Trabajo Práctico Nº1
Aritmética de punto flotante
66.17 Sistemas Digitales - FIUBA
2do Cuatrimestre - 2018

José F. González - 100063 - <jfgonzalez@fi.uba.ar>

1. Objetivos

El objetivo de este trabajo práctico consiste en implementar la operación de multiplicación de una unidad de punto flotante en lenguaje descriptor de hardware VHDL.

2. Diseño de Unidad de Multiplicación

Figura 1: Representación de simple precisión en punto flotante

Un número puede ser representado en formato de punto flotante mediante un signo, una precisión p, una base b y un exponente e. De forma tal que la mutiplicación de dos numéros se reduce a:

$$(\pm a.aaa\cdots a \times b^{e_a}) \cdot (\pm b.bbb\cdots n \times b^{e_b}) = \pm(a.aaa\cdots a \times b.bbb\cdots b) \times b^{e_a+e_b} \tag{2.1}$$

El estándar IEEE 754 nos da un formato de representación de números reales en punto flotante de base dos, en el cuál se usa un bit de signo, seguido de ϵ bits del exponente y una mantissa de m bits de precisión, tal cómo se indica en la figura 1. Luego podemos implementar la multiplicación en VHDL tomando dos números de $e + m + 1$ bits y recreando la ecuación (2.1).

2.1. Exponente

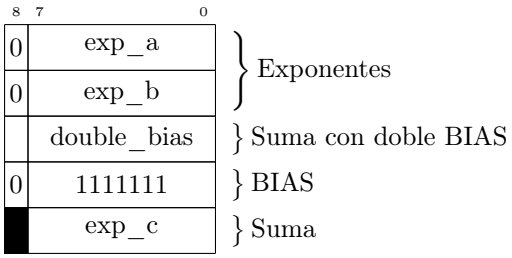

Figura 2: Manejo del exponente

El exponente se almacena como un valor sin signo con un *BIAS* de valor $2^{n-1} - 1$ para facilitar la comparación de exponentes. Luego según (2.1) el exponente será la suma de los exponentes, operación que implementamos en binario con un sumador de $\epsilon + 1$ bits. El bit adicional tiene la función de guardar un valor de exponente que incluye un *BIAS* doble, que restamos utilizando un restador de $\epsilon + 1$ bits como se muestra en la figura 2 para el caso de un exponente de 8bits. Este resultado se compara con los límites de *under/overflow* que serán fuera del rango $[1, 2^e - 1]$, si el exponente es menor a 1 el resultado se aproxima a cero y si es mayor al límite máximo se lo guarda como $2^e - 1$ reservado para mayor número normalizado. Si está dentro del rango de representación se guarda los primeros ϵ *bits*.

2.2. Mantissa

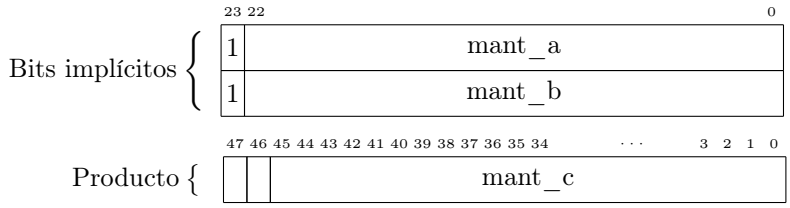

Figura 3: Manejo de la mantissa

El producto de mantissas se implementa en VHDL como un producto de enteros sin signo, siendo el resultado almacenado en $2 \cdot (m+1)$ bits como se indica en la figura 3 para $m = 23$. Donde se debe tener en cuenta el *bit oculto* dado por la normalización de la mantissa del estándar IEEE 754 donde el bit más significativo será siempre un 1. Finalmente el resultado se debe truncar y normalizar. Se implementa una descripción funcional donde dependiendo de los dos primeros dígitos se realiza un corrimiento de bits o un corrimiento de coma con un ajuste del exponente.

2.3. Signo

El signo del producto será computado simplemente por una descripción funcional de una compuerta EX-OR para los casos generales, llevando execpciones a 0 o 1 según corresponda.

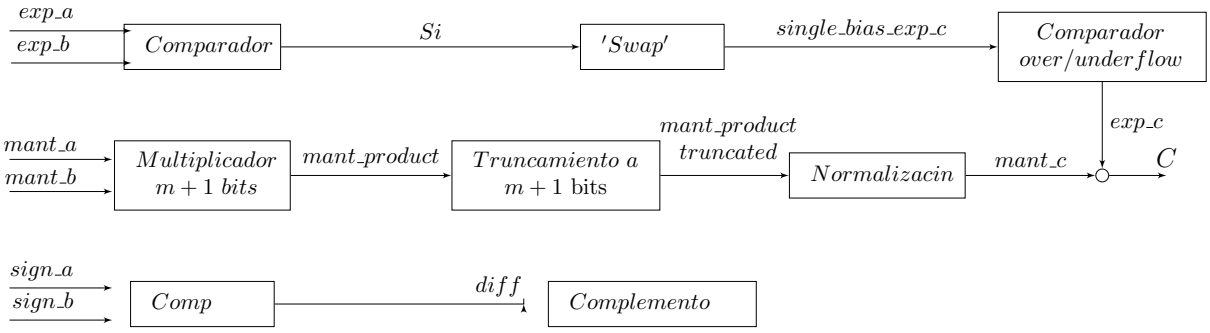

Figura 4: Diagrama de bloques unidad de multiplicación

3. Simulación

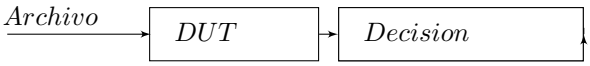

Figura 5: Diagrama de la simulación

Se implementó un banco de pruebas alimentado por archivos con vectores que contienen valores de prueba con sus respectivos resultado para comparación, si la comparación presenta un error se aumenta un contador de errores. En la figura 6 se muestra un ejemplo del resultado de ejecución de la simulación para el archivo **"test_mul_float_26_7.txt"**, donde se aprecia la ejecución sin errores de la FPU. El proyecto por defecto tiene cargado el archivo **test_mul_float_24_6.txt** para simular, para ejecutar simulaciones de otros archivos se debe modificar, en el archivo **testbench.vhd**, el nombre del archivo que se desea y cambiar las constantes **WORD_SIZE_T**, **MANTISSA_SIZE_T**, **EXP_SIZE_T** por los correspondientes al nuevo archivo. Por ejemplo,el archivo 26_7 lleva 26, 18 y 7 respectivamente.

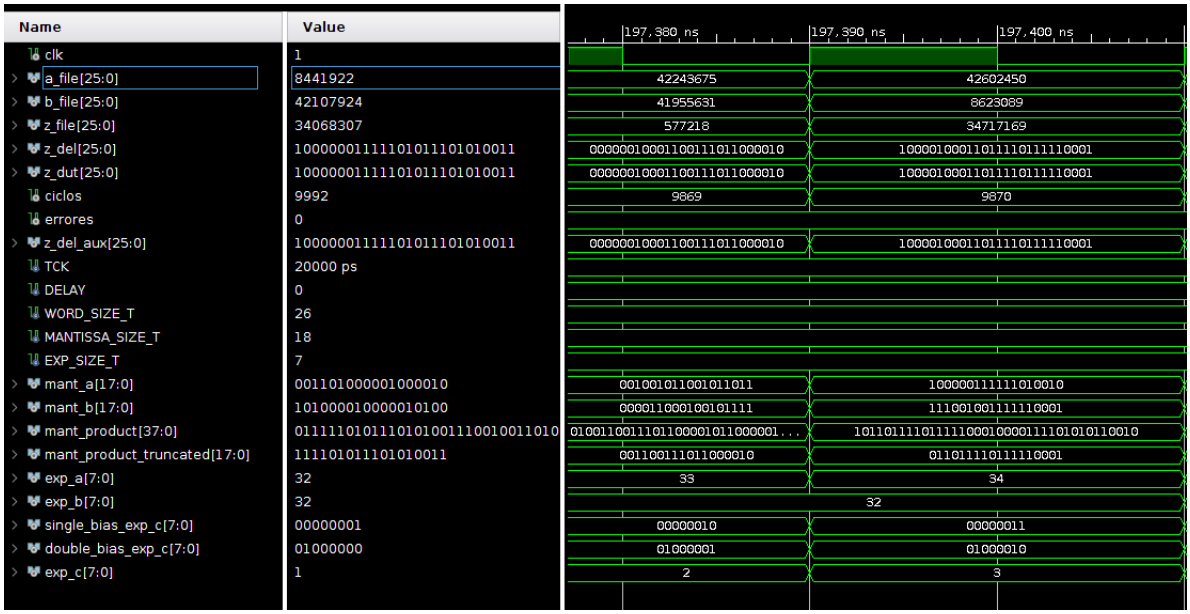

Figura 6: Ejemplo de la simulación para el archivo **26_7**

4. Diseño de Unidad de Suma

Se implementa el algoritmo representado por el diagrama de bloques de la figura 1. Las consideraciones de simulación son las mismas que en el caso de la multiplicación.

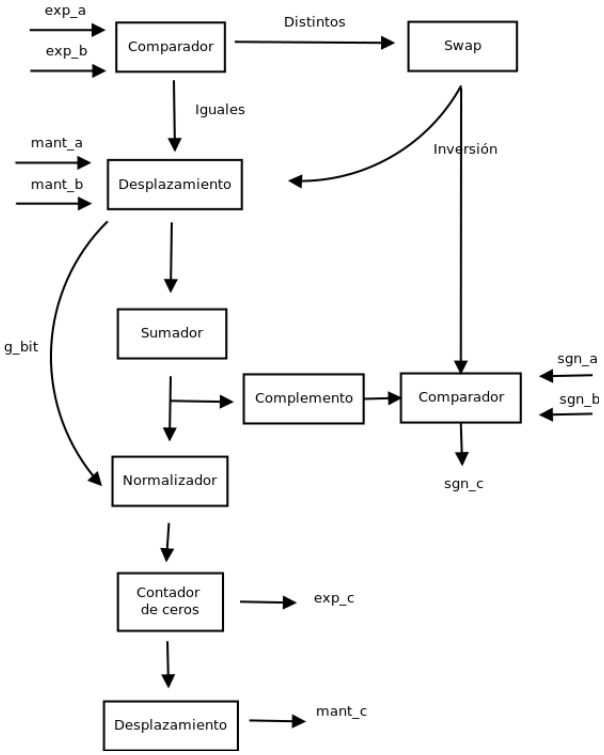

Figura 7: Diagrama de bloques unidad de suma

5. Síntesis

En los archivos **fp_mult_utilization_synth.rpt** se adjuntan los resultados de la síntesis en Vivado (para cada formato el circuito difiere) para una placa Arty Z7. En la figura 7 se detallan la cantidad de Slices, Flip-Flops y LUT's utilizados para el formato 26_7. Notar que no se usan registros como Flip-Flops pues el circuito es puramente combinacional. Para correr la síntesis en Vivado del circuito sin que tenga en cuenta la etapa de simulación se debe colocar al archivo **fp_mult.vhd** con maxíma jerarquía: **set as top**. Los valores por defecto de síntesis e implementación son del formato 26_7.

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	114	0	41000	0.28
LUT as Logic	114	0	41000	0.28
LUT as Memory	0	0	13400	0.00
Slice Registers	18	0	82000	0.02
Register as Flip Flop	0	0	82000	0.00
Register as Latch	18	0	82000	0.02
F7 Muxes	0	0	20500	0.00
F8 Muxes	0	0	10250	0.00

Figura 8: Slice Logic de multiplicación del formato **26_7** para una placa Arty Z7.

6. Implementaciónes

Se implementa el sistema sintetizado para una placa Arty Z7 con las opciones por defecto de Vivado. En el archivo **impl_1_place_report_utilization_0** se adjunta el reporte de utilización de recursos.

Site Type	Used	Fixed	Available	Util%
Slice LUTs	113	0	41000	0.28
LUT as Logic	113	0	41000	0.28
LUT as Memory	0	0	13400	0.00
Slice Registers	18	0	82000	0.02
Register as Flip Flop	0	0	82000	0.00
Register as Latch	18	0	82000	0.02
F7 Muxes	0	0	20500	0.00
F8 Muxes	0	0	10250	0.00

Figura 9: Slice Logic de la implementación de multiplicación del formato **26_7** para una placa Arty Z7.

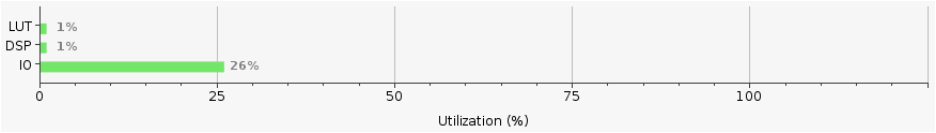

Figura 10: Uso de recursos para la multiplicación en placa Arty Z7.

7. Conclusión

Se logró implementar el sistema digital diseñado en VHDL comprobando su correcto funcionamiento mediante las herramientas de Vivado, logrando un primer acercamiento a los lenguajes descriptores de hardware y a su entorno de desarrollo en el contexto de la aritmética de punto flotante.

8. Referencias

[1] Computer Arithmetic, David Goldberg, 2003, Elsevier Science US.

[2] Material de Cátedra, Sistemas Digitales, FIUBA.