2022

Informe de obligatorio

ARQUITECTURA DE SISTEMAS

PROFESOR/A: VALERIA EMANUELLI

UNIVERSIDAD ORT URUGUAY

Federico Gutiérrez (262684)

Santiago González (238993)

Tabla de contenidos

Semáforo

Introducción:

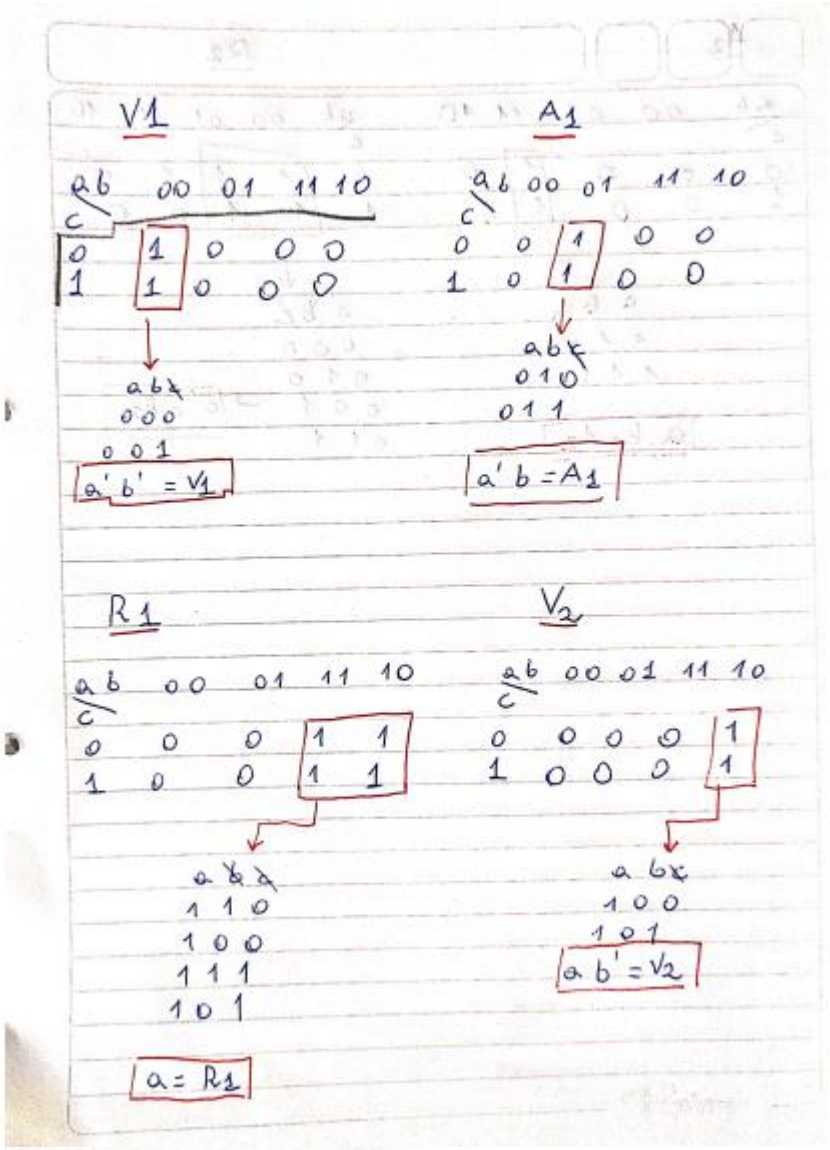
La primera tarea era realizar un circuito que controle 4 semáforos, de la intersección de 2 calles.

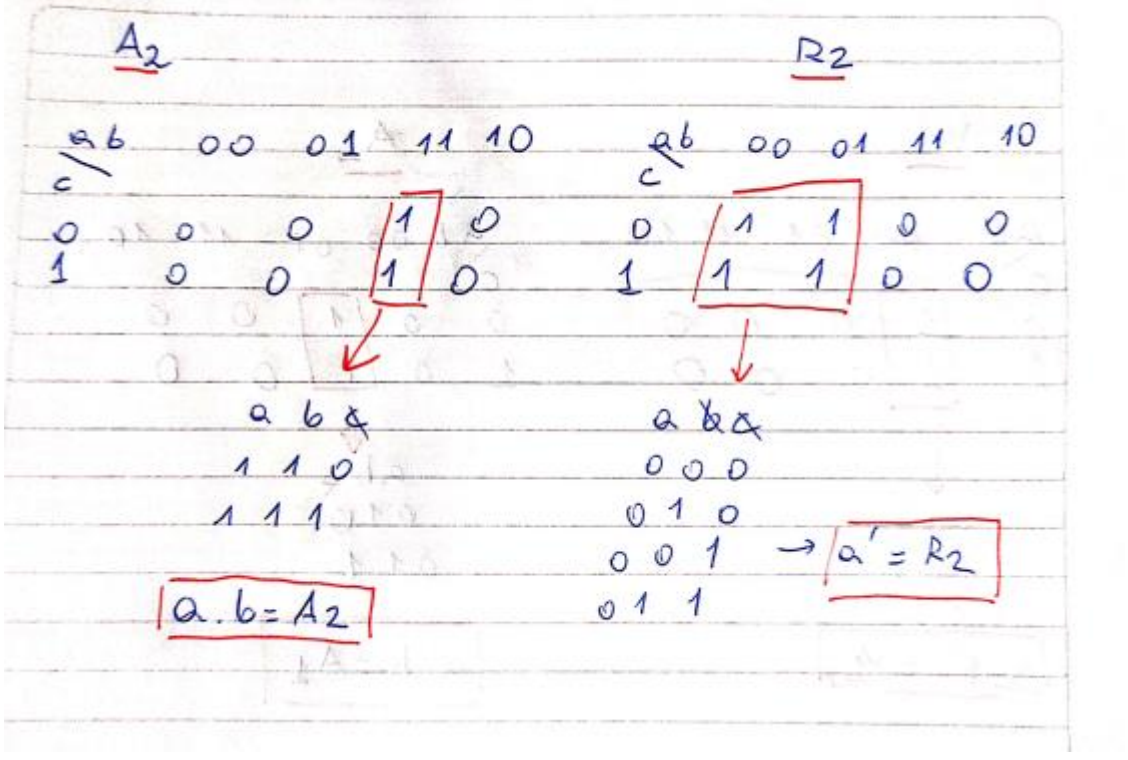
Como muestra la siguiente imagen:

Nuestra solución al problema fue con un circuito combinacional utilizando la siguiente tabla de verdad:

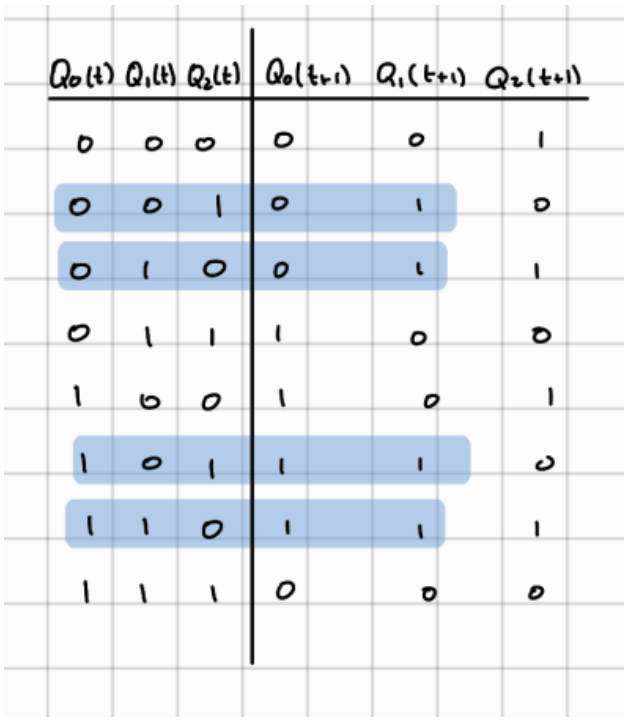
a	b	c		V1	A1	R1		V2	A2	R2
0	0	0		1	0	0		0	0	1
0	0	1		1	0	0		0	0	1
0	1	0		0	1	0		0	0	1
0	1	1		0	1	0		0	0	1
1	0	0		0	0	1		1	0	0
1	0	1		0	0	1		1	0	0
1	1	0		0	0	1		0	1	0
1	1	1		0	0	1		0	1	0

Consiguiendo la siguiente simplificación mediante mapas de Karnaugh



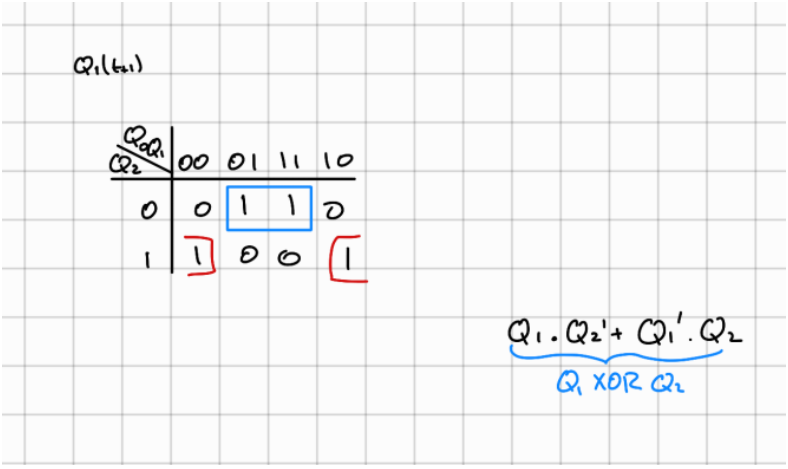


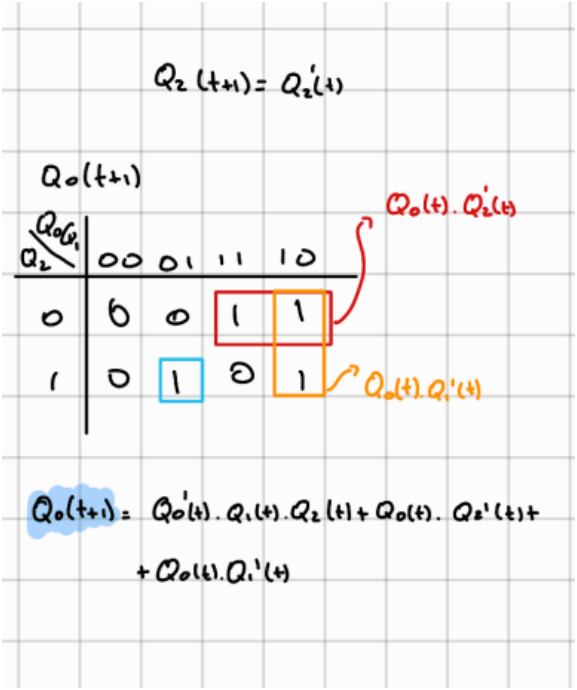
Para cambiar los estados del semáforo utilizamos un contador de 3 bits (del 0 al 7) que tiene la siguiente tabla de verdad:



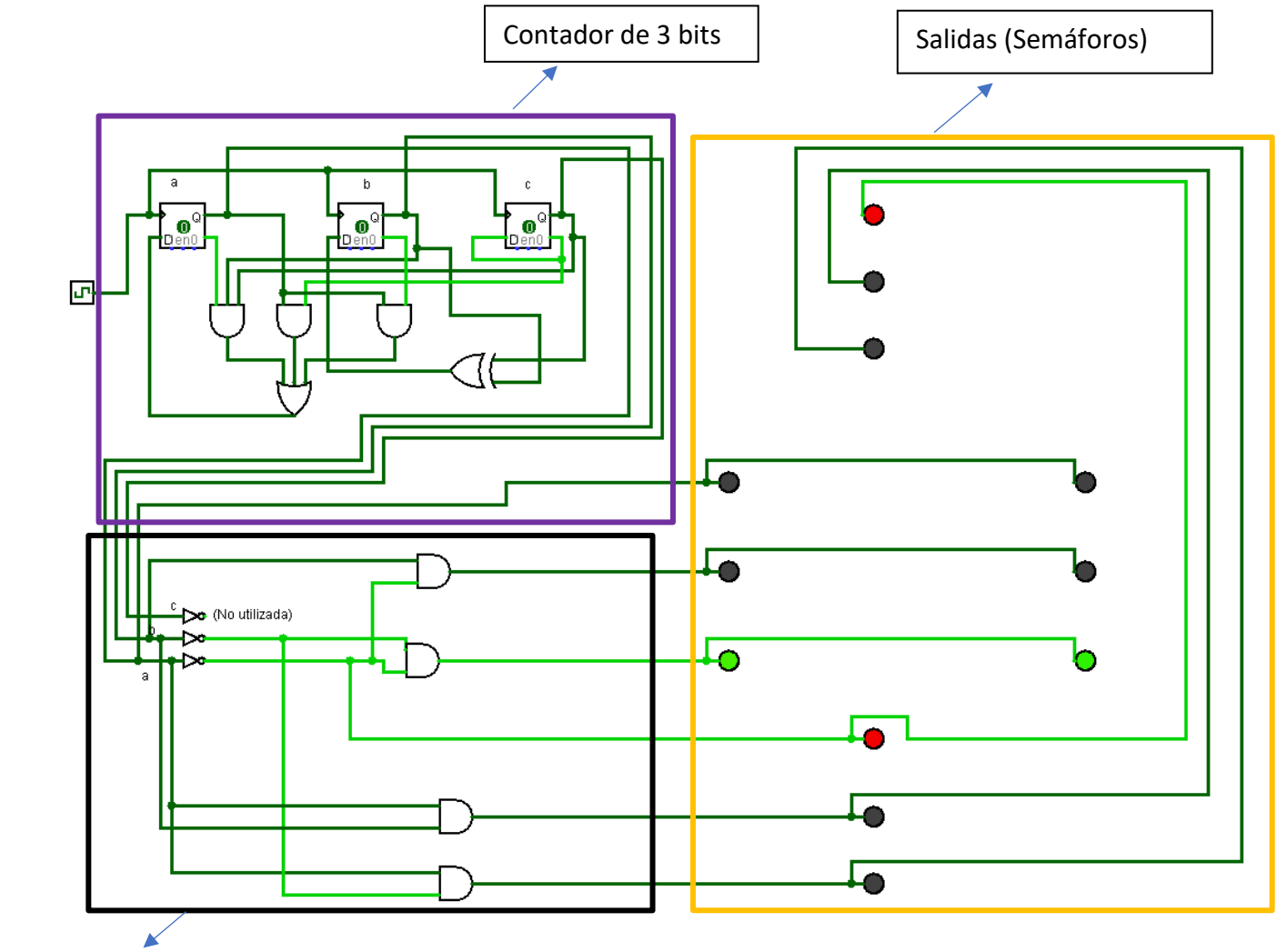
$Q_0(t)$	$Q_1(t)$	$Q_2(t)$	$Q_0(t+1)$	$Q_1(t+1)$	$Q_2(t+1)$
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

Simplificando por mapas de Karnaugh:





El circuito resultante fue el siguiente:



Máquina

Introducción:

En esta parte del obligatorio se pedía una máquina capaz de procesar una palabra genérica ABCDEFG, donde ABC indica la operación a realizar y DEF GHI son los parámetros ingresados.

Dichas operaciones son:

1) ABC = 000: Guardar la palabra GHI en la dirección DEF.
2) ABC = 001: Muestra en la salida la palabra almacenada en la dirección DEF.
3) ABC = 010: Muestra en la salida el resultado de la suma binaria de 3 bits de GHI + DEF.
4) ABC = 011: Resolver la función DEF/G+ DEFG+ /DEF/G + D/E/FG +/D/E/F/G, utilizando la menor cantidad de compuertas posibles.
5) ABC = 100: Se borra cada palabra de la memoria interna.
6) ABC = 101: Muestra 1 en caso de que la palabra DEFGHI (de 6 bits) contenga una cantidad impar de dígitos 1.
7) ABC = 110: Invierte DEF y le suma 1 (procedimiento equivalente a buscar el complemento), muestra el resultado en la salida de 3 bits.
8) ABC = 111: Resta DE - FG. Para el caso de que el valor sea negativo, se prende la bandera de overflow y se descarta el resultado.

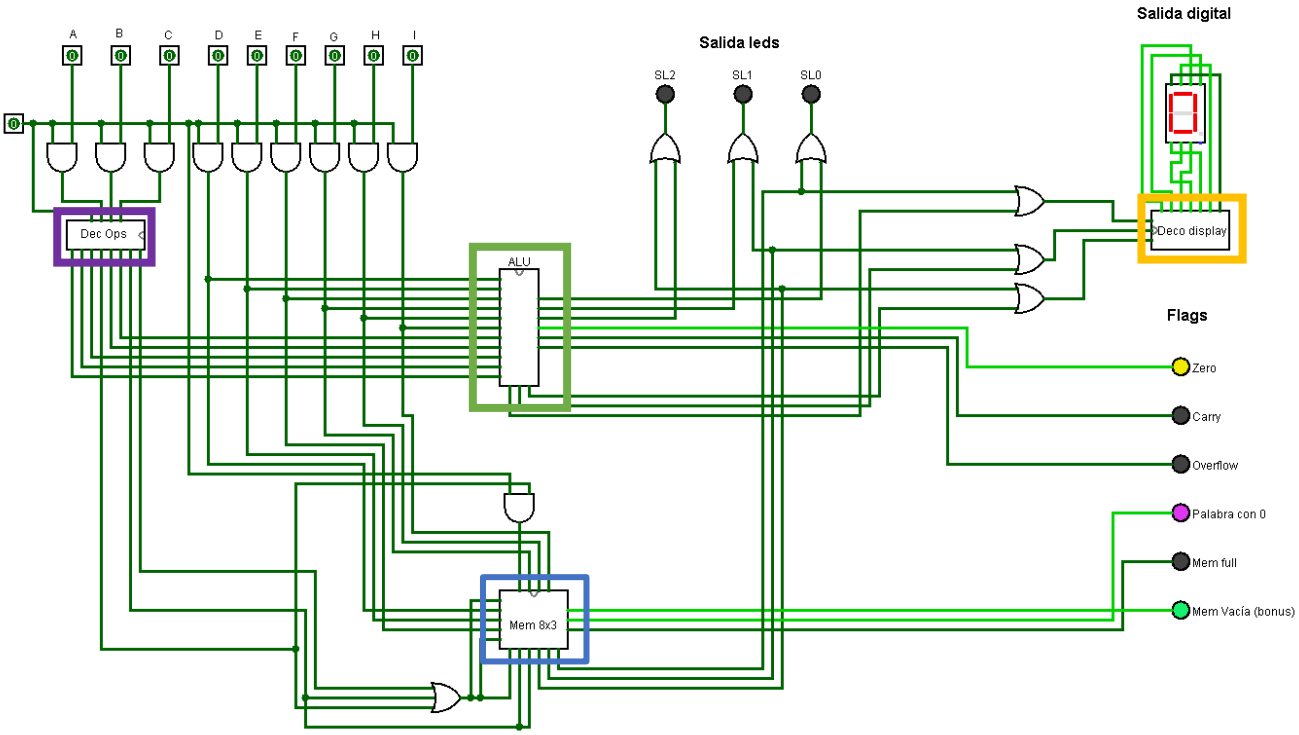
También se pide que la maquina la maquina cuenta con las siguientes flags:

1) Carry Flag: Si ha ocurrido carry en el procesamiento de la operación. (solo en la operación 3)
2) Overflow Flag: Se enciende de existir overflow en la operación 3.
3) Zero Flag: Se enciende si el resultado de la operación es cero.
4) “Existe palabra vacía” Flag: Es true en caso de que exista una palabra de la memoria con todos sus valores.
5) “Memoria llena de unos” Flag: Se enciende si todos los bits de la memoria tienen valor 1.

Los resultados de las operaciones son mostrados (dependiendo de cual se haya seleccionado) en un display de 7 segmentos o en una salida de 3 leds.

Para implementar lo solicitado, el equipo utilizó una estrategia del estilo “divide and conquer” separando las operaciones en “bloques” más chicos.

Así se llegó al siguiente circuito:



Fundamentación de circuitos:

Dec Ops:

Este circuito cumple la función de selección de operaciones, dado lo ingresado en ABC.

Diagrama:

Este circuito es un decodificador de 3 bits el cual cuenta con una cuarta entrada (del botón) con el fin de que solo se seleccione la salida cuando realmente se desee.

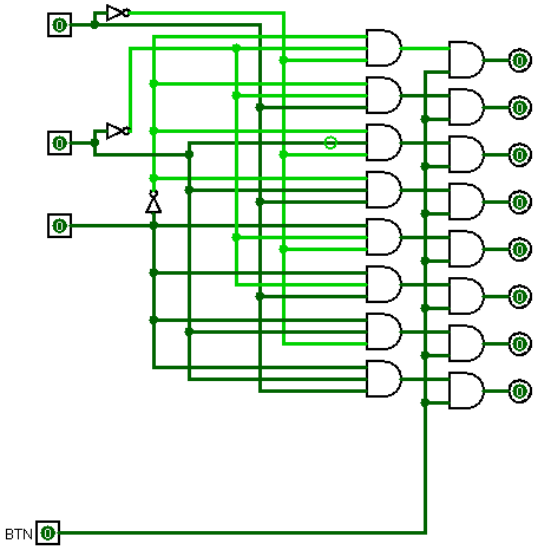
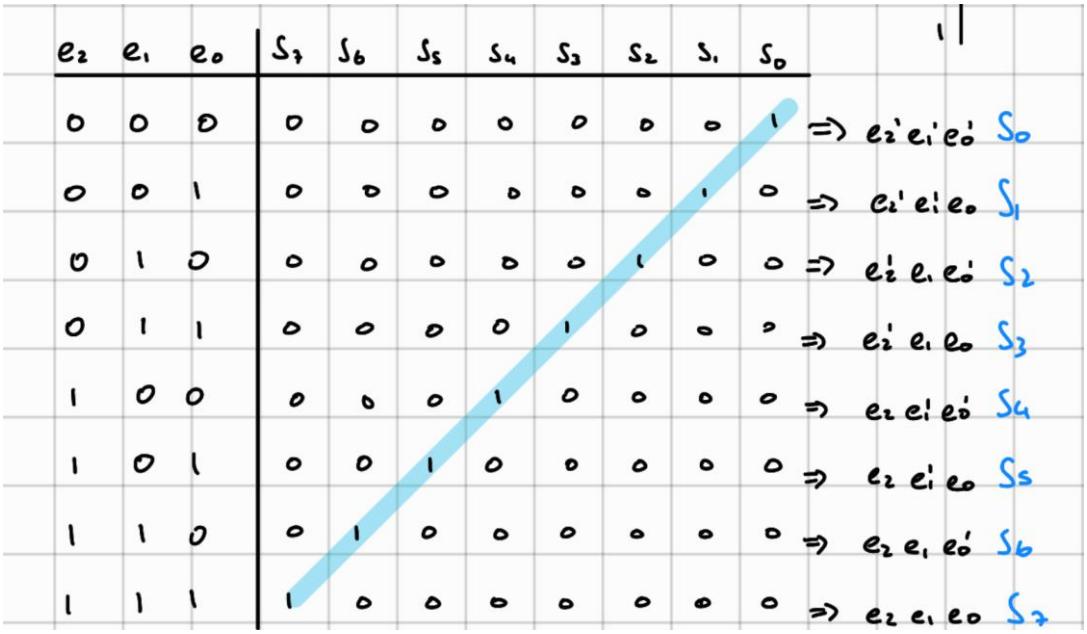


Tabla de verdad:

Se realiza únicamente la tabla del decodificador de 3 bits dado que siempre que el botón no esté habilitado, no selecciona ninguna salida.



Memoria:

Se realizó una memoria RAM de 8 direcciones con palabras de 3 bits (memoria de 2^3 x 3). Donde se almacenan las palabras según lo ingresado en la operación 000, se muestra la palabra deseada según la operación 001 y se borra la memoria (operación 100)

A esta memoria se le agregaron salidas de las flags correspondientes, así como una flag de memoria vacía que el equipo consideró pertinente agregar para fases de pruebas y se decidió dejarla como un añadido que aporta más información sobre el estado de la memoria en todo momento.

Para la construcción se utilizaron circuitos biestables D (proporcionados por la librería de logisim) los cuales fueron conectados tal cual lo visto en clase.

Debido a un funcionamiento extraño del programa logisim, se utilizó el pin "enable CLK" el cual, si ingresa un 0, anula la entrada del CLK; cuando ingresa un 1 habilita la misma. Con el fin de habilitar el CLK únicamente para cuando se seleccionan operaciones de memoria.

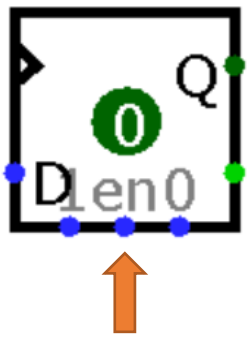
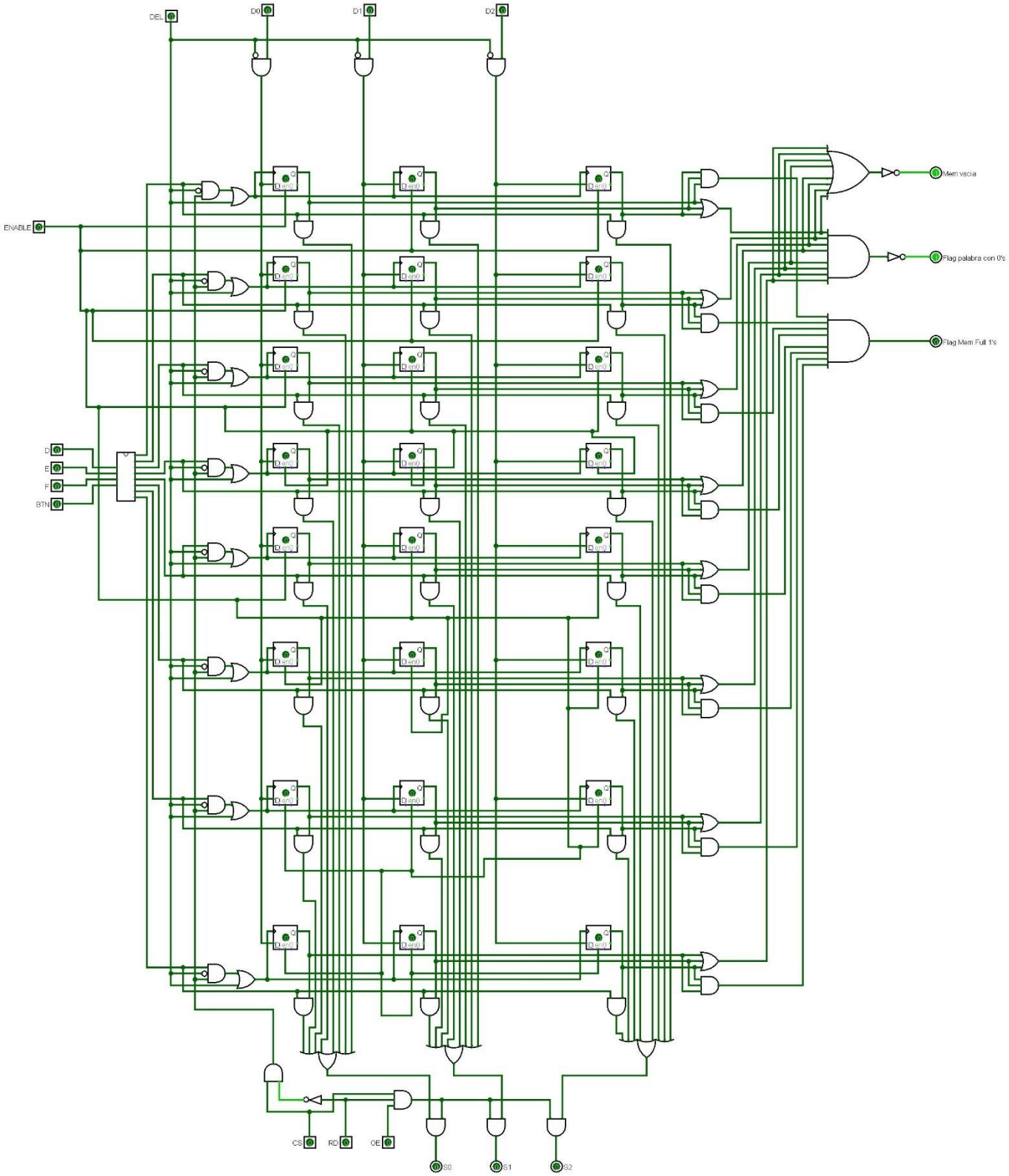


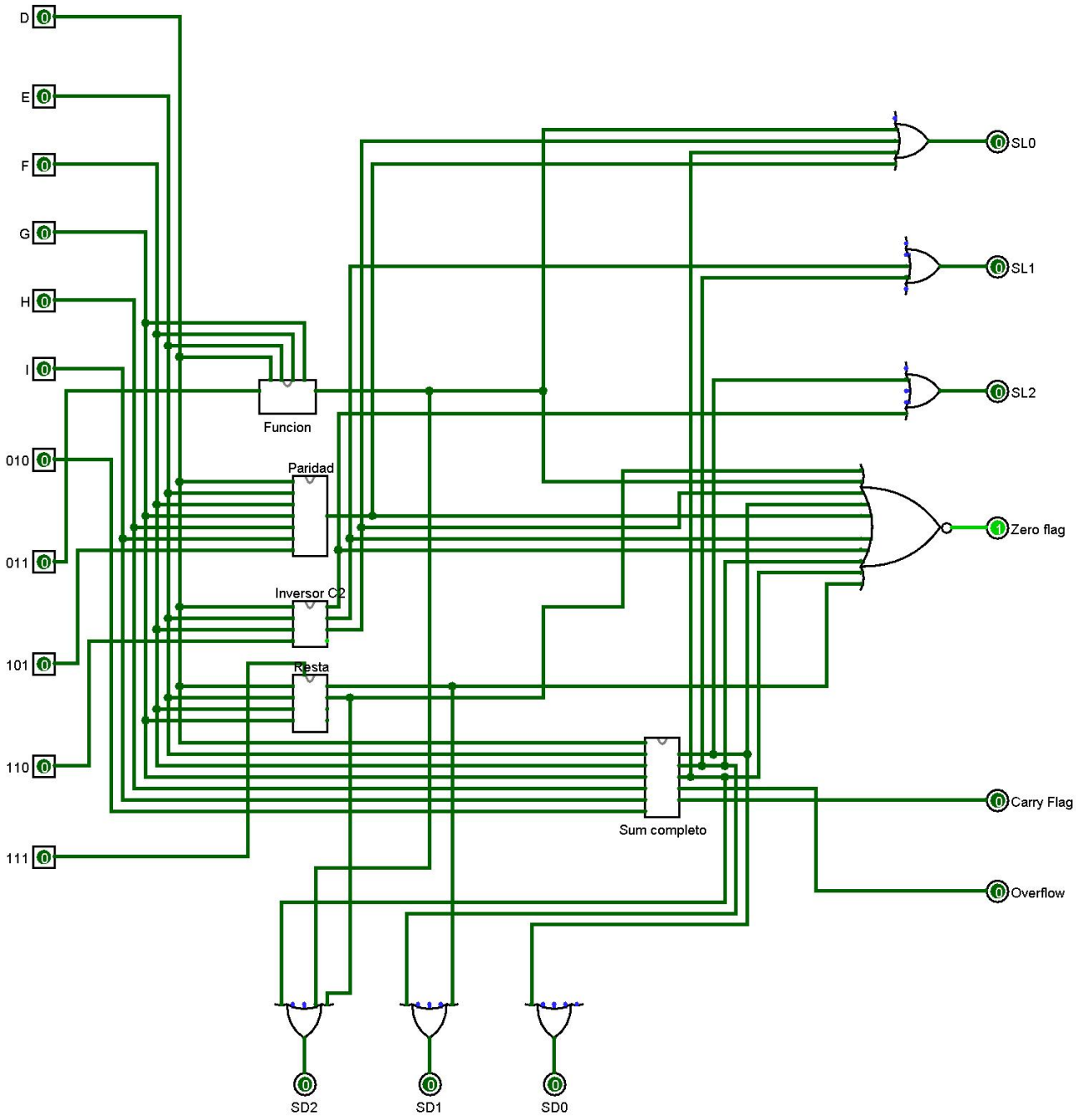
Diagrama:


DEL
D0
D1
D2
ENABLE
D
E
F
BTN
Mem vacia
Flag palabra con 0's
Flag Mem Full 1's
CS
RD
OE
S0
S1
S2

ALU (Arithmetic Logic Unit):

Aquí se realizan todas las operaciones que no involucran la memoria, para ello se subdividieron las operaciones con el fin de simplificar, utilizándose así, un circuito para cada operación de la ALU.

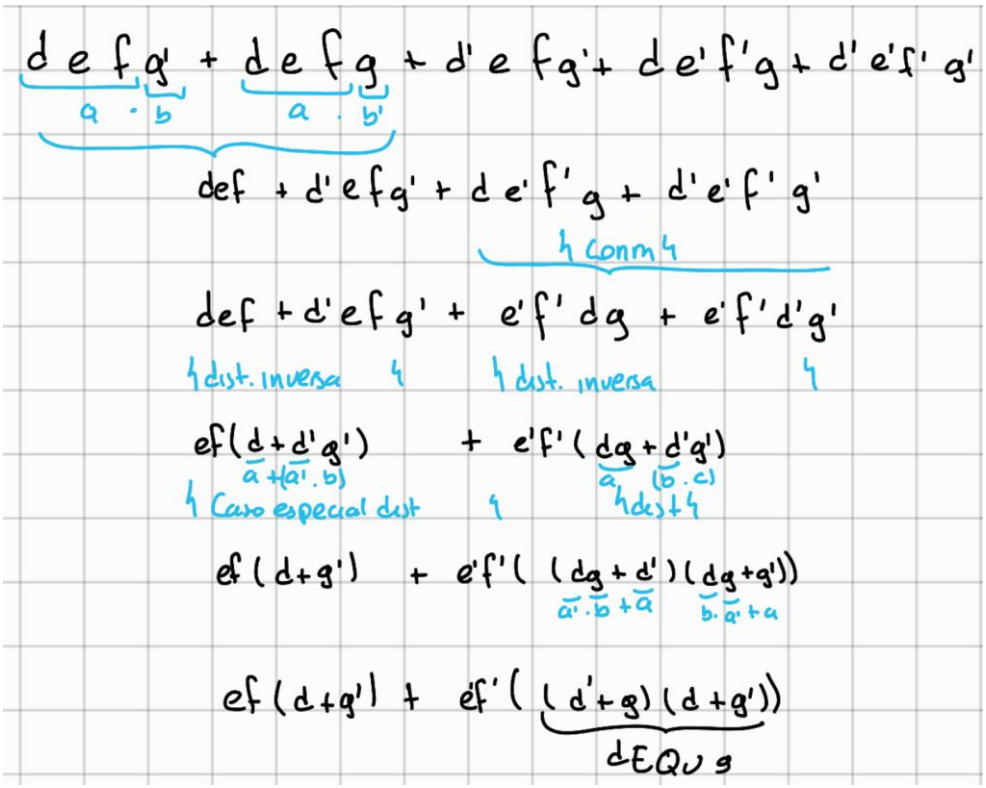
Diagrama:



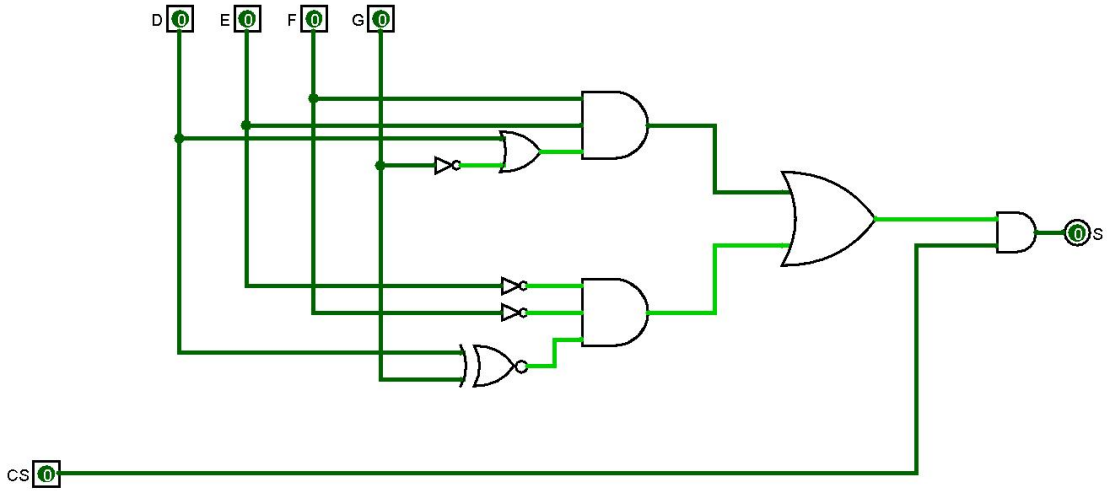
Para este diseño se optó por separar la salida de los leds (SLx) de la salida del display (SDx) y agregar las salidas de las flags pertinentes. Además, cada circuito de las operaciones cuenta con una entrada de “selección” (etiquetado como CS) la cual habilita o no la salida dependiendo de si la operación fue seleccionada.

Función:

Para esta operación se simplificó la función DEF/G+ DEFG+ /DEF/G + D/E/FG +/D/E/F/G de la siguiente manera:



Llegando al siguiente diagrama:



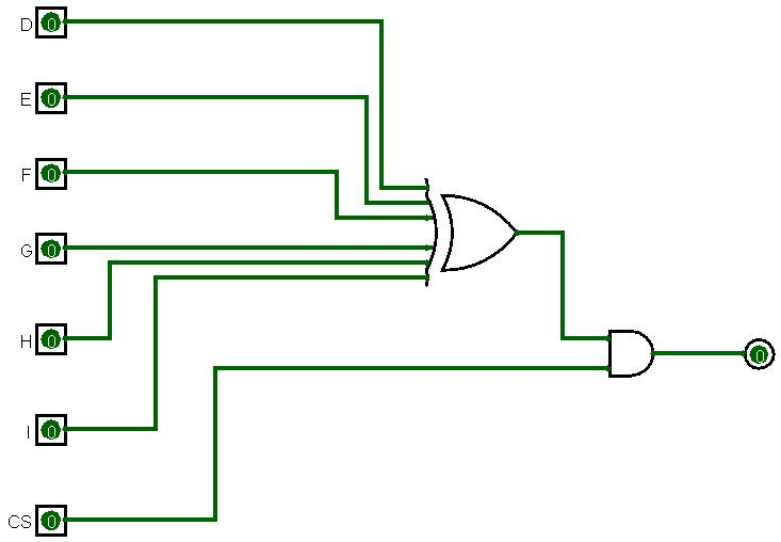
Paridad:

Para esta operación realizamos la siguiente tabla de verdad

D	E	F	G	H	I	Impar
0	0	0	0	0	0	0
0	0	0	0	0	1	1
0	0	0	0	1	0	1
0	0	0	0	1	1	0
0	0	0	1	0	0	1
0	0	0	1	0	1	0
0	0	0	1	1	0	0
0	0	0	1	1	1	1
0	0	1	0	0	0	1
0	0	1	0	0	1	0
0	0	1	0	1	0	0
0	0	1	0	1	1	1
0	0	1	1	0	0	0
0	0	1	1	0	1	1
0	0	1	1	1	0	1
0	0	1	1	1	1	0
0	1	0	0	0	0	1
0	1	0	0	0	1	0
0	1	0	0	1	0	0
0	1	0	0	1	1	1
0	1	0	1	0	0	0
0	1	0	1	0	1	1
0	1	0	1	1	0	1
0	1	0	1	1	1	0
0	1	1	0	0	0	0
0	1	1	0	0	1	1
0	1	1	0	1	0	1
0	1	1	0	1	1	0
0	1	1	1	0	0	1
0	1	1	1	0	1	0
0	1	1	1	1	0	0
0	1	1	1	1	1	1
1	0	0	0	0	0	1
1	0	0	0	0	1	0
1	0	0	0	1	0	0
1	0	0	0	1	1	1
1	0	0	1	0	0	0
1	0	0	1	0	1	1
1	0	0	1	1	0	1
1	0	0	1	1	1	0
1	0	1	0	0	0	0
1	0	1	0	0	1	1
1	0	1	0	1	0	1
1	0	1	0	1	1	0
1	0	1	1	0	0	1
1	0	1	1	0	1	0

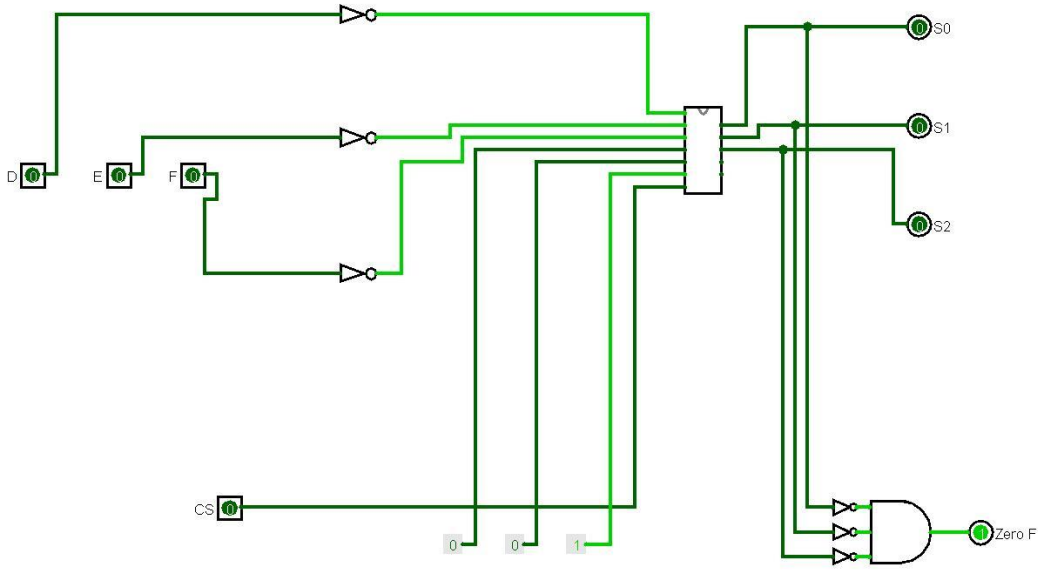
Mediante simplificación por Mapas de Karnaugh se obtuvo un XOR entre todas las entradas.

Llegando al siguiente diagrama



Inversor C2:

Para dicha operación invertimos (negación) la entrada DEF y luego, mediante el uso del circuito sumador (también utilizado en la operación 010) al cual se le agregaron las constantes 001, de forma que la suma sea entre el negado de DEF + 001, obteniendo así el resultado deseado.



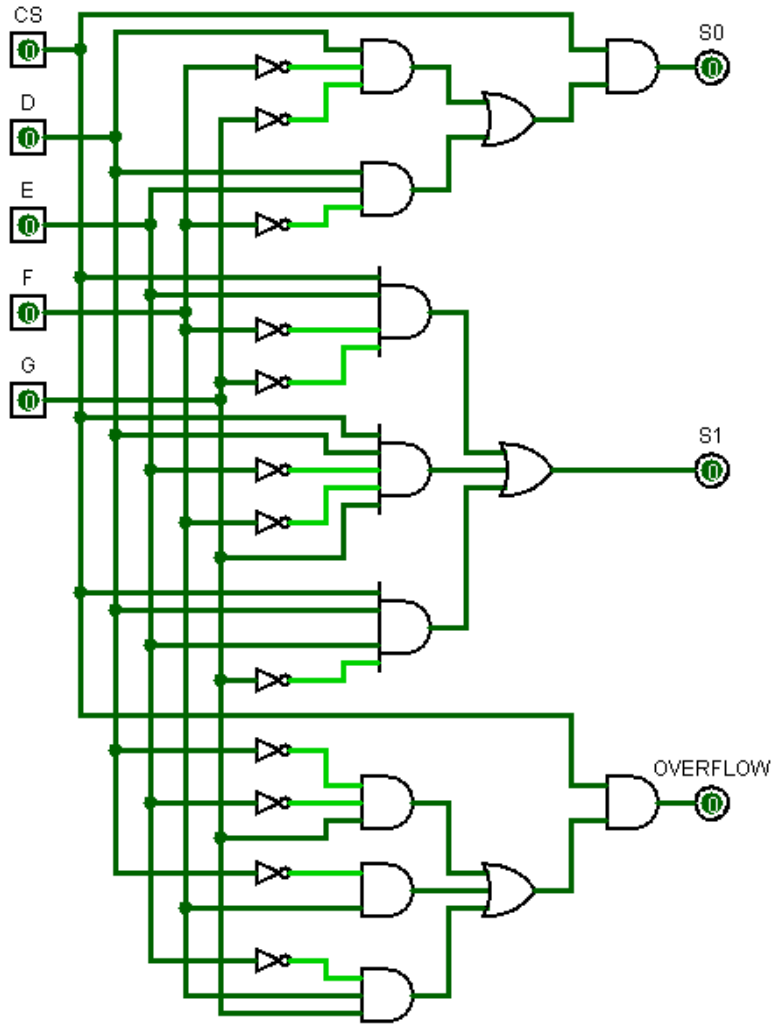
Resta:

Se partió de la siguiente tabla de verdad

D	E	F	G		(DE-FG)	OVERFLOW FLAG
0	0	0	0		00	0
0	0	0	1		00	1
0	0	1	0		00	1
0	0	1	1		00	1
0	1	0	0		01	0
0	1	0	1		00	0
0	1	1	0		00	1
0	1	1	1		00	1
1	0	0	0		10	0
1	0	0	1		01	0
1	0	1	0		0	0
1	0	1	1		00	1
1	1	0	0		11	0
1	1	0	1		10	0
1	1	1	0		01	0
1	1	1	1		00	0

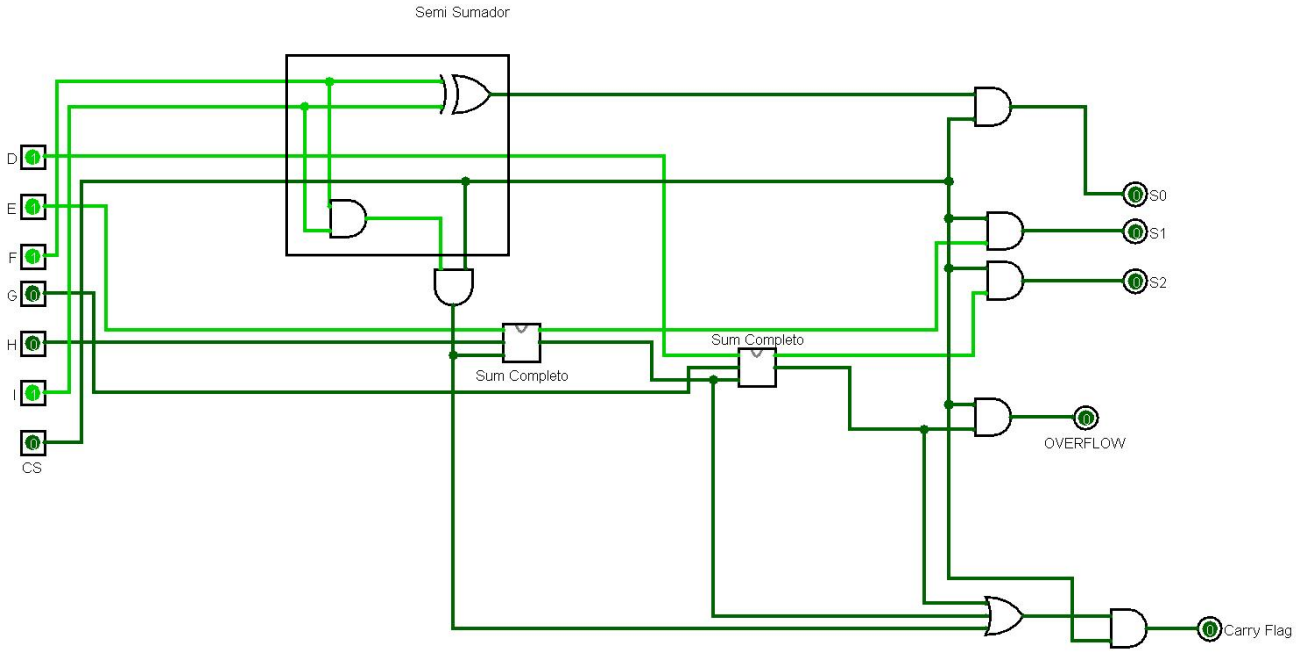
Se tomó la decisión de que en los casos donde el resultado es negativo la operación devuelve 0.

Se llego al siguiente diagrama

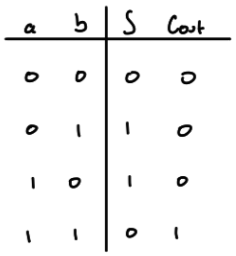


Sum Comp:

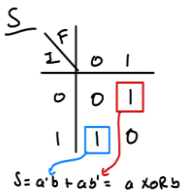
Este circuito se encarga de sumas del estilo DEF + GHI, el cual se encuentra compuesto por un semi sumador y 2 sumadores completos; contempla acarreo y overflow. Se llego al siguiente diagrama:

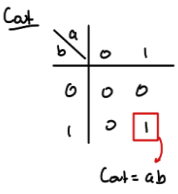


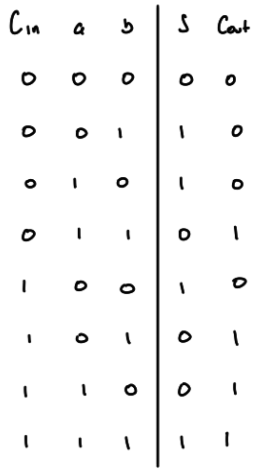
Se proporcionan las tablas de verdad de los circuitos utilizados



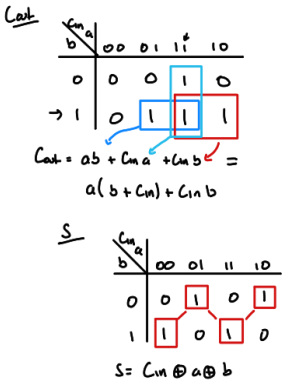
a	b	S	Cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

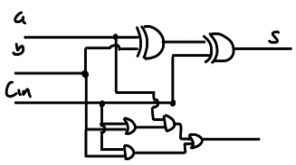






Cin	a	b	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

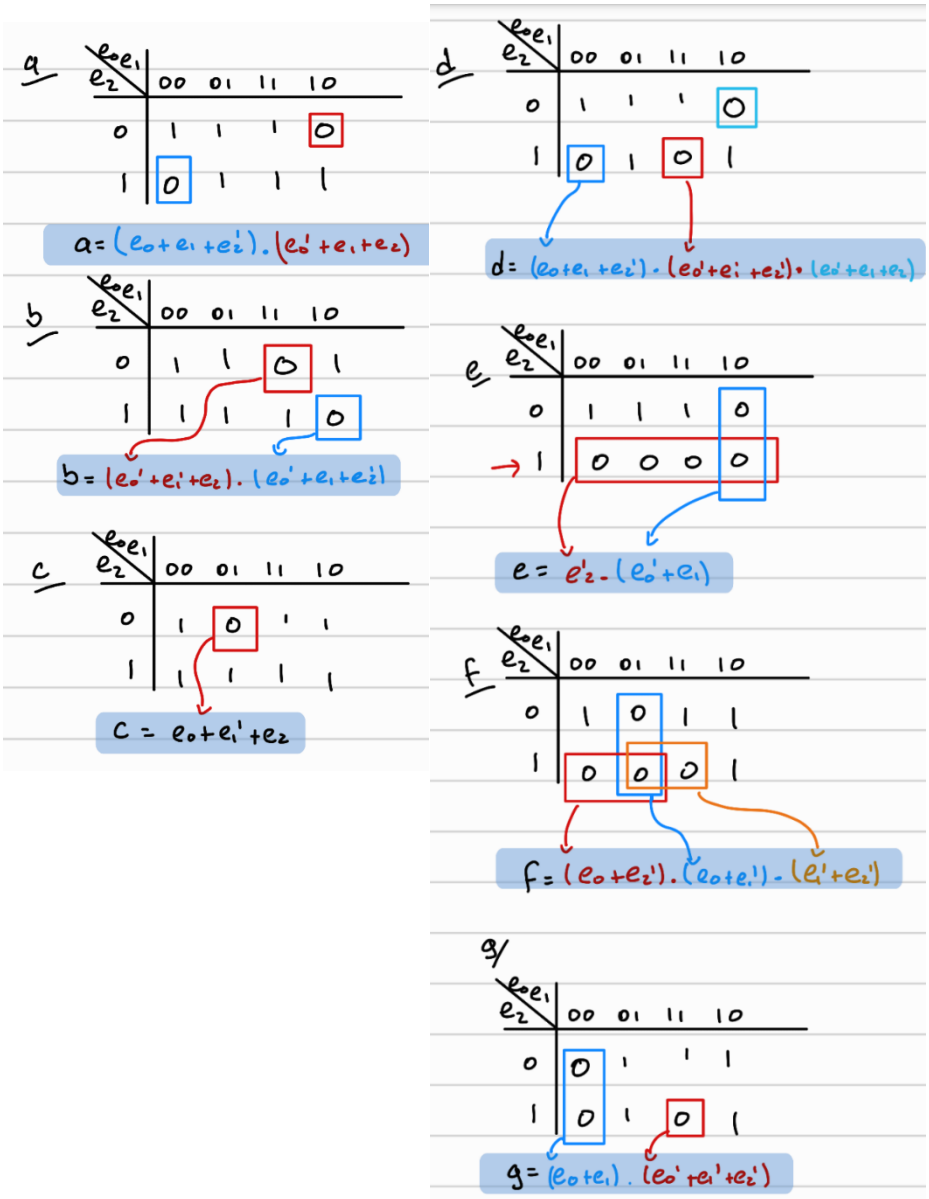




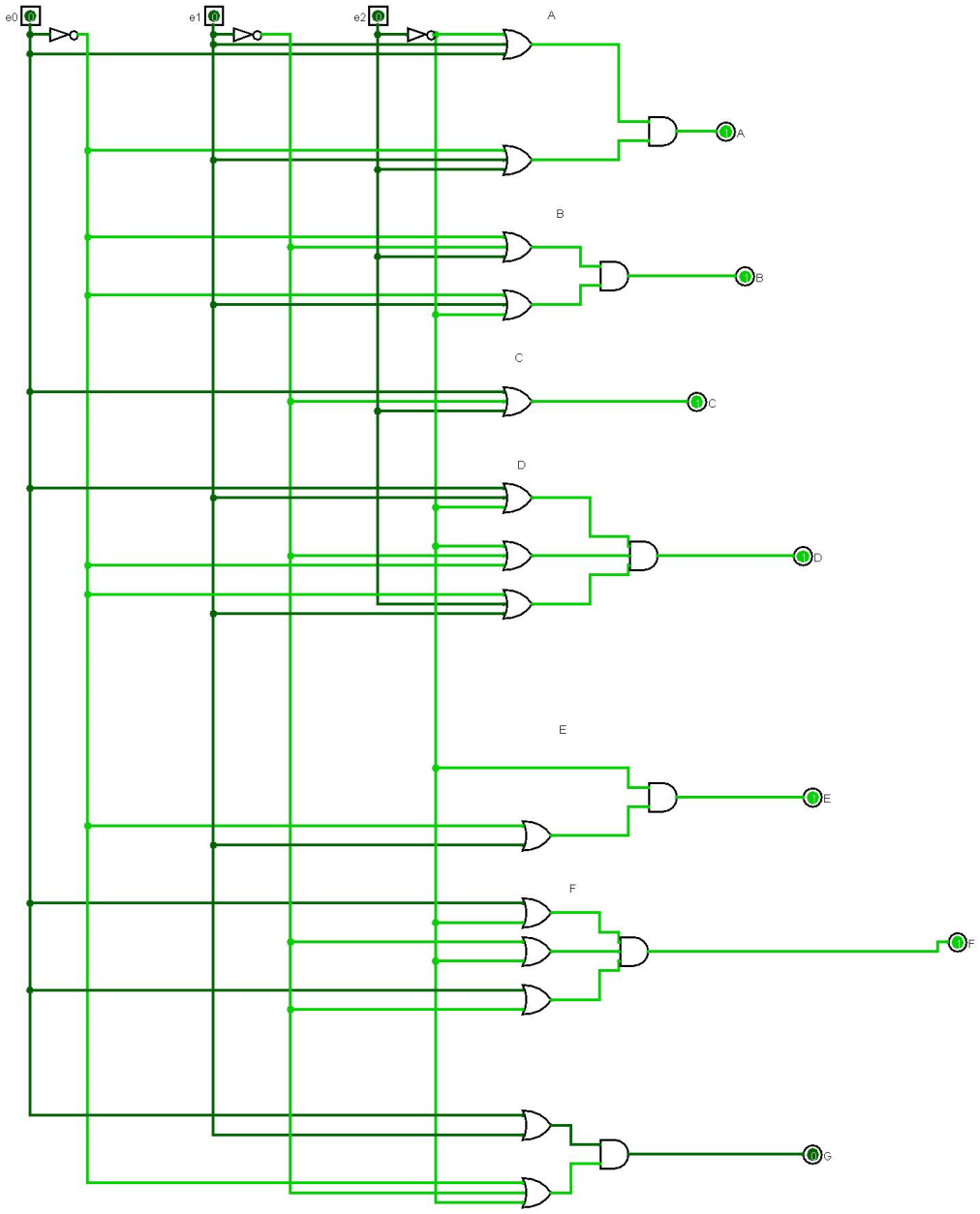
Deco display:

Para este decodificador se realizó la siguiente tabla de verdad

e_0	e_1	e_2	a	b	c	d	e	f	g
0	0	0	1	1	1	1	1	1	0
0	0	1	0	1	1	0	0	0	0
0	1	0	1	1	0	1	1	0	1
0	1	1	1	1	1	1	0	0	1
1	0	0	0	1	1	0	0	1	1
1	0	1	1	0	1	1	0	1	1
1	1	0	1	0	1	1	1	1	1
1	1	1	1	1	1	0	0	0	0



Se llego al siguiente circuito



Conclusión:

Mediante el uso del programa Logisim y los conocimientos aprendidos en clase se logró realizar todos los objetivos propuestos por la letra.