República Bolivariana de Venezuela

Universidad de Carabobo

Facultad de Ingeniería

Escuela de Ingeniería Eléctrica y de Telecomunicaciones

Departamento de Sistemas y Automática

Cátedra de Lógica Digital

**Práctica # 3 – Sesión Única**

**Objetivo: Implementar circuitos combinacionales aritméticos utilizando VHDL y la tarjeta de desarrollo BASYS2**

Sección #05 de Laboratorio

**Integrantes:**

Carlos Hernández

C.I.: 25.829.471

Gianfranco Gasbarri

C.I.: 26.654.860

Fecha de entrega: 26/02/19

**Pre-Laboratorio**

1. **Actividad 1: Sumador Completo (Full Adder) en LogicWorks**

Ver **Anexo 1**

1. **Actividad 2: Algoritmo de la resta de 2 números binarios con complemento a 2**

Para restar 2 números enteros binarios con aritmética de complemento, se debe seguir los siguientes pasos:

* Se debe establecer un orden de magnitud para ambos operandos. El mismo puede adaptarse a cierto problema planteado o simplemente al operando con mayor cantidad de bits.
* Se obtiene el complemento a 2 del sustraendo (B) con tanta cantidad de bits como indique el orden de magnitud. En caso de ser necesario, completar con 0´s en las cifras más significativas del operando A para adecuarse al orden de magnitud.
* Realizar la suma del operando A con el complemento a 2 de B.
* Una vez hecha la suma, se deberá tomar una de las siguientes alternativas de acuerdo al acarrea por fuera del orden de magnitud:
  + Si el acarreo es igual a 1, la resta es positiva y la magnitud de la misma se leerá directamente.
  + Si el acarreo es 0, la resta es negativa y para leer la magnitud del resultado de la misma se deberá obtener su complemento a 2

1. **Actividad 3: Componente Comparador4 con VHDL**

Ver **Anexo 2**

**Post-Laboratorio**

1. **Tabla de resultados de las pruebas del sumador completo:** Ver **Anexo 1.5**
2. **Diseño y programa VHDL del sumador completo, diseño y programa VHDL del sumador de 4 bits, simulación:** Ver **Anexo 3**
3. **Diagrama de bloque y programa VHDL del comparador de magnitud, simulación y tabla de resultados:** Ver **Anexo 2**
4. **Diagrama de bloque y programa VHDL de la unidad aritmética; diseño y programa VHDL de alto nivel, tabla de resultados:** Ver **Anexo 4**

**Conclusiones**

En la presente práctica, se desarrollaron múltiples componentes. Uno de ellos fue el **Sumador de 4 bits**, con el cual podemos, no únicamente sumar, sino también restar 2 números de 4 bits. Para lograr esto último, debemos realizar la resta aplicando el algoritmo del complemento a 2, descrito anteriormente en el presente informe. Debemos invertir las entradas del operando **B** del Sumador y conectarle la fuente de tensión por el acarreo de entrada **Ci** (esto es equivalente a obtener el complemento a 1 del número y sumar 1).

Luego, cuando se realice la suma debemos pasar el resultado de la misma por un inversor controlado, el cual va a invertir dicha cantidad únicamente cuando el acarreo de salida **Co** del Sumador anterior sea igual a 0 (es decir, no hay acarreo). Finalmente, debemos sumar a dicha cantidad 1 o 0 dependiendo de la variable anterior. Esto se loga con un segundo componente Sumador de 4 bits.

Dicho componente puede ser utilizado para realizar operaciones de números mayores de 4 bits. Por ejemplo, si deseáramos sumar 2 números de 8 bits, emplearíamos 2 de estos componentes, en donde cada uno de ellos operaría un par de la mitad de bits de ambos operandos. Simplemente se tendría que conectar el **Co** del Sumador que operaría los bits menos significativos hacia el **Ci** del próximo componente y agrupamos todas las salidas en el mismo orden de conexión, arrojando finalmente un resultado.

Los circuitos integrados 7483 y 74283 poseen la funcionalidad del componente Sumador 4 bits. Ambos admiten las mismas entradas y salidas que el componente. Para el caso del 7483, las entradas correspondientes de A están en los puertos 10, 8, 3 y 1, de menos a más significativos, las de B: 11, 7, 4 y 16, el acarreo de entrada en el pin 13 y el de salida en el 14. En el caso del 74283, las entradas de A: 5, 3, 14, 12; las de B: 6, 2, 15 y 11; las de S: 4, 1, 13 y 10; el acarreo de entrada en el 7 y el de salida en el 9.

Otro de los componentes que se diseñaron, fue el **Comparador de 4 bits**. El mismo tiene la capacidad de comparar la magnitud de 2 operandos **A y B** directamente, donde podremos obtener información de la comparación a través de sus 3 salidas, encendiéndose únicamente la que cumple con la condición: mayor, igual o menor. También podremos comparar números mayores de 4 bits, de la siguiente manera: debemos rediseñar el componente de manera tal que acepte 3 nuevas entradas (una para cada caso posible en la comparación de magnitud) que se van a conectar entre sí con otras copias del mismo componente. Cada Comparador que opere un par de bits debe conectar sus salidas con las nuevas entradas del Comparador de los siguientes bits más significativos sucesivamente (y en las entradas del Comparador menos significativo, la entrada de la condición de **igual** debe conectarse a fuente y las demás a tierra).

El circuito integrado 7485 realiza la implementación del componente Comparador modificado. Las entradas de A: 10, 12, 13 y 15; las de B: 9, 11, 14 y B3; las entradas 2, 3 y 4 corresponden a las salidas de la comparación para el par de bits actual y las entradas 5, 6 y 7 van a las salidas de las comparaciones para el próximo circuito integrado que opere con bits menos significativos.

El componente **UA** fue diseñado con la finalidad de sumar o restar dos operandos **A y B**, controlando la operación que deseamos realizar con una señal **Op** y con salidas **R** para el resultado, **C** para el acarreo de salida (en el caso de la suma) y **Signo**. En el código de dicho componente, podemos observar que el segundo bloque **if/then/else** evalúa el caso en que se esté realizando una resta y el bit más significativo de la suma con complemento a 2 es igual a 1, lo que es equivalente a decir que existió un acarreo por fuera del orden de magnitud.

Este último componente se puede diseñar con varios componentes **Sumador** de 4 bits y compuertas lógicas, sería muy similar al descrito hace unos párrafos atrás para restar, sólo que al momento de invertir el operando **B**, debemos asegurarnos de que en realidad se desee hacerlo. Lo cual se lograr con un inversor controlado.

Se pudo comprobar, a través de las múltiples tablas de resultados y simulaciones anexadas al presente informe, que las pruebas realizadas en la sesión de Laboratorio cumplían con los valores esperados de acuerdo a la funcionalidad de cada uno de los componentes que fueron diseñados.

El lenguaje **VHDL** ofrece muchas facilidades a la hora de diseñar componentes aritméticos. Una de ellas es que a la hora de diseñar el componente **Comparador** de 4 bits, pudimos realizar la comparación de ambos vectores completos, sin la necesidad de comparar bit por bit. Además, a la hora de diseñar el componente **UA**, el bloque instruccional **if/then/else** nos permitía resolver el problema a través de algunas instrucciones de asignación. Cosa que no sucedía cuando realizamos el diseño en diagrama de bloques, para lo cual debíamos construir el circuito paso a paso, siguiendo la teoría y la lógica de componentes anteriores.

**ANEXOS**

**Anexo 1**

**Anexo 1.1**

**Diagrama de caja negra**

**A**

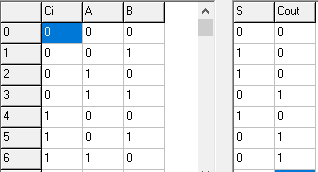
**S**

**B**

**Full Adder**

**Anexo 1.2**

**Tabla de la verdad**

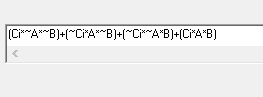


**Anexo 1.3**

**Expresiones lógicas**

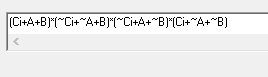
**Anexo 1.3.1**

**SOP Simplificada**



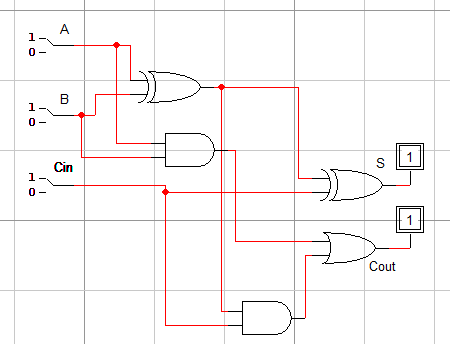
**Anexo 1.3.2**

**POS Simplificada**



**Anexo 1.4**

**Diseño**



**Anexo 1.5**

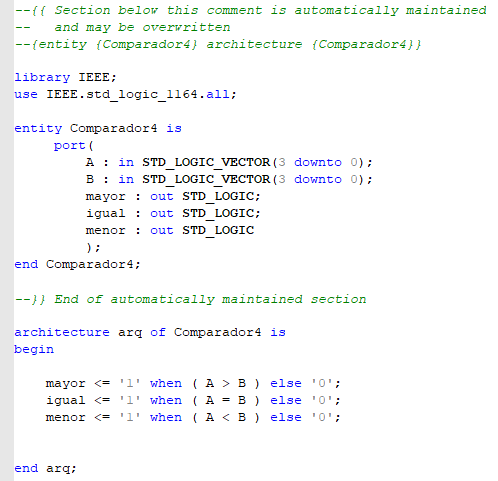
**Tabla de resultados**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **BTN3** | **SW1** | **SW0** | **LD1** | **LD0** |
| **OFF** | **OFF** | **OFF** | **OFF** | **OFF** |
| **OFF** | **OFF** | **ON** | **OFF** | **ON** |
| **OFF** | **ON** | **OFF** | **OFF** | **ON** |
| **OFF** | **ON** | **ON** | **ON** | **OFF** |
| **ON** | **OFF** | **OFF** | **OFF** | **ON** |
| **ON** | **OFF** | **ON** | **ON** | **OFF** |
| **ON** | **ON** | **OFF** | **ON** | **OFF** |
| **ON** | **ON** | **ON** | **ON** | **ON** |

**Anexo 2**

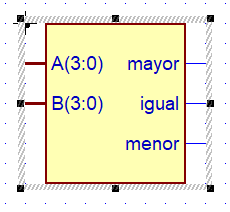
**Anexo 2.1**

**Código VHDL**



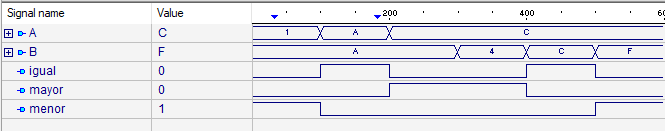
**Anexo 2.2**

**Diagrama de bloques**



**Anexo 2.3**

**Simulación**



**Anexo 2.4**

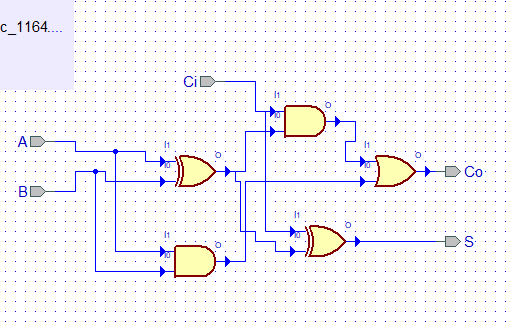
**Tabla de resultados (binario)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **Mayor** | **Igual** | **Menor** |
| **0001** | **1010** | **0** | **0** | **1** |
| **1010** | **1010** | **0** | **1** | **0** |
| **1100** | **1010** | **1** | **0** | **0** |
| **1100** | **0100** | **1** | **0** | **0** |
| **1100** | **1100** | **0** | **1** | **0** |
| **1100** | **1111** | **0** | **0** | **1** |

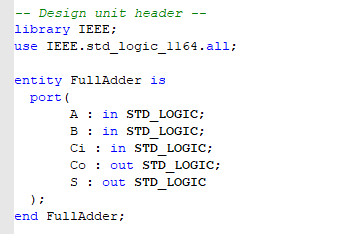
**Anexo 3**

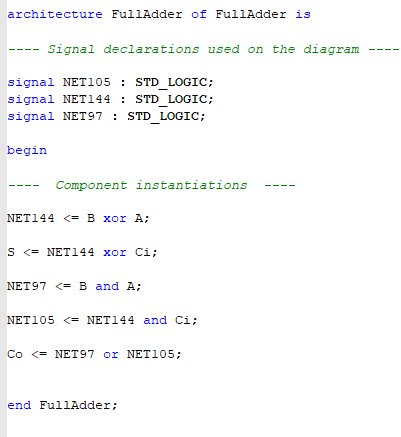
**Anexo 3.1**

**Diseño del sumador completo**



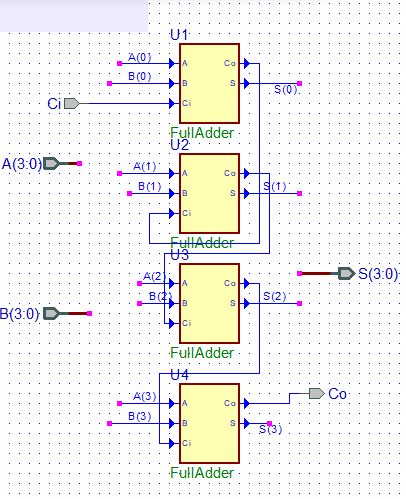
**Anexo 3.2**

**Código VHDL del sumador completo**



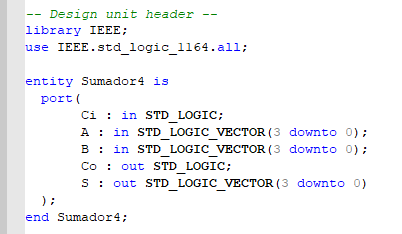
**Anexo 3.3**

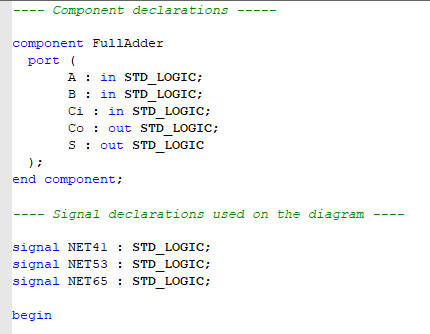
**Diseño del sumador de 4 bits**

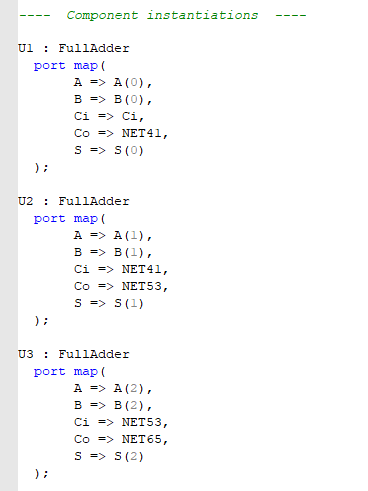


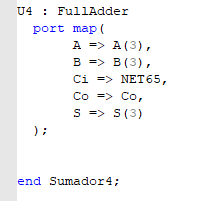
**Anexo 3.4**

**Código VHDL del sumador de 4 bits**

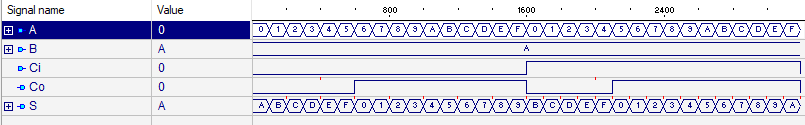








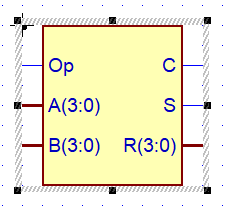
**Anexo 3.5**

**Simulación**

**Anexo 4**

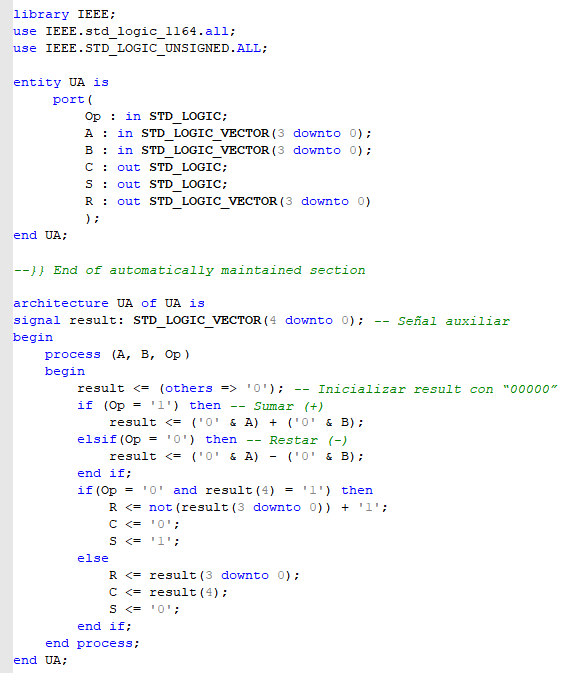
**Anexo 4.1**

**Diagrama de bloque UA**



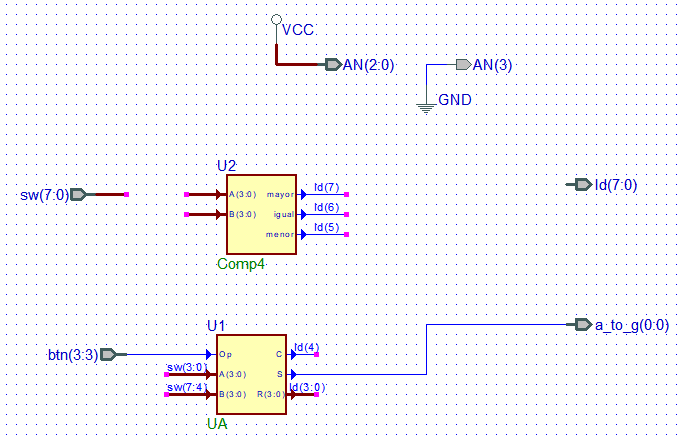
**Anexo 4.2**

**Código VHDL UA**

****

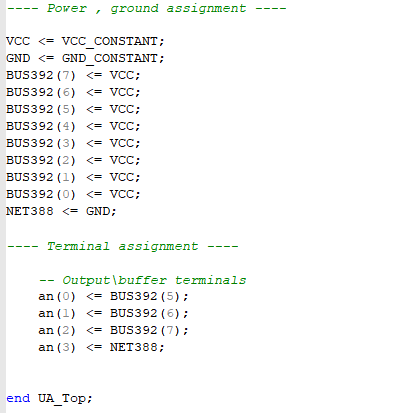
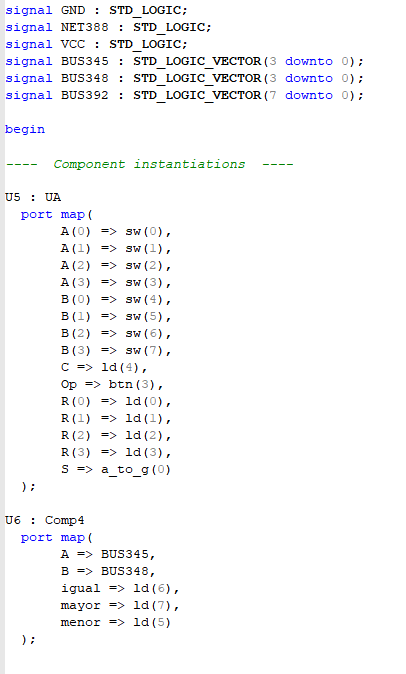
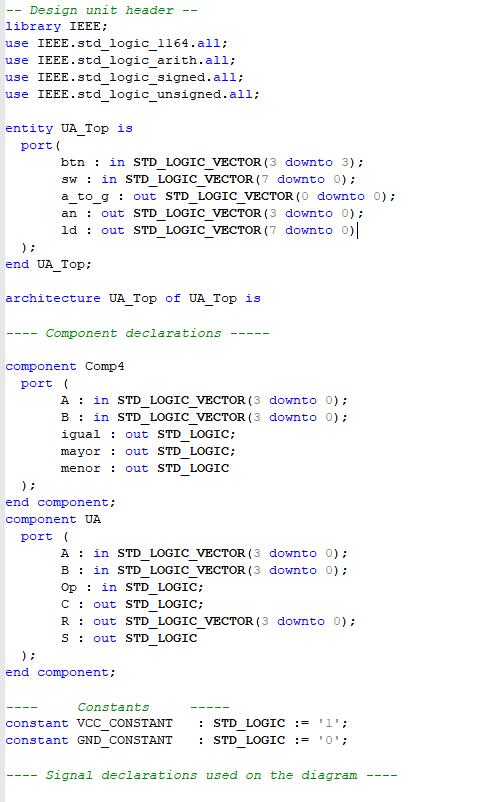
**Anexo 4.3**

**Diseño de alto nivel UA**



**Anexo 4.4**

**Código alto nivel UA**



**Anexo 4.5**

**Tabla de Resultados**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SW7** | **OFF** | **OFF** | **OFF** | **OFF** | **OFF** | **OFF** | **OFF** | **OFF** | **ON** | **ON** |
| **SW6** | **OFF** | **OFF** | **OFF** | **OFF** | **ON** | **ON** | **OFF** | **OFF** | **OFF** | **OFF** |
| **SW5** | **OFF** | **OFF** | **OFF** | **OFF** | **ON** | **ON** | **ON** | **ON** | **ON** | **ON** |
| **SW4** | **OFF** | **OFF** | **ON** | **ON** | **ON** | **ON** | **OFF** | **OFF** | **OFF** | **OFF** |
| **SW3** | **OFF** | **OFF** | **ON** | **ON** | **OFF** | **OFF** | **OFF** | **OFF** | **ON** | **ON** |
| **SW2** | **OFF** | **OFF** | **OFF** | **OFF** | **ON** | **ON** | **OFF** | **OFF** | **OFF** | **OFF** |
| **SW1** | **OFF** | **OFF** | **OFF** | **OFF** | **OFF** | **OFF** | **ON** | **ON** | **ON** | **ON** |
| **SW0** | **OFF** | **OFF** | **OFF** | **OFF** | **OFF** | **OFF** | **OFF** | **OFF** | **OFF** | **OFF** |
| **BTN3** | **OFF** | **ON** | **OFF** | **ON** | **OFF** | **ON** | **OFF** | **ON** | **OFF** | **ON** |
| **LD7** | **OFF** | **OFF** | **OFF** | **OFF** | **ON** | **ON** | **OFF** | **OFF** | **OFF** | **OFF** |
| **LD6** | **ON** | **ON** | **OFF** | **OFF** | **OFF** | **OFF** | **ON** | **ON** | **ON** | **ON** |
| **LD5** | **OFF** | **OFF** | **ON** | **ON** | **OFF** | **OFF** | **OFF** | **OFF** | **OFF** | **OFF** |
| **LD4** | **OFF** | **OFF** | **OFF** | **OFF** | **OFF** | **OFF** | **OFF** | **OFF** | **ON** | **OFF** |
| **LD3** | **OFF** | **OFF** | **ON** | **OFF** | **ON** | **OFF** | **OFF** | **OFF** | **OFF** | **OFF** |
| **LD2** | **OFF** | **OFF** | **OFF** | **ON** | **OFF** | **OFF** | **ON** | **OFF** | **ON** | **OFF** |
| **LD1** | **OFF** | **OFF** | **OFF** | **ON** | **ON** | **ON** | **OFF** | **OFF** | **OFF** | **OFF** |
| **LD0** | **OFF** | **OFF** | **ON** | **ON** | **ON** | **ON** | **OFF** | **OFF** | **OFF** | **OFF** |
| **AN3** | **ON** | **ON** | **ON** | **ON** | **ON** | **ON** | **ON** | **ON** | **ON** | **ON** |
| **a\_to\_g(0)** | **OFF** | **OFF** | **OFF** | **ON** | **OFF** | **OFF** | **OFF** | **OFF** | **OFF** | **OFF** |