



Informe 3:

The Neanderthal Memory

Arquitectura y Organización de Computadores

Profesor: Mauricio Solar

Estudiante: Diego Rosales León – Rol: 201810531-7

25 de Junio de 2021

1. Resumen

En este informe se confeccionó los circuitos necesarios para una consola de videojuegos, con el objetivo de crear circuitos secuenciales con memoria primitiva, es decir, que un valor que quede almacenado en la memoria interna del circuito pueda ser utilizado.

Para lo anterior se analizaron distintos tipos de circuitos con sus respectivos requisitos (*M de 2 Bits*, *X de 8 Bits* y un *Reloj de 1 Bit*). Con todo lo anterior, se distinguió entre circuitos, en específico se trabajó con Flip Flops, y un circuito que es equivalente a lo que es una *ALU* (definida posteriormente) y finalmente completar con lo pedido en la tarea.

Finalmente, el circuito Main cumple con su misión y los resultados son los esperados.

2. Introducción

El presente trabajo trata de una tercera tarea entregada por la misma empresa, *Oculus™*, donde su objetivo principal es la creación de un circuito con memoria primitiva, para que la consola de videojuegos *Abyss* funcione sin ningún error, todo esto a través de la utilización del software Logisim. Para trabajar con este último, y resolver la tarea se entregan tres inputs¹ (pins), *M, X, CLK*:

Tabla 1: Inputs para la consola Abyss

Input	Bits
M	2 Bits
X	8 Bits
CLK	1 Bit (Reloj ²)

En términos generales, lo que se buscó implementar fueron circuitos secuenciales que permiten la creación de memoria primitiva para tener una consola de videojuegos funcional. Algunas fórmulas utilizadas para el proceso fueron:

$$\bullet \quad A + B \quad (1)$$

$$\bullet \quad A - B = A + \bar{B} + 1 \quad (2)$$

¹ Input: entrada del programa.

² El reloj conmuta su valor de salida de forma periódica siempre y cuando se active la conmutación a través del menú de Logisim

3. Desarrollo

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

A. SR-Latch³

Lo primero que se desarrolló fue el SR-Latch, donde las siglas *SR* equivalen a los estados *Set* y *Reset* respectivamente. Se construyó a través de compuertas de tipo *NOR*, con la excepción de que se eliminó la salida \bar{Q} (no se utiliza):

S	R	Q	\bar{Q}
0	0	latch	latch
0	1	0	1
1	0	1	0
1	1	0	0

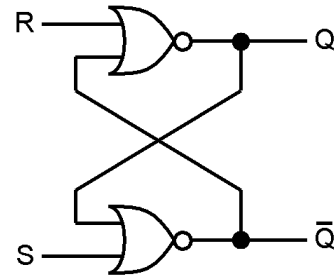


Imagen 1: Tabla de verdad y Circuito SR-Latch

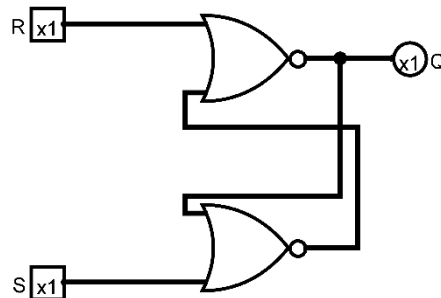


Imagen 2: Circuito SR-Latch utilizado en la Tarea

B. Auxiliar

Con el circuito anterior se crea el circuito Auxiliar utilizado para ordenar de manera visual y para crear la relación Master-Slave⁴ entre las funciones siguientes (FF1 y FF2):

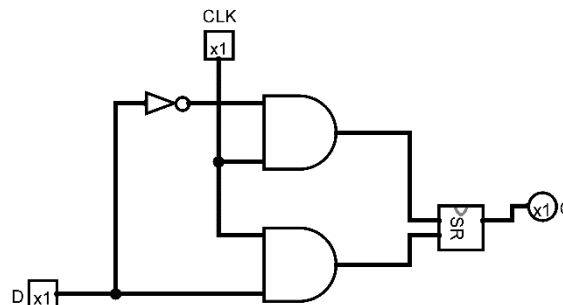


Imagen 3: Circuito Auxiliar para mantener el orden

³ Un Latch (Late Memory) es un circuito electrónico biestable asíncrono usado para almacenar información en sistemas lógicos digitales

⁴ El Master se comporta como un Latch de entrada con SR. Cuando el reloj se hace 1, el Slave se deshabilita y mantiene su estado previo, mientras que el Master está habilitado con su respectiva entrada. Cuando es 0 ocurre lo mismo, pero al revés.

C. FF1 (Master)

Con todo lo anterior se pudo crear este circuito para trabajar un Flip Flop⁵ SR de tipo *Maestro (Master)* con :

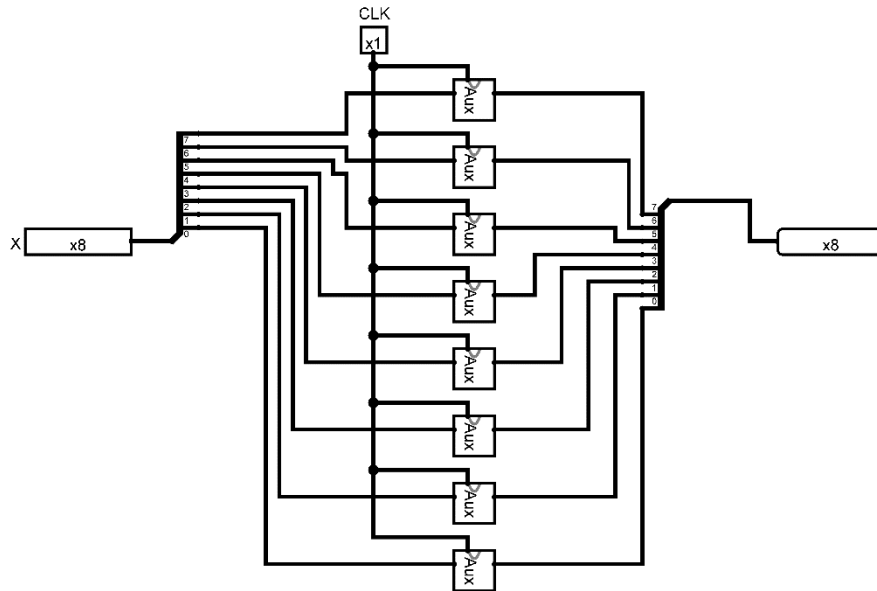


Imagen 4: Flip Flop Maestro (Estará en el Anexo)

D. FF2 (Slave)

Es idéntico al circuito anterior, con la diferencia de que el input del Reloj está negado (Se podrá apreciar en el circuito Main), siendo así un Flip Flop SR de tipo *Esclavo (Slave)*:

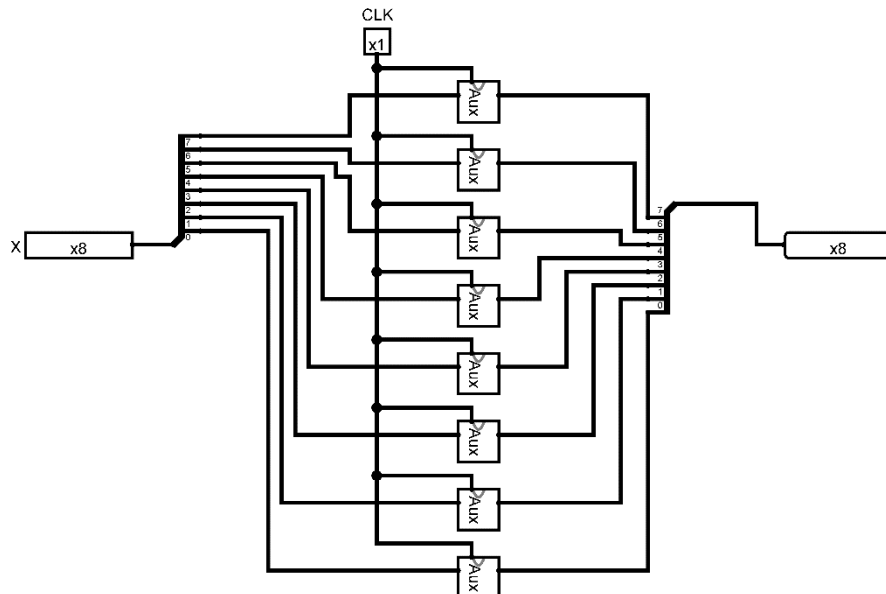


Imagen 5: Flip Flop esclavo (Estará en el Anexo)

⁵ El flip flop es el nombre común que se le da a los dispositivos de dos estados (biestables), que sirven como memoria básica para las operaciones de lógica secuencial

E. Suma Binaria.

Luego se trabajó con el circuito de la Suma binaria (1) que fue planteado en la tarea pasada con la diferencia de que fue modificado para ser una suma binaria con valores de 8 Bits:

a. AuxSum:

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.

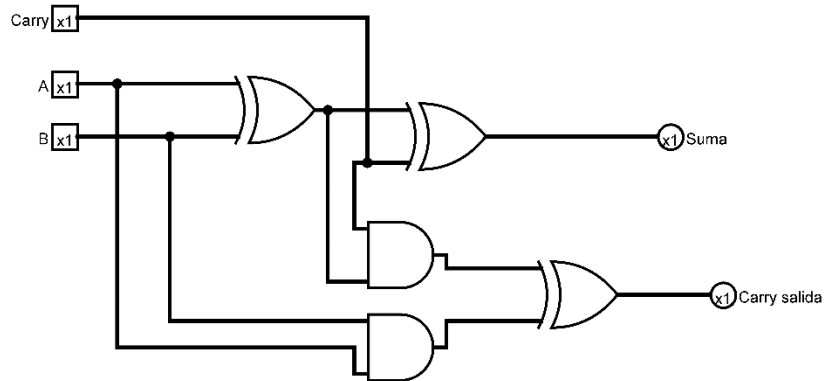


Imagen 6: Circuito de ayuda para la suma de números binarios

b. SumBinary:

El circuito anterior es utilizado para cada uno de los bits recibidos provenientes de X y $P(X)$, además de una constante con valor constante igual a 0, obteniendo como resultado:

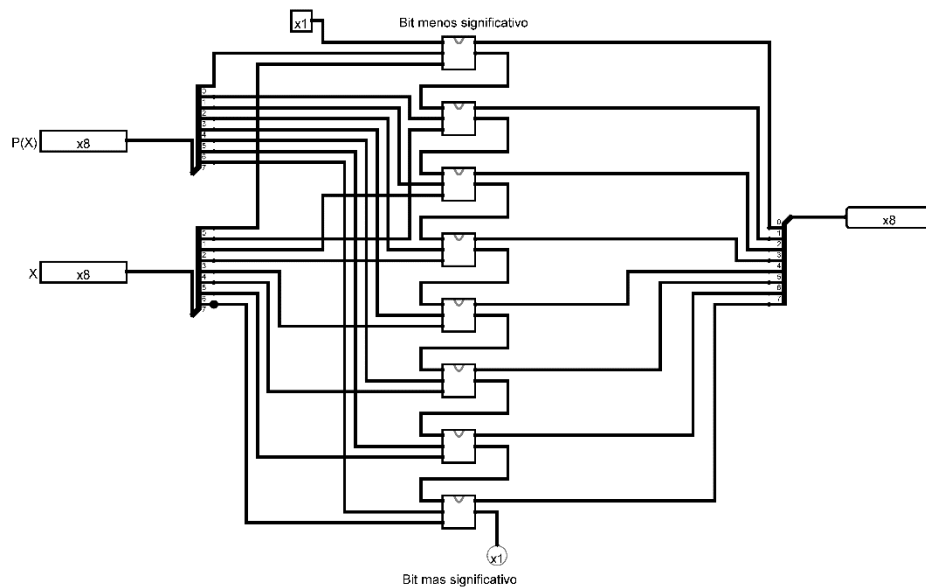


Imagen 7: Circuito para la suma de números binarios

Finalmente, se obtiene la suma de todos los bits, con una salida de tipo Ver proveniente de la carpeta Wiring de Logisim.

F. Resta

Para este caso se utiliza la ecuación (2) y el circuito anterior AuxSum, debido a que facilita la resta binaria entre dos valores. Tiene los mismos inputs, con las excepciones de que el valor constante es equivalente a 1 y el valor de $P(X)$ fue negado:

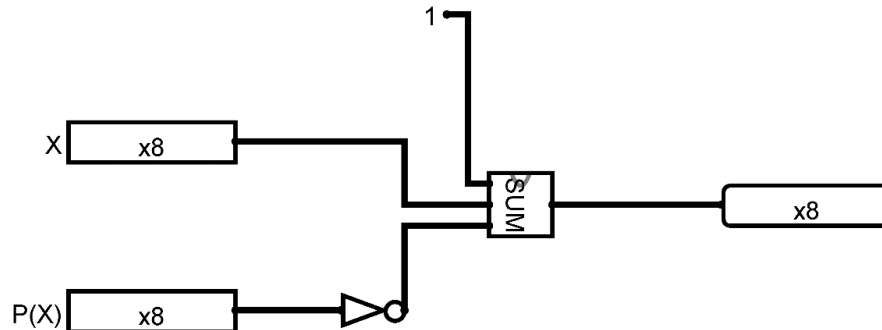


Imagen 8: Circuito para la resta binaria

G. S-Logic

Este circuito representa lo que es una **ALU**, una unidad aritmética lógica que es parte de la CPU⁶ y resuelve cálculos aritméticos. Recibe los valores de X y $P(X)$:

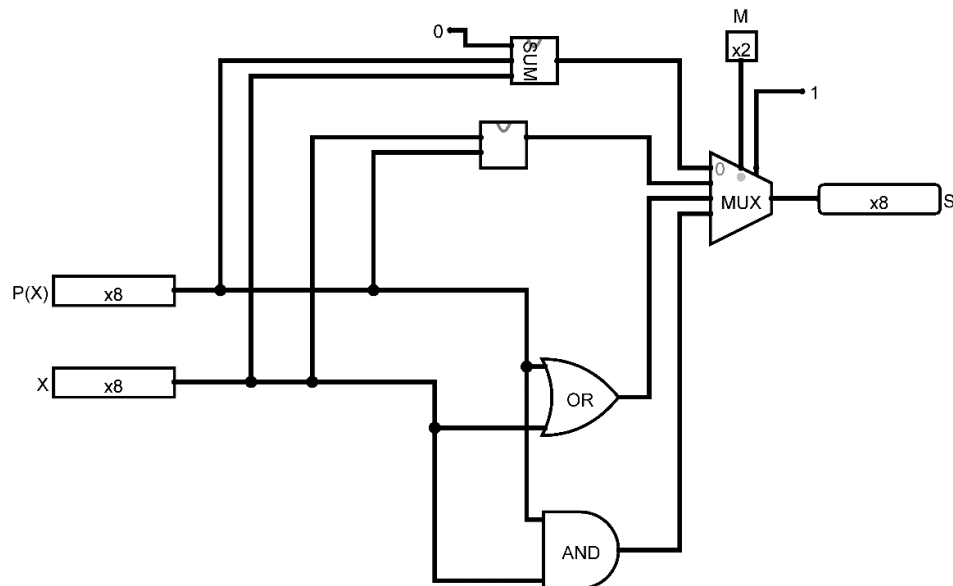


Imagen 9: Circuito S-Logic que equivale a una ALU

Utiliza los circuitos de Suma, Resta y a su vez, dos compuertas lógicas, una compuerta **OR** y **AND** unidas a un **Multiplexor (MUX)**⁷. Finalmente tiene como Output el valor de S .

⁶CPU son las siglas de Central Processing Unit, lo que traducido significa Unidad Central de Procesamiento.

⁷ Los MUX son circuitos combinacionales con varias entradas y una única salida de datos

H. Circuito Principal (Main).

Finalmente, con todos los casos expuestos hasta ahora se construye el circuito principal, con entradas M, X y CLK , la relación Master-Slave entre FF1 y FF2 respectivamente y como output lo pedido por la tarea en forma de LEDs:

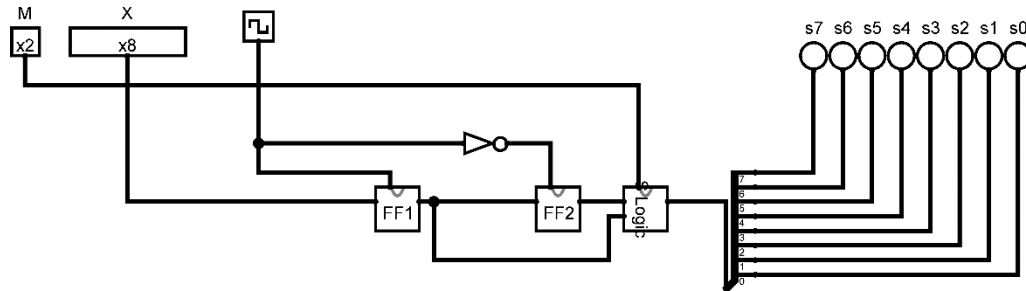
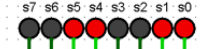
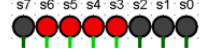
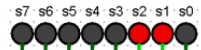
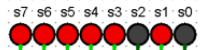
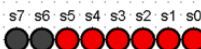
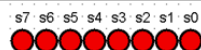
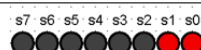



Imagen 10: Circuito Main

4. Resultados

Para los resultados se utilizó los casos de ejemplo entregados en la Tarea, agregando a la derecha otra columna con los LEDs formados. Para lo anterior, se registra primero el valor de $P(x)$ para ser guardado en la memoria de la función FF2 y luego se inserta el valor de X correspondiente:

Tabla 2: Inputs, $P(X)$ y Output en LEDs (Se adjunta en el anexo)

M	X	P(X)	S	Leds
00	00110011	00000000	00110011	
00	00001111	01101001	01111000	
01	00010000	00001010	00000110	
01	00001010	00010000	11111010	
10	00001111	00110011	00111111	
10	11111111	00000000	11111111	
11	00001111	00110011	00000011	
11	11111111	00000000	00000000	

Se puede apreciar que los resultados dados por los LEDs son equivalentes a la Salida (S).

5. Análisis

Primero, se debe resaltar que los resultados obtenidos fueron correctos y equivalentes a lo que solicitaba en la tarea, aunque cabe decir que a veces la resta binaria puede ser inexacta, debido a que cuando se indagó acerca del tema, hay casos donde el signo de la Resta podía agregar un valor más, es decir, que pasa de un valor de 8 Bits a uno de 9 Bits lo cual cambiaría el resultado pedido.

Segundo, una dificultad encontrada tuvo relación con la forma que se usó el circuito Main, debido a que al momento de terminar la tarea y trabajar con los distintos inputs, funcionaba de manera incorrecta, llevando a errores en el resultado final. Al indagar sobre el porqué de esta situación, se entiende que esto ocurre por el orden en el que se ingresan los datos, primero se debe ingresar el valor de $P(X)$, presionar el reloj para que quede guardado en la memoria del circuito, y luego ingresar el valor correspondiente a X , de esta forma funcionará de manera normal y correcta.

Finalmente, y como tercera dificultad fue el uso de la herramienta de Multiplexor para armar circuitos, debido al poco conocimiento práctico de utilizar el mismo, a su vez, el trabajo se hubiese facilitado utilizando otras herramientas de Logisim, como las encontradas en las carpetas de Aritmética y Memoria, pero están **prohibidas**.

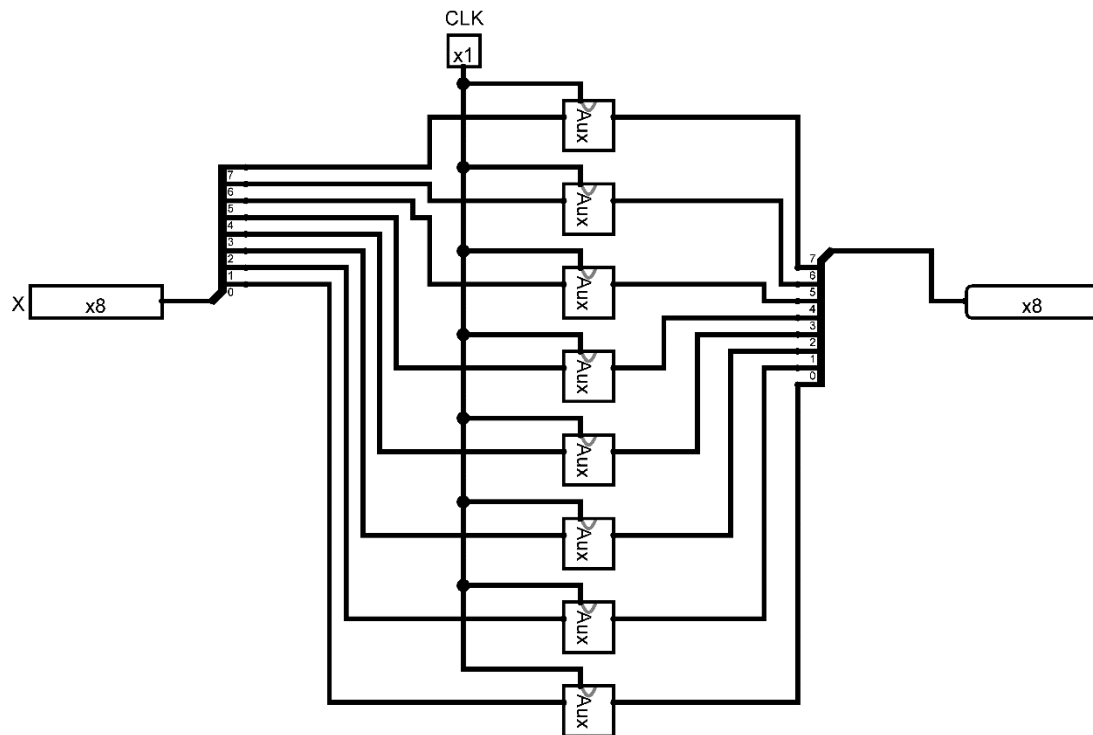
6. Conclusión

Finalizado el análisis y con todo lo anterior, se debe concluir acerca de un circuito secuencial, el cual está compuesto de circuitos combinacionales y, además, elementos de memoria de la parte combinacional, aceptando así entradas externas y entradas internas desde los elementos de la memoria, concluyéndose así, que los circuitos secuenciales son útiles para trabajar con distintos estados del circuito.

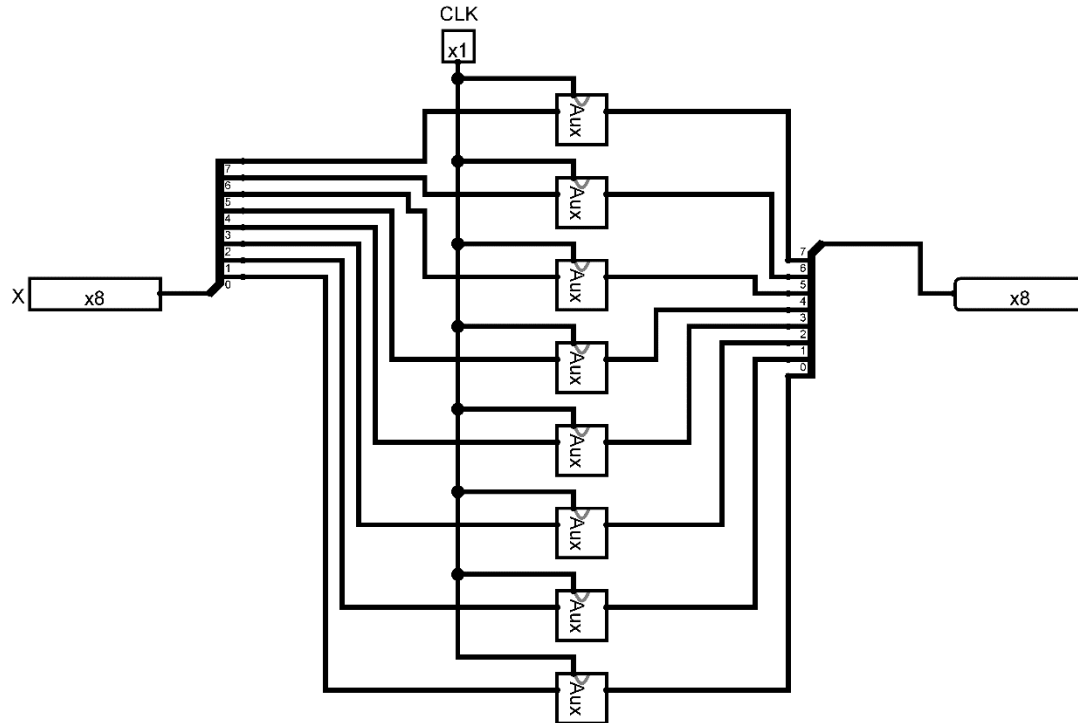
Finalmente, como se mencionó en el análisis, el caso de la resta puede ser inexacto, debido a su signo, pero una solución es utilizar un LED más que represente el signo correspondiente, si es negativo o positivo.

7. Bibliografía/Anexo

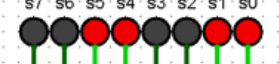

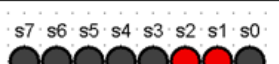


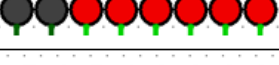

- Brown, Barry. [Barry Brown]. (2017, Febrero, 9). *Logisim 4-Bit Ripple-Carry Adder* [Archivo de vídeo]. Recuperado de: https://www.youtube.com/watch?v=OkzoRnjOuNw&ab_channel=BarryBrown
- Cburch. (s.f). *Clock*. Obtenido de: <http://www.cburch.com/logisim/docs/2.1.0-es/libs/base/clock.html>
- Fernández, Yubal. (2021,Marzo,17). *CPU: Qué es, cómo es para que sirve*. Obtenido de <https://www.xataka.com/basics/cpu-que-como-sirve>
- Ingeniería Mecafenix (2017, Abril, 24). *Flip Flop ¿Qué es y cómo funciona?* .Obtenido de: <https://www.ingmecafenix.com/electronica/flipflop/>
- Circuito Maestro



- Circuito Esclavo:



Resultados:

M	X	P(X)	S	Leds
00	00110011	00000000	00110011	
00	00001111	01101001	01111000	
01	00010000	00001010	00000110	
01	00001010	00010000	11111010	
10	00001111	00110011	00111111	
10	11111111	00000000	11111111	
11	00001111	00110011	00000011	
11	11111111	00000000	00000000	