



Informe 2:

Abyss, disección de circuitos

Arquitectura y Organización de Computadores

Profesor: Mauricio Solar

Estudiante: Diego Rosales León – Rol: 201810531-7

07 de Junio de 2021

1. Resumen

En este informe se confeccionó los circuitos necesarios para una consola de videojuegos, con el objetivo de que al ingresar una entrada binaria entregue el valor relacionado en base Hexadecimal. (Un circuito funcional)

Para lo anterior se analizaron distintos casos que cumplieran con ciertos requisitos (1 bit (T), 4 Bits (A y B)). Una vez hecho esto se distinguió entre circuitos y cada uno de sus requerimientos para el funcionamiento completo de *Abyss*.

En específico se trabajó con un Selector, Funciones varias como pasar de binario a hexadecimal y finalmente completar el circuito pedido en la tarea.

Por último, el circuito final cumple con su misión y los resultados son tal cual los esperados.

2. Introducción

El presente trabajo trata sobre una segunda tarea entregada por la empresa *OculusTM*, en donde su objetivo dice relación con la creación de un circuito para que la consola *Abyss* funcione de manera correcta, a través de la utilización del software Logisim¹. Para que poder trabajar con este último, se entregaron tres inputs² (pins), T , A y B :

Tabla 1: Inputs para la consola Abyss

Input	Bits ³
T	1 Bit
A	4 Bits
B	4 Bits

En términos generales, lo que se buscó implementar fueron circuitos combinacionales que permitieran realizar operaciones binarias⁴ para tener una consola de videojuegos funcional. Algunas fórmulas utilizadas para el proceso fueron:

$$\bullet \quad f(x, y, z) = (x \cdot y \cdot z + \bar{x} \cdot \bar{y} + \bar{y} \cdot \bar{z} + \bar{x} \cdot \bar{z}) \oplus (\overline{x + y + z}) \quad (1)$$

$$\bullet \quad g(w, x, y, z) = \begin{cases} f(x, y, z), & \text{si } w=0 \\ \overline{f(x, y, z)}, & \text{si } w=1 \end{cases} \quad (2)$$

¹ Herramienta de libre distribución de diseño y simulación de circuitos lógicos digitales

² Input: entrada del programa.

³ Bit es la abreviación de Binary Digit, o Dígito Binario

⁴ El código binario es el sistema numérico usado para la creación de procesadores de instrucciones de computadora, usando el sistema binario, que es un sistema numérico de dos dígitos o bits, el 0 y el 1

3. Desarrollo

Para entender mejor el trabajo realizado, se dividirá la información en varios apartados, debido a la creación de distintos tipos de circuitos **necesarios** para lograr el objetivo propuesto.

A. Condición1 (Selector)

Lo primero que se trabajó, fue la creación de un circuito Selector, el cual **decide** cual función (circuitos) debe ser usada como output⁵, dependiendo de un Input T , donde si $T=0$, el output será la *Suma Binaria* ($F1$), y si $T=1$, el output será $f(x)$ o $g(x)$ ($F2$), respectivamente.

Para ello, se ingresaron los siguientes valores en *Logisim*, obteniendo:

Tabla 2: Tabla con los valores del Selector

T	F1 (Suma Binaria)	F2 ($f(x)$ o $g(x)$)	Output O
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Luego de ingresar los valores, el software generó un *mapa de Karnaugh*, que se muestra a continuación:

		F1, F2			
		00	01	11	10
T	0	0	0	1	1
	1	0	1	1	0

Imagen 1: Mapa de Karnaugh para el Selector

Finalmente, con todo lo anterior se creó el siguiente circuito:

⁵ Output: salida de un cálculo o proceso.

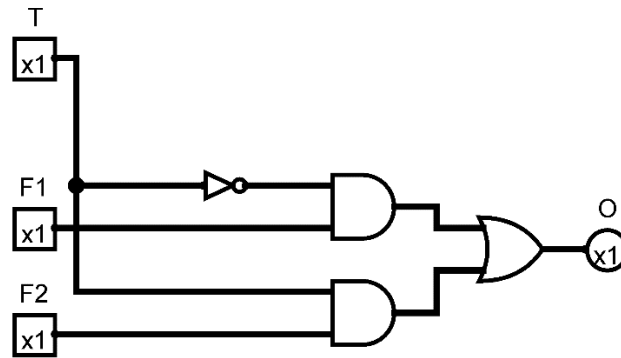


Imagen 2: Circuito de la condición1 (Selector)

B. Funciones.

Luego de armar el circuito Selector, se construyó el caso más extenso de este informe, que corresponde a cuando $T=I$, se consideran dos posibles opciones de salida, las ecuaciones (1) o (2) definidas anteriormente. Para el Output O, se tiene el siguiente orden, $[O_0, O_1, O_2, O_3]$, donde:

- $O_0 = f(x_A, y_A, z_A)$
- $O_1 = g(w_A, x_A, y_A, z_A)$
- $O_2 = f(x_B, y_B, z_B)$
- $O_3 = g(w_B, x_B, y_B, z_B)$

Posteriormente, se desarrolló el circuito $f(x)$, que requirió ingresar la fórmula directamente en Logisim, resultando:

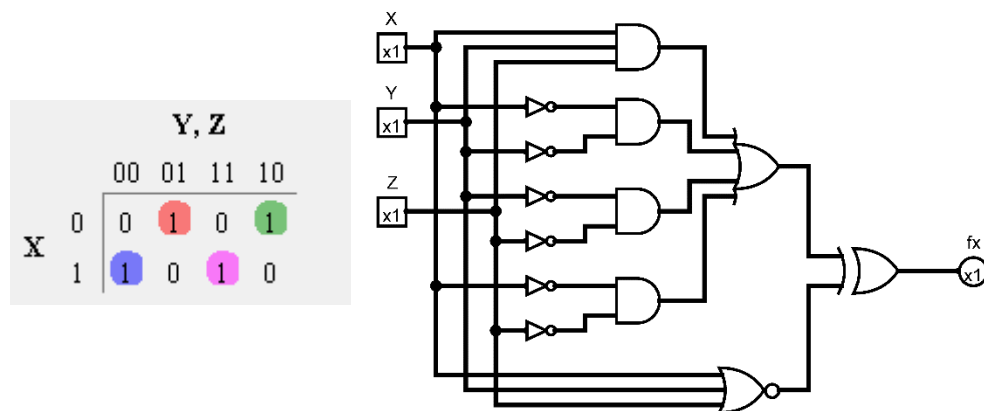


Imagen 3: Tabla de Karnaugh y Circuito correspondiente con Output $f(x)$

Luego se observó que para el caso $g(x)$, la única diferencia existente fue que el valor de w define cual caso debe usarse (2), si $w=0$ se usa $f(x)$ de manera normal, pero si $w=1$, se usa un $f(x)$ negado:

Tabla 3: Casos de $g(x)$

w	f(x)	g(x)
0	0	0
0	1	1
1	0	1
1	1	0

Por lo tanto, al circuito $f(x)$ se le agregó una puerta **XOR**⁶, arreglando el input para que fuera más ordenado y se obtuvo el valor $g(x)$:

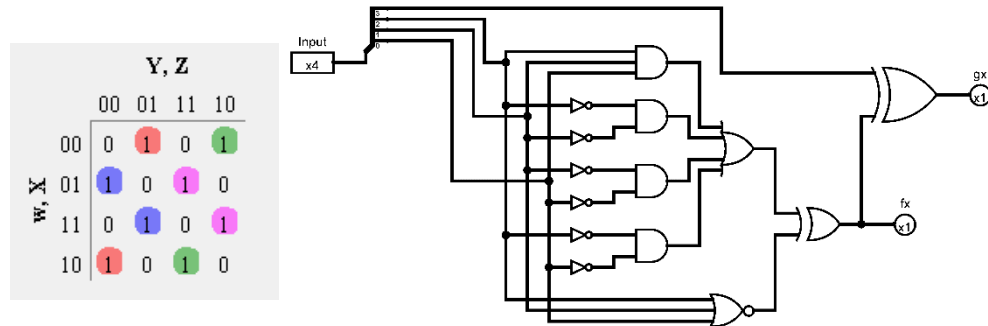


Imagen 4: Mapa de Karnaugh y Circuito correspondiente a $g(x)$ y $f(x)$

C. Suma Binaria.

Por otro lado, se trabajó con otro caso cuando $T=0$, donde a diferencia de lo planteado antes, solamente se hace una suma binaria entre A y B , sin tomar en cuenta el carry. Lo anterior se enmarcó en el video de Brown (2017), donde se explica gráficamente como trabajar los siguientes circuitos en Logisim:

a. Ayuda para la suma:

Es un circuito que recibe los bits del input y un carry de entrada (en caso de que exista) y se entrega un carry de salida con la suma correspondiente entre bits.

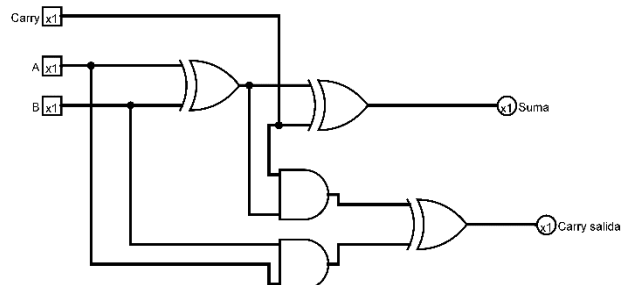


Imagen 5: Circuito de ayuda para la suma de números binarios

⁶ Es una puerta lógica digital que implementa el O exclusivo.

b. Suma con bits separados:

El circuito anterior es utilizado para cada uno de los bits recibidos provenientes de A y B , obteniendo como resultado:

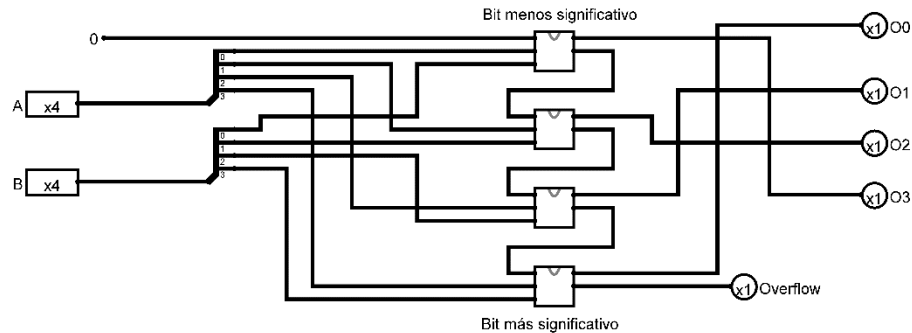


Imagen 6: Circuito para la suma de números binarios

Finalmente, se obtiene la suma de todos los bits, en su respectiva posición, con una salida de Overflow.

D. Hexadecimal

Otro caso ocurre cuando el output entregado por el Selector es el input para esta nueva función hexadecimal, la cual recibe 4 bits en binario y los transforma a un valor hexadecimal utilizando LEDs, como se muestra en la *Tabla 5(Anexo)* donde cada fila o columna de LEDs se prenderá dependiendo del valor correspondiente. Todo lo anterior fue posible gracias a la idea proporcionada por un video sin autor (*Desconocido, 2018*), que sirvió de base para entender él que hacer. Finalmente, se insertó el contenido de la *Tabla 5* en una Tabla de Logisim, con un total de 7 outputs, obteniendo el circuito que se puede ver en la parte final del anexo.

E. Circuito Principal (Main).

Finalmente, con todos los casos expuestos hasta ahora se construye el circuito principal, donde se reciben las entradas T , A y B y como output se entrega lo pedido por la tarea a través de los LEDs. A esto se le suma una salida para el Overflow:

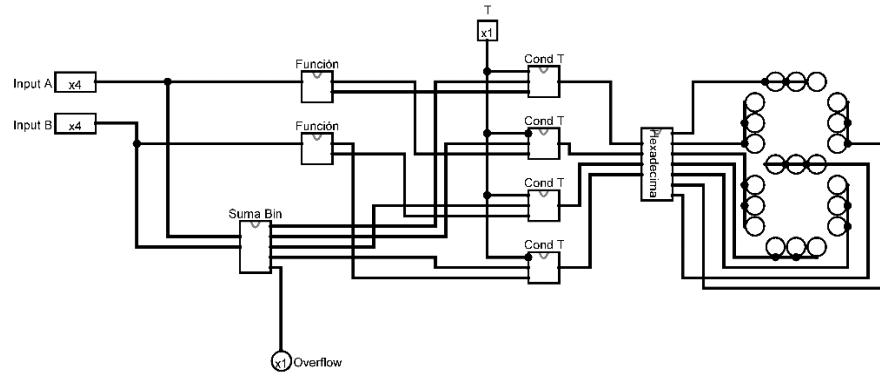










Imagen 7: Circuito Main Completo (Estará en el anexo con mayor tamaño)

4. Resultados

Para los resultados, se utilizaron los ejemplos entregados en la Tabla de la Tarea añadiendo otra columna con los LEDs formados. Para lo anterior, se registraron los valores de T , A y B , a través de los inputs entregados y se revisó si el resultado era correcto, obteniendo:

Tabla 4: Inputs, Output Binario, Output Hex y LEDs

T A B	O Binario	O Hex	LEDs
0 1101 0001	1110	E	
0 0011 0110	1001	9	
0 1111 0001	0000	0	
0 1001 0001	1010	A	
1 1101 0001	0111	7	
1 0011 1000	0001	1	
1 1111 1001	1010	A	
1 1001 0111	1011	B	

Aquí se puede apreciar que los resultados de los LEDs con los valores Hex, son equivalentes y que los valores de T son distintos en algunos casos.

5. Análisis

Primero, se debe destacar que los resultados fueron correctos, satisfactorios e idénticos a lo que se solicitaba en la tarea. La única excepción fue el caso de las letras B y b debido a que la letra B en los LEDs es similar a lo entregado por el valor de 8 (por eso se usa b). Por otro lado, se observa que en la *Tabla 4*, hay casos donde el valor de A y B son idénticos, pero el valor O binario resulta distinto, y al indagar en el porqué de esta situación, se entiende que ocurre principalmente debido al valor de T , el cual afecta directamente a la función que se debe utilizar.

Segundo, Logisim en sí es un software *friendly* para aprender a armar circuitos, sin embargo, se debe tener conocimiento previo de todas las herramientas que entrega, si no, la dificultad aumenta considerablemente, impidiendo la continuación del trabajo, como por ejemplo al relacionar distintos circuitos entre sí. A su vez, deben existir otras formas más simples de resolver esto, quizás con otro tipo de herramientas o funciones que entregue el software (sin contar Hex Display o Sumadores, que están **prohibidos**)

Por último, una tercera dificultad observada tuvo relación con el uso de la función $FI(f(x) \text{ o } g(x))$, debido a que al trabajar con inputs de 4 bits y utilizar los separadores, el orden de cada bit importa, es decir, si uno de ellos está mal posicionado podría provocar que todo el programa estuviese incorrecto, ya sea por estar superpuestos o en una ubicación errónea, incluso, cualquier output generaba un resultado, pero no el esperado.

6. Conclusión

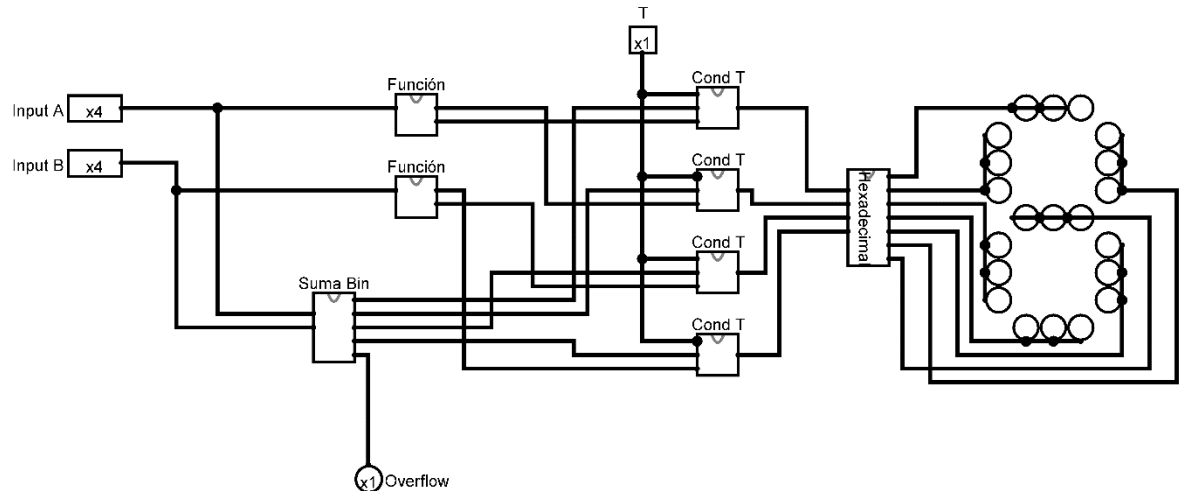
Finalizado el análisis, se puede concluir que Logisim es una herramienta fuerte para el desarrollo de circuitos, debido a que facilita varios procesos, por ejemplo, la minimización de funciones, donde desde algo complejo se puede llegar a algo más simple a través de los Mapas de Karnaugh, los cuales impactan directamente en el desarrollo del circuito.

Como se mencionó anteriormente, hay casos donde los LEDs pueden ser inexactos o poco parecidos a las letras correspondientes (B con b), quizás se debería utilizar una mayor cantidad de estos para obtener un mejor resultado, aunque esto implique mayor complejidad, incluso si los resultados obtenidos, en base a la *Tabla 4*, fueron correctos.

Finalmente, es necesario tener en mente que el paso del tiempo aumenta, mejora y evoluciona la tecnología que conocemos y todo lo que se desarrolla en base a la misma está en constante cambio, a excepción de los circuitos, del Logisim y los Mapas de Karnaugh que se han mantenido por mucho tiempo vigentes. Ahora bien, y a modo de cierre, cabe preguntarse qué pasará en el futuro lejano con ellos si es que seguimos desarrollándonos con la misma velocidad y diversidad de recursos.

7. Bibliografía/Anexo

- Brown, Barry. [Barry Brown]. (2017, Febrero, 9). *Logisim 4-Bit Ripple-Carry Adder* [Archivo de vídeo]. Recuperado de: https://www.youtube.com/watch?v=OkzoRnjOuNw&ab_channel=BarryBrown
- Autor Desconocido [Tutorials & Tricks]. (2018, Diciembre, 30). *Binary to decimal conversion (7 segment display) using logisim software*. [Archivo de Vídeo]. Recuperado de: https://www.youtube.com/watch?v=albJ0iuz1wg&ab_channel=Tutorials%26Tricks
- Cburch. (s.f). *7-Segment Display*. Obtenido de: <http://www.cburch.com/logisim/docs/2.3.0/libs/io/7seg.html>
- Usuario juramgon. (8 de Septiembre). Sistema Binario. *Blog: Tecnología e Informática*. Sistema Binario. Obtenido de: <https://www3.gobiernodecanarias.org/medusa/ecoblog/juramgon/2017/09/08/sistema-binario/>
- Circuito Main Completo



- Tabla de representación Hexadecimal:

Binario	HEXADEC	1	2	3	4	5	6	7
0000	0	1	1	1	1	1	1	0
0001	1	0	0	0	0	1	1	0
0010	2	1	0	1	1	0	1	1
0011	3	1	0	0	1	1	1	1
0100	4	0	1	0	0	1	1	1
0101	5	1	1	0	1	1	0	1
0110	6	1	1	1	1	1	0	1
0111	7	1	0	0	0	1	1	0
1000	8	1	1	1	1	1	1	1
1001	9	1	1	0	0	1	1	1
1010	A	1	1	1	0	1	1	1
1011	B	0	1	1	1	1	0	1
1100	C	1	1	1	1	0	0	0
1101	D	0	0	1	1	1	1	1
1110	E	1	1	1	1	0	0	1
1111	F	1	1	1	0	0	0	1

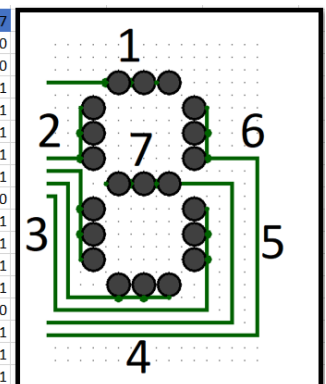


Tabla 5: Representación de los valores hexadecimales

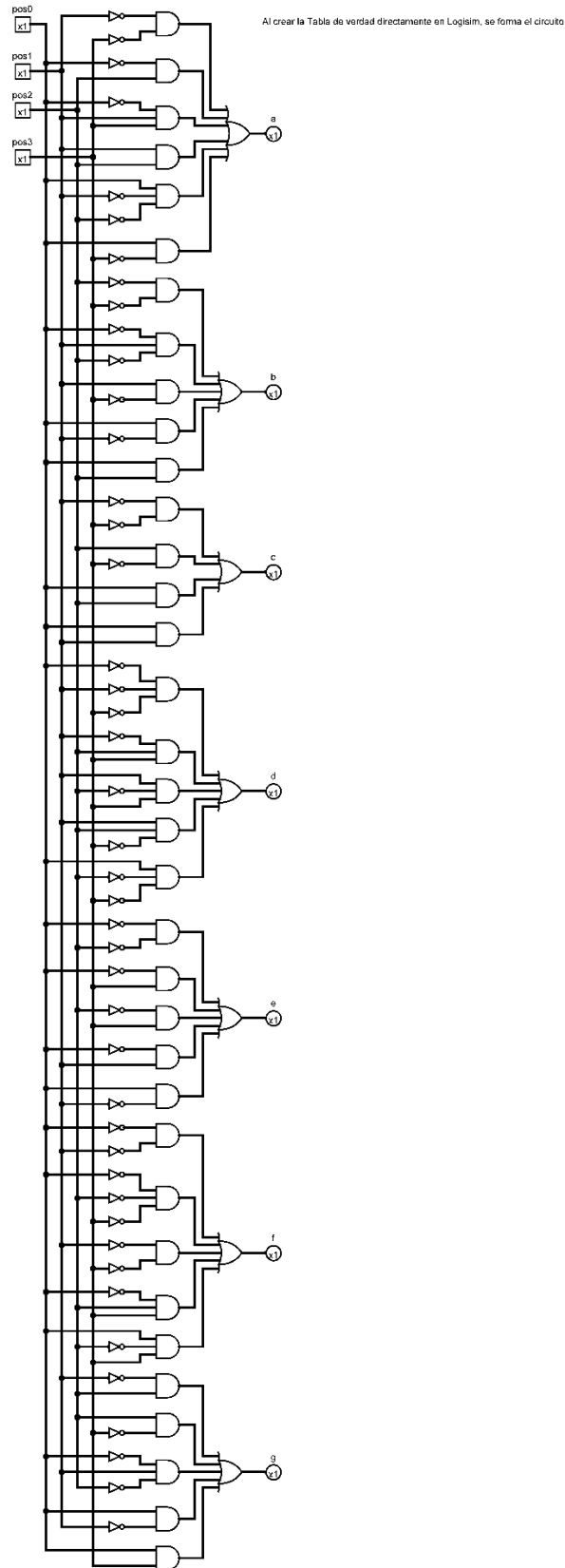


Imagen 8: Circuito Hexadecimal