16 Bits Carry Look Ahead

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 optimizar de otra manera difrente al carry skip los tiempos en los sumadores, con el sumador de 16 bits no se logra percibir una diferencia, pero a grandes escalas los tiempos de retardo realmente importan.

Keywords: VHDL, Adder, bits, Tiempos, Sumador, FPGA, Look Ahead

1 Introducción

A la hora de diseñar un circuitos es importante considerar cuestiones como tiempo, área y energía. Mientras que el Look ahead requiere un hardware más complejo, es más rápido, también a considerar es que la energía incrementaría de acorde al tamaño

Característica	**CLA**	**CSA**
Velocidad	Muy rápida $O(\log n)$	Moderada
Propagación	Paralela	Por bloques
Complejidad	Alta	Menor
Uso	Sumadores pequeños/medianos	Sumadores grandes
Energía	Mayor	Menor

Table 1 Comparación entre sumadores CLA y CSA.

2 Metodología

2.0.1 Diseño de los bloques de 4 bits

A la hora de hacer el sumador de 16 bits con look ahead se tomo la decisión generar bloques de 4 bits para facilitar las operaciones y el tamaño del código.

Esto hace mas modular el sumador permitiendo que a la hora de expandir sea mas fácil, practico y eficiente.

Este tipo de metodologia tambien se hace para la facildad de comprension a la hora de generar y implementar los componentes.

2.1 Optimización de código

La ecuación se podría simplificar de la siguiente forma, esto queda de una forma más reducida en cuanto a código, como se puede apreciar en la (Figura).

Para comprobar que ambos códigos realizan lo mismo podemos observar el RTL, y como coincide podemos deducir que realiza el mismo funcionamiento.

```
carry(2) <=  G(1)  or
             (P(1) and G(0))  or
             (P(1) and P(0) and carry(0));


carry(3) <=  G(2)  or
             (P(2) and G(1)) or
             (P(2) and P(1) and G(0)) or
             (P(2) and P(1) and P(0)  and carry(0));

carry(4) <=  G(3) or
             (P(3) and G(2)) or
             (P(3) and P(2) and G(1)) or
             (P(3) and P(2) and P(1) and G(0)) or
             (P(3) and P(2) and P(1) and P(0)  and carry(0));

carry(1) <=  G(0)  or (P(0) and carry(0));
carry(2) <=  G(1)  or (P(1) and carry(1));
carry(3) <=  G(2)  or (P(2) and carry(2));
carry(4) <=  G(3)  or (P(3) and carry(3));
```

Fig. 1 Código escrito por dos formas diferentes

3 Resultados

En este caso, la salida termina siendo la misma, la diferencia radicaría en la optimización de tiempos.

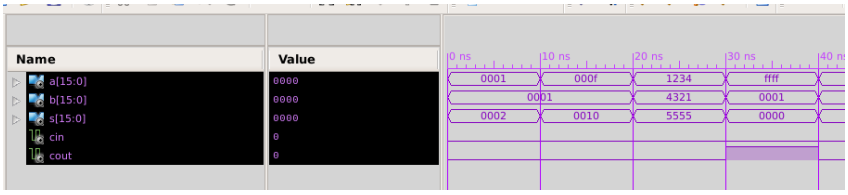

Fig. 2 Testbench realizado por EDA Playground

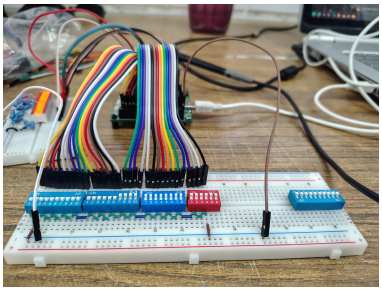

Fig. 3 Circuito Armado

Device Utilization Summary				
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	0	11,440	0%	
Number of Slice LUTs	16	5,720	1%	
Number used as logic	16	5,720	1%	
Number using O6 output only	8			
Number using O5 output only	0			
Number using O5 and O6	8			
Number used as ROM	0			
Number used as Memory	0	1,440	0%	
Number of occupied Slices	10	1,430	1%	
Number of MUXCYs used	0	2,860	0%	
Number of LUT Flip Flop pairs used	16			
Number with an unused Flip Flop	16	16	100%	
Number with an unused LUT	0	16	0%	
Number of fully used LUT-FF pairs	0	16	0%	
Number of slice register sites lost to control set restrictions	0	11,440	0%	
Number of bonded IOBs	49	102	48%	
Number of LOCed IOBs	49	49	100%	
Number of RAMB16BWERs	0	32	0%	
Number of RAMB8BWERs	0	64	0%	
Number of BUFIO2/BUFIO2_2CLKs	0	32	0%	
Number of BUFIO2FB/BUFIO2FB_2CLKs	0	32	0%	
Number of BUFG/BUFGMUXs	0	16	0%	
Number of DCM/DCM_CLKGENs	0	4	0%	
Number of ILOGIC2/ISERDES2s	0	200	0%	
Number of IODELAY2/IODRP2/IODRP2_MCBs	0	200	0%	
Number of OLOGIC2/OSERDES2s	0	200	0%	
Number of BSCANs	0	4	0%	
Number of BUFHs	0	128	0%	

Fig. 4 Summary report generated by Xilinx

4 Conclusiones

Nos pareció que esta práctica fue bastante sencilla, pues, aunque tuvo algunas modificaciones, en esencia era el mismo circuito anterior. Habría sido más interesante ver un incremento más notable en la complejidad.