IV 8 Bits Multiplier

Contreras R Orlando[1,2*] and Garcia B Iker[1,2*†]

[3]Departamento de Sistemas Electrónicos, UAA, Av. Universidad 940, Aguascalientes, 20131, Aguascalientes, México.

*Corresponding author(s). E-mail(s): {al348390,al307630}@edu.uaa.mx;
†Ambos autores contribuyeron de forma equitativa a este trabajo.

Abstract

El objetivo de la siguiente práctica es diseñar un multiplicador mediante productos parciales para un producto de 8 bits por 8 bits, el presente documento busca optimizar en cuanto a descripción los bloques de descripción.

Keywords: VHDL, FPGA, Productos Parciales, Multiplicador

1 Introducción

El procesamiento eficiente de señales digitales y aplicaciones de alto rendimiento requieren operaciones aritméticas rápidas y eficientes. Entre ellas, la multiplicación es una de las más importantes, ya que es fundamental en algoritmos de procesamiento de imágenes, inteligencia artificial y sistemas embebidos.

En este contexto, el uso de circuitos diseñados en **FPGA** ofrece ventajas significativas en términos de velocidad, paralelismo y consumo energético en comparación con soluciones basadas en procesadores convencionales.

Este trabajo presenta el diseño e implementación de un **multiplicador de 8×8 bits** en FPGA.

2 Metodología

2.0.1 Productos Parciales

Para realizar el correcto diseño de un multiplicador, al igual que en la primer práctica, se debe analizar de una forma empírica. Podemos observar que en el producto realizado, se hace un desplazamiento a la izquierda cuando se multiplica por el segundo dígito (espacio en blanco) y después de realizar dicho producto, se hace la suma de los productos parciales, la misma analogía se aplicará para bits, generalizándolo de una mejor forma quedaría como se aprecia en la figura 2

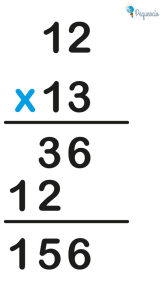

Fig. 1 Multiplicación de forma manual

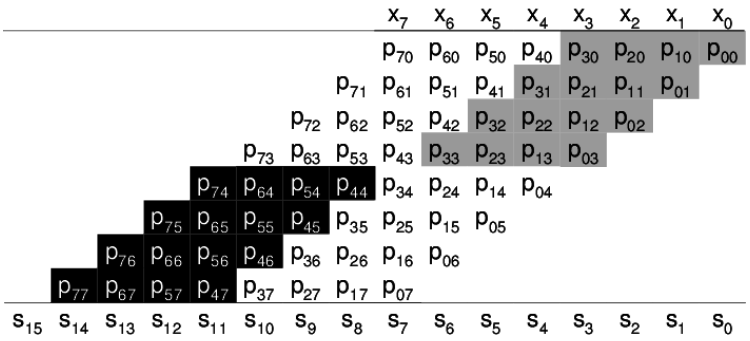

Fig. 2 Productos Parciales

2.1 Diseño de los bloques de 8 bits

Por cuestiones de optimización de código. Se tomó la decisión de implementar cada producto parcial en forma de vector, creando así un componente como se puede observar en la siguiente figura, esto genera el mismo circuito que si se hiciera bloque por bloque como se observa en la figura 3. Esto hace mas modular el multiplicador permitiendo que a la hora de expandir sea mas fácil, práctico y eficiente.

Fig. 3 Construcción bloque a bloque

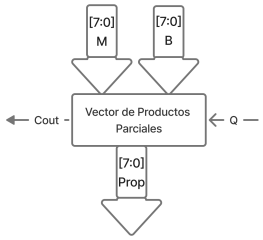

Fig. 4 Bloque de vector

3 Resultados

Se puede observar en la simulación como M multiplicado por Q en todos los casos dan el resultado correcto en P como lo es $2 \times 3 = 6$, $15 \times 15 = 225$, $2 \times 128 = 256$, $255 \times 255 = 65025$ (En este caso si da el numero bien pero como el numero es mas grande que el vector le falta 1 bit para expresar el 65025(1111111000000001) y pone el 32257 (111111000000001)) pero ese bit no se pierde, se expresa como el Cout.

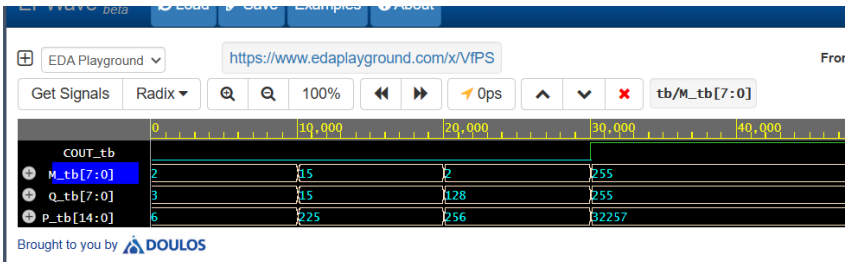

Fig. 5 Testbench realizado por EDA Playground

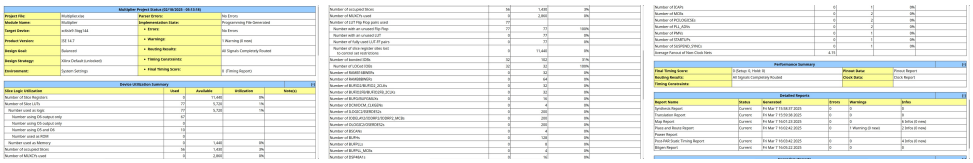

Fig. 6 Summary report generated by Xilinx

4 Conclusiones

Esta practica aumentó un poco la complejidad comparada a las pasadas y nos pareció un poco mas difícil a la hora de hacer el código, en esta ocasión se tuvo que implementar de otra forma de forma que quedara más óptimo en cuanto a código, lo que requirió que comprendiéramos enteramente el funcionamiento del multiplicador. El otro problema que nos encontramos y uno de nuestros puntos débiles fue el modificar la constraints file ya que en la documentación de la FPGA MIMAS vienen algunos errores con los pines y pues es hacer prueba y error hasta que funcione.