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 # 4 – Sesión #02**

**Objetivo: Diseñar e implementar circuitos combinacionales con Multiplexores y Demultiplexores 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: 26/03/19

**Pre-Laboratorio**

1. **Desarrollo de componente MUX21:** Ver **Anexo 1**
2. **Desarrollo de componente MUX41 mediante árbol de multiplexores 2:1:** Ver **Anexo 2**
3. **Desarrollo de componente DEMUX14:** Ver **Anexo 3**
4. **Desarrollo de componente Codificador3:2:** Ver **Anexo 4**
5. **Problema “Sistema”:** Ver **Anexo 5**

**Laboratorio**

1. **Diseño de alto nivel:** Ver **Anexo 6**

**Conclusiones**

Se pudo observar en la presente práctica que diseñar componentes de multiplexores y demultiplexores mediante el lenguaje VHDL es muy sencillo. La sencillez con las que el mismo nos permite variables compuestas de tipo vector o array, es muy bien aprovechada cuando se definen las arquitecturas mediante la sentencia **when/others**. Lo cual es un hecho que nos permite realizar diseños muy rápidamente.

Otra facilidad observada es que mediante el estilo de flujo de datos, podemos obtener las expresiones lógicas de cada una de las salidas mediante un software auxiliar como el **Boole Deusto** y expresarlas mediante instrucciones de asignación utilizando VHDL.

En la presente práctica, diseñamos un componente **MUX 4:1** mediante un árbol de multiplexores **2:1** construidos previamente. Podemos repetir esta operación para construir componentes de orden mayor, como unos posibles **MUX** **8:1, MUX 16:1 o MUX 32:1**, respetando el orden de las variables de selección en cada uno de los componentes, como bien se apoya en sus fundamentos teóricos. De la misma manera, se puede llevar la idea a los demultiplexores.

En la sesión anterior, trabajamos con los multiplexores y demultiplexores a nivel de circuitos integrados, apoyándonos con el uso del **Protoboard** y el software **LogicWorks**. Nos encontramos que en dicha oportunidad ya existían los componentes que realizaban dichas labores, tales como eran los integrados **74153/75151/74139/74138**. En esta oportunidad, dichos componentes no existían. Debíamos crearlos, a través de lenguaje VHDL y el diagrama de bloques, y una vez a la mano, también contábamos con la opción de reutilizarlos. En ambas sesiones, se propuso el diseño de un sistema que satisfacía una serie de problemas planteados, en el cual se empleaban codificadores y decodificadores para darles solución.

Similar al problema de la sesión anterior, el problema planteado bien podría subdividirse en varios subproblemas para su simplificación. Uno de ellos, era la codificación de la entrada compuesta **E** para su posterior salida mediante un display 7 segmentos. Para ello, empleamos el componente **Codificador32**, creado en esta sesión, y el componente **Conversor BCD7Seg,** creado en sesiones anteriores.

Otro de ellos se trataba de la codificación de las entradas de los 4 dispositivos, con 2 bits para cada uno de ellos, mediante **multiplexores 4:1** empleando las salidas del componente **Codificador32** como variables de selección de los mismos. Las salidas de los multiplexores corresponden a los 2 últimos **LEDs** de la tarjeta BASYS y representarían los bits del dato seleccionado.

Finalmente, debíamos crear un detector de multiplicidad de 3 y mostrarlo a través del siguiente **LED** de la tarjeta. Nuestra propuesta fue diseñada sin tomar el 0 como múltiplo de 3 y se realizó con 4 componentes **MUX 2:1** para cada uno de los dispositivos de entrada del sistema, luego algunas de esas salidas pasarían por la función lógica que modela la detección de la multiplicidad según la tabla de la verdad y finalmente, esas salidas pasarían a un último multiplexor **4:1** que tendría como salida la activación o no de la multiplicidad de la entrada de 4 bits seleccionada mediante el **LED** mencionado.

**ANEXOS**

**Anexo 1**

**Anexo 1.1**

**Diagrama de caja negra**

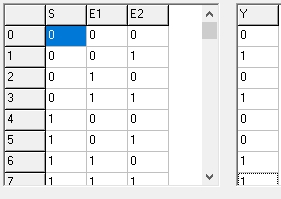
**S**

**MUX 2:1**

**Y**

**Anexo 1.2**

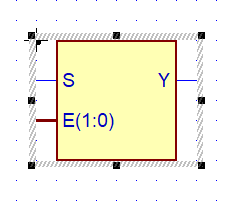
**Tabla de la verdad**





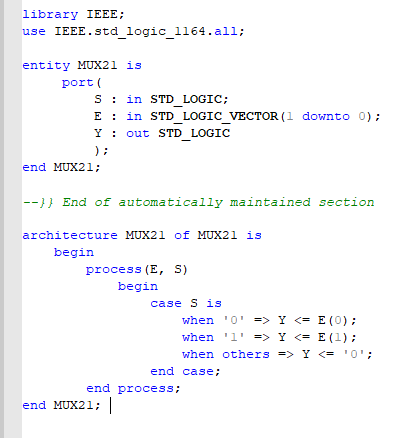
**Anexo 1.3**

**Símbolo**



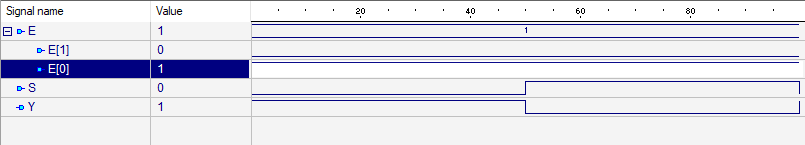
**Anexo 1.4**

**Código VHDL**



**Anexo 1.5**

**Simulación**



**Anexo 2**

**Anexo 2.1**

**Diagrama de caja negra**

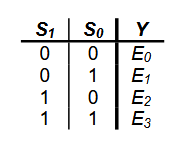
**S**

**MUX 4:1**

**Y**

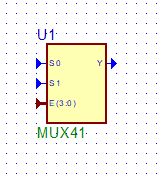
**Anexo 2.2**

**Tabla de la verdad**



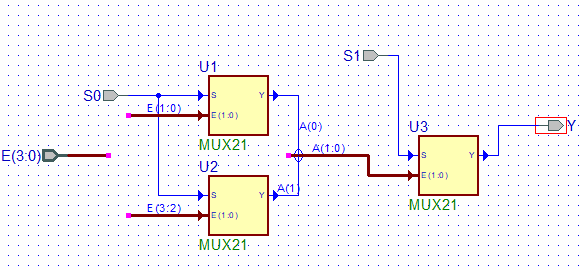
**Anexo 2.3**

**Símbolo**



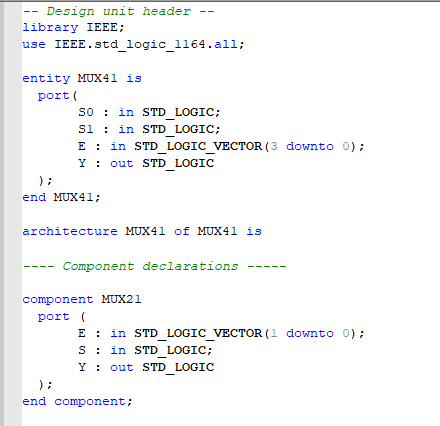
**Anexo 2.4**

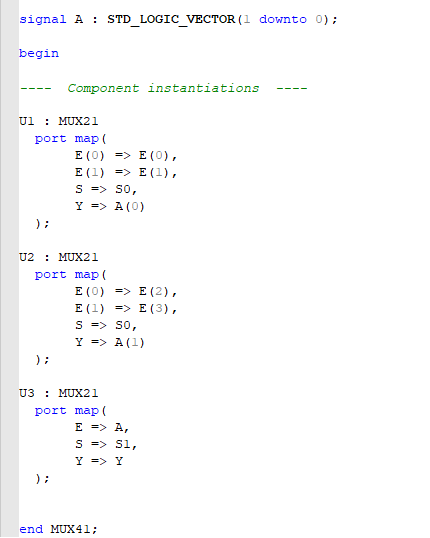
**Diseño**



**Anexo 2.5**

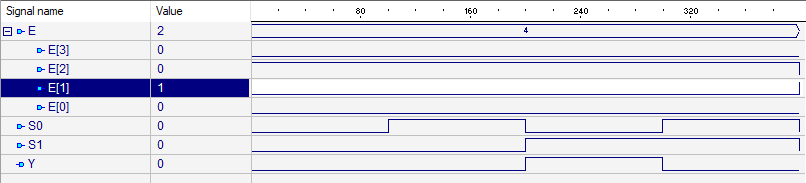
**Código VHDL**





**Anexo 2.6**

**Simulación**



**Anexo 3**

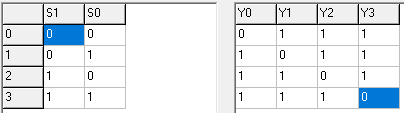
**Anexo 3.1**

**Diagrama de caja negra**

**DEMUX 1:4**

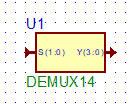
**Anexo 3.2**

**Tabla de la verdad**



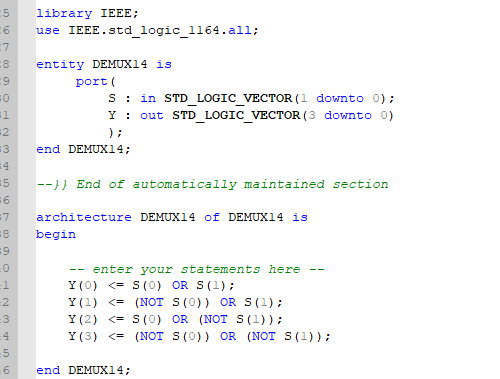
**Anexo 3.3**

**Símbolo**



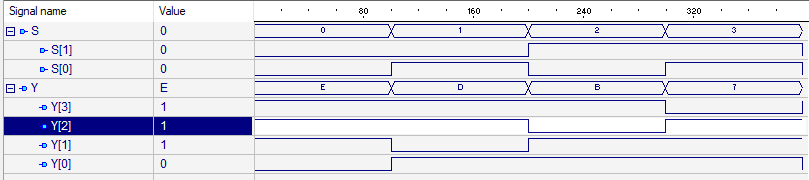
**Anexo 3.4**

**Código VHDL**



**Anexo 3.5**

**Simulación**



**Anexo 4**

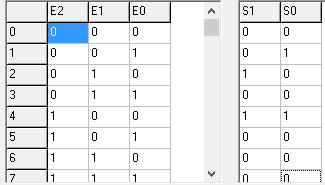
**Anexo 3.1**

**Diagrama de caja negra**

**Codificador3:2**

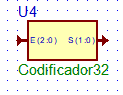
**Anexo 3.2**

**Tabla de la verdad**



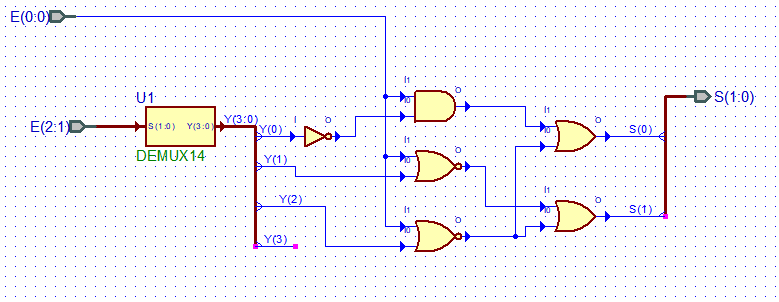
**Anexo 3.3**

**Símbolo**



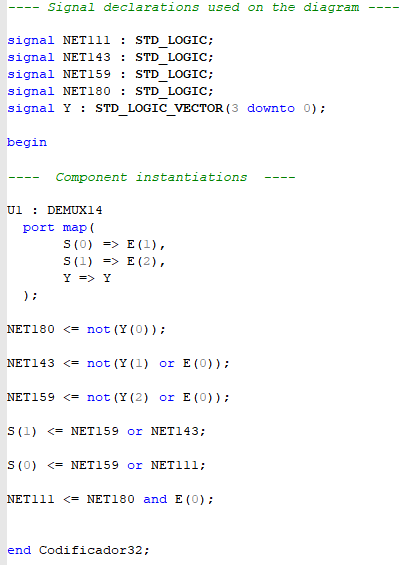
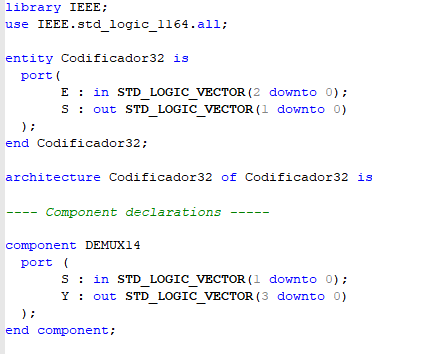
**Anexo 3.4**

**Diseño**



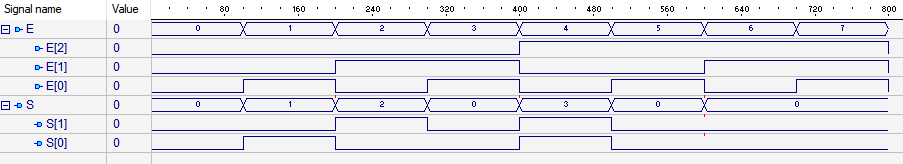
**Anexo 3.5**

**Código VHDL**



**Anexo 3.6**

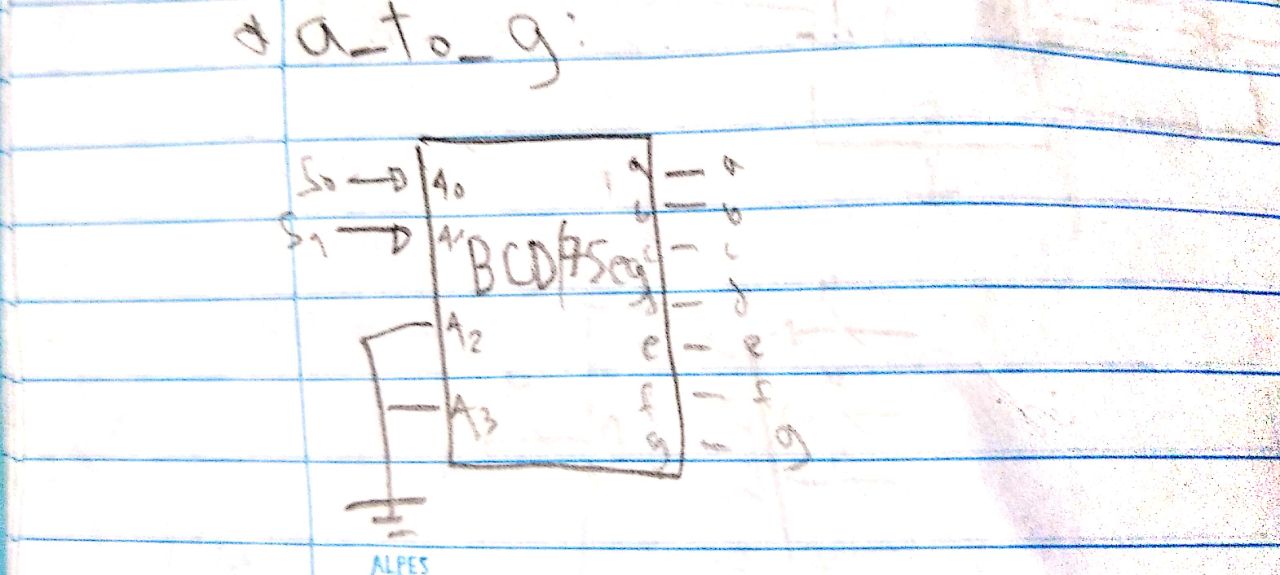
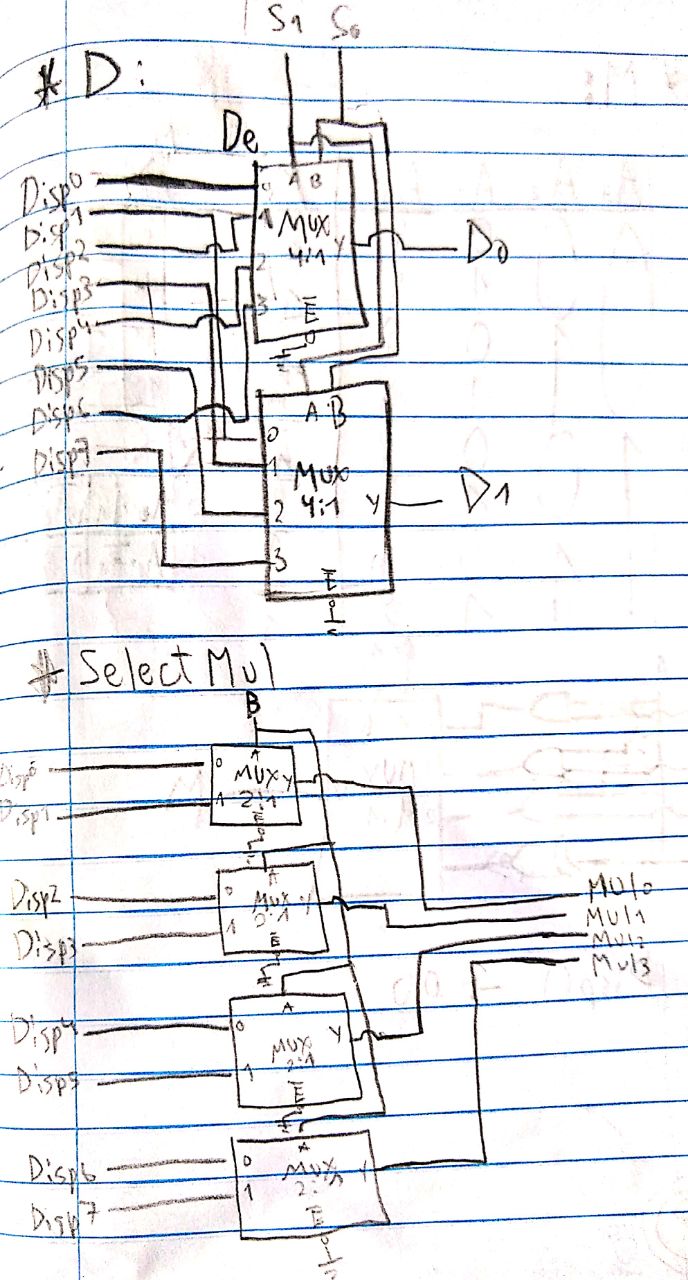
**Simulación**

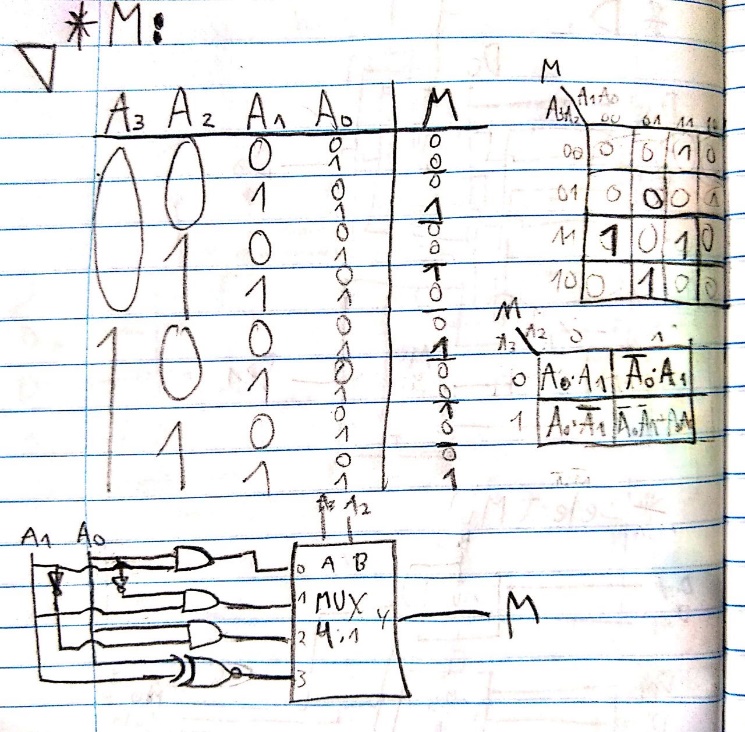


**Anexo 5: Problema “Sistema”**

**Anexo 5.1**

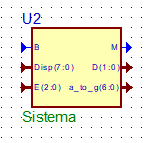
**Análisis**

****

****

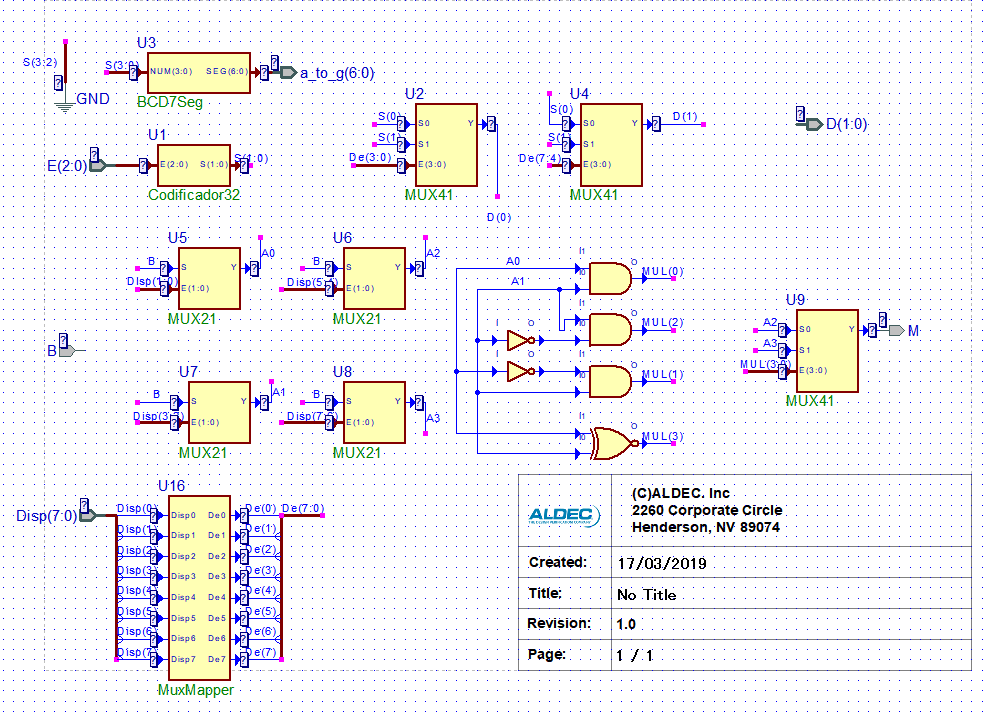
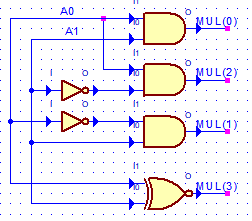
**Anexo 5.2**

**Símbolo**



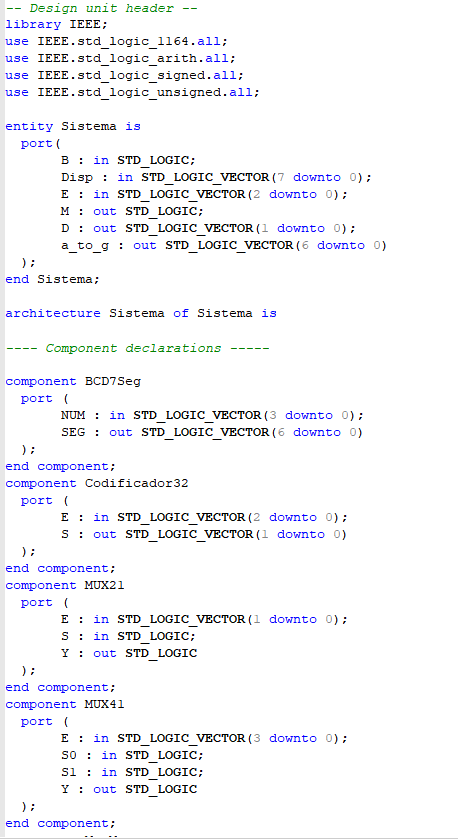
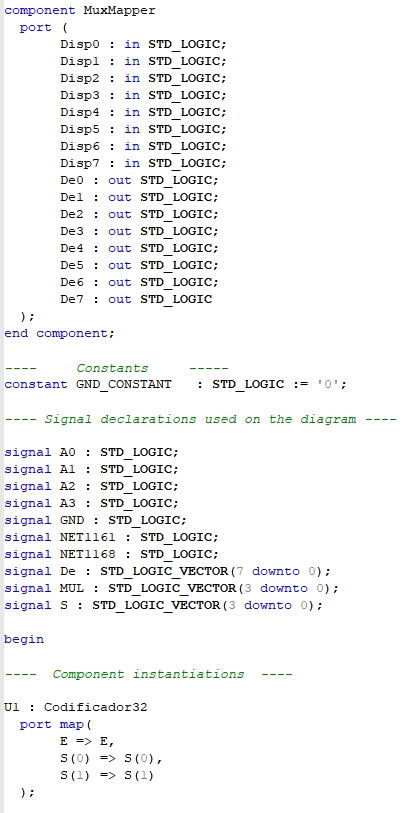
**Anexo 5.3**

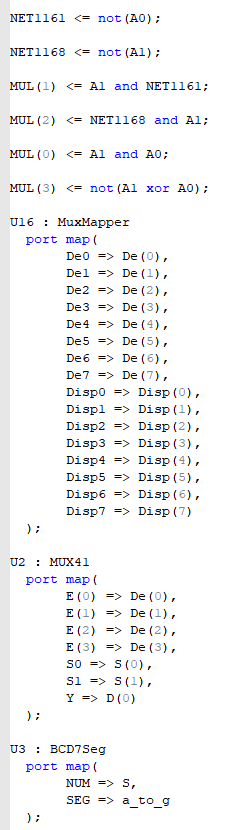
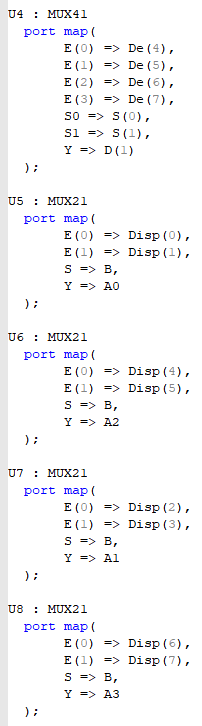
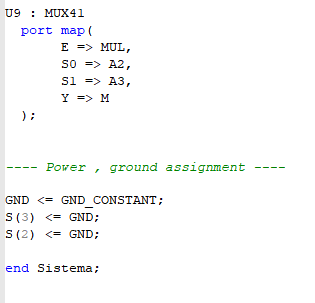
**Diseño**



**Anexo 5.4**

**Código**

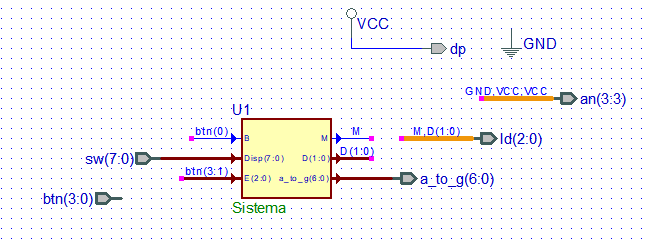




**Anexo 6: Diseño de alto nivel**

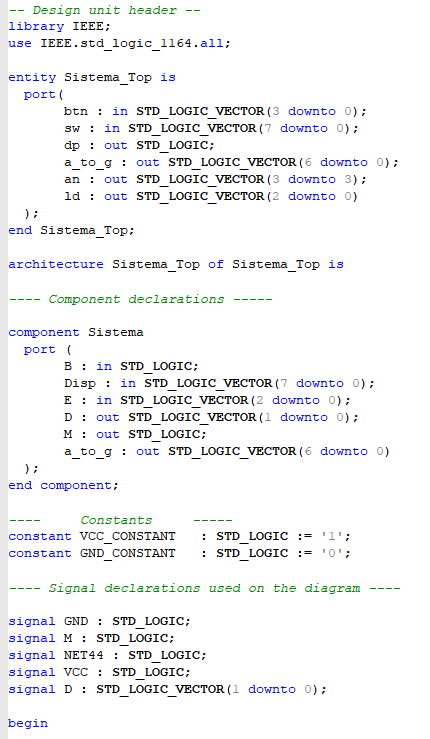
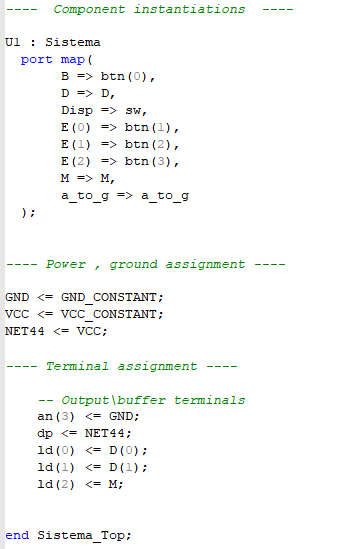
**Anexo 6.1**

**Diseño**



**Anexo 6.2**

**Código VHDL**



**Anexo 6.3**

**Tabla de resultados**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SW7** | **OFF** | **OFF** | **OFF** | **OFF** | **ON** |
| **SW6** | **OFF** | **OFF** | **OFF** | **ON** | **OFF** |
| **SW5** | **OFF** | **OFF** | **OFF** | **OFF** | **ON** |
| **SW4** | **OFF** | **OFF** | **OFF** | **OFF** | **OFF** |
| **SW3** | **OFF** | **OFF** | **OFF** | **OFF** | **ON** |
| **SW2** | **OFF** | **ON** | **ON** | **ON** | **OFF** |
| **SW1** | **OFF** | **OFF** | **OFF** | **OFF** | **ON** |
| **SW0** | **OFF** | **OFF** | **ON** | **OFF** | **OFF** |
| **BTN3** | **OFF** | **OFF** | **OFF** | **ON** | **OFF** |
| **BTN2** | **OFF** | **OFF** | **ON** | **OFF** | **OFF** |
| **BTN1** | **OFF** | **ON** | **OFF** | **OFF** | **OFF** |
| **BTN0** | **OFF** | **OFF** | **OFF** | **OFF** | **ON** |
| **LD2** | **OFF** | **OFF** | **ON** | **OFF** | **ON** |
| **LD1** | **OFF** | **OFF** | **OFF** | **OFF** | **ON** |
| **LD0** | **OFF** | **ON** | **OFF** | **ON** | **OFF** |
| **AN3** | **ON** | **ON** | **ON** | **ON** | **ON** |
| **a\_to\_g** | **0** | **1** | **2** | **3** | **0** |