Universidad Nacional de San Luis

Trabajo Práctico N° 2: Diseño de una ALU

Arquitectura de Computadoras

ALUMNO: PARISI PABLO

Desarrollo

El trabajo práctico cuenta de 5 ejercicios, los cuales deberán ser presentados con sus respectivos códigos de simulación. Además, se deberá demostrar su funcionamiento en clase.

Ejercicios

1) Describir en VHDL un full-adder (Sumador Completo). Escribir un testbench para simular el comportamiento del full adder.

2) Utilizando el full-adder del Ejercicio 1, describa un sumador con acarreo de carry (Ripple Carry Adder). Utilice generics para poder especificar el tamaño. Simule todos los valores posibles para 8 bits mediante un testbench.

3) En base al Ejercicio 2, diseñe un módulo que sume o reste números en complemento a dos, con tamaño definido por generics. Simule para todos los valores posibles para 8 bits.

4) Describir en VHDL tres módulos de hardware que realicen las siguientes operaciones: AND, OR, NOT bit a bit de dos vectores de tamaño genérico, de por lo menos 2 bits. Simule cada uno de los módulos para 10 valores seleccionados al azar.

5) Integre los módulos descriptos anteriormente para formar una ALU de tamaño genérico que posea las operaciones: SUMA, RESTA, AND, OR y NOT. Simule todos los valores posibles de las entradas en 7 bits, para las cinco operaciones.

6) Realice la síntesis de la ALU diseñada en el ejercicio anterior. En base a los resultados de síntesis, indique cuál es la máxima frecuencia de operación de la ALU. Elimine la operación de resta y verifique si se mejora la frecuencia máxima de operación. En caso de que se mejore, ¿se justifica la eliminación de esta operación para mejorar la velocidad de la ALU?

Respuestas:

Todos los ejercicios han sido realizados por medio del programa Quartus Primer 17.1.

Para la realización del ejercicio 6 fueron creados 3 códigos vhdl con sus respectivos test bench:

1) El primer código consta de la modificación de la ALU creada en el ejercicio 5 al sustraerle la operación de resta, dicho código posee el nombre de alu_mod y su test bench posee el nombre de alu_mod_tb.
2) El segundo código posee el nombre de alu_clock y su test bench es alu_clock_tb; al código alu_clock se le agregan flip-flop al código del ejercicio 5 para verificar la frecuencia máxima de operación.
3) El tercer código posee el nombre de alu_mod_clock y su test bench correspondiente es alu_mod_clock_tb; al código alu_mod_clock se le agregan flip-flop al código alu_mod (del ítem 1) para verificar la frecuencia máxima de operación.

Por medio de el ítem TimeQuest Timing Analysis en el ítem Task->Compilation en Quartus, se obtiene la frecuencia máxima de operación para la ALU generada con restador y sin restador; a partir de lo que se puede observar que:

- Para la ALU con restador incluido, se obtiene una frecuencia máxima de 295.86 MHz en el ítem Slow 1100mV OC Model. Esto se puede observar en la siguiente imagen:


Slow 1100mV 0C Model

	Fmax	Restricted Fmax	Clock Name	Note
1	295.86 MHz	295.86 MHz	clock	

Fig. 1 – Frecuencia máxima de ALU con restador.

- Para la ALU sin restador incluido, se obtiene una frecuencia máxima de 436.68 MHz en el ítem Slow 1100mV OC Model. Esto se puede observar en la siguiente imagen:


Slow 1100mV 0C Model

	Fmax	Restricted Fmax	Clock Name	Note
1	436.68 MHz	436.68 MHz	clock	

Fig. 2 – Frecuencia máxima de ALU sin restador.

Como se puede observar la frecuencia máxima de operación de la ALU sin restador es mayor que la frecuencia máxima de operación para la ALU con restador. Por lo que se concluye que se justifica la eliminación de la operación para mejorar los la velocidad de la ALU.