Universidad de Costa Rica

Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
IE0523 – Circuitos Digitales II
I ciclo 2018

Tarea 2

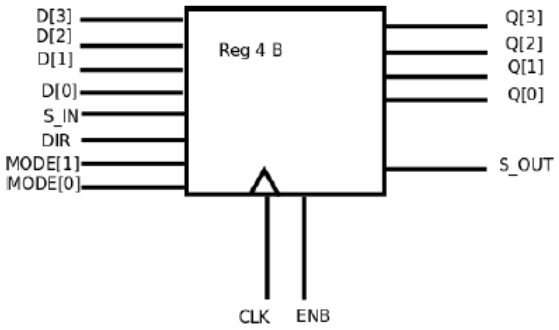
Andrés Alvarado Velazquez, B30316

Profesor: Jorge Soto.

4 de abril del 2018

Resumen:
El siguiente reporte presenta los resultados del experimento realizado con la finalidad de obtener un objeto registro de 4 bits capaz de rotar en círculo o línea y también con la posibilidad de cargar datos o leer de forma paralela, se realizó la creación del módulo registro el cual cuenta con las señales de entrada y salida requeridas, el registro de un nibble se encuentra conformado por cuatro registros de un bit o flip-flops los cuales a la hora de una lectura en paralelo leen la respectiva señal o aportan esta en caso de que se lea la salida Q, cuando se coloca el registro en modo escritura 00, este replica a la salida paralela el valor que se encuentre en la entrada y conserva este hasta el proximo cambio de entrada y flanco ascendente, se cuenta también con el modo rotación el cual hará rotar los datos en el registro una vez cada flanco ascendente de reloj, en la dirección indicada por la entrada DIR.
Para la parte del desarollo del registro de 32 bits se utilizaron dos registros de 16 bits conformados a su vez por dos registros de 8 bits, los cuales se han construido con dos registros de 4 bits.
Se realizaron varias pruebas de SET de valores en los registros y rotación de datos, los resultados fueron observables con la utilización del software GTKwave.

1. Descripción arquitectónica:



1.1 Registro de cuatro bits:
Se posee un registro unidad cuya función principal es el operar sobre los cuatro valores que son posible de setear en paralelo a través de las entradas D[3:0] o inclusive la entrada serial S_IN, la entrada DIR se encarga de señalar por medio de un valor boleano la dirección de rotación deseada, sea 0 a la izquierda o 1 a la derecha, las entradas de modo, MODE[1:0] permiten el selecionar la función a realizar por el registro, rotación izquierda, derecha, rotación circular, set de los valores en paralelo.
El circuito cuenta con una entrada de habilitación ENB, la cual es al igual que DIR un valor boleano con el único fin de activar las entradas de modo y lectura en paralelo en el caso en que la señal sea ENB == 1, caso contrario el registro ignora todas las señales a excepción de esta última.

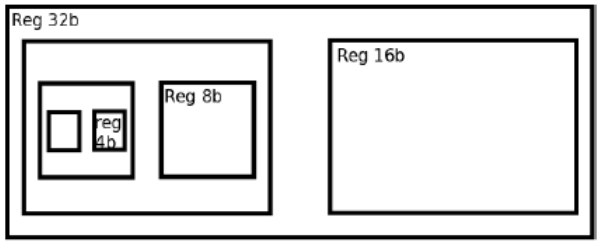
Las salidas en paralelo son Q[3:0] se definen como las salidas propias de los registros individuales de un bit.

La salida S_OUT es propiamente el bit desplazado de la estructura en caso de realizar una rotación, el valor de S_OUT se mantiene en cero en caso de que no se estén realizando rotaciones lineales.

Lo que se desea es en sintesis el operar el registro en 3 modos básicos, operación en paralelo, desplazamiento serial y rotación circular.

1.2 Registro de treinta y dos bits:
Se realiza a partir de la unión de dos registros de 16 bits, los cuales a su vez se conforman de 2 de 8

bits, y estos últimos se crean con dos registros de 4 bits cada uno.



Basicamente el registro de 8b se encarga de entregar cuatro bits a un registro y 4 al sobrante, el registro de 8b entregará también la señal de CLK, ENB y DIR, a los registros slave, sin embargo entregará un modo interno decidido por el registro master, en caso de rotación circular el registro de 8b hará a los slave rotar de forma lineal uniendo la salida de uno con la entrada del otro y viceversa esto permitirá que el registro de 8b en sí tenga una rotación circular, en caso de desear una rotación lineal el registro master unirá la salida ya sea del registro con el bit más significativo cuando la rotación es a la izquierda, o unirá la salida a la salida del registro con el bit menos significativo en rotación a la derecha, y ambos casos la entrada a la entrada del otro registro, se unirán entre estos para permitir una rotación lineal del registro como un total, en ambos casos, tanto en rotación lineal como en rotación circular, los registros internos únicamente rotan linealmente, pero el registro como un todo lo hace de ambas formas, finalmente este mismo algoritmo master-slave se implementa para el registro de 16b usando registros 8b y para el registro de 32 bits usando dos de 16b, el código perteneciente a estos registros es idéntico con la excepción de que cambian la cantidad de hilos de entrada y salida en paralelo, y el nombre de los registros slave internos, por lo demás los métodos se conservan intactos.

2. Plan de pruebas:
Se planea en primer lugar el realizar la rotación a la izquierda, para esto se usa un modulo "Probador" ubicado en el fichero probador.v, este módulo cuenta con todas las señales de salida necesarias para operar sobre las entradas del módulo registro, en primer lugar se inicia la pueba con un tiempo muerto igual a 10 unidades de tiempo de simulación.

Prueba 1: Rotación a la izquierda:

Se colacan las entradas ENB =1, DIR =0, S_IN= 1, se utiliza el modo 10 para setear el registro en 0000, posteriormente se regresa al modo 00 en t6, es posible el observar como los bits se desplazan llenando el nibble de unos.

Prueba 2: Rotación a la derecha:

10 tiempos discretos después se hace DIR=1; con lo cual se a cambiado la direción de rotación, se coloca S_IN = 0, un tiempo discreto después se coloca en uno, y se alterna de esta manera varias veces.

Prueba 3: Rotación circular:

Por medio del probador se coloca el modo en 01 y se setea la direción en 0 para girar a la izquierda, se observa una vuelta completa y media al esperar t6.

Prueba 4:

Pasados 6t se cambia el sentido de rotación al variar DIR y se observa lo sucedido.

Prueba 5:
Se demuestra la función de escritura en paralelo por el resto de la simulación junto con la verificación del ENB. Para esto se hace que D tome varios valores y se observa su comportamiento a la salida Q pasado un flanco ascendente.

Prueba 6:
Con ayuda del probador "Probador" presente en el fichero probador_32b.v y el banco_de_trabajo_32.v se espera el observar el funcionamiento del registro implementado con registros de menor capacidad.

Tanto probador.v como probador_32b.v cuentan con comentarios en el código que muestran las distintas etapas de verificación del funcionamiento del registro.

Se realiza la siguiente prueba en este caso. Inicialmente el reloj permanece por 10t en un estado desconocido, el modo se setea en 20t con lo cual el registro se lleva a un estado conocido, se deshabilita el registro y se hace cambiar varias veces la entrada, la cual debe ser ignorada, se habilita nuevamente el registro y en este caso la salida Q debe reflejar la entrada pasado un ciclo de reloj.

Se introducen varios valores de D en el registro y se observa la salida Q.
Desde 172t a 232t se realiza una rotación circular manteniendo DIR en 0, con lo cual se mueven hacia la izquierda los valores, posteriormente se cambia el modo a rotación lineal y se mantiene la rotación dirección izquierda, no deben aparecer nuevos valores ya que S_IN se encuentra para entonces en 0, en 272t se hace el cambio de direción, con lo cual ahora el registro gira hacia la derecha, para este punto el registro a limpiado gran parte de su memoria. En el tiempo 312t se regresa el registro al modo 10, con lo cual este escucha nuevamente las entradas paralelas y setea Q.

En 362t se cambia el modo a 01 con DIR=1, con lo cual el registro rota de forma cerrada hasta 412t donde DIR = DIR, posteriomente se mantiene un rotado circular hasta el final de la simulación en tiempo=500t.

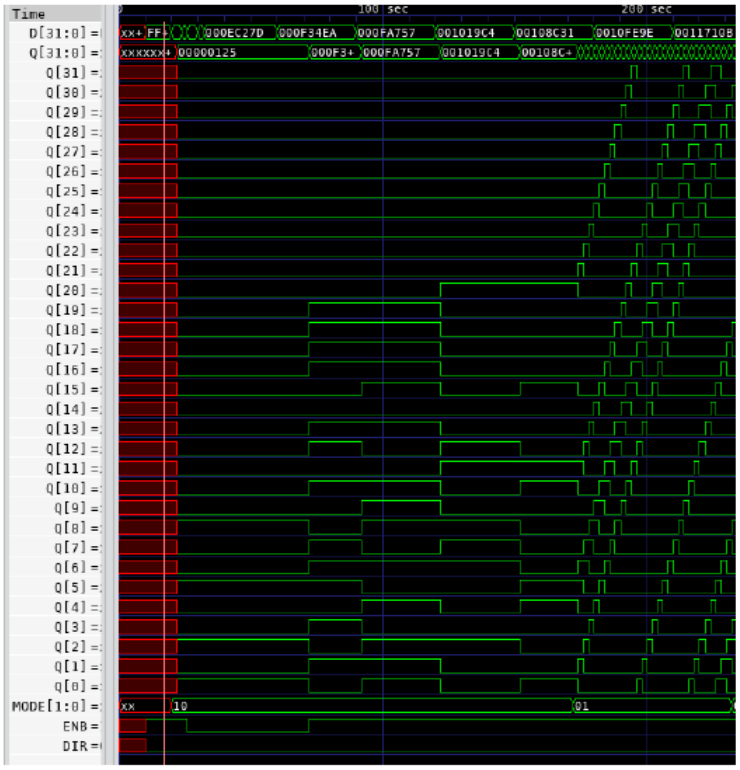
3. Utilización de la simulación:

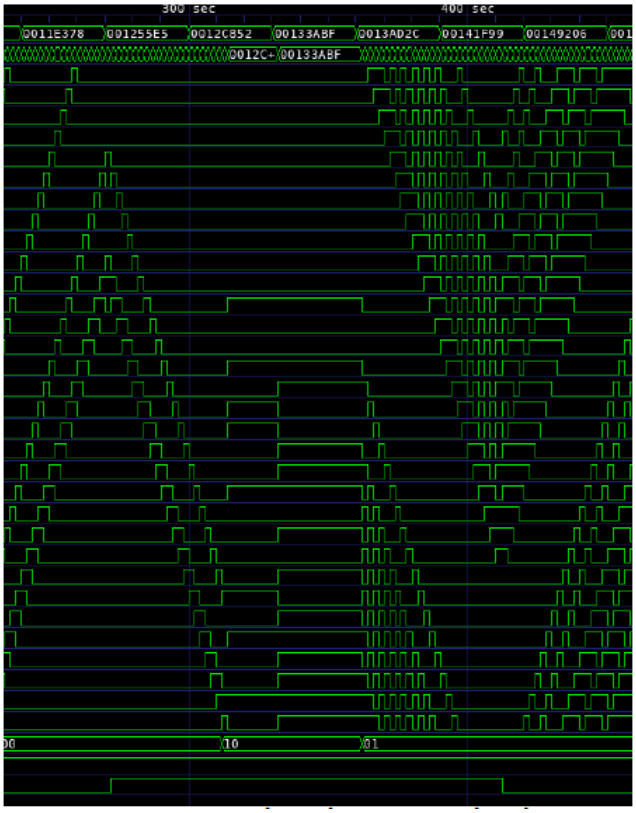
En el directorio raíz del proyecto se encuentran los ficheros fuente:
banco_de_pruebas_16b.v
banco_de_pruebas_32b.v
banco_de_pruebas_8b.v
banco_de_pruebas.v
probador_16b.v
probador_32b.v
probador_8b.v
probador.v
registro_16b.v

registro_32b.v
registro_8b.v
registro.v
Para cada respectivo caso el procedimiento es el siguente:
$ iverilog -o Salida_X banco_de_pruebasX.v
$ vvp Salida_X
$gtkwave ./senales.vcd
Siendo 'X' el número de registro a simular.

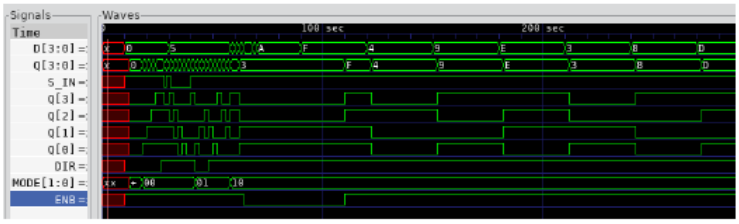
También se cuenta con un script Makefile con lo cual se puede observar la simulación en GTKwave con:
Simulación registro de 4 bits:
$ make
Simulación registro de 8 bits:
$ make 8bits
Simulación registro de 16 bits:
$ make 16bits
Simulación registro de 32 bits:
$ make 32bits

4. Ejemplo de los resultados:
Ejemplos de resultados obtenidos de la simulación del registro de 32 bits que engloba todas las demás verificaciones:





Es posible observar de las imagenes el funcionamiento de ENB, inicialmente se setea el registro con las entradas en paralelo y posteriormente se cambia el modo a rotación circular en el sentido izquierdo, en la cuarta imagen se observa que al colocar el modo en rotación lineal, la salida S_OUT ya no se toma en cuenta y se sustituyen los valores por ceros provenientes de S_IN, posteriormente se cargan valores en paralelo nuevamente y por último se hace rotar el registro nuevamente en los dos sentidos al cambiar el modo.



En la figura 5 se observa primero como se setea en cero el arreglo del registro de 4 bits, posteriormente se hace rotar el mismo hacia izquierda y luego hacia derecha, se introduce un uno por S_IN y luego varios ceros, se cambia el modo a rotación circular, luego se regresa al modo set en paralelo y se deshabilita el registro, para ser habilitado más tarde y setear el mismo entonces con el valor actual de D.

6. Conclusiones y recomendaciones:

Se logró implementar los dos casos de registro, sin embargo en mi carencia de conocimientos sobre verilog creo que es posible el optimizar muchisimo más la construción de los mismos, la mayor limi-

tante fue quizás el saber poco de la sintaxis del lenguaje, hubieron varios errores debido a retrasos a la salida serial, sin embargo se solucionaron, en el caso de todos los registros, el problema consistia en que al usar dos registros en cascada, dado que ambos funcionan con el flanco ascendente es necesario esperar un ciclo del reloj para que el segundo registro se entere de lo que se encuentra a la salida del primero debido a la operación desplazamiento del primero, sin embargo esto se solucionó al mostrar la salida con los pulsos negativos y leerla con los positivos, de esta manera ambos registros leían con el flanco ascendente pero el que entregaba el dato lo aportaba durante el ciclo y no acabado este, de esta manera se acoplaban sin pérdida de tiempo.

Para el próximo proyecto planeo el utilizar las plantillas realizadas durante este primer proyecto para ahorrar de esta manera tiempo, e invertir mejor el tiempo anterior a la entrega.

7. Bitácora:

Tiempo total : 00:07:19 : Comienzo la tarea leyendo el enunciado de la misma
Tiempo total : 00:16:49 : Comienzo a implementar el registro de cuatro bits
Tiempo total : 00:08:16 : Investigo como hacer un IF en verilog
Tiempo total : 00:45:23 : Investigo sobre sintaxis de verilog
Tiempo total : 01:02:23 : El registro opera bien, lo siguiente es documentar las pruebas
Tiempo total : 00:57:36 : Crear un registro de 8 bits
Tiempo total : 00:05:44 : Crear un registro de 16 bits
Tiempo total : 00:00:24 : Crear un registro de 32 bits
Tiempo total : 00:00:35 : Demostrar funcionamiento del registro de 32 bits
Tiempo total : 01:23:41 : Realizar pruebas de correción en el registro de 8 bits
Tiempo total : 01:01:00 : Realizar el reporte escrito
Tiempo total : 00:02:37 : Realizar el fichero Makefile
Tiempo total : 00:19:13 : Realizar el plan de pruebas
Tiempo total : 00:19:49 : Experimentar con el registro de 8 Bits optimizar
Tiempo total : 00:08:41 : Aplicar las correciones hechas en el diseño de 8 bits al de 32 bits
Tiempo total : 02:05:52 : Aplicar las correciones hechas en el diseño de 8 bits al de 16 bits y probarlo
Tiempo total : 00:12:31 : Probar funcionamiento de la última implementación hecha en el registro de 32 bits
Tiempo total : 00:19:14 : Corregir la rotación circular
Tiempo total : 00:01:21 : Obtener evidencias del funcionamiento del registro de 32 bits
Tiempo total : 01:20:34 : Conclusión del informe escrito
Total absoluto: 10:39:02