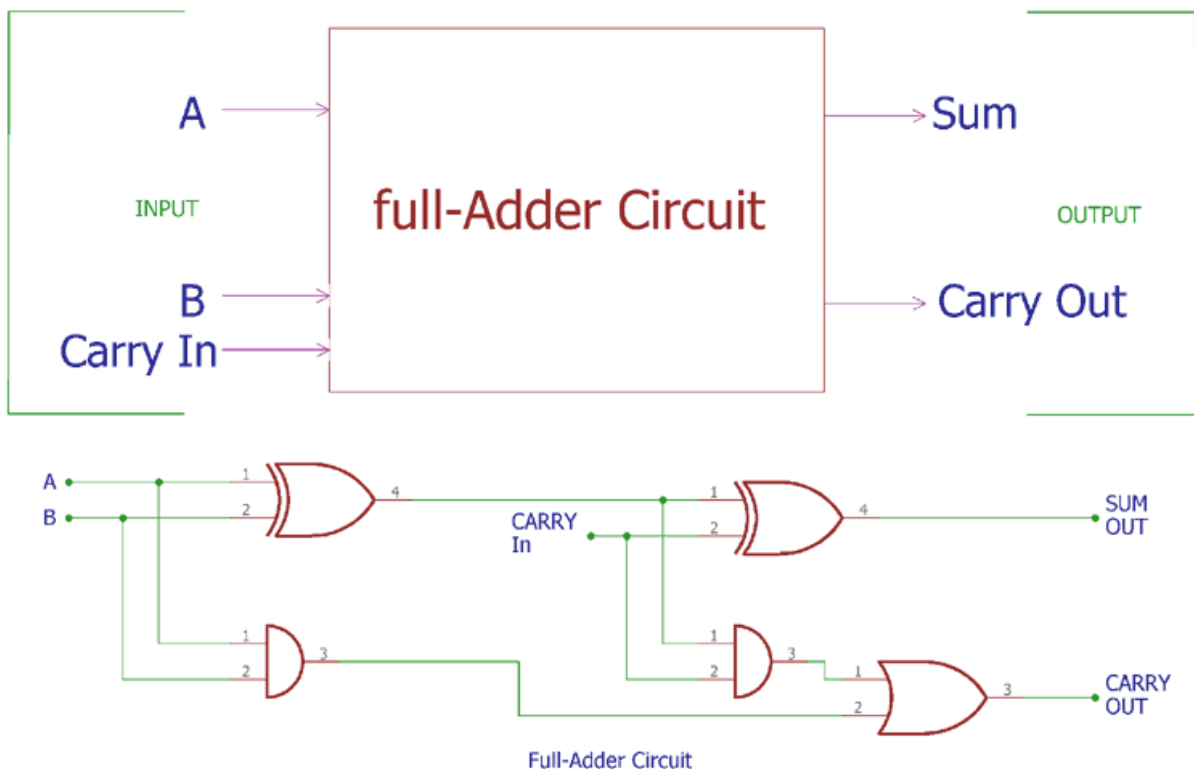


**Instituto Tecnológico y De Estudios
Superiores de Monterrey
Laboratorio - Arquitectura de Computadoras
TE-2031.1**

Lab 06 - Simulation using ModelSim



Entrega: 16 / 10 / 2022

Equipo 07:

Héctor Javier Pequeño Chairez A01246364

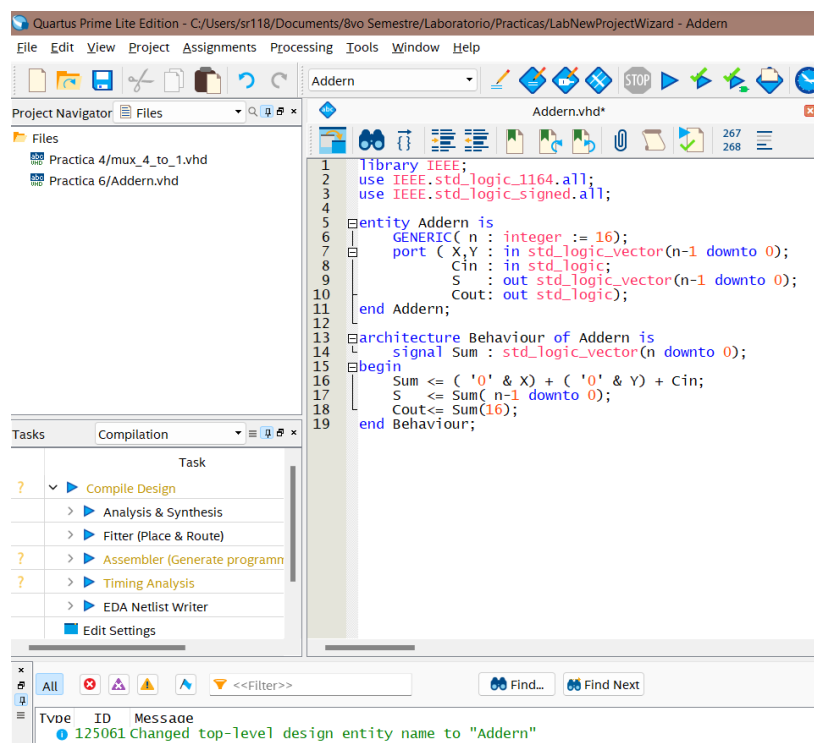
Gabriela Jazmín Álvarez Espinoza A00825719

Introducción

En la siguiente práctica, tenemos como objetivo el entender el funcionamiento de la aplicación ModelSim, esto con la finalidad de poder hacer un “Test” de nuestros hardware programado. Teniendo como introducción la configuración que debemos realizar dentro de la aplicación para poderla utilizar con nuestros archivos realizados en Quartus Prime.

Desarrollo

1. Creamos una nueva revisión con el nombre “Addern” según el proyecto que configuramos inicialmente con el nombre “switch_to_led”. Después de esto insertamos el código descrito en la práctica dentro de un archivo de vhd llamado “Addern.vhd”.



Código dentro del archivo Addern.vhd, asignado como “Top-level design”.

Funcionamiento del circuito

Iniciando con las librerías, tenemos una diferencia principal y es que ahora definimos una librería llamada “use IEEE.std_logic_signed.all;”, lo cual nos permitira trabajar con números con signos, que a diferencia de prácticas anteriores, no habíamos utilizado.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_signed.all;
```

Librerías en nuestro proyecto. (Se utilizó Visual Studio Code, para realizar la programación más rápidamente, pero la ejecución dentro de Quartus Prime).

Después utilizamos la *entity* Addern, en donde definimos como Generic la constante n, como mencionamos, Generic nos permite trabajar con datos que funcionan como constantes, por lo que en este caso tenemos la constante n con un valor de 16. Luego continuamos con la definición de puertos entrada y salida de nuestro hardware, en este caso tenemos:

Entrada	Tamaño (n-1) = 15	Salida	Tamaño (n-1) = 15
X	16 bits (n-1 downto 0)	S	16 bits (n-1 downto 0)
Y	16 bits (n-1 downto 0)	Cout	1 bit
Cin	1 bits	-	-

- **X, Y** : Representan los dos números de entrada de 16 bits, que serán utilizados para ser sumados.
- **Cin** : Nos permite introducir a la suma un bit de Carry que exista en la suma.
- **S** : Contiene la salida de la suma de los valores X y Y.
- **Cout** : En caso de existir un *carry* de salida, esta se encontrará en este puerto.

Cabe mencionar que existe una variable llamada “**Sum**” que nos permite realizar las sumas, esta variable no es un puerto de entrada ni de salida, solamente nos permite manejar los bits de la suma de X y Y con facilidad y asignarlo a los puertos.

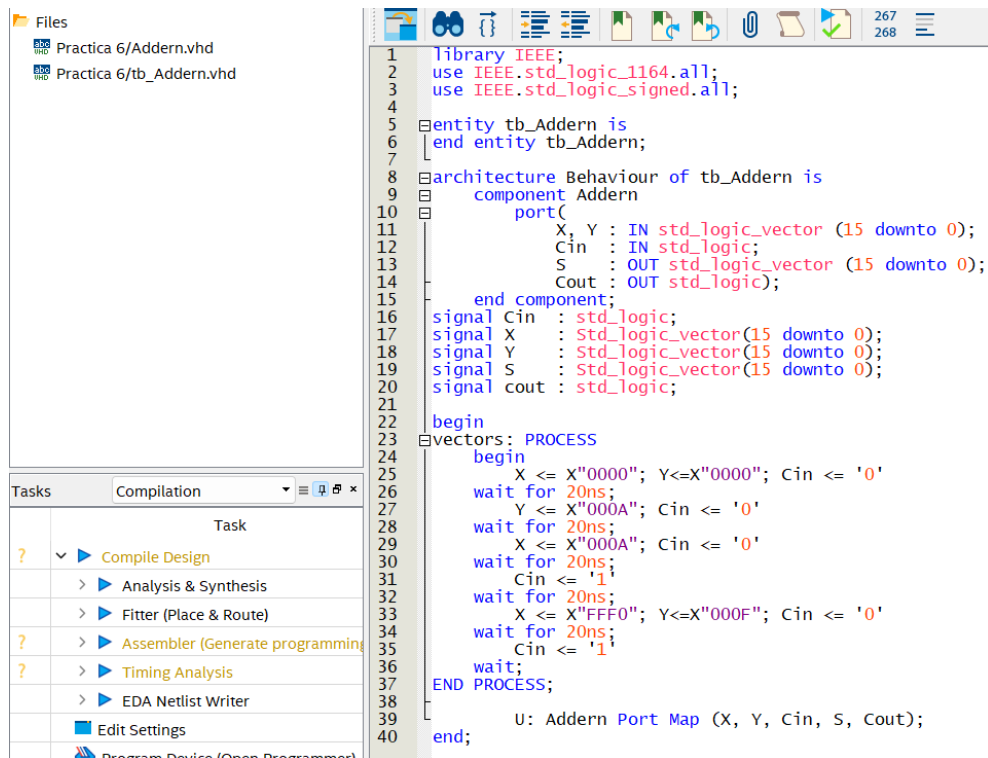
```
entity Addern is
    GENERIC( n : integer := 16);
    port ( X,Y : in std_logic_vector(n-1 downto 0);
          Cin : in std_logic;
          S   : out std_logic_vector(n-1 downto 0);
          Cout: out std_logic);
end Addern;

architecture Behaviour of Addern is
    signal Sum : std_logic_vector(n downto 0);
begin
    Sum <= ( '0' & X) + ( '0' & Y) + Cin;
    S   <= Sum( n-1 downto 0);
    Cout<= Sum(16);
end Behaviour;
```

Código descrito anteriormente.

Test Bench VHDL code

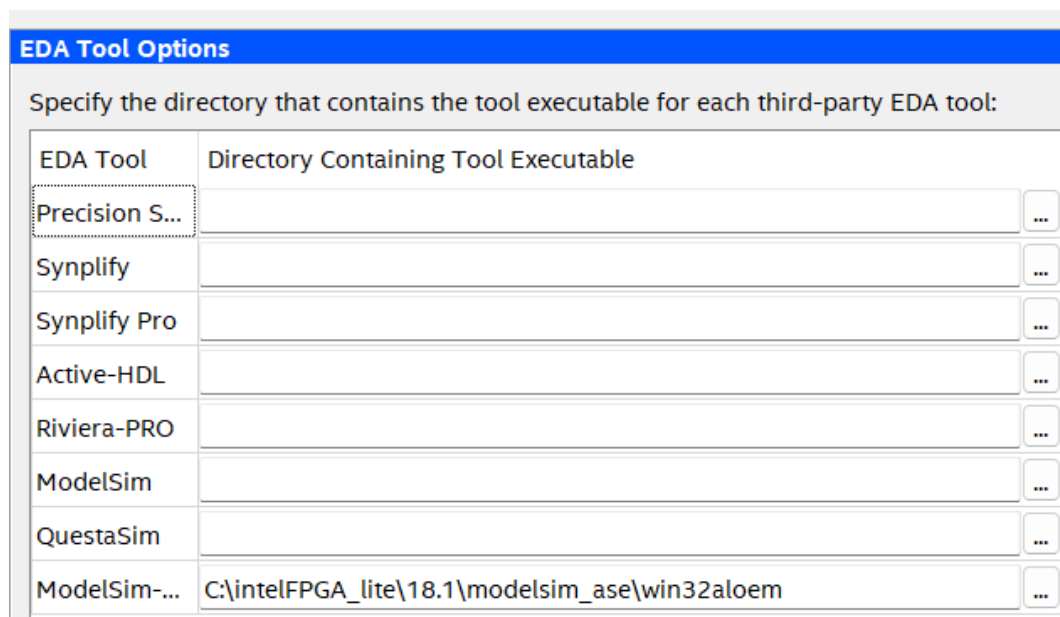
Después de esto escribimos el código tb_Addern, el cual básicamente es un código que indica la prueba que realizaremos sobre nuestro hardware en el ambiente simulado, dando diferentes valores a través del tiempo, dicho test bench, se podrá ver dentro de la aplicación de Modelsim.



Código Test Bench añadido al proyecto.

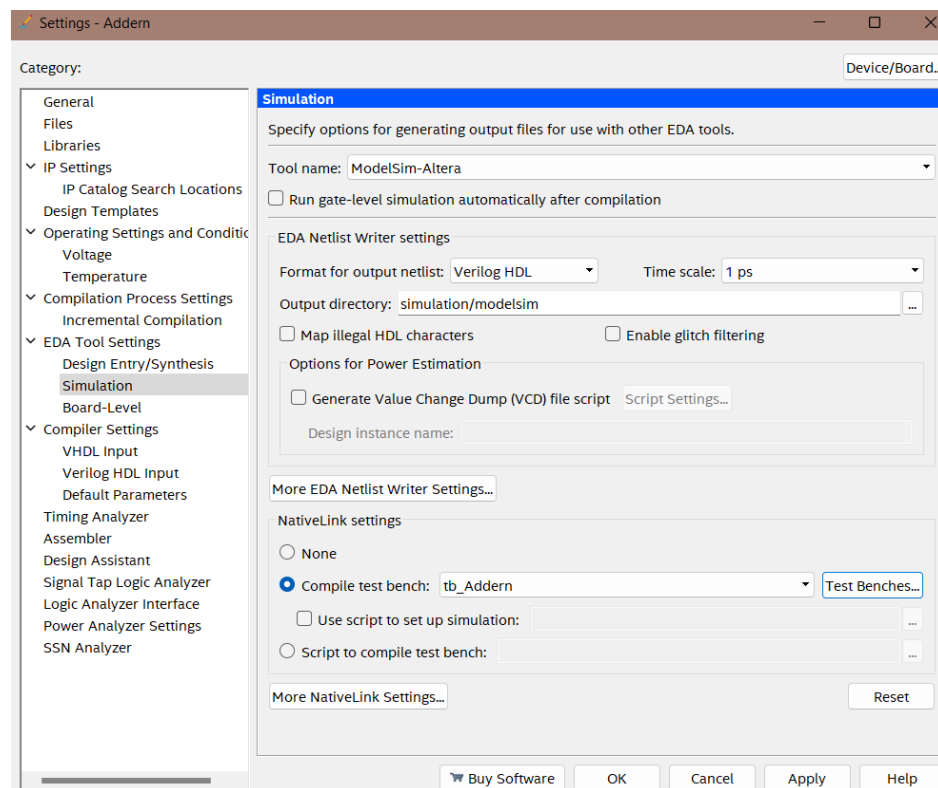
Configuración de Modelsim en Quartus

1. Debemos primero configurar dentro de Quartus el uso de Modelsim, para que se abra a la hora de que queramos simular dentro de ModelSim, por lo que debemos agregar la ruta de nuestra aplicación en **Tools** → **Options** → **EDA Tool Options**, esto nos abre la siguiente ventana emergente en donde debemos colocar la ruta mencionada en el apartado “ModelSim-Altera”.



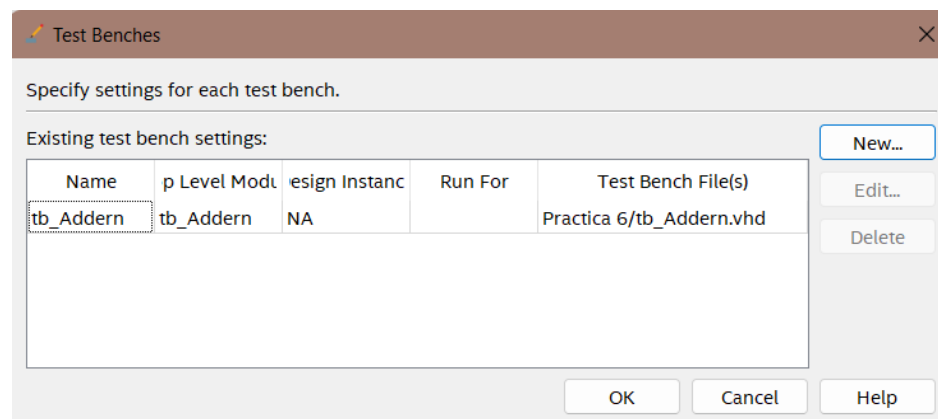
Ubicación actualizada.

2. Ahora debemos agregar el Test Bench para realizar la prueba por lo que deberemos dirigirnos a **Assignments** → **Settings** → **EDA Tool Settings** → **Simulation**. Aquí asignaremos ModelSim-Altera como la herramienta.



Test Bench configurado.

Ahora en Native Link Settings, pulsamos en “New” y añadimos el test bench que creamos.



Añadimos como Test Bench.

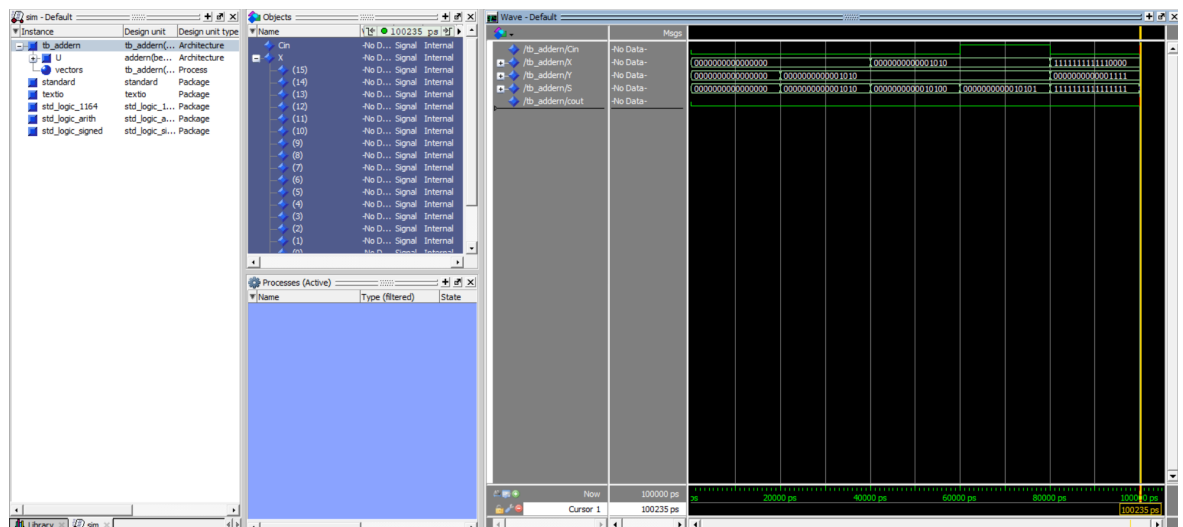
Ejecutamos la simulación

Para ejecutar nuestra simulación debemos compilar correctamente nuestros archivos.

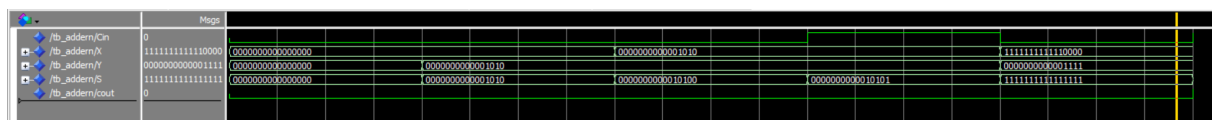
Flow Summary	
<<Filter>>	
Flow Status	Successful - Wed Oct 19 19:28:11 2022
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	Addern
Top-level Entity Name	Addern
Family	MAX 10
Device	10M50DAF484C7G
Timing Models	Final
Total logic elements	19 / 49,760 (< 1 %)
Total registers	0
Total pins	50 / 360 (14 %)
Total virtual pins	0
Total memory bits	0 / 1,677,312 (0 %)
Embedded Multiplier 9-bit elements	0 / 288 (0 %)
Total PLLs	0 / 4 (0 %)
UFM blocks	0 / 1 (0 %)
ADC blocks	0 / 2 (0 %)

Ejecución de la compilación correcta.

Después deberemos dirigirnos a **Tools** → **Run Simulation Tool** → **RTL Simulation**, esto nos desplegara una pantalla como la siguiente, en donde podremos encontrar el comportamiento de nuestro hardware según el Test bench que le mandamos.



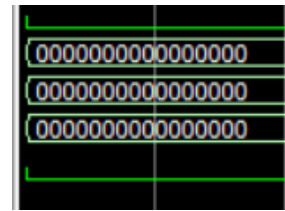
Pantalla de la herramienta Modelsim.



Comportamiento de nuestro hardware en el tiempo.

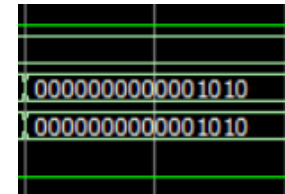
Explicación de las pruebas

Como podemos observar, nuestros valores iniciales en el Test bench fueron ambos '0' para X y Y, los cuales se ven así durante la prueba.



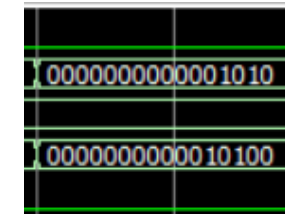
0000000000000000
0000000000000000
0000000000000000

Después en el nanosegundo 20, cambiamos el valor de la variable Y a X"000A", esto de igual manera se despliega correctamente en la gráfica. Podemos ver como Cin y X se mantienen en 0 (Señales respectivas de arriba hacia abajo) y la tercera señal muestra el valor de ... 1010, el cual es nuestra A que le mandamos en el test bench. Debido a que nuestro hardware representa un sumador, el resultado que se representa en la penúltima columna, tiene el mismo valor que Y, ya que Representa el $S = X + Y = 0x000A + 0x0000$.



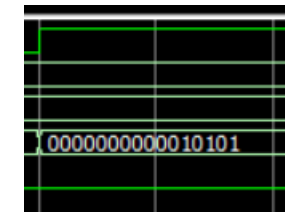
0000000000001010
0000000000001010

Lo mismo sucede ahora con la variable X que toma un valor de 0x000A, el cual al sumarse con 0x000A + 0x000A (... 1010 + ... 1010) = 0x0014 (... 0001 0100). Esto de igual manera se ve representado en el nanosegundo 40 en la simulación.



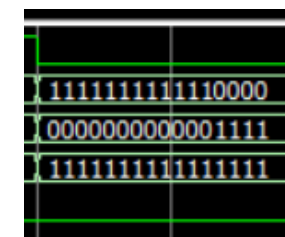
0000000000001010
0000000000001010

Luego agregamos un Cin con valor de '1', por lo que modifica la suma a (... 0001 0100) + 1 = ... 0001 0101, dicho resultado se ve en el nanosegundo 60 y el resultado coincide correctamente.



00000000000010101

Por último en el nanosegundo 80, modificamos tanto el valor de X y de Y, lo cual modifica la salida de la suma, los valores otorgados a X y Y fueron para X = 0xFFF0 y para Y = 000F, lo cual nos debería dar como resultado S = 0xFFFF, lo cual se representa en puros 1s en el resultado, tal y como se muestra en la gráfica. Además se apagó el Cin, por lo que el Cout, de igual manera tiene un valor de 0, ya que no existe.



1111111111110000
0000000000001111
1111111111111111

Problemas y soluciones

Reflexiones Individuales

Hector Pequeno: En esta práctica tuve la oportunidad de poder practicar la generación de código para programar Test Bench y estos incorporarlos a la herramienta Quartus Prime, además de recordar el funcionamiento de un Adder con signo, sus puertos de entrada y sus puertos de salida. Esto me pareció útil ya que en la clase de Sistemas Digitales avanzados utilizamos herramientas poco prácticas para visualizar el comportamiento de nuestro hardware, por lo que se agradece que exista una forma de enlazar una herramienta para visualizar fácilmente nuestro comportamiento.

Gabriela Álvarez: Dentro de esta práctica, fue de gran ayuda el poder generar los test bench para poder hacer diferentes pruebas con el Adder que generamos en la aplicación de quartus prime, me recordó a la materia de sistemas digitales avanzados donde de igual manera teníamos que generar los test bench para probar todo el funcionamiento de nuestro esquemático. Lo único de diferencia es que fue más fácil poder visualizar el comportamiento, a diferencia de cuando usábamos modelsim.

Bibliografía

^[1] Guía de la práctica:

<https://docs.google.com/document/d/1Zfh-XAKp1pPh4zRmS7gXqMhJrW2DZOsQWTfrk2Fx4es/edit>