Proyecto Final - FSM y máquina microprogramada

1st José Julián Camacho Hernández
Ingeniería en Computadores
2019201459
jjulian.341@gmail.com

2nd Leonardo Guillén Fernández
Ingeniería en Computadores
2019031688
leonardoguillen946@gmail.com

3rd Luis Diego Sandí Quesada
Ingeniería en Electronica
2017098994
luisdiegosandiquesada@gmail.com

***Resumen*—En este proyecto se desarrolla una unidad que permita el cálculo de multiplicaciones binarias con operandos de 8 bits, mediante dos unidades de control distintas y una ruta de datos. Para ello se seguió un proceso de diseño que incluye diagramas y la codificación de los bloques en el HDL de Verilog. Se presentarán los resultados de diferentes pruebas que verifican el funcionamiento del sistema, y se van a exponer las principales conclusiones respecto a la utilización de cada tipo de unidad de control.**

***Palabras clave*—Datapath, FSM, microprogramada, multiplicador, verilog**

I. Proceso de diseño

I-A. Comprensión y planteo del problema

El presente proyecto tiene como objetivo desarrollar una unidad que permita el cálculo de multiplicaciones binarias con operandos de 8 bits. Estas operaciones se realizan por medio de un algoritmo iterativo, que contiene un vector de resultado al que se le aplican sumas y corrimientos a la derecha.

El primer paso para el diseño de esta unidad fue la comprensión del problema. Para esto, se leyó en repetidas ocaiones la especificación de los requerimientos necesarios para el funcionamiento del sistema. Además, se analizó el algoritmo solicitado a detalle, con el fin de comprenderlo y lograr implementarlo de manera adecuada.

Adicionalmente, se analizaron los diagramas de bloques proporcionados con el fin de identificar las diferentes salidas y entradas de cada uno de ellos. Se observaron y se definieron tanto las señales exteriores de salida y entrada para la unidad vista como una caja negra, como de manera interna para cada uno de los bloques. Este análisis se detallará en la siguiente sección.

I-B. Ruta de datos

Posteriormente, se procedió a la implementación de la ruta de datos. Para esto, en primer lugar se realizó un diagrama de flujo, con el fin de observar y validar que los componentes por implementar son capaces de ser utilizados para llevar a cabo el algoritmo. Seguidamente, se procedió a la codificación de los mismos en Verilog.

Como su implementación es en forma estructural y jerárquica, el primer paso para esto fue codificar cada uno de los registros necesarios con las características que se detallan en la siguiente sección. De esta manera se realizaron dos módulos de registro, así como un módulo para el sumador y otro para el contador hacia abajo.

En segundo lugar, se implementó un módulo superior, en el cual se realizaban las instancias de todos los registros, una del contador y una del sumador. En el mismo archivo, se definieron cables para la conexión interna entre los módulos. Adicionalmente, se realizaron varios bloques always para modificar el comportamiento de los módulos según las señales de entrada proporcionadas por el controlador. Para concluir con el Datapath, se agregó otro always para el control de las salidas del bloque, como el producto resultado, y el bit Q[0].

Para verificar la funcionalidad del Datapath, se construyó un testbench en el cual se realiza una instancia del módulo superior y se varían las señales que van a llegar desde el controlador según el algoritmo. Se realizaron varias pruebas para diferentes números de multiplicando y multiplicador, y de esta manera, fue posible verificar que los resultados que se obtenían eran los esperados.

I-C. Unidad de control - Máquina de estados

Para el diseño de la Máquina de estados primero se identificaron las entradas y salidas, entradas como son el Reset, Zero y LSB y como salidas utilizamos un one hot para poder ingresar los bits necesarios a la ruta de datos, entonces dependiendo del comportamiento de las entradas se ira tomando el rumbo en el diagrama de estados y de esa manera se obtiene una salida de bits la cual controlará el curso de los datos controlados por la ruta de datos y dependiendo de estos estados la ruta tendrá diferentes instrucciones en cada ciclo de reloj. Se realiza un diagrama de flujo, el cual toma para sus decisiones los valores de entrada de la Máquina de estados y retorna la activación de variables ,las cuales se encargan de modificar los registros y de manejar el inicio y la finalización del algoritmo, de igual manera se creó un Diagrama de estados, con la misma base, tomando como salidas números de 5 bits, utilizamos la arquitectura de One Hot para poder controlar la ruta de datos enviando una señal de la instrucción a ejecutar, y el diagrama de estados toma sus decisiones de la misma forma que el diagrama de flujo con los valores de entrada. Para comprobar el funcionamiento de estos diagramas, se procede a probarlos en el código de verilog y comprobar que tanto sus entradas como salidas están correctas.

I-D. Unidad de control - Microprogramada

Para realizar el diseño de la unidad de control microprogramada se siguió el proceso que se detalla a continuación. El primer paso fue comprender el funcionamiento de la máquina

microprogramada controlada por la memoria ROM interna. Seguidamente, se definieron las entradas y salidas del bloque, estas se detallan en la explicación del diagrama de bloques de la unidad microprogramada en la siguiente sección.

Luego, se procedió a modificar el diagrama de flujo y el diagrama de estados según las nuevas entradas y salidas particulares del bloque. Esta modificación se explica en la sección III.

Seguidamente, como dentro de los insumos del proyecto se encontraba la máquina desarrollada en Verilog, solo se realizaron las instrucciones para colocar en la memoria, esto con el fin de codificar los saltos y salidas necesarias para ejecutar el algoritmo. Para obtener estas instrucciones, se utilizó una tabla en Excel donde se colocaba cada bit del ancho de la memoria ROM. Esos bits se convirtieron en hexadecimal para coincidir con la estructura propuesta de la memoria.

Dentro de las instrucciones desarrolladas se encuentran las siguientes. Las primeras direcciones se utilizan para esperar la señal de Start, si esta no se encuentra en alto, no se avanza en las instrucciones. Si da la señal de Start, se pasa a la dirección donde indica que se deben cargar los operandos.

Luego, se verifica si el bit Q[0] es uno. Si es cero, se salta directamente a la dirección donde la salida indica que se debe realizar un corrimiento a la derecha. Si es uno, se salta a la dirección que da la señal de realizar la suma y luego se salta incondicionalmente a la del corrimiento. Finalmente, se salta incondicionalmente a la dirección que indica un decremento en el contador del datapath.

En este proceso, se verifica constantemente si se da la indicación de reset del sistema. En ese caso, se regresa a esperar la señal de start. Además, al inicio de cada iteración se verifica que el datapath no haya terminado aún con la señal de zero. Si es así, se salta a la dirección que indica que el sistema finalió el algoritmo con la señal de Ready.

Una vez se programaron estas instrucciones, se probaron en LogiSim en un circuito que simulaba la máquina microprogramada. Como se obtuvo el resultado esperado, se colocaron en la memoria ROM del modelo en Verilog. A continuación, se implementó un módulo TOP para unir la ruta de datos con esta unidad de control y se instanció el mismo en un testbench donde se verificó su funcionalidad. Para este caso fue posible utilizar el mismo testbench que se usó para la unidad de control con máquina de estados (FSM), ya que las entradas al sistema completo deben ser las mismas sin importar el tipo de controlador que se utilice. Es decir, para el usuario la implementación interna debe ser transparente.

II. Diagramas de bloques

A continuación se explicarán los dos diagramas que se presentan a continuación, primero observamos en la figura 1 un diagrama de bloques el cual consta de dos partes, un controlador y una ruta de datos, el controlador se encarga de enviar bits de activación de las variables que controlan la ruta de datos y a su vez reciben dos datos , un contador el cual indicara cuando llegue a cero y un Q[0] que es el último valor del multiplicando y como estados principales tiene un start y un ready que indica el inicio y fin el circuito. Por otro lado, está la ruta de datos, la cual recibe los números a multiplicar y 4 estados que le indican que ejecutar y como salida un estado del contador para saber cuándo llega a cero y también Q [0] que es el LSB de la operación.

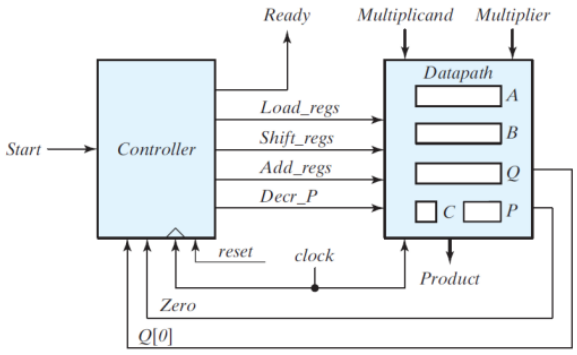

Figura 1. Diagrama de bloques de la estructura general de la unidad.

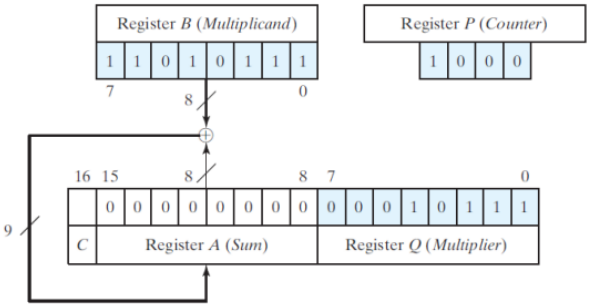

Figura 2. Diagrama de bloques de la ruta de datos de la unidad.

Los registros del Datapath deben tener las características que se listan a continuación. En primer lugar, respecto al ancho de cada bloque, los registros A, B y Q deben ser de 8 bits, mientras que C debe ser de 1 bit. El registro P debe ser de mínimo 3 bits para que sea posible realizar una cuenta hacia atrás desde siete hasta cero, en binario. El sumador debe ser de números de 8 bits y su salida debe ser de 9 bits debido a un posible acarreo de salida.

El registro B es el encargado de almacenar el multiplicando, debido a ello, debe tener una entrada paralela con el fin de poder cargar y almacenar el dato directamente. Como el multiplicando debe sumarse con la parte alta del vector de resultado, este registro tiene una salida paralela que hace posible obtener todo el número completo para ser operado. Este registro no necesita un corrimiento, por lo que cuenta con esta característica.

Los registros del vector de resultado C, A y Q son registros distintos al B. Estos cuentan con una entrada paralela, ya que en Q se necesita cargar el multiplicador directamente, y en A se requiere colocar el resultado de la suma de la misma manera. Además, tienen dos salidas: una serie y una paralela. La salida serie se utiliza para realizar los corrimientos hacia la

derecha, ya que de esta manera, es posible pasar el último bit de cada registro al siguiente, y así conectarlos entre sí. Esta salida también permite obtener el bit Q[0] que es fundamental para la toma de decisiones del algoritmo. La salida paralela se utiliza para obtener los datos en A para ser operados y para sacar todos los datos de los registros para obtener el resultado de salida.

Finalmente, el registro P es un contador hacia abajo. Su salida es paralela y corresponde a la cuenta que tiene en ese momento. Además, posee una señal de control habilitadora que indica si se detiene o continúa contando hacia abajo.

Todos los registros anteriores cuentan con una señal común de reset en bajo, que retorna los bloques a un estado inicial. Además, cuentan un la misma señal de reloj, que sincroniza el sistema y hace que los cambios se realicen únicamente en los flancos positivos del mismo. Al igual que P, los registros A,B,C y Q poseen un enable para controlar que funcionen o dejen de hacerlo. Todas estas características que se detallaron de los elementos se manejan por medio de las señales que proporciona el bloque controlador.

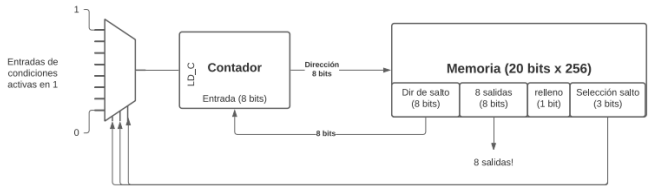

Figura 3. Diagrama de la unidad de control microprogramada.

III. Diagramas de flujo y estados

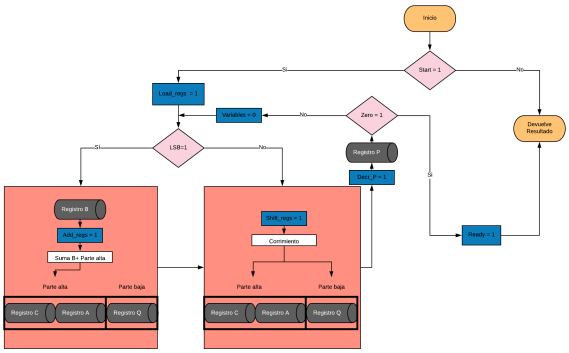

Figura 4. Diagrama de Flujo de la unidad de control.

Como se observa en el diagrama de flujo de la máquina de estados, tiene 3 valores los cuales son importantes para la dirección de las instrucciones, luego de que el start este activo pone el valor Load regs en 1 y tomando el último valor del multiplicando como LSB el cual si es 1 se sumara a la parte alta del resultado pasando luego a la parte del corrimiento, si LSB está en 0, se pasa directamente al corrimiento, luego se pone la variable Drec P que hace decrecer a zero y se valora zero, si este es 0 se continua el ciclo con las variables en cero, pero si este está en 1 significa que el contador termino, pasando al estado ready y activándolo para dar fin al algoritmo.

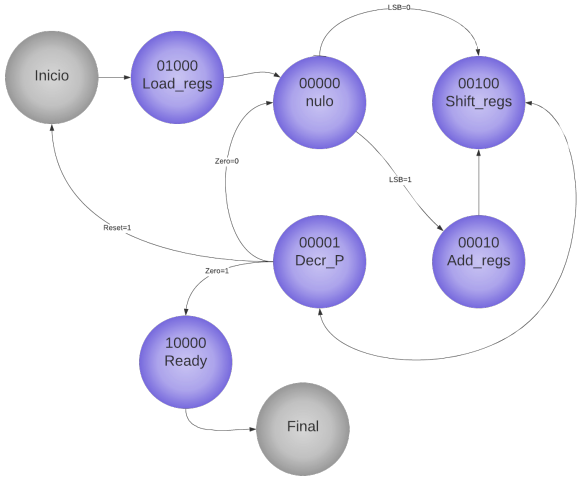

Figura 5. Diagrama de estados de la Máquina de estados con arquitecttura Moore.

Con el diagrama de estados, como se menciona anteriormente se realizó un One Hot, y cada salida lleva un bit activo el cual dará sentido a las instrucciones de la ruta de datos, se tienen 3 bits para controlar el flujo, primero un LSB que decide si se suma o si se hace un corrimiento , luego si se suma directamente pasa a hacer un corrimiento luego pasa directamente al estado de decrecer y este valora Zero y reset, si zero esta inactivo , repite el ciclo otra vez , si está activo, termina el ciclo con un resultado pero si reset está activo, vuelve al inicio de los estados.

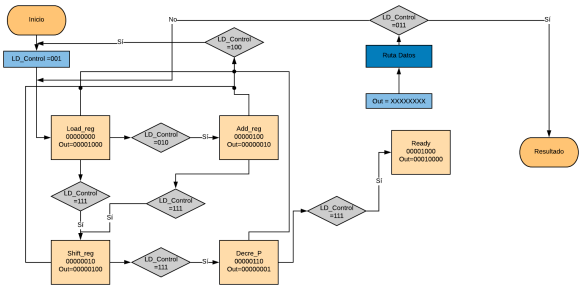

Figura 6. Diagrama de Flujo de la arquitectura Microprogramada.

El diagrama de flujo se la Máquina Micro programada, toma sus decisiones basadas en la señal de tres bits llamada LD Control, cada selección se hace en cada ciclo, otorgando un Out para cada estado y esa salida, es la entrada de la ruta de datos y esta garantiza si debe seguir en el ciclo o si debe terminarlo con el LD Control 011, el cual significa que debe analizar el valor de zero iniciando con el valor de start y finalizando con el valor de ready.

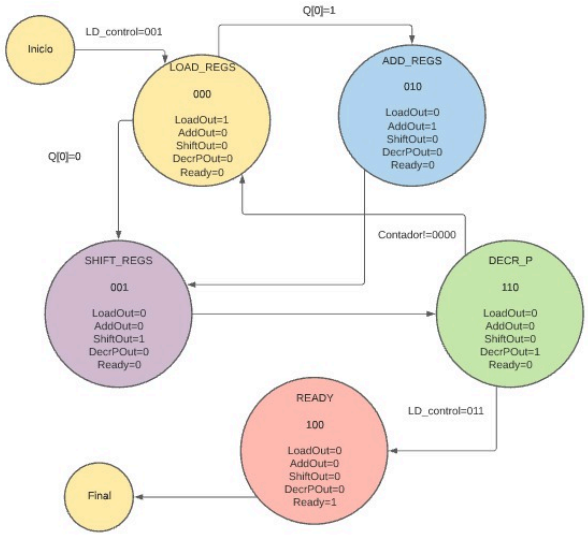

Figura 7. Diagrama de estado de la unidad microprogramada.

En el diagrama de estados de la maquina microprogramada, se realizó una codificacion binaria, y cada salida lleva un bit activo el cual dará sentido a las instrucciones de la ruta de datos, se tienen 3 bits para controlar el flujo, el LDControl es una señal de control de 3 Bits que indica cuando cambiar de estado basandose en Q[0], Zero, el estado del contador, entre otras señales. El ciclo de suma y shift o solo shift se repite hasta que el contador del registro P llega a cero, una vez ahi llega al estado Ready mediante la misma maquina microprogramada.

IV. Resultados

Se diseño la ruta de datos de forma satisfactoria, se logro replicar el algoritmo mencionado en instructivo del proyecto a partir de cinco registros, A, B, C, P y Q. El sumador utiliza los estados de los registros A y B para luego guardarlos en el registro B, tambien guarda el carry out en el registro C. Los registros C, B y Q estan conectados de modo que se puedan desplazar los datos hacia la derecha cuando la condicion lo indique.

Para la elaborar la maquina de estados se baso en una maquina de Moore, con 7 estados en donde se define que la maquina debe comenzar en el estado inicio de ahi debe ir al estado Load regs que carga el estado en el load del contador,apartir de esto se programo para que el bit menos significativo del registro Q se desempenara como una condicion para el siguiente estado, si el bit menos significativo es uno el siguiente estado es Add regs si el bit menos significativo es cero, entonces pasa al estado Shift regs que es donde se desplazan los bits hacia el bit menos significativo. El siguiente estado luego de el estado add regs es el estado Shift regs despues de esto se encunetra el estado Decr P que indica al contador que debe decrecer en uno. En el caso de que el contador ya llego a cero entonces el siguiente estado es Ready y luego final, pero si el contador aun no llego a cero, entonce el programa de verilog hace que se vuelva al estado load regs y asi generando un sistema iterativo.

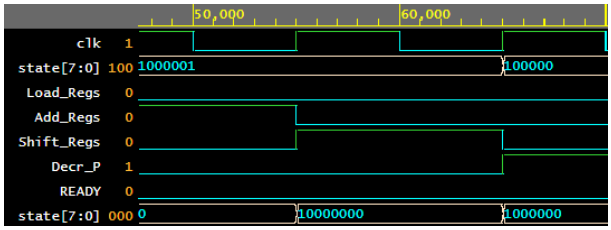

Figura 8. Diagrama de Tiempo de la maquina de estados.

En la figura anterior se evidencia una suma y un corriemiento hacia la derecha, donde el primer dato con 8 bits es el registo Q y el segundo dato con 8 bits es el registo A.

La unidad de control microprogramada se programo para replicar el comportamiento de la maquina de estados anteriormente diseñada pero mediante la memoria y combinaciones de bits especificos para lograr el funcionamiento deseado. Tanto para la maquina de estados como para la maquina microprogramada se utilizo el mismo testbench.

V. Conclusiones

Las maquinas de estado difieren de las maquinas micro programadas en que las maquinas de estado una vez impresas no se pueden reconfigurar para otro sistema, en cambio la maquina microprogramada puede reutilizarse si el sistema cambiara de alguna manera o si se desea utilizar la unidad de control para algun sistema diferente. Las maquinas microprogramadas suelen ser mas lentas que las maquinas de estado debido a que las maquinas de estado estan hechas para una tarea muy especifica y las microprogramadas no. Es importante tener en cuenta la cantidad de microinstrucciones y el largo de las mismas ya que esto podria afectar el rendimiento del sistema.

VI. Código fuente y recursos

Repositorio en GitHub: https://github.com/JulianCamacho/Proyecto_Dise-o_Logico