

**DEPARTAMENTO DE ELECTRÓNICA Y AUTOMÁTICA**

**FACULTAD DE INGENIERÍA – UNIVERSIDAD NACIONAL DE SAN JUAN**

Informe de Laboratorio Nº 2

**Asignatura:** Temas Especificos de Electronica Digital I

**Electronca**

***Autor:***

*Villafañe Hausser, Mario Ezequiel – Registro 27857*

**1º Semestre**

**Año 2021**

# Índice de Contenidos

1. Objetivos Página 3
2. Parte A Página 3
   1. Objetivo Página 3
   2. Desarrollo Página 3
   3. Problemas y Soluciones Página 4
3. Parte B Página 4
   1. Objetivo Página 4
   2. Desarrollo Página 4
   3. Problemas y Soluciones Página 7
4. Parte C Página 7
   1. Objetivo Página 7
   2. Desarrollo Página 7
   3. Problemas y Soluciones Página 8
5. Parte D Página 8
   1. Objetivo Página 8
   2. Desarrollo Página 9
   3. Camino Critico Página 11
   4. Problemas y Soluciones Página 12
6. Carta de Gantt Página 12
7. Conclusiones Página 13
8. Apéndices Página 14
   1. Código Decodificador 3 a 8 Página 14
   2. Código Decodificador BCD a 7 segmentos Página 14
   3. Código Decodificador HEXA a 7 segmentos Página 16
   4. Código Multiplexor 4 a 1 Página 16
   5. Código Laboratorio 2 Página 17

# Índice de Figuras

Figura 01. Decodificador de 3 a 8 Página 3

Figura 02. Simulación del Decodificador de 3 a 8 Página 4

Figura 03. Decodificador de BCD a 7 segmentos Página 5

Figura 04. RTL Viewer Decodificador BCD a 7 segmentos Página 6

Figura 05. Simulación del Decodificador de BCD a 7 segmentos Página 6

Figura 06. Simulación del Decodificador de HEXA a 7 segmentos Página 8

Figura 07. Esquemático Laboratorio 2 Página 8

Figura 08. Multiplexor 4x1Página 9

Figura 09. RTL Viewer Laboratorio 2 Página 9

Figura 10. RTL Viewer Multiplexer Página 10

Figura 11. RTL Decodificador BCD a 7 segmentos Página 10

Figura 12. Data path Página 11

Figura 13. Locate Path Technology View Página 11

Figura 14. Locate Path Chip Planner Página 12

# Índice de Tablas

Tabla 01. Tabla de verdad del Decodificador de 3 a 8 Página 4

Tabla 02. Tabla de verdad del Decodificador de BCD a 7 segmentos Página 5

Tabla 03. Tabla de verdad del Decodificador de Hexa a 7 segmentos Página 7

Tabla 04. Carta de Gantt Página 13

# Objetivos

* Utilización de las instrucciones concurrentes conditional signal assignment (CSA), selected signal assignment (SSA), component-declaration y component-instantiation.
* Familiarización con el software Quartus y con el software ModelSim.
* Uso de sub-componentes para crear un sistema jerárquico.
* Uso de la guía de usuario del baord DE2-115
* Uso de archivo de restricción para pines E/S del FPGA vinculados a hardware del board DE2-115.
* Realizar simulación funcional.
* Realizar estudio y determinación de camino crítico combinacional.
* Generar archivo de configuración de FPGA.
* Configuración del FPGA con el código VHDL correspondiente.

# Parte A

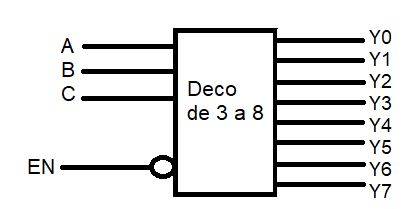
## Objetivo

Usando (solo) instrucciones concurrentes describa el funcionamiento de un decodificador genérico de 3 a 8. Implemente el diseño en la placa DE2-115, usando los switches como entradas del decodificador y los LEDs como salida.

## Desarrollo

Los decodificadores tienen como función detectar la presencia de una determinada combinación de bits en sus entradas y señalar la presencia de este código mediante un cierto nivel de salida. Un decodificador posee N líneas de entrada para gestionar N bits y en una de las 2N líneas de salida indica la presencia de una o más combinaciones de n bits. Para cualquier código dado en las entradas solo se activa una de las 2N posibles salidas.

Para este caso en particular se poseen tres entradas, por lo que se va a controlar 8 salidas. Además, posee una entrada de habilitación que se activa en bajo, caso contrario todas las salidas se ponen en un valor de cero lógico.

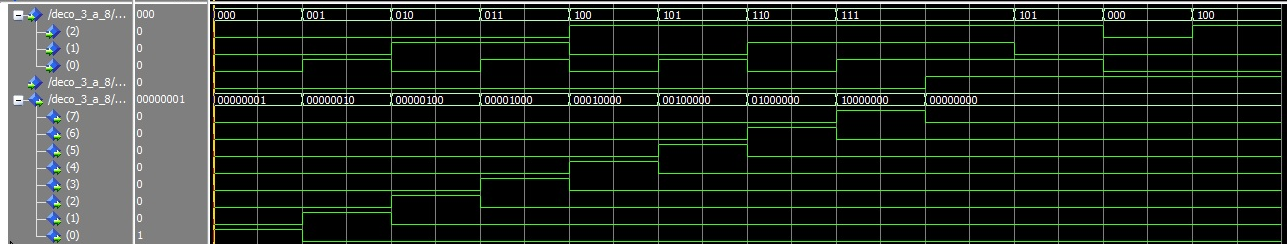


*Figura 01. Decodificador de 3 a 8*

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TABLA DE VERDAD DECODIFICADOR 3 A 8 | | | | | | | | | | | |
| A | **B** | **C** | **EN** | **Y0** | **Y1** | **Y2** | **Y3** | **Y4** | **Y5** | **Y6** | **Y7** |
| X | X | X | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

*Tabla 01*

En la Figura 01 se puede ver un esquemático de las entradas y salidas del decodificador mientras que en la Tabla 01 se observa la tabla de verdad del decodificador. A continuación, se puede ver la simulación con la herramienta MultiSim del funcionamiento del código VHDL implementado. Dicho código se encuentra en el Apéndice A.



*Figura 02. Simulación del Decodificador de 3 a 8*

## Problemas y Soluciones

En este código implementado no se encontraron errores ni advertencias significativas que requieran alguna revisión.

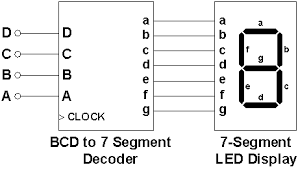
# Parte B

## Objetivo

Realice un decodificador de BCD a 7-segmentos. El decodificador tiene cuatro bits de entrada que codifica un número entre 0 y 9. Como salida, el decodificador tiene siete salidas, segmentos a, b, c, d, e, f y g’, correspondiendo cada letra a un segmento del display. Cada segmento es activo en bajo.

## Desarrollo

Es un elemento digital que funciona a base de estados lógicos, con los cuales indica una salida determinada basándose en un dato de entrada característico, su función operacional se basa en la introducción a sus entradas de un número en código binario correspondiente a su equivalente en decimal para mostrar en los siete pines de salida establecidos para el integrado, una serie de estados lógicos que están diseñados para conectarse a un elemento alfanumérico en el que se visualizará el número introducido en las entradas del decodificador. El elemento alfanumérico que se conecta a las siete salidas del decodificador también está diseñado para trabajar con estados lógicos, es un dispositivo elaborado con un arreglo de LED de tal manera que muestre los números decimales desde el cero hasta el nueve dependiendo del dato recibido desde el decodificador, a este elemento se le conoce con el nombre de display ó dispositivo alfanumérico de 7 segmentos.



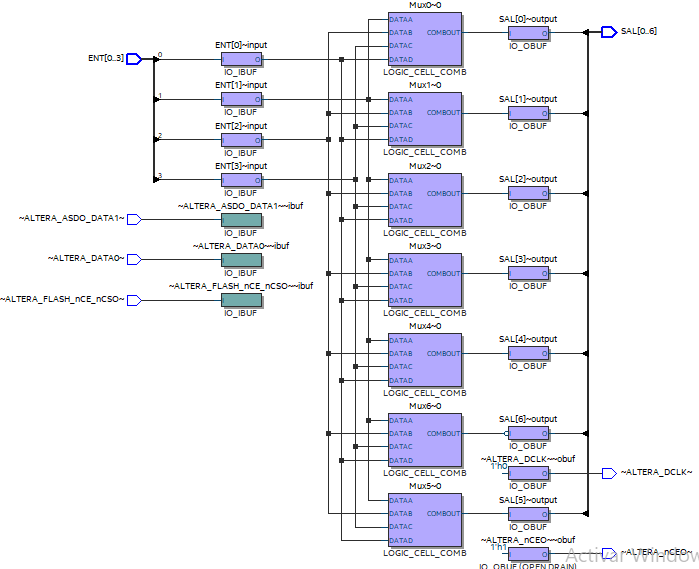
*Figura 03. Decodificador de BCD a 7 segmentos*

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TABLA DE VERDAD DECODIFICADOR BCD A 7 SEGMENTOS | | | | | | | | | | | |
| A | **B** | **C** | **D** | **a** | **b** | **c** | **d** | **e** | **f** | **g** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | X | X | X | X | X | X | X |
| 1 | 0 | 1 | 1 | X | X | X | X | X | X | X |
| 1 | 1 | 0 | 0 | X | X | X | X | X | X | X |
| 1 | 1 | 0 | 1 | X | X | X | X | X | X | X |
| 1 | 1 | 1 | 0 | X | X | X | X | X | X | X |
| 1 | 1 | 1 | 1 | X | X | X | X | X | X | X |

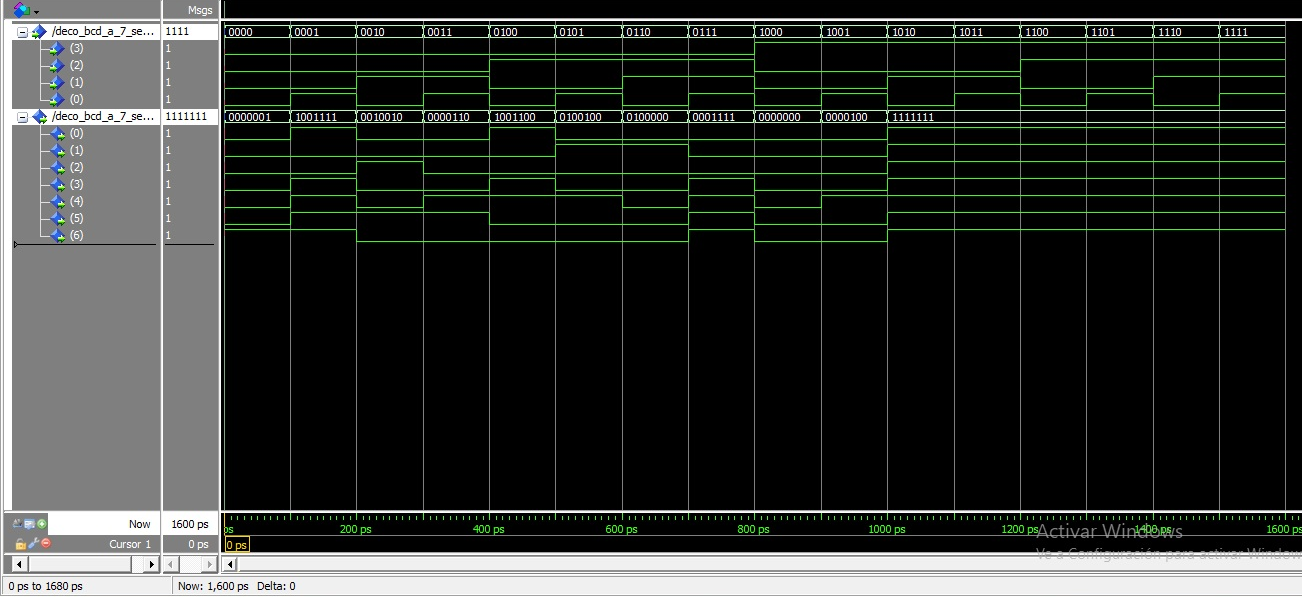
*Tabla 02*

En la Figura 03 se puede ver un pequeño esquema del diseño propuesto para la práctica. En la Tabla 02 se puede apreciar la tabla de verdad que se deberá programar en el código VHDL. Resultado de ese código y de su correspondiente simulación se puede ver en la Figura 04, donde observamos el diseño de la herramienta Quartus. La misma implementa 7 LUT (LookUp Tables).

Además, en la Figura 05 vemos el resultado de la simulación. Esta como todas las simulaciones fueron realizadas en la herramienta MultiSim, usando los comandos “force” y “run”. Este tipo de simulación es muy útil cuando se requiere una simulación rápida y el diseño no exige muchas líneas de control en lógica combinacional. Para diseños más complejos, con lógica secuencial o cuando se requiere hacer un control completo del funcionamiento es mucho mejor realizar el TestBench.



*Figura 04. RTL Viewer Decodificador BCD a 7 segmentos*



*Figura 05. Simulación del Decodificador de BCD a 7 segmentos*

## Problemas y Soluciones

En este código implementado no se encontraron errores ni advertencias significativas que requieran alguna revisión.

# Parte C

## Objetivo

Realice un decodificador de Hexadecimal a 7-segmentos. El decodificador tiene cuatro bits de entrada que codifica un número entre 0 y 15. Como salida, el decodificador tiene siete salidas, segmentos a, b, c, d, e, f y g, correspondiendo cada letra a un segmento del display. Cada segmento es activo en bajo.

## Desarrollo

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TABLA DE VERDAD DECODIFICADOR HEXA A 7 SEGMENTOS | | | | | | | | | | | |
| A | **B** | **C** | **D** | **a** | **b** | **c** | **d** | **e** | **f** | **g** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |

Este circuito es similar al desarrollado en el apartado 3, la diferencia es que no solo codifica los valores del 0 al 9, sino además también los valores A a F, del código Hexadecimal. Es decir que vamos a tener 16 posibilidades en vez de 10, como en el caso anterior. Para poder representar las letras A, C, E y F no existe mayor inconveniente. Pero resulta confuso con las letras B y D con el 8 y 0 respectivamente. Para ello se decidió representar a dichas letras en minúscula, para evitar así cualquier tipo de confusión.

La tabla 03 muestra el valor de las salidas al presentarse las combinaciones correspondientes en la entrada y en la Figura 06 la simulación correspondiente.

*Tabla 03*



*Figura 06. Simulación del Decodificador de HEXA a 7 segmentos*

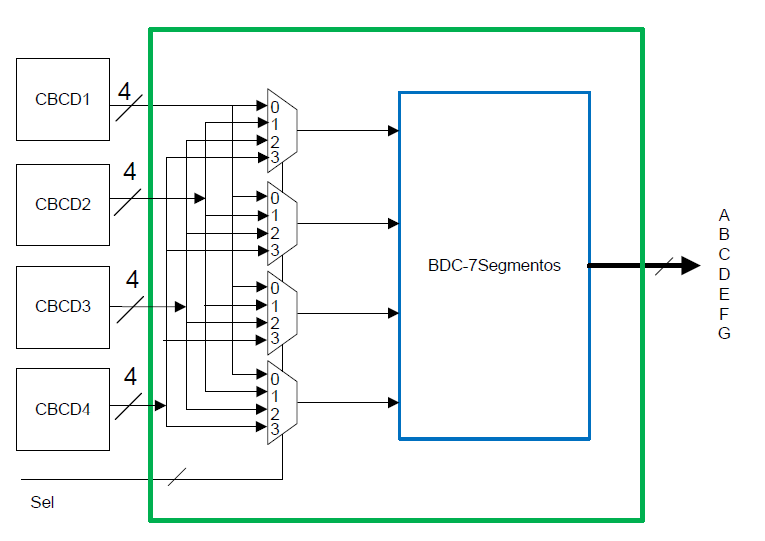
## Problemas y Soluciones

En este código implementado no se encontraron errores ni advertencias significativas que requieran alguna revisión.

# Parte D

## Objetivo

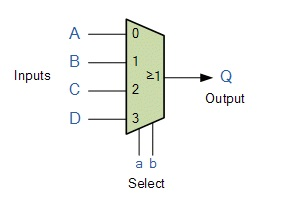
Describa el siguiente sistema en VHDL:



*Figura 07. Esquemático Laboratorio 2*

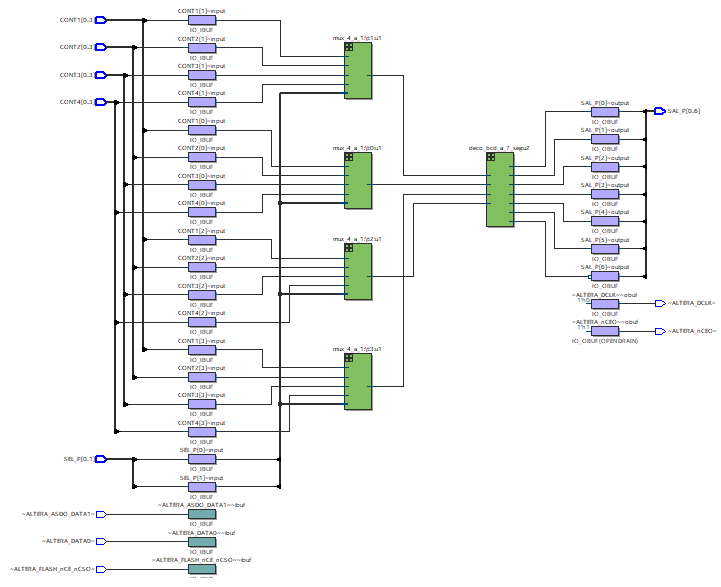
## Desarrollo

Para implementar el circuito de la Figura 07 se debe diseñar en código VHDL un multiplexor y un decodificador de BCD a 7 segmentos. Para este último se va a implementar el mismo del apartado 3. Los multiplexores son circuitos combinacionales con varias entradas y una única salida de datos. Están dotados de entradas de control capaces de seleccionar una, y sólo una, de las entradas de datos para permitir su transmisión desde la entrada seleccionada hacia dicha salida. En el campo de la electrónica el multiplexor se utiliza como dispositivo que puede recibir varias entradas y transmitirlas por un medio de transmisión compartido. Para ello lo que hace es dividir el medio de transmisión en múltiples canales, para que varios nodos puedan comunicarse al mismo tiempo.

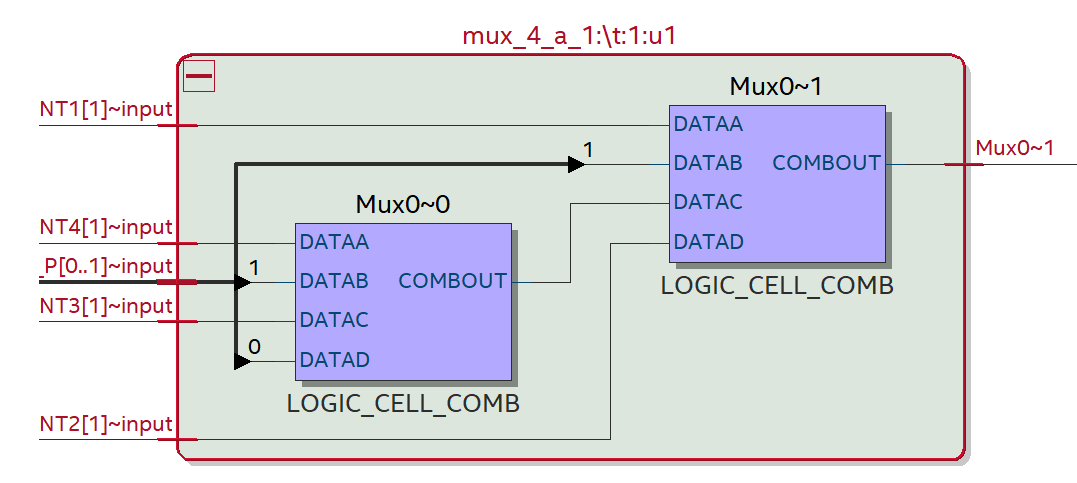


*Figura 08. Multiplexor 4x1*

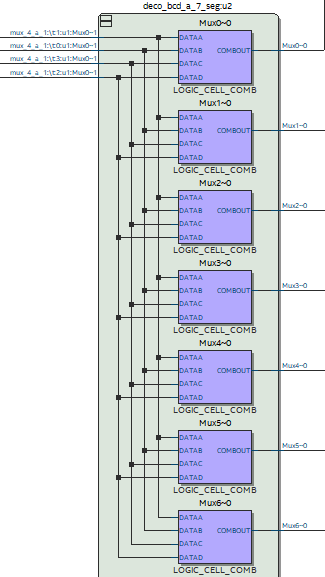
En la siguiente Figura podremos ver el diagrama completo producto del diseño en VHDL. Vamos a ver en particular en las composiciones de un bloque multiplexor y el decodificador. En la cuenta se podrá ver un total de 15 CLB (bloque lógico configurable).



*Figura 09. RTL Viewer Laboratorio 2*



*Figura 10. RTL Viewer Multiplexer*

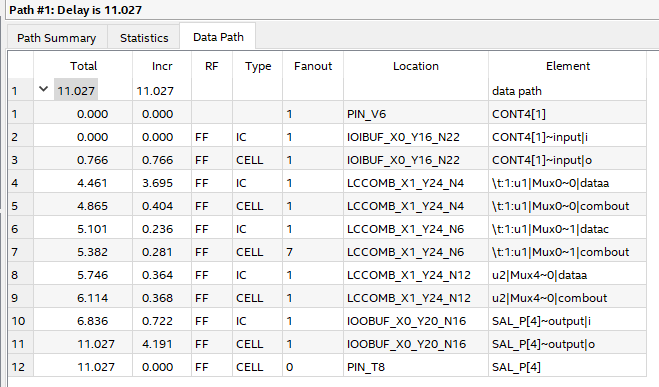


*Figura 11. RTL Decodificador BCD a 7 segmentos*

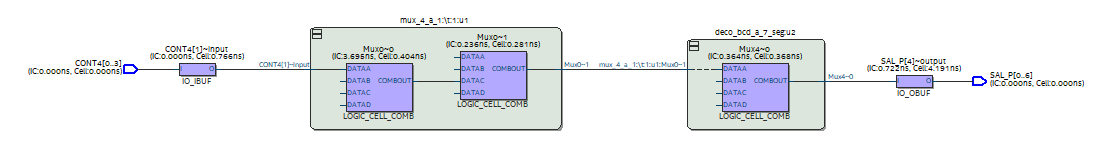
## Camino Critico

A continuación, en tres Figuras (12, 13 y 14) se muestra el resultado del procedimiento para tener y analizar el camino critico combinacional. Es decir, aquel camino que más tardara un cambio en la entrada, en verse reflejado en salida.

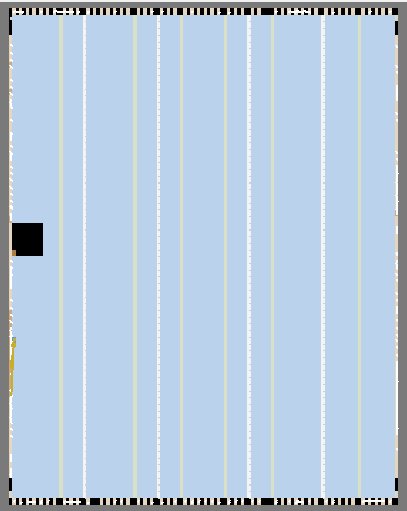
Este análisis se realizará solo para el camino que requiera más tiempo. En la Figura 12 muestra información del Data Path. La Figura 13 muestra Technology View, mostrando solamente los elementos involucrados en el camino crítico. Por ultimo en la figura 14 se ve la implementación en el Chip Planer como quedaría implementado en la placa.



*Figura 12. Data path*



*Figura 13. Locate Path Technology View*



*Figura 14. Locate Path Chip Planner*

## Problemas y Soluciones

En este código implementado presento la siguiente lista de errores a la hora de ser compilados. Los mismos se presentaron en el código del multiplexor. La solución requería que se corrigiera los errores de sintaxis que se describen a continuación. El resto de código no presento ningún problema.

Error (10500): VHDL syntax error at mux\_4\_a\_1.vhd(15) near text "architecture"; expecting ";"

Error (10500): VHDL syntax error at mux\_4\_a\_1.vhd(19) near text "when"; expecting "(", or "'", or "."

Error: Quartus Prime Analysis & Synthesis was unsuccessful. 2 errors, 1 warning

Error: Peak virtual memory: 4787 megabytes

Error: Processing ended: Wed Apr 21 02:23:24 2021

Error: Elapsed time: 00:00:44

Error: Total CPU time (on all processors): 00:01:11

Error (293001): Quartus Prime Full Compilation was unsuccessful. 4 errors, 1 warning

# Carta de Gantt

A continuación, se describe la carta de Gantt utilizada en este laboratorio. Posee cuatro partes independientes entre sí, correspondiente a cada parte de este laboratorio. A su vez cada sección está compuesta por cuatro partes también, Interpretación, Diseño, Corrección y Simulación, las cuales deben respetar cierto orden, pero no implica que algunas se realizan al simultaneo. En el margen superior de la tabla están las horas dedicadas al Laboratorio y en el margen derecho las tareas. Con amarillo destacado el tiempo dedicado a cada tarea.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| A-Interpretar |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| A-Diseñar |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| A-Corregir |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| A-Simular |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| B-Interpretar |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| B-Diseñar |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| B-Corregir |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| B-Simular |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| C-Interpretar |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| C-Diseñar |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| C-Corregir |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| C-Simular |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| D-Interpretar |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| D-Diseñar |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| D-Corregir |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| D-Simular |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

*Tabla 04. Carta de Gantt*

# Conclusiones

* Existen más de una manera de encarar las especificaciones de un circuito con VHDL
* Es importante entender la funcionalidad del sistema requerido para llevar a cabo un diseño correcto
* La Herramienta de Quartus brinda muchas herramientas para la optimización de nuestro circuito
* Es importante conocer la distribución de pines en la placa a la hora de realizar la implementación de nuestro circuito
* Es más efectivo usar códigos genéricos a la hora de definir un componente que luego puede ser utilizado en otras implementaciones.

# Apéndices

## Código Decodificador 3 a 8

---Laboratorio 2 SDA

---Villafane Mario

--- Parte A DECO 3 a 8

-----------------------------------

library ieee;

use ieee.std\_logic\_1164.all;

-----------------------------------

entity deco\_3\_a\_8 is

port(

ENT: in std\_logic\_vector(2 downto 0); ---Entradas

HAB: in std\_logic; ---Habilitacion

SAL: out std\_logic\_vector(7 downto 0)); ---Salida

end deco\_3\_a\_8;

-----------------------------------

architecture DEC of deco\_3\_a\_8 is

signal sal\_int: std\_logic\_vector(7 downto 0); ---Señal de salida interna

begin

with ENT select ---tabla de verdad

sal\_int <= "00000001" when "000",

"00000010" when "001",

"00000100" when "010",

"00001000" when "011",

"00010000" when "100",

"00100000" when "101",

"01000000" when "110",

"10000000" when "111",

"00000000" when others;

SAL<= sal\_int when (HAB= '0') else "00000000";

end DEC;

## Código Decodificador BCD a 7 segmentos

---Laboratorio 2 SDA

---Villafane Mario

--- Parte B DECO BCD a 7seg 0 a 9

------------------------------------------

library ieee;

use ieee.std\_logic\_1164.all;

------------------------------------------

entity deco\_bcd\_a\_7\_seg is

port(

ENT: in std\_logic\_vector(3 downto 0); ---Entradas BCD

SAL: out std\_logic\_vector(0 to 6)); ---Salidas ABCDEFG

end deco\_bcd\_a\_7\_seg;

------------------------------------------

architecture DEC of deco\_bcd\_a\_7\_seg is

begin

with ENT select

SAL <= "0000001" when "0000",

"1001111" when "0001",

"0010010" when "0010",

"0000110" when "0011",

"1001100" when "0100",

"0100100" when "0101",

"0100000" when "0110",

"0001111" when "0111",

"0000000" when "1000",

"0000100" when "1001",

"1111111" when others;

end DEC;

## Código Decodificador HEXA a 7 segmentos

## ---Laboratorio 2 SDA

## ---Villafane Mario

## --- Parte C DECO BCD a 7seg 0 a F

## ------------------------------------------

## library ieee;

## use ieee.std\_logic\_1164.all;

## ------------------------------------------

## entity deco\_hexa\_a\_7\_seg is

## port(

## ENT: in std\_logic\_vector(3 downto 0); ---Entradas BCD

## SAL: out std\_logic\_vector(0 to 6)); ---Salidas ABCDEFG

## end deco\_hexa\_a\_7\_seg;

## ------------------------------------------

## architecture DEC of deco\_hexa\_a\_7\_seg is

## begin

## with ENT select

## SAL <= "0000001" when "0000",

## "1001111" when "0001",

## "0010010" when "0010",

## "0000110" when "0011",

## "1001100" when "0100",

## "0100100" when "0101",

## "0100000" when "0110",

## "0001111" when "0111",

## "0000000" when "1000",

## "0000100" when "1001",

## "0001000" when "1010",

## "1100000" when "1011",

## "0110001" when "1100",

## "1000010" when "1101",

## "0110000" when "1110",

## "0111000" when "1111",

## "1111111" when others;

## 

## end DEC;

## Código Multiplexor 4 a 1

## ---Laboratorio 2 SDA

## ---Villafane Mario

## --- Parte D MUX 4 entradas a 1 salida

## ------------------------------------------

## library ieee;

## use ieee.std\_logic\_1164.all;

## ------------------------------------------

## entity mux\_4\_a\_1 is

## port(

## ENT: in std\_logic\_vector(3 downto 0);

## SAL: out std\_logic;

## SEL: in std\_logic\_vector(1 downto 0));

## end mux\_4\_a\_1;

## -------------------------------------------

## architecture MUX of mux\_4\_a\_1 is

## begin

## with SEL select

## SAL<= ENT(0) when "00",

## ENT(1) when "01",

## ENT(2) when "10",

## ENT(3) when "11",

## '-' when others;

## end MUX;

## Código Laboratorio 2

## library ieee;

## use ieee.std\_logic\_1164.all;

## entity Lab\_2 is

## port(

## CONT1,CONT2,CONT3,CONT4: in std\_logic\_vector (3 downto 0);

## SEL\_P: in std\_logic\_vector(1 downto 0);

## SAL\_P: out std\_logic\_vector(0 to 6));

## end Lab\_2;

## architecture PRINCIPAL of Lab\_2 is

## component mux\_4\_a\_1 is

## port(

## ENT: in std\_logic\_vector(3 downto 0);

## SAL: out std\_logic;

## SEL: in std\_logic\_vector(1 downto 0));

## end component;

## component deco\_bcd\_a\_7\_seg is

## port(

## ENT: in std\_logic\_vector(3 downto 0); ---Entradas BCD

## SAL: out std\_logic\_vector(0 to 6)); ---Salidas ABCDEFG

## end component;

## type entradas\_4X4 is array (0 to 3) of std\_logic\_vector(3 downto 0);

## signal ENTRADA: entradas\_4x4;

## signal SEN\_INT: std\_logic\_vector(3 downto 0);

## begin

## t: for i in CONT1'range generate

## ENTRADA(i) <= CONT1(i) & CONT2(i) & CONT3(i) & CONT4(i) ;

## u1: mux\_4\_a\_1 port map(

## ENT=> ENTRADA(i),

## SAL=> SEN\_INT(i),

## SEL=> SEL\_P);

## end generate t;

## u2: deco\_bcd\_a\_7\_seg port map(

## ENT=> SEN\_INT,

## SAL=> SAL\_P);

## end PRINCIPAL;