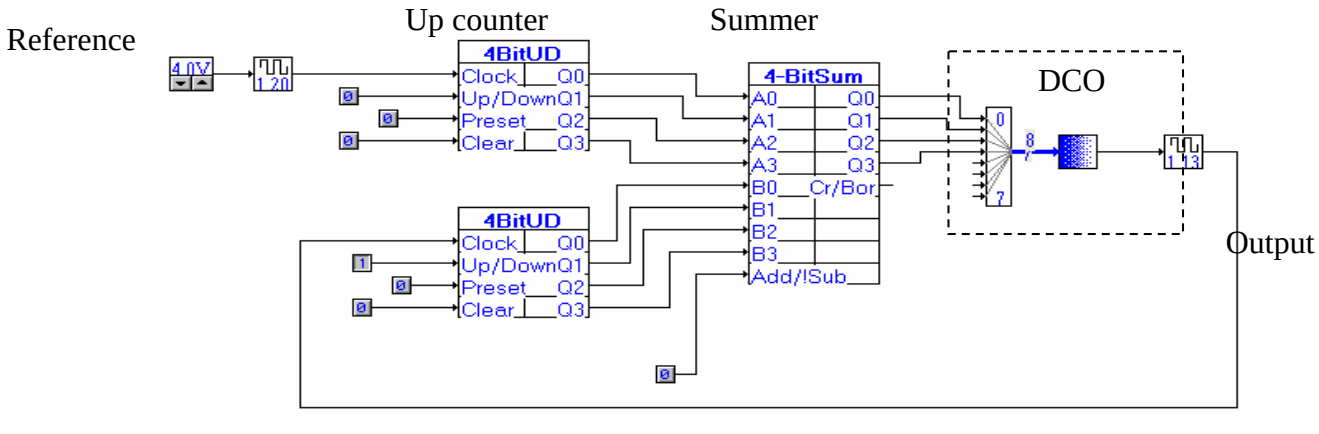
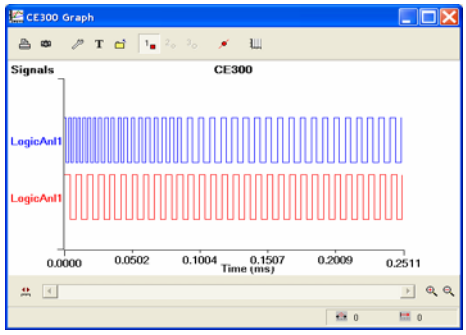


Figura 21:CE300 simulación DPLL: DPLLSim.ic3



DCO output

Reference

Figura 22: DPLL, seguro en referencia (baja).

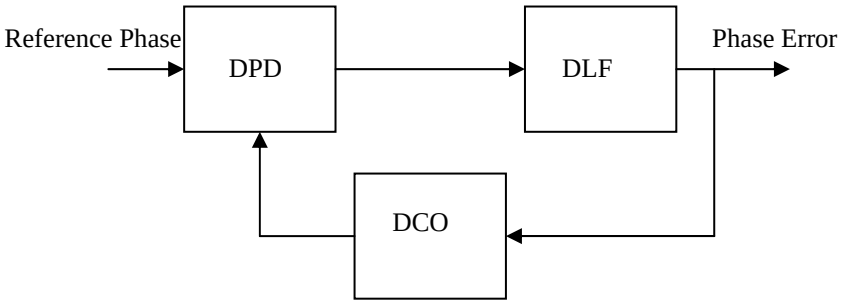


Figura 20: Diagrama a bloques del DPLL.

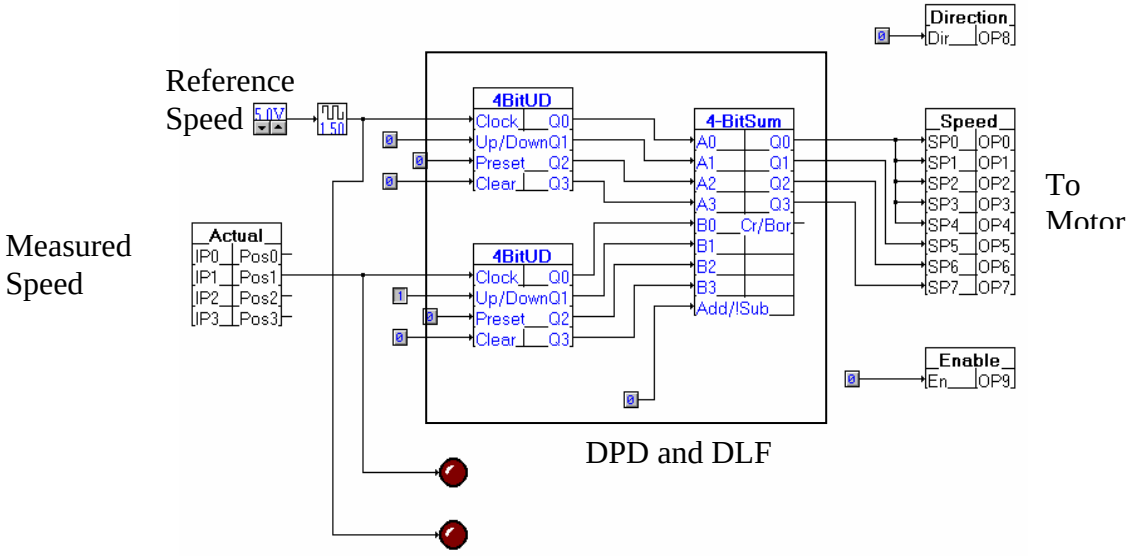


Figura 23. CE300 DPLL control velocidad de motor

4.1 Aplicación: Control de la Velocidad de Motor ADPLL.

Una aplicación típica de un PLL es un controlador de velocidad de un motor. Hemos visto en el manual de control de posición digital [8] cómo el control de la posición angular de un sistema auxiliar usando un codificador y un descodificador. Aquí es usado un codificador para generar pulsos proporcionales a la velocidad del motor y las acciones del motor como un DCO. Remplazando el VCO y el convertidor AD en la simulación de la figura 21, por el motor auxiliar CE300 nos permite controlar la velocidad del motor auxiliar. El codificador de posición relativa es fijado al motor auxiliar y la salida del sumador es usada para manejar el motor. El programa del CE300 es mostrado en la figura 23. Dos LED's son usados para dar una indicación visual de que la velocidad del motor auxiliar se ha cerrado sobre la velocidad de referencia. El bloque de alcance puede ser para registrar y mostrar la referencia y la señal de salida del motor, por lo que ellos pueden ser observados para ser cerrados (Figura 24). La solución es restringida porque solamente usamos un sumador de 4 bits dando 16 velocidades posibles.

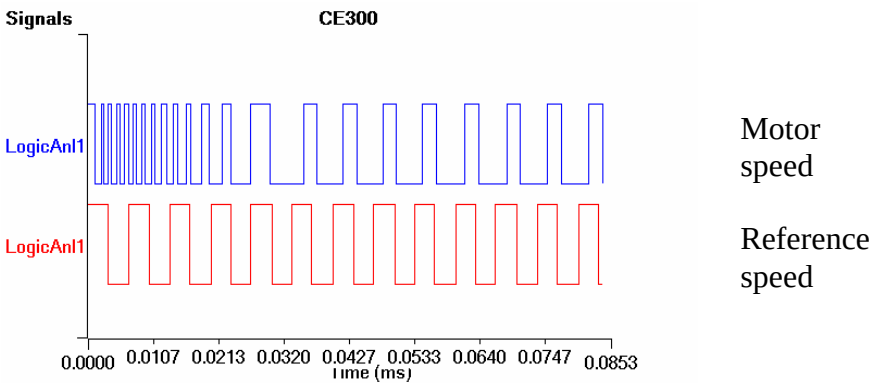


Figura 24. CE300, seguro (lock) en la referencia del motor servo.

Con un PLL analógico el motor agiliza la función de transferencia del motor/codificador actuando como un DCO. Éste puede ser aproximado por:

$$P_m(s) = \frac{K_m K_z}{s} \qquad (7)$$

donde K_m es la constante del motor y K_z es el número de segmentos en el codificador [1].

Conclusiones.

En este manual hemos mostrado cómo el Circuito Entrenador Lógico CE300 puede ser usado como un una ayuda en la enseñanza para aprender y observar algunas aplicaciones básicas de circuitos lógicos combinacionales y secuenciales. Los circuitos lógicos son cruciales para la correcta operación de muchos sistemas de control. Combinado con sistemas de control tradicionales tenemos una combinación compleja de sistemas de control que son un tema recurrente de investigaciones. El control retroalimentado de sistemas digitales como controlar el tráfico de Internet, es un problema de control digital. Esto es comúnmente llamado sistema de evento discreto.

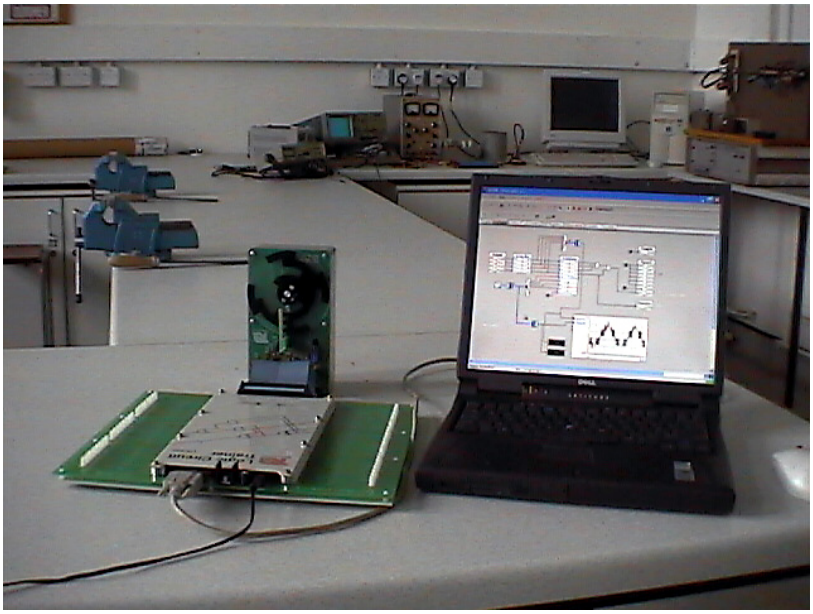


Figura 25. Entrenador lógico CE300 con el Servo motor CE300b.

6. Palabras finales.

No es posible responder preguntas sobre nuestro manual, a menos que tengamos un contrato con su organización. Para más información acerca del CE300 Control y del programa de simulación dirigirse al sitio de internet de TQ Education and Training usando la liga de nuestra página web www.control-systems-principles.co.uk o la dirección de correo electrónico info@tq.com. Existen muchos libros sobre lógica digital y sistemas digitales, así como sitios web dedicados a estos temas. Nosotros estamos agradecidos particularmente con las siguientes referencias.

7. Referencias.

1. Abramovitch D Y, Phase-Locked Loops: A Control Centric Tutorial, Proc 2002 ACC, Anchorage/

2. Healey, Martin, 1935-. - Principles of automatic control. – 3rd ed. - London : English Universities Press, 1975

3. B. Holdsworth, and C. Woods., Digital Logic Design, 4th Edition, Newnes 2002.

4. Horowitz and Hill, The Art Of Electronics, 2nd Ed, CUP 1989.

5. Readman Mark., Servo Control Systems 2: Digital Servomechanisms, www.control-systems-principles.co.uk

6. CE300 Logic Circuit Trainer, Students Guide. TQ Education and Training Ltd, 2000.

7. CE300 Logic Circuit Trainer, Instructors Manual. TQ Education and Training Ltd, 2000.

8. Readman Mark, Phase Locked Loops, www.control-systems-principles.co.uk