Interpretación de la Unidad de Coma flotante

Curso: Arquitectura de Computadores I
Profesor: M.Sc. Luis León
Problema: Implementación de la Unidad de Coma Flotante de 16 bits de dos líneas
Entregable: Documento de ejecución y guía de módulos
Simulación: EDA Playground (SystemVerilog/Verilog)

Este documento describe la estructura de la unidad especial de coma flotante de 16 bits (dos lanes) y las instrucciones para su ejecución en EDA Playground.

Resumen

En el ejercicio del segundo problema descrito para el primer proyecto del curso de Arquitectura de Computadores I, se nos pide realizar una implementación de una extensión adicional que, para este grupo corresponde una unidad de coma flotante de 16 bits de dos líneas segun el formato IEEE-754. Como parte de los entregables de dicho trabajo, se nos pide un documento de texto con instrucciones para correr la simulación y scripts para construir y reproducir el proyecto; este documento representa dicho entregable. Hablaremos un poco sobre la idea de la unidad especial, los diferentes módulos que se realizaron para cada operación tanto de un lane como de doble lane, y como todos estos en conjunto logran calcular lo solicitado haciendo uso de un testbench familiar con el usuario para que estimule las entradas a modo de prueba y corroborar la funcionalidad de toda la unidad.

Introducción

Una unidad de punto flotante como la desarrollada en este proyecto, sirve a la necesidad de expresar números muy grandes o muy pequeños en un formato simple. Se parte desde la idea de expresar este tipo de números en notación científica, donde tenemos un número multiplicado por una base (10 para el caso decimal, 2 en binario), y esta última elevada a un cierto exponente; de esta forma, tenemos un amplio rango de posibilidades para escribir números grandes y pequeños y hasta con parte fraccionaria.

Las operaciones solicitadas para esta unidad son 3: suma, multiplicación y recíproco. La idea es lograr que la unidad especial reciba dos operandos de 32 bits; la parte alta de un operando se opera con la parte alta del otro, y lo mismo con las partes bajas. Teniendo un multiplexor al inicio que podamos manipular para seleccionar la operación deseada y que los módulos se encarguen de calcular el resultado de la operación usando los vectores de entrada. El caso del recíproco solo utiliza uno de los operandos, o sea que solo acepta una entrada de 32 bits a la cual se le calcula el recíproco a su parte baja y su parte alta para luego regresar los resultados en el mismo empaquetado.

Módulos

Para cada operación se desarrollaron distintos módulos que sirven a la causa, la implementación de la suma parte desde la construcción de multiplexores, sumadores, shift registers, etc. En cambio la implementación de la multiplicación y el recíproco parten desde la teoría, con módulos más complejos que se encargan de las diferentes partes en la construcción de los resultados.

Suma

- diferencial5bits.v
- muxExpM.v
- mux_2x10bits.v
- shift_regis_right.v
- sumador_Alubig.v
- shift_left_right.v
- sumador_Alusmall.v
- special_cases.v
- control.v
- fp16_add_lane.v
- fp16_add2lanes.v

Para esta operación se cuenta con un restador, multiplexores, sumadores, shift registers, un módulo que identifica y manipula casos especiales de la operación y una unidad de control. En el módulo **fp16_add_lane.v** se unen todos estos componentes para realizar la suma de un lane, y con esto ya podemos replicar la idea para generar los dos lanes requeridos lo cual se hace en **fp16_add2lanes.v**.

Multiplicación

- fp16_unpack.v
- fp16_classify.v
- fp16_special_mul.v
- fp16_expacc.v
- fp16_mantmul.v
- fp16_normround.v
- fp16_mul_lane.v
- fp16_mul2lanes.v

Por medio de **fp16_unpack.v** se separa el vector en signo, exponente y fracción. Con **fp16_classify.v** asignamos el tipo de número que entra, generando valores por defecto para casos especiales o prohibidos en **fp16_special_mul.v**. Con **fp16_expacc.v**, **fp16_mantmul.v** y **fp16_normround.v** realizamos la suma de exponentes, multiplicación de mantisas y normalización más redondeo del producto final. Los últimos dos módulos descritos se encargan de la multiplicación en un solo lane y para dos lanes.

Recíproco

- fp16_unpack.v
- fp16_classify.v
- fp16_reciprocal.v
- fp16_reciprocal_dual.v

Los módulos de **fp16_unpack.v** y **fp16_classify.v** se usan igual que en el cálculo de la multiplicación. En **fp16_reciprocal.v** se llama a estos módulos anteriores, se consideran un par de casos especiales y luego se realiza el cálculo correspondiente para casos normales. El signo de entrada se mantiene en el resultado, y a través de una división de alta precisión más pasos de normalización y redondeo, otenemos un lane que duplicamos para generar el doble lane.

Manager

Con las operaciones de suma, multiplicación y recíproco funcionales, se procede a unirlas en una sola raíz que nos permita introducir los operandos de 32 bits y por medio del selector de un multiplexor, elegir la operación deseada. Este módulo conocido como el **Manager.v** une las operaciones de doble lane, utiliza banderas para detectar entradas prohibidas y se encarga de utilizar cada módulo anterior para la operación seleccionada. Básicamente es en este módulo donde converge todo lo descrito anteriormente; usando includes de algunos módulos en otros módulos clave como si se tratara de bibliotecas, logramos conectar todas las piezas para completar la unidad especial para este grupo. Con testbench's por separado para cada operación y uno general que prueba el **Manager.v** logramos corroborar la funcionalidad de cada módulo, y por ende, cada operación de forma paralela.

Ejecución

La realización de esta unidad especial fue gracias a la página EDA Playground, este es un IDE online para escribir y simular hardware, justo lo que ocupamos para ejecutar todos estos módulos. Copiando y pegando este link: `https://edaplayground.com/` en un navegador, el usuario deberá registrarse si no lo ha hecho; puede usar una cuenta de Google pero es recomendable que lo haga con una cuenta estudiantil o de empresa, esto para acceder a ventajas como guardar proyectos y la utilización de simuladores comerciales como **Cadence Xcelium 23.09**, aunque también se puede usar simuladores libres como **Icarus Verilog 12.0**.

Con la página de EDA abierta, se pueden ver tres paneles principales ubicados sobre el centro de la pantalla, así como una serie de pestañas y herramientas para comenzar a desarrollar proyectos.

- **Panel izquierdo(Testbench)**: Acá tenemos el primer archivo con extensión .sv el cual sirve para escribir los testbenches que prueban los módulos creados en el otro panel.
- **Panel derecho(Design)**: Al otro lado tenemos la sección de diseño, donde el archivo por defecto se llama design.sv, que para este proyecto solo se usa para llamar a la raíz principal de toda la implementación.
- **Panel inferior**: En este panel se puede dar un nombre como título de proyecto, así como una descripción del mismo. Esto sirve para etiquetar el trabajo y que no se pierda entre los demás cuando se acceda de nuevo a él mediante la pestaña de "Your Playgrounds".
- **Sección izquierda de la pantalla**: A la izquierda de todo, se pueden ver varias pestañas como la de "Languages & Libraries"donde puedes seleccionar el sintetizador y lenguaje de preferencia, en este caso usamos **SystemVerilog/Verilog**. En la pestaña de "Tools & Simulators"tenemos varios tipos de simuladores como los ya mencionados, opciones de compilación, etc.
- **Pestañas superiores**: Arriba de los paneles tenemos opciones para crear nuevos proyectos, ejecutar, guardar y copiar; opciones normales donde el nombre indica todo lo que hay que saber de ellas. En la pestaña de "Playgrounds"se puede acceder a los proyectos guardados y en "Profile"se encuentran los datos del perfil del usuario.

En https://edaplayground.com/x/j4Zg se encuentra la implementación de toda la unidad lista para solo darle al botón de Run. De forma empírica sabemos que pueden haber problemas para ejecutar la unidad siguiendo el enlace, por eso el usurio debe comprobar que está logueado en la página y que ha seleccionado un simulador acorde a su status, la forma segura es elegir **Cadence Xcelium 23.09** en la pestaña ya que es un simulador libre.

Otra forma para lograr la ejecución de la unidad (camino largo) es seguir el link hacia el repositorio del proyecto, dicho link recide en el documento principal del proyecto en la sección 'Repositorio'. En el branch **Unidad-Punto-Flotante-16-bits** se puede ver el archivo titulado 'ComaFlotante.zip' el cual se deberá descargar y extraer sus archivos para copiarlos y pegarlos en el panel de Diseño de EDA Playground.

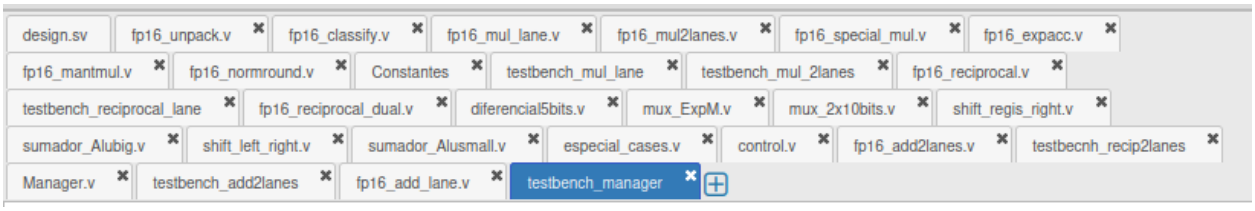

Figura 1: Pestaña de diseño en EDA

En la figura 1 se puede ver el acomodo de todos los módulos en el panel Design. Con el botón de '+' se crean nuevos archivos (módulos), que luego de nombrarlos con alguna extensión como '.v' o '.sv' son archivos ejecutables para el sintetizador (Si no escribes la extensión el archivo creado solo es de texto como el caso de los archivos nombrados 'testbench_...'). Al extraer los archivos de la carpeta 'ComaFlotante.zip' solo falta crear todos los módulos con el mismo nombre con el que aparecen para proceder a copiar y pegar los contenidos de la carpeta en estos nuevos módulos. El archivo design.sv no hace falta copiarlo tal cual, ya que este es un archivo creado por defecto en la página, por lo cual solo se necesita un 'include' del Manager.v y listo. Los archivos solo de texto con el nombre testbench guardan en su interior lo que ya se puede suponer, testbenches para probar los módulos:

- **testbench_add2lanes:** Suma en dos lanes.
- **testbench_mul_lane:** Multiplicación de un lane.
- **testbench_mul2lanes:** Multiplicación de dos lanes.
- **testbench_reciprocal_lane:** Recíproco de un lane.
- **testbench_recip2lanes:** Recíproco de dos lanes.
- **testbench_manager:** Implementación completa en dos lanes para las 3 operaciones.

Por último, el archivo **Constantes** contiene algunos valores quemados que sirven para que el usuario los pruebe en los testbenches, estos valores ya vienen con el formato fp16 por supuesto. Aun así, una mejor comprobación de valores se puede obtener mediante el link https://numeral-systems.com/ieee-754-converter/ donde tenemos la opción de pasar números de decimal a IEEE-754 y viceversa.