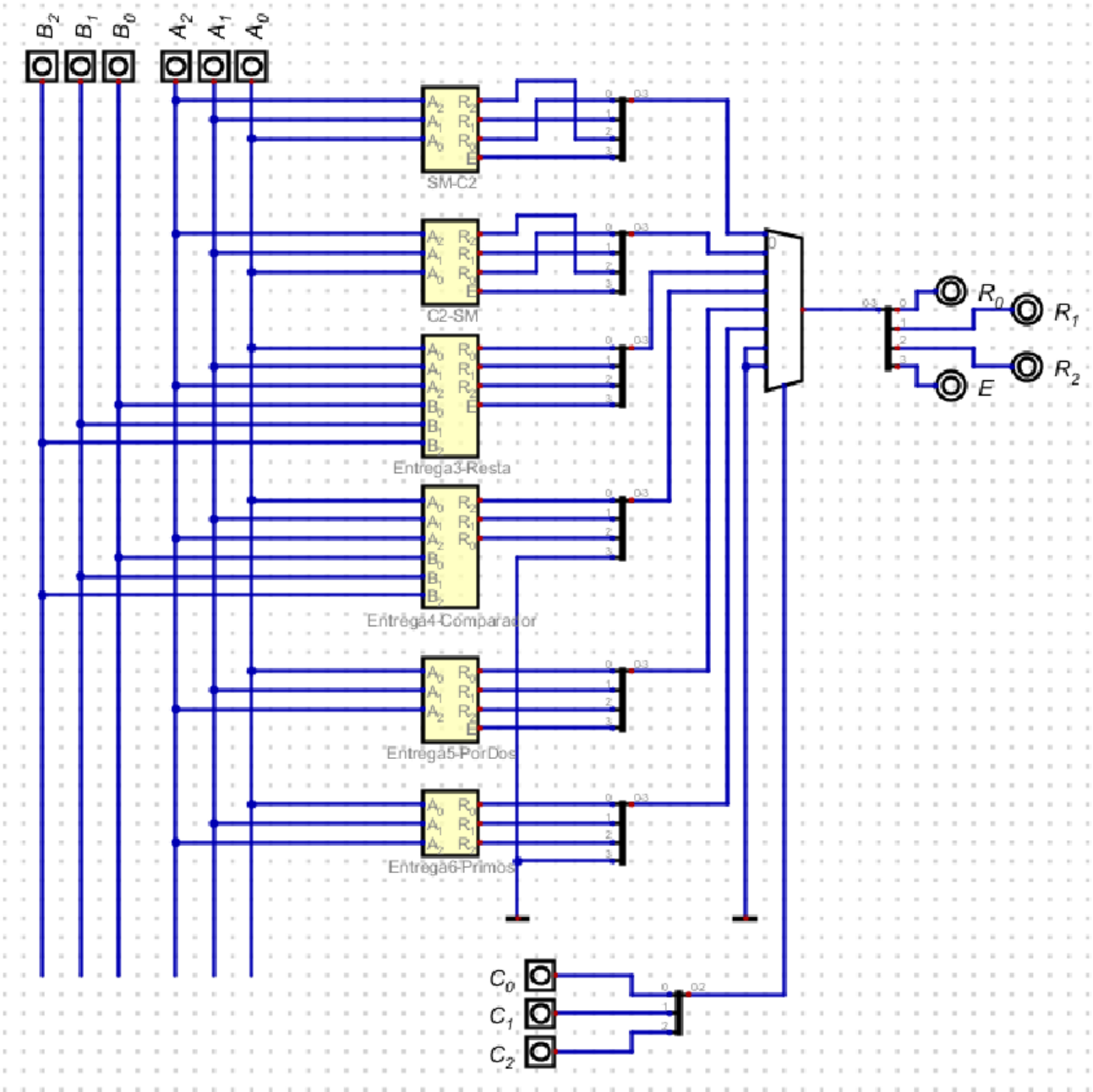
Sistemas Digitales: Práctica Combinacionales



GORDIOLA FRAU, ANTONI 43482642-S

SAYES SÁNCHEZ, ALEJANDRO 43215170-X

ÍNDICE

INTRODUCCIÓN:

A lo largo de la práctica de combinacionales, hemos desarrollado una ALU (Unidad Aritmético-Lógica o *Arithmetic Logic Unit* en inglés), con las instrucciones del profesorado y siguiendo las pautas marcadas en la guía de la práctica. Destacamos el uso de los apuntes proporcionados por el profesorado.

Como una ALU está compuesta de una serie de entradas y salidas que hacen la función de dar valores y hacer unas operaciones determinadas, cabe indicar que durante todo el desarrollo hemos usado las identificaciones de la guía. Como esta requiere de dos entradas de datos de 3 bits, estas son A(A_2, A_1 y A_0) y B(B_2, B_1, B_0), una entrada de selección C(C_2, C_1, C_0) y dos de salida, R(R_2, R_1, R_0) y E, de 1 bit.

La entrada C, que indica la operación a realizar, cumple su funcionalidad según la guía, siendo de los valores 000 hasta el 101 las operaciones indicadas: transformar de Signo y magnitud a Complemento de 2, transformar de Complemento de 2 a Signo y magnitud, realizar la resta de A – B con indicador de Overflow con la E siendo todos los valores en C2, comparar los valores de A y B para saber si uno es mayor, menor o igual al otro siendo también en C2, realizar la multiplicación de A * 2 en BNSS e indicar si el valor de A en BNSS es primo o no, respectivamente. De esta forma, los valores de C que se excedan del rango útil ([000,101]) tendrán salida nula (R 000 y E 0).

DESCRIPCIÓN DE LAS PARTES IDENTIFICADAS:

Transformación de Signo y Magnitud a Complemento a 2:

Este circuito se encarga de, a través de la introducción de un número de 3 bits representado en Signo y Magnitud, transformar este a Complemento a 2. Cómo el rango en SM de 3 bits va de [-3,3] y el rango de C2 va de [-4, 3], nos apoyaremos en el bit de salida “E” para saber si el número introducido es representable (E = 0) o por el contrario, no es representable (E = 1).

A2	A1	A0	R2	R1	R0	E
0	0	0	0	0	0	0
0	0	1	0	0	1	0
0	1	0	0	1	0	0
0	1	1	0	1	1	0
1	0	0	0	0	0	0
1	0	1	1	1	1	0
1	1	0	1	1	0	0
1	1	1	1	0	1	0

Tras ver el resultado de las tablas de verdad, nos dispusimos a realizar los respectivos mapas de Karnaugh. Estos nos devolvieron los siguientes resultados:

R2 = (A2 * A1) + (A2 * A0)
R1 = (A2’ * A1) + (A1 * A0’) + (A2 * A1’ * A0)
R0 = A0
E = 0

Transformación de Complemento a 2 a Signo y Magnitud:

Este circuito se encarga de, a través de la introducción de un número de 3 bits representado en Complemento a 2, transfórmalo a Signo y Magnitud. Cómo hemos visto en el caso anterior los rangos de C2 y SM son diferentes. Por lo que también utilizaremos el bit de salida "E" para saber si el número introducido es representable (E = 0) o por el contrario, no es representable (E = 1).

A2	A1	A0	R2	R1	R0	E
0	0	0	0	0	0	0
0	0	1	0	0	1	0
0	1	0	0	1	0	0
0	1	1	0	1	1	0
1	0	0	X	X	X	1
1	0	1	1	1	1	0
1	1	0	1	1	0	0
1	1	1	1	0	1	0

Para este circuito no realizamos ningún mapa de Karnaugh para las salidas R2, R1 y R0 ya que iban a tener el mismo resultado que las salidas del circuito anterior. Sin embargo, si tuvimos que realizar uno para la salida E ya que esta iba a ser diferente por el hecho de que el número 4 (100) no puede ser representado.

R2 = (A2 * A1) + (A2 * A0)
R1 = (A2' * A1) + (A1 * A0') + (A2 * A1' * A0)
R0 = A0
E = A2 * A1' * A0'

A – B:

Este circuito se encarga de, a través de la introducción de dos números de 3 bits representados en C2, realizar su resta. Para realizarlo hemos realizado la suma de A + (-B), para así poder utilizar Full Adders integrados ya en Digital. En este caso también utilizamos el bit de salida "E", pero con una función distinta. E = 1, representará que en la resta se produce Overflow, mientras que E = 0 representará que no.

A2	A0	A1	B2	B1	B0	R2	R1	R0	E
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	1	1	0
0	0	0	0	1	0	1	1	0	0
0	0	0	0	1	1	1	0	1	0
0	0	0	1	0	0	1	0	0	1
0	0	0	1	0	1	0	1	1	0
0	0	0	1	1	0	0	1	0	0
0	0	0	1	1	1	0	0	1	0
0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	1	0	0	1	0
0	0	1	0	1	0	0	0	0	0
0	0	1	0	1	1	1	1	1	0
0	0	1	1	0	0	1	1	0	1
0	0	1	1	0	1	1	0	1	1
0	0	1	1	1	0	1	0	0	1
0	0	1	1	1	1	0	1	1	0
0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	1	0	0	0	0
0	1	0	0	1	0	1	1	1	0
0	1	0	0	1	1	1	1	0	0
0	1	0	1	0	0	1	0	1	1
0	1	0	1	0	1	1	0	0	1
0	1	0	1	1	0	0	1	1	0
0	1	0	1	1	1	0	1	0	0
0	1	1	0	0	0	0	1	1	0
0	1	1	0	0	1	0	1	0	0
0	1	1	0	1	0	0	0	1	0
0	1	1	0	1	1	0	0	0	0
0	1	1	1	0	0	1	1	1	1
0	1	1	1	0	1	1	1	0	1
0	1	1	1	1	0	1	0	1	1
0	1	1	1	1	1	1	0	0	1
1	0	0	0	0	0	1	0	0	0
1	0	0	0	0	1	0	1	1	1
1	0	0	0	1	0	0	1	0	1
1	0	0	0	1	1	0	0	1	1
1	0	0	1	0	0	0	0	0	0
1	0	0	1	0	1	1	1	1	0
1	0	0	1	1	0	1	1	0	0
1	0	0	1	1	1	1	0	1	0

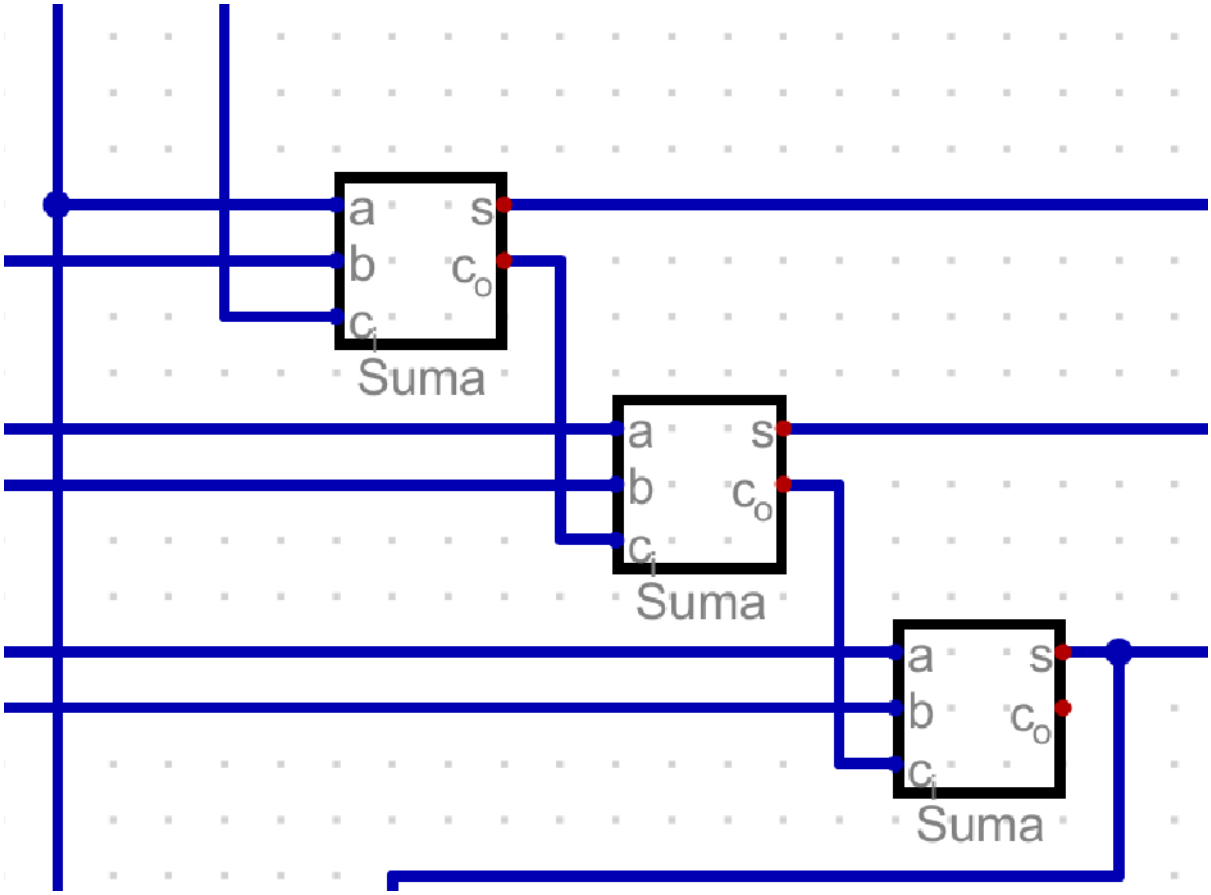
1	0	1	0	0	0	1	1	0	0
1	0	1	0	0	1	1	0	1	0
1	0	1	0	1	0	1	0	0	0
1	0	1	0	1	1	0	1	1	1
1	0	1	1	0	0	0	1	0	0
1	0	1	1	0	1	0	0	1	0
1	0	1	1	1	0	0	0	0	0
1	0	1	1	1	1	1	1	1	0
1	1	0	0	0	0	1	0	1	0
1	1	0	0	0	1	1	0	0	0
1	1	0	0	1	0	0	1	1	1
1	1	0	0	1	1	0	1	0	1
1	1	0	1	0	0	0	0	1	0
1	1	0	1	0	1	0	0	0	0
1	1	0	1	1	0	1	1	1	0
1	1	0	1	1	1	1	1	0	0
1	1	1	0	0	0	1	1	1	0
1	1	1	0	0	1	1	1	0	0
1	1	1	0	1	0	1	0	1	0
1	1	1	0	1	1	1	0	0	0

1	0	0	1	0	0	0	0	0	0
1	0	0	1	0	1	1	1	1	0
1	0	0	1	1	0	1	1	0	0
1	0	0	1	1	1	1	0	1	0
1	0	1	0	0	0	1	1	0	0
1	0	1	0	0	1	1	0	1	0
1	0	1	0	1	0	1	0	0	0
1	0	1	0	1	1	0	1	1	1
1	0	1	1	0	0	0	1	0	0
1	0	1	1	0	1	0	0	1	0
1	0	1	1	1	0	0	0	0	0
1	0	1	1	1	1	1	1	1	0
1	1	0	0	0	0	1	0	1	0
1	1	0	0	0	1	1	0	0	0
1	1	0	0	1	0	0	1	1	1
1	1	0	0	1	1	0	1	0	1
1	1	0	1	0	0	0	0	1	0
1	1	0	1	0	1	0	0	0	0
1	1	0	1	1	0	1	1	1	0
1	1	0	1	1	1	1	1	0	0

1	1	1	0	0	0	1	1	1	0
1	1	1	0	0	1	1	1	0	0
1	1	1	0	1	0	1	0	1	0
1	1	1	0	1	1	1	0	0	0
1	1	1	1	0	0	0	1	1	0
1	1	1	1	0	1	0	1	0	0
1	1	1	1	1	0	0	0	1	0
1	1	1	1	1	1	0	0	0	0

Para hacer este circuito optamos por realizar en vez de la operación A – B, hacer A + (-B). Tras cambiar de signo B, gracias a los apuntes proporcionados del Tema 1 supimos que para hacer un sumador de 3 bits necesitaríamos 3 Full Adders. El primer Full Adder tiene como entradas A0, B0 y C_In= 1 (Debido al cambio de signo realizado en B); como salidas tiene un C_Out y R0. El segundo Full Adder tiene como entradas A1, B1 y C_In = C_Out del 1º Full Adder; como salidas tiene un C_Out y R1. Por último

tenemos el tercer Full Adder el cual cuenta con las entradas A2, B2 Y C_In = C_Out del 2º Full Adder; como salidas tiene C_Out y R2.



Para saber si la resta tendrá o no Overflow, nos acogemos a las características de la suma en C2 (ya que cambiamos el signo de B), y es que la operación únicamente tendrá Overflow si el signo de los dos números a sumar es el mismo pero el del su resultado es diferente.

Comparador:

Este circuito se encarga de, a través de la introducción de dos números de 3 bits representados en C2, realizar su comparación. En caso de que el número A > B, la salida será R = 001. Si A = B, la salida será de R = 010, mientras que si A < B, la salida será de R = 100. En este circuito no necesitaremos el bit de salida "E" por lo que siempre será igual a 0.

A2	A1	A0	B2	B0	B1	R2	R1	R0
0	0	0	0	0	0	0	1	0
0	0	0	0	0	1	1	0	0
0	0	0	0	1	0	1	0	0
0	0	0	0	1	1	1	0	0
0	0	0	1	0	0	1	0	0
0	0	0	1	0	1	0	0	1
0	0	0	1	1	0	0	0	1
0	0	0	1	1	1	0	0	1
0	0	1	0	0	0	0	0	1
0	0	1	0	0	1	1	0	0
0	0	1	0	1	0	0	1	0
0	0	1	0	1	1	1	0	0
0	0	1	1	0	0	1	0	0
0	0	1	1	0	1	0	0	1
0	0	1	1	1	0	1	0	0
0	0	1	1	1	1	0	0	1
0	1	0	0	0	0	0	0	1
0	1	0	0	0	1	0	1	0
0	1	0	0	1	0	0	0	1
0	1	0	0	1	1	1	0	0
0	1	0	1	0	0	1	0	0
0	1	0	1	0	1	1	0	0
0	1	0	1	1	0	1	0	0
0	1	0	1	1	1	0	0	1
0	1	1	0	0	0	0	0	1
0	1	1	0	0	1	0	0	1
0	1	1	0	1	0	0	0	1
0	1	1	0	1	1	0	1	0
0	1	1	1	0	0	1	0	0
0	1	1	1	0	1	1	0	0
0	1	1	1	1	0	1	0	0
0	1	1	1	1	1	1	0	0
1	0	0	0	0	0	1	0	0
1	0	0	0	0	1	0	0	1
1	0	0	0	1	0	0	0	1
1	0	0	0	1	1	0	0	1
1	0	0	1	0	0	0	1	0
1	0	0	1	0	1	1	0	0
1	0	0	1	1	0	1	0	0
1	0	0	1	1	1	1	0	0

1	0	1	0	0	0	1	0	0
1	0	1	0	0	1	0	0	1
1	0	1	0	1	0	1	0	0
1	0	1	0	1	1	0	0	1
1	0	1	1	0	0	0	0	1
1	0	1	1	0	1	1	0	0
1	0	1	1	1	0	0	1	0
1	0	1	1	1	1	1	0	0
1	1	0	0	0	0	1	0	0
1	1	0	0	0	1	1	0	0
1	1	0	0	1	0	1	0	0
1	1	0	0	1	1	0	0	1
1	1	0	1	0	0	0	0	1
1	1	0	1	0	1	0	1	0
1	1	0	1	1	0	0	0	1
1	1	0	1	1	1	1	0	0
1	1	1	0	0	0	1	0	0
1	1	1	0	0	1	1	0	0
1	1	1	0	1	0	1	0	0
1	1	1	0	1	1	1	0	0

1	1	1	1	0	0	0	0	1
1	1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0	1
1	1	1	1	1	1	0	1	0

Para realizar este circuito decidimos crear la siguiente tabla de verdad con el objetivo de hacer más fácil nuestro trabajo:

S2	S1	S0	R2	R1	R0
0	0	0	0	1	0
0	0	1	0	0	1
0	1	0	0	0	1
0	1	1	0	0	1
1	0	0	1	0	0
1	0	1	1	0	0
1	1	0	1	0	0
1	1	1	1	0	0

Dónde S está basada en los posibles resultados que nos puede dar la resta A -B. En esta tabla de verdad podemos ver que si el resultado de la resta es igual a 0 querrá decir que **A = B**, entonces el resultado a representar será **010**. Si el resultado de la resta nos devuelve un número positivo querrá decir que el número **A > B** por lo que a resultado será **001**. Mientras que si el resultado de la resta es negativo significará que **A<B** por lo que el valor del resultado será **100**. Tras crear esta tabla realizamos sus respectivos mapas de Karnaugh que nos devolvieron estos resultados:

R2 = S2
R1 = S2’ *S1’ * S0’
R0 = S2’ * S1 + S2’ * S0
E = 0

A x 2:

Este circuito se encarga de, a través de la introducción de un número de 3 bits representado en BNSS, multiplicarlo por 2 (010). El resultado también será representado en BNSS y precisaremos de la ayuda del bit de salida "E" para que haga la función de cuarto bit ya que algunos resultados no se pueden representar solamente con 3 bits (5 (1010) · 2(010) = 10 (1010)).

A2	A1	A0	2_2	2_1	2_0	E	R2	R1	R0
0	0	0	0	1	0	0	0	1	0
0	0	1	0	1	0	0	0	0	1
0	1	0	0	1	0	0	0	0	1
0	1	1	0	1	0	0	0	0	1
1	0	0	0	1	0	1	1	0	0
1	0	1	0	1	0	1	1	0	0
1	1	0	0	1	0	1	1	0	0
1	1	1	0	1	0	1	1	0	0

Para realizar este circuito implementamos una nueva variable que es el numero 2 (010). Tras esto para saber los valores de R y 0 hicimos las respectivas multiplicaciones. Por último, llevamos a cabo los mapas de Karnaugh que nos devolvieron los siguientes resultados:

R2 = A1
R1 = A0
R0 = 0
E = A2

Número primo:

Este circuito se encarga de, a través de la introducción de un número de 3 bits representado en BNSS, comprobar si este es un número primo o no. EN caso de que sí la salida será de R = 111 y en caso contrario de R = 000. En este circuito no necesitaremos la ayuda del bit de salida "E", por lo que siempre tendrá el valor 0.

A2	A1	A0	R2	R1	R0	E
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	1	1	1	0
0	1	1	1	1	1	0
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	1	1	1	0
1	1	1	1	1	1	0

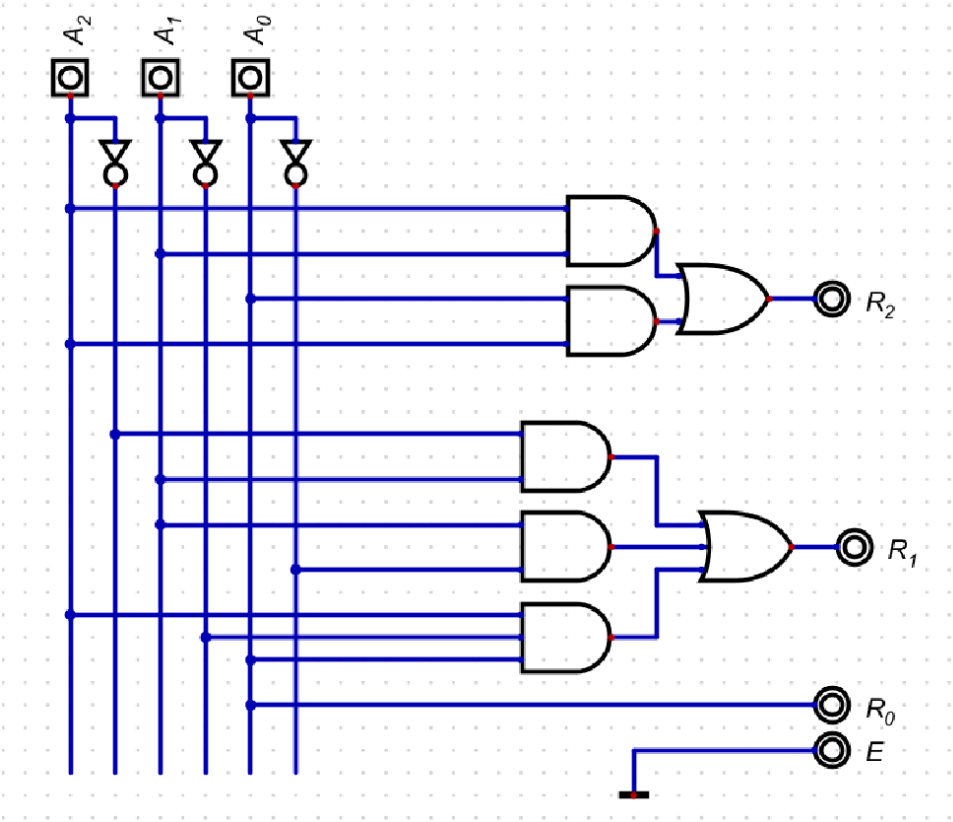
Para realizar este circuito realizamos un único mapa de Karnaugh ya que como podemos ver en la tabla las salidas R2, R1 y R0 son iguales por lo que obtenemos los siguientes resultados:

R2, R1 y R0 = (A2' * A1) + (A2 * A0)

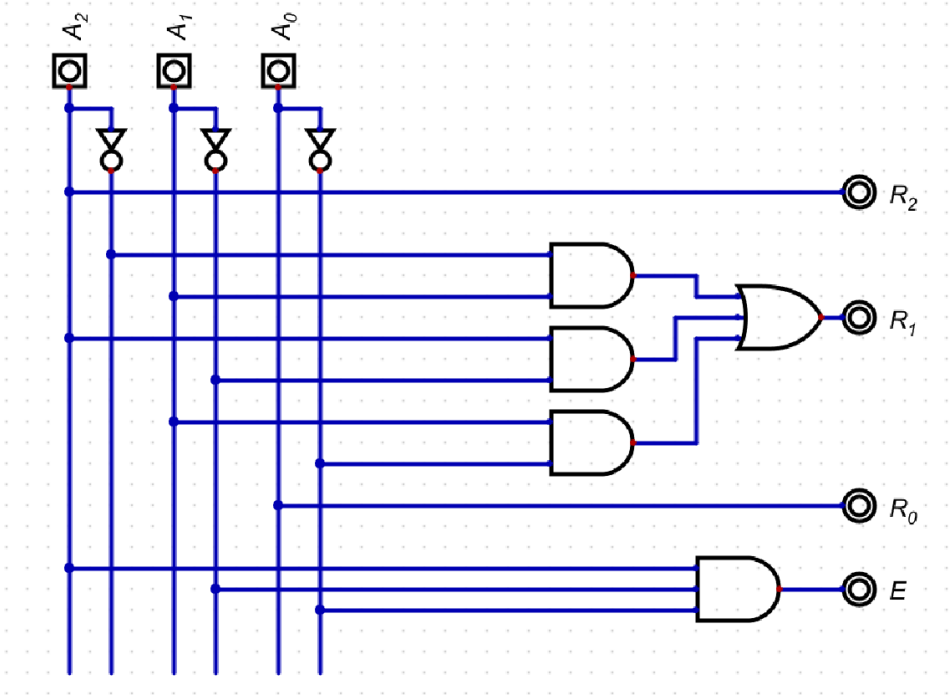
E = 0

CIRCUITO DE CADA UNA DE LAS PARTES IDENTIFICADAS:

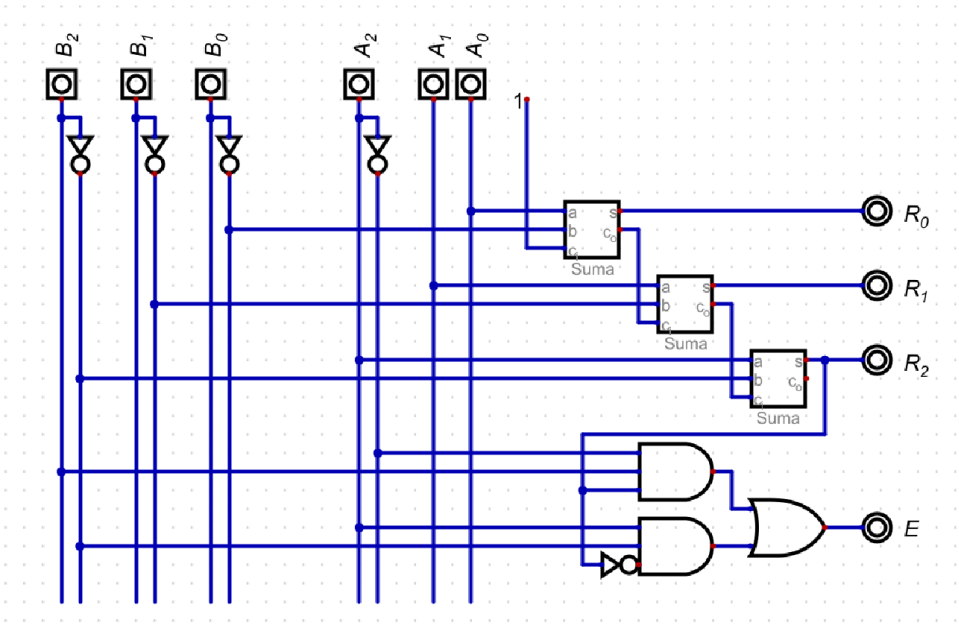
Transformación de Signo y Magnitud a Complemento a 2:



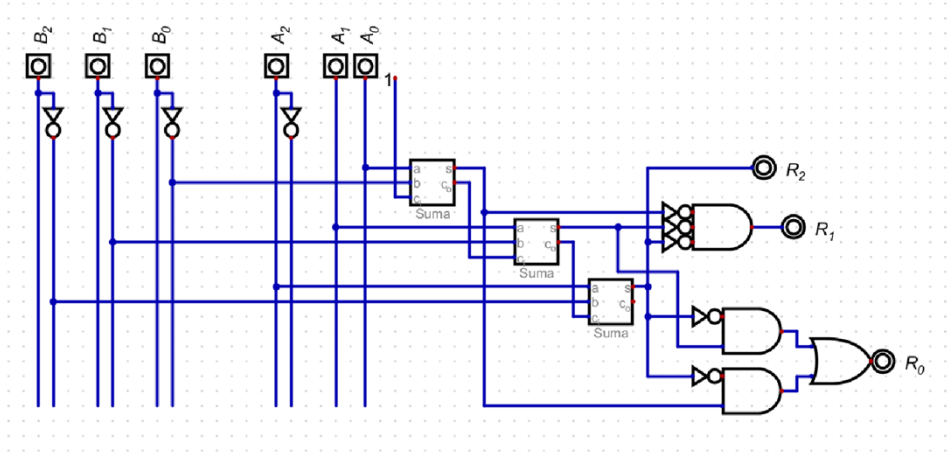
Transformación de Complemento a 2 a Signo y Magnitud:



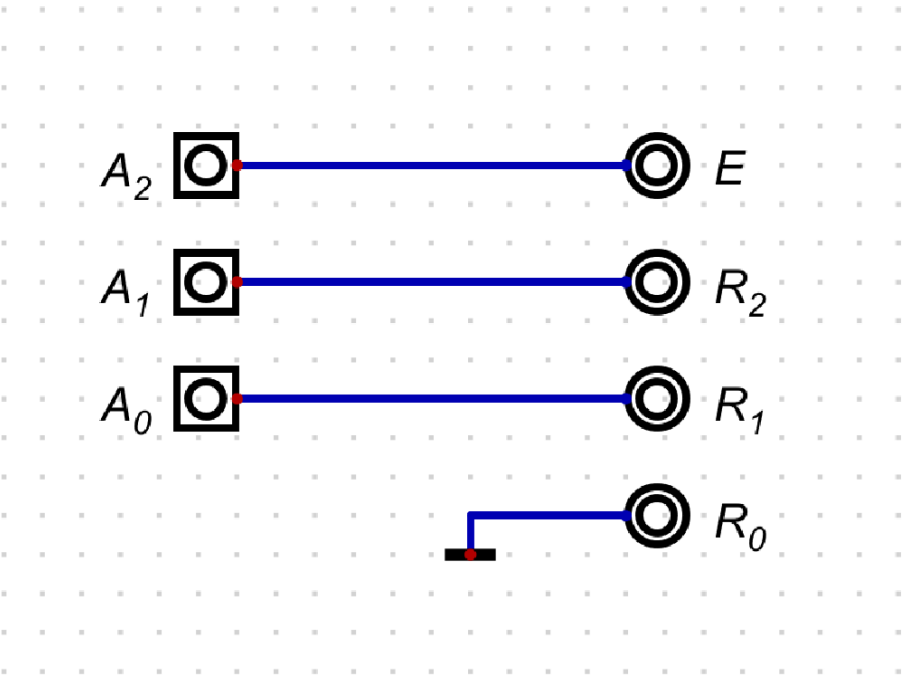
A – B:



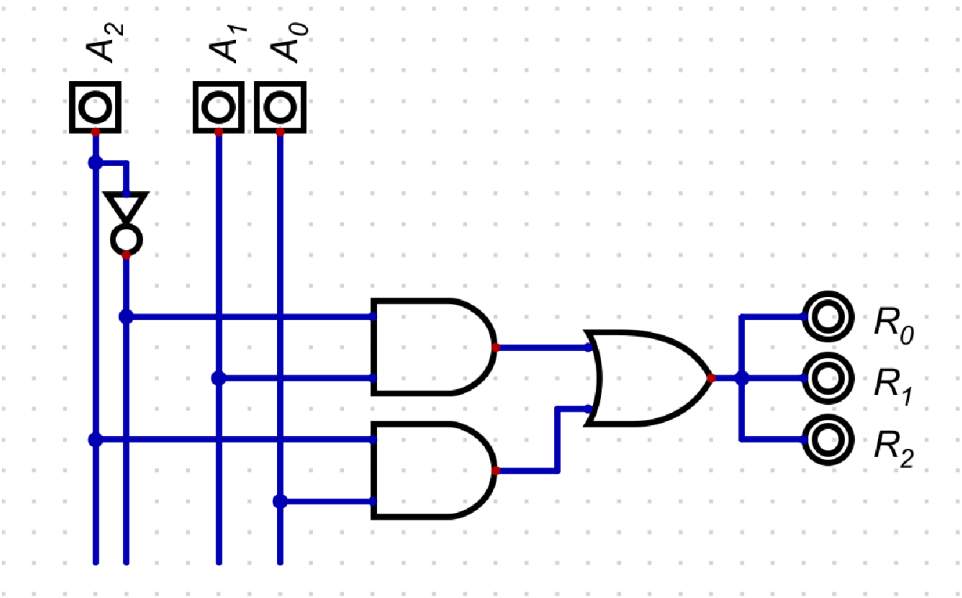
Comparador:



A x 2:

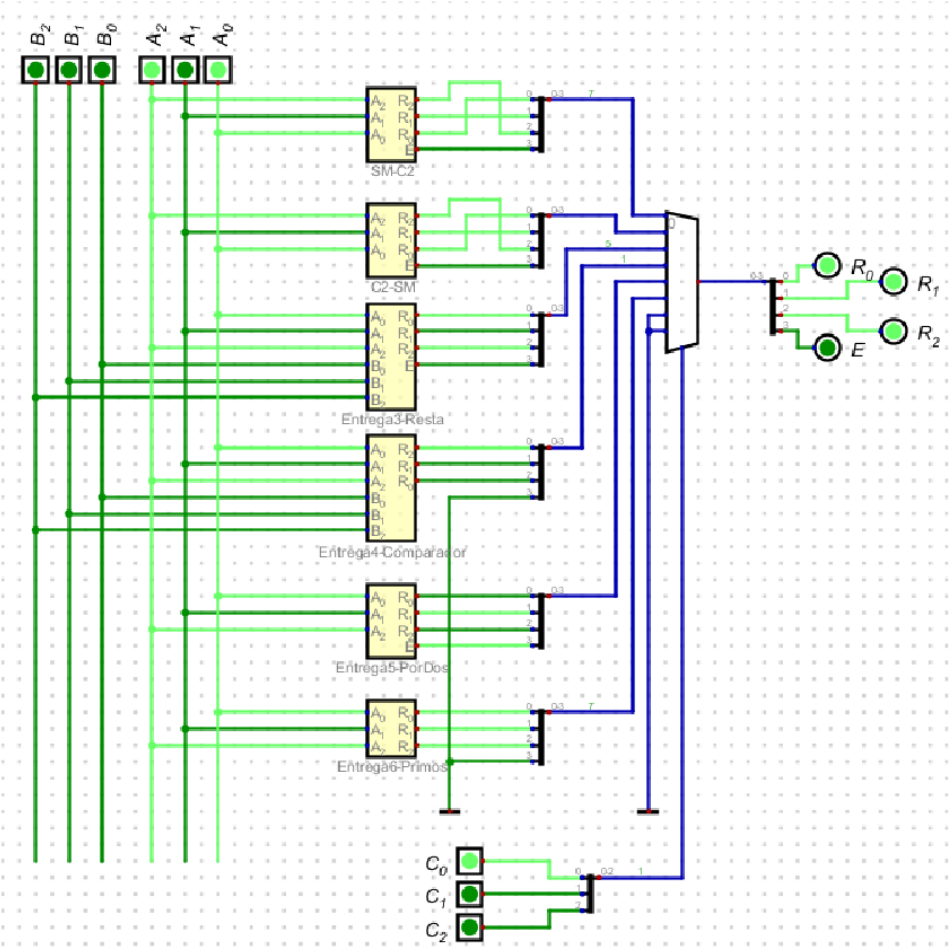


Número primo:

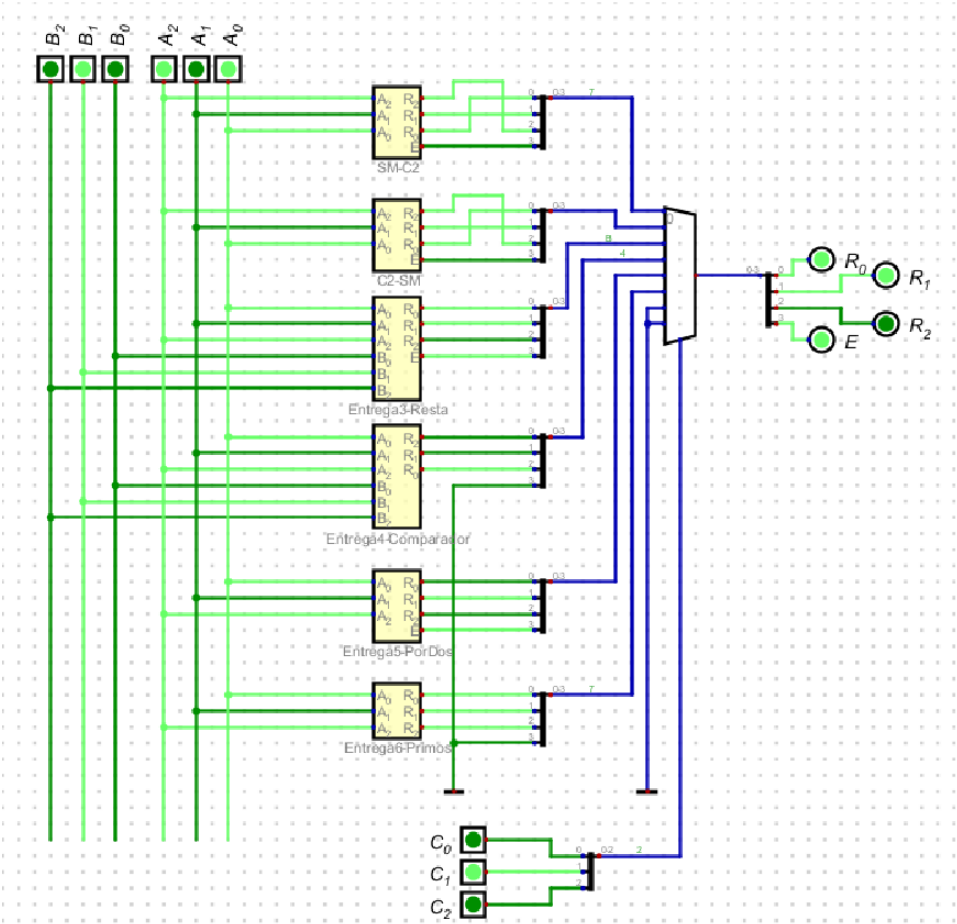


JUEGO DE PRUEBAS (POR PARTES Y PARA EL CIRCUITO COMPLETO):

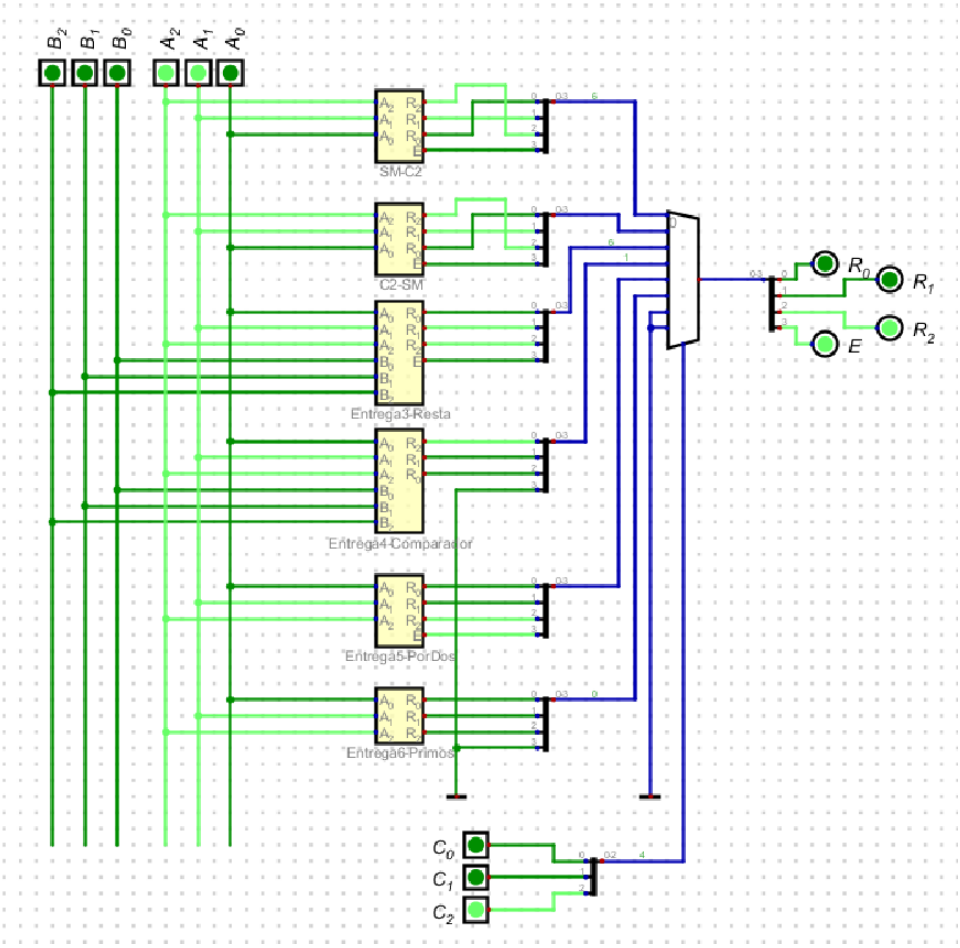
Circuito completo:



En la primera prueba del juego completo se ve como con los dígitos de A 101, B 000 y C 001 resulta en 111 y E 0, lo cual es correcto con la operación indicada de transformar de C2 a Signo y magnitud. Esto es visto en que A en decimal representa el –3, al igual que el resultado. Se ve también que el bit E es 0 porque no interfiere en el proceso.

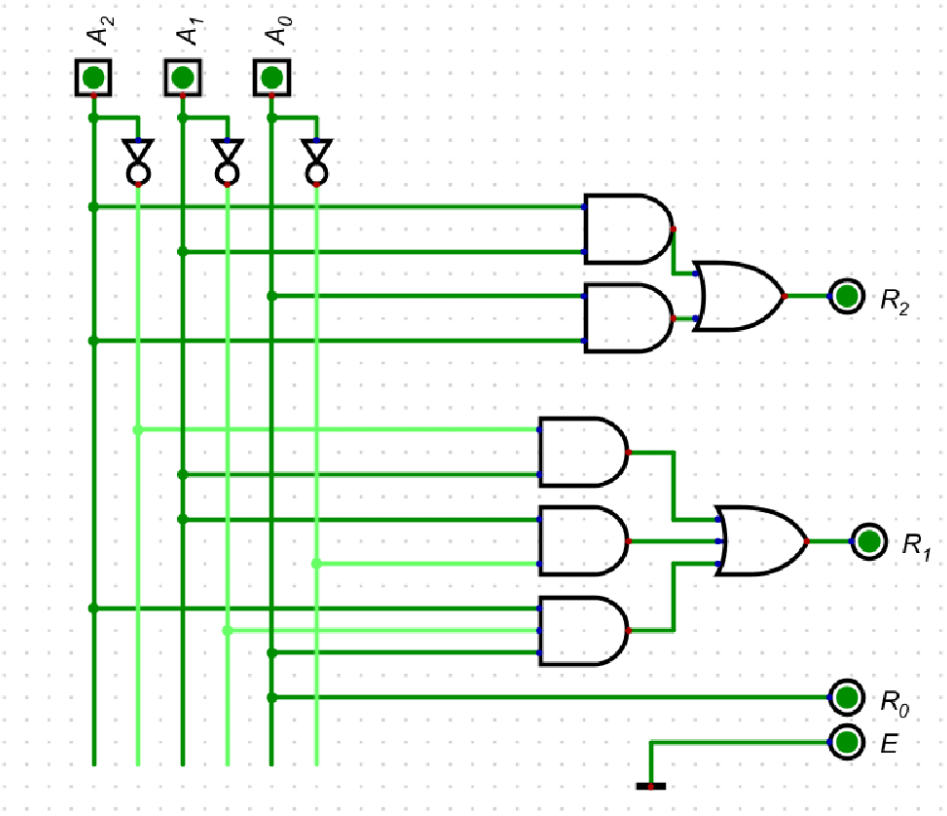


En la segunda prueba del juego completo se ve como con los dígitos de A 101, B 010 y C 010 resulta en 011 y E 1, lo cual es correcto con la operación indicada de restar A - B. Esto es visto en que A en decimal representa el –3 y B el 2 por tanto el resultado 5, pero como no entra en el rango del C2 el valor no tiene sentido. Entra en juego el bit E con 1 porque indica que hay overflow.

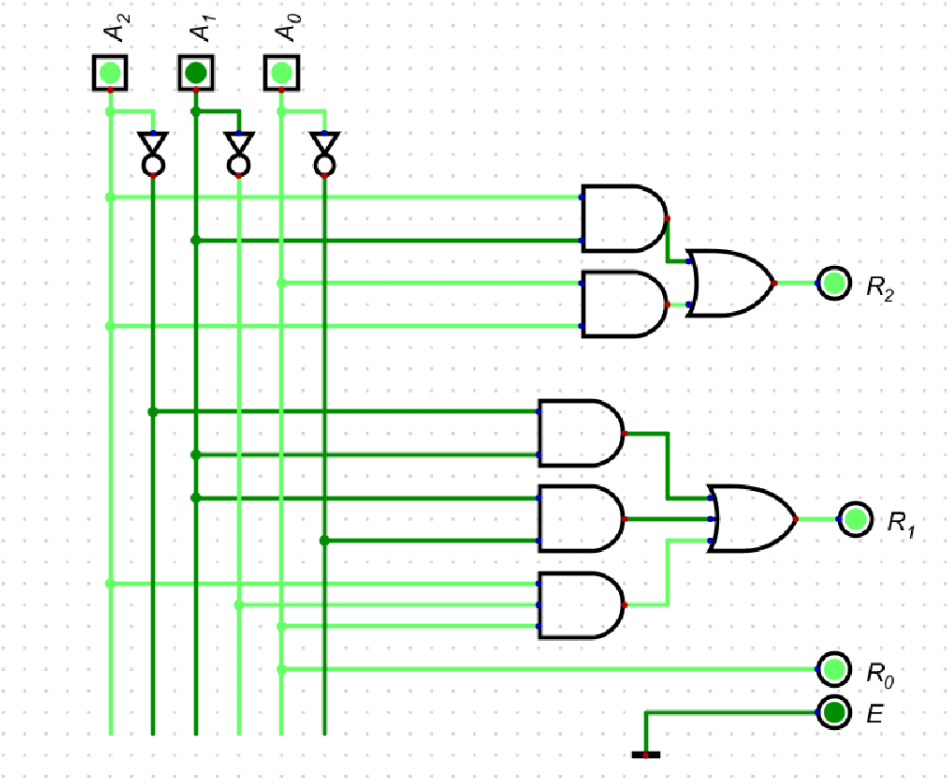


En la tercera prueba del juego completo se ve como con los dígitos de A 110, B 000 y C 100 resulta en 100 y E 1, lo cual es correcto con la operación indicada de multiplicar A por 2. Esto es visto en que A en decimal representa el 6 y el resultado 12 teniendo en cuenta el bit E para evitar el overflow. El bit E ayuda convirtiéndose en el más significativo, valiendo 1 en este caso.

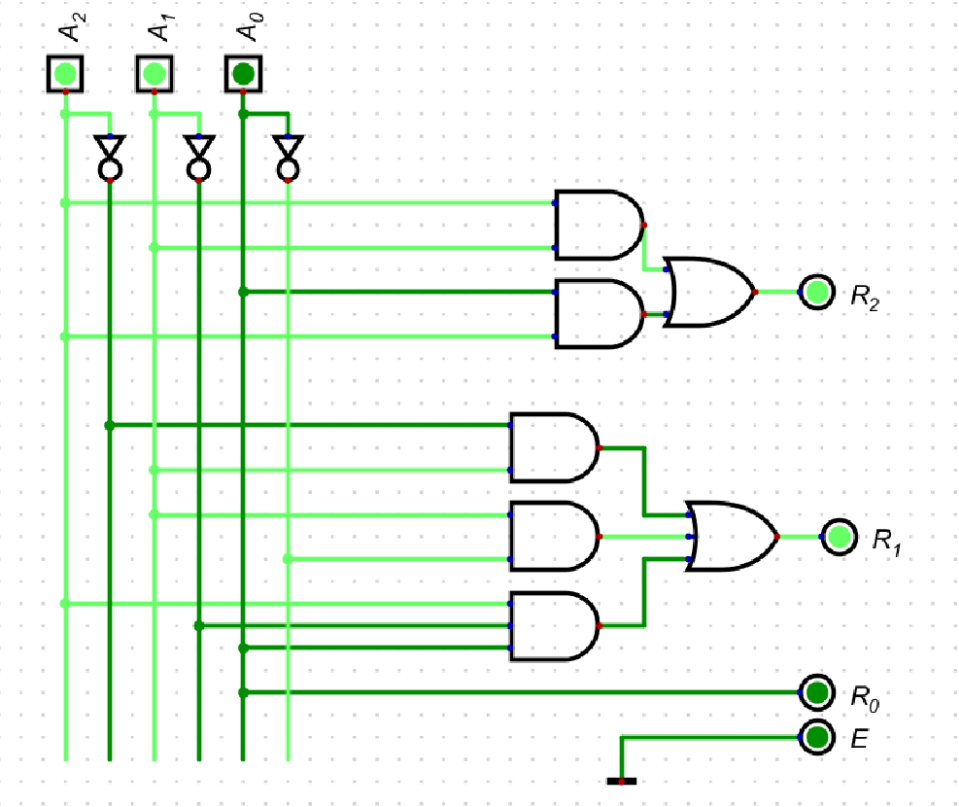
Transformación de Signo y Magnitud a Complemento a 2:



En la primera prueba del primer módulo se ve como con los dígitos 000 resulta en 000, lo cual es correcto como operación de Signo y magnitud a Complemento de 2. Se ve también que el bit E es 0 porque no afecta en nada.

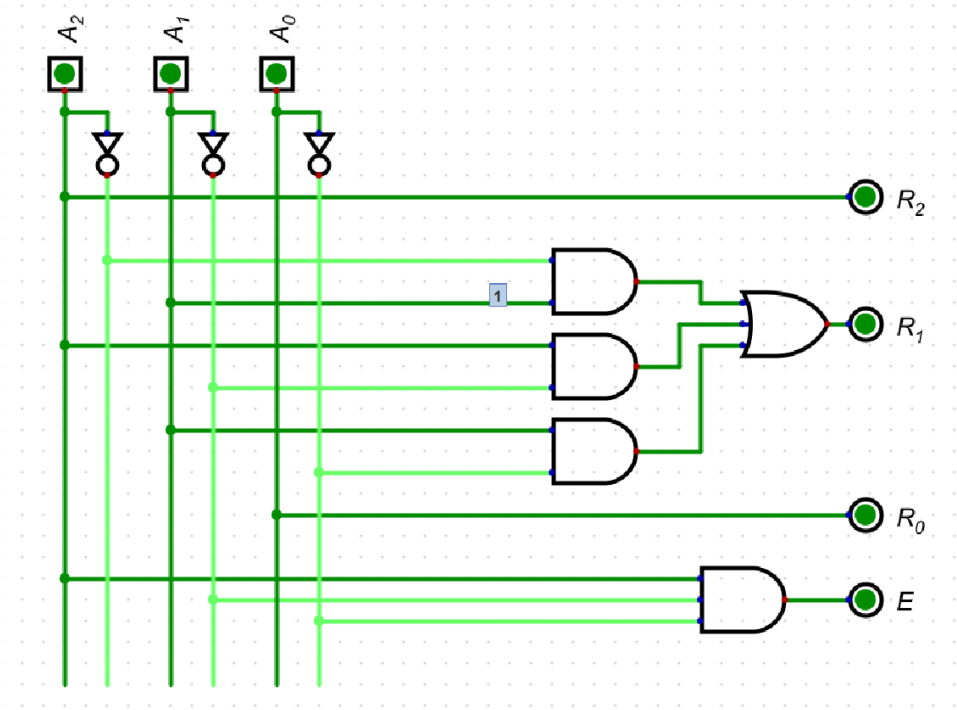


En la segunda prueba del primer módulo se ve como con los dígitos 101 resulta en 111, lo cual también es correcto como operación de Signo y magnitud a Complemento de 2, ya que en decimal, ambos equivalen al -1. Se ve también que el bit E es 0 porque no afecta en nada.

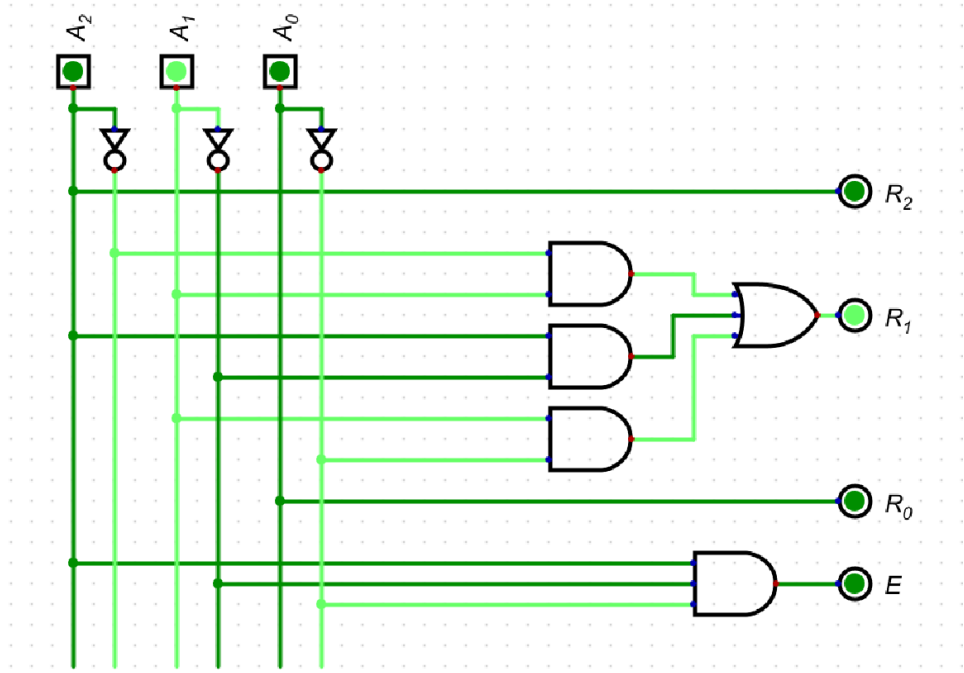


En la tercera prueba del mismo módulo se ve como con los dígitos 110 resulta en 110, lo cual sigue siendo correcto de Signo y magnitud a Complemento de 2 porque visto en decimal el valor de ambos es -2. Se ve también que el bit E es 0 porque no afecta en nada.

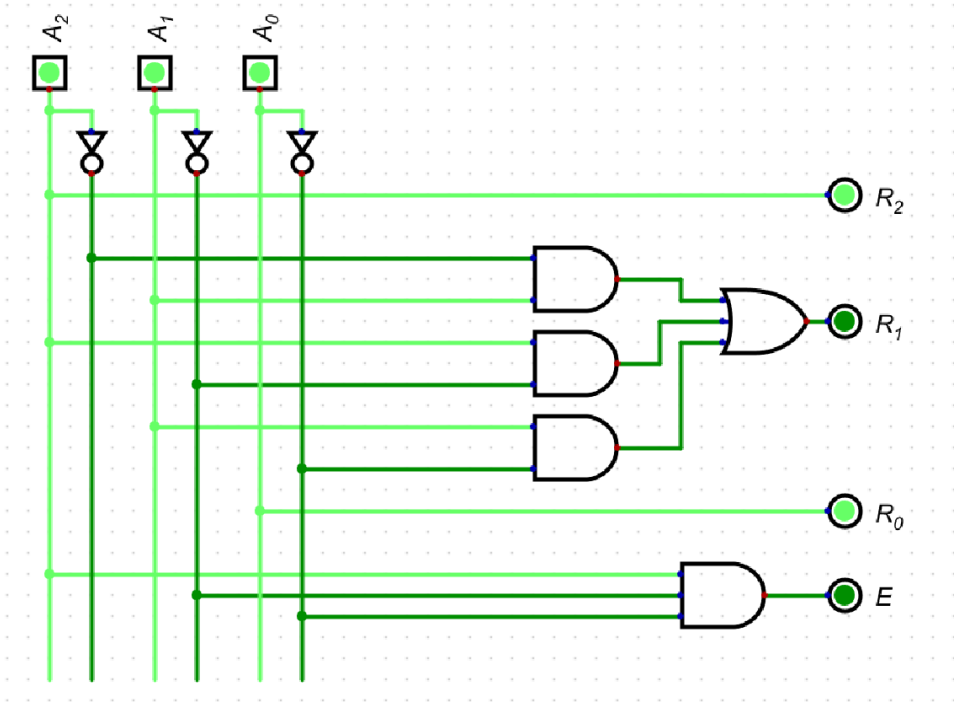
Transformación de Complemento a 2 a Signo y Magnitud:



En la primera prueba del primer módulo se ve como con los dígitos 000 resulta en 000, lo cual es correcto como operación de Complemento a 2 a Signo y magnitud. Se ve que el bit E es 0 también porque no afecta en nada a este módulo.

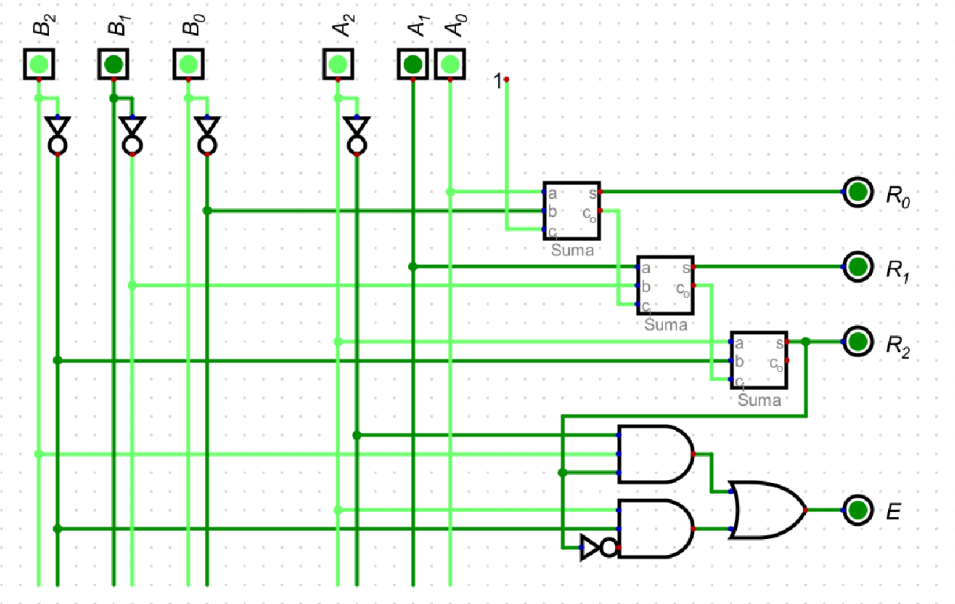


En la segunda prueba del módulo se ve como con los dígitos 010 resulta en 010, lo cual es correcto como operación de Complemento a 2 a Signo y magnitud sabido porque los positivos en ambos casos son iguales. El bit E sigue valiendo 0.

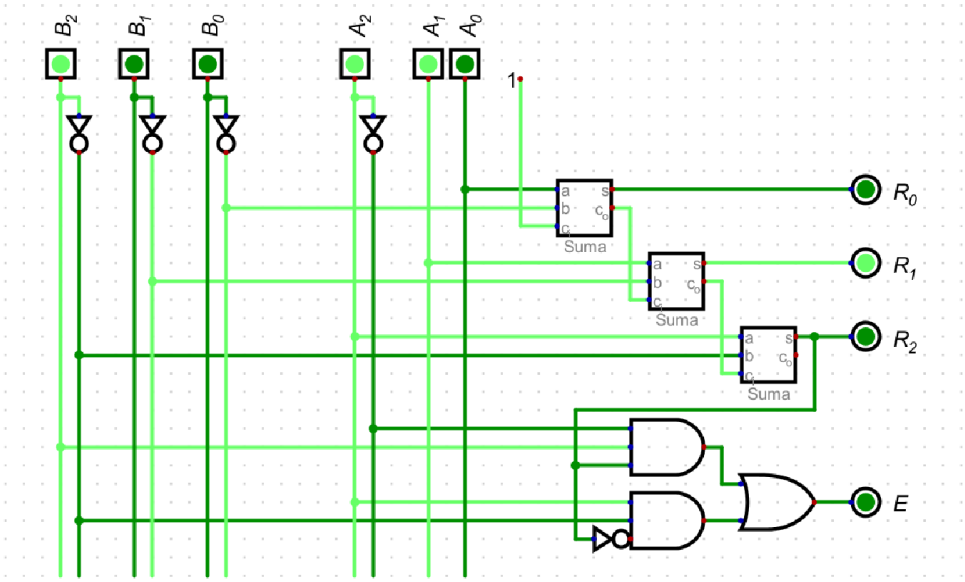


En la tercera prueba del módulo se ve como con los dígitos 111 resulta en 101, lo cual es correcto como operación de Complemento a 2 a Signo y magnitud visto porque en decimal el valor de ambos es -3. El bit E sigue valiendo 0.

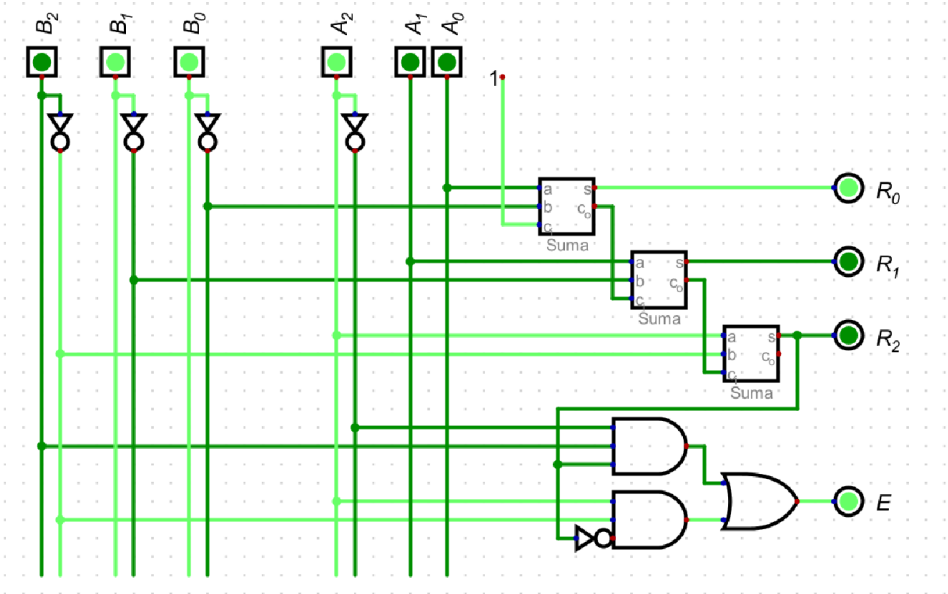
A – B:



En la primera prueba del tercer módulo se ve como con los dígitos de A,101 y de B,101 resultan en 000, lo cual es correcto como resta de A - B y se ve que es correcto porque se restan dos valores iguales y el resultado es 0. El bit E vale 0, indicando que no hay overflow, resultado correcto al sumar realmente un valor negativo a uno positivo (A-B -> A+(-B)).

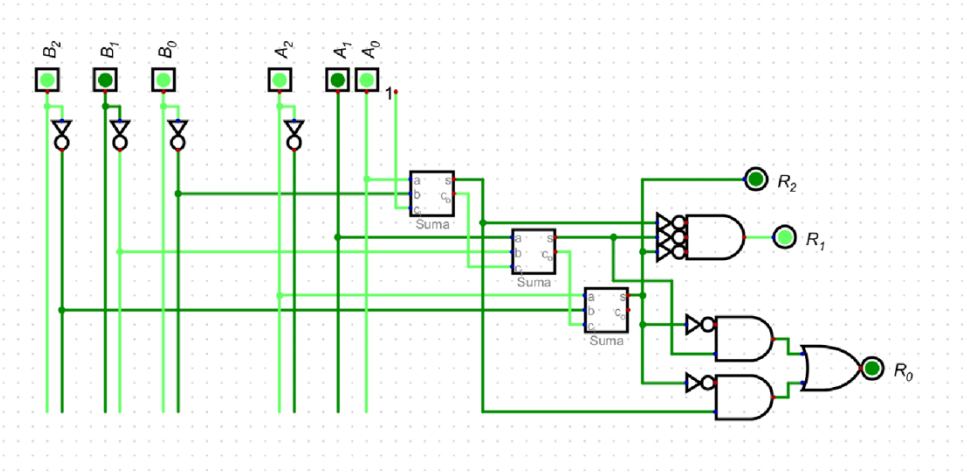


En la segunda prueba del módulo se ve como con los dígitos de A,110 y de B,100 resultan en 010, lo cual es correcto como resta de A - B y se ve que lo es porque se restan dos valores que en decimal equivalen a –2 y –4 con resultado 2. El bit E vale 0, indicando que no hay overflow, resultado correcto visto al saber que 2 no se sale del rango de Complemento a 2.

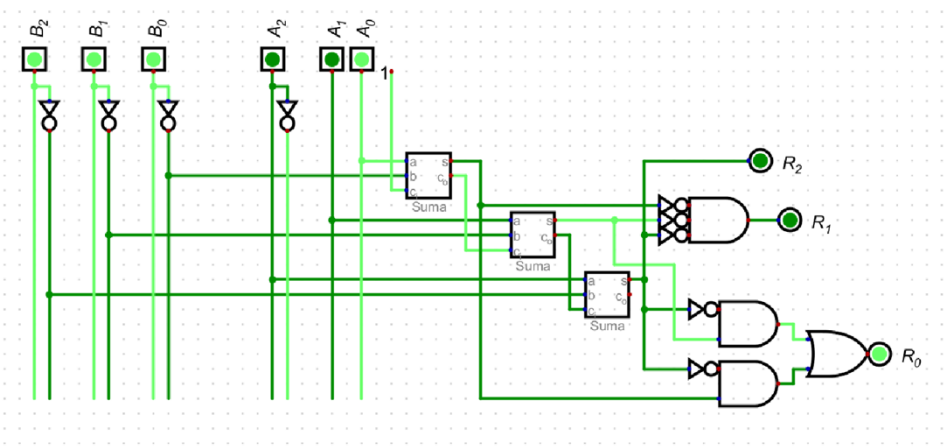


En la tercera prueba del módulo se ve como con los dígitos de A,100 y de B,011 resultan en 001, lo cual es correcto y se ve porque se restan dos valores que en decimal equivalen a –4 y 2 con resultado -6. El bit E vale 1, indicando que sí hay Overflow, resultado correcto al saber que -6 se sale del rango de Complemento a 2.

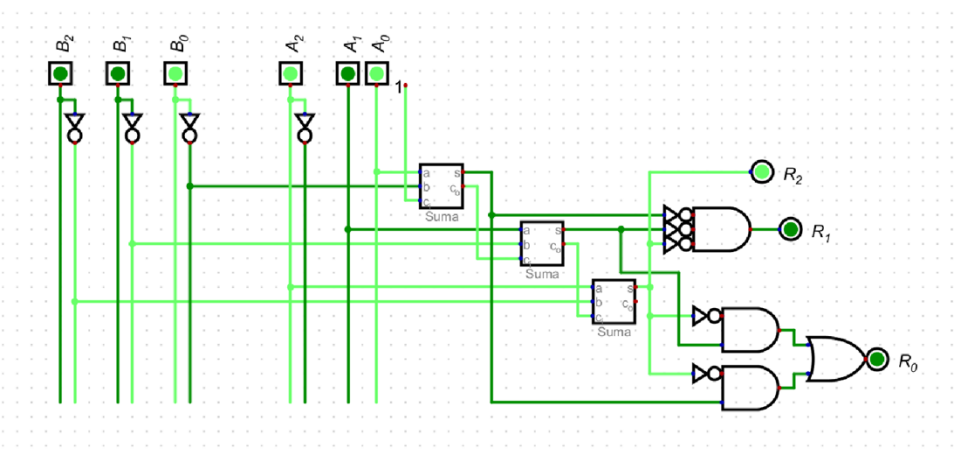
Comparador:



En la primera prueba del cuarto módulo se ve como con los dígitos de A,101 y de B,101 resultan en 010, lo cual es correcto ya que indica que son iguales, visto al pasar los valores a decimal siendo -3 en ambos casos. No hay bit E porque no era necesario en el módulo.

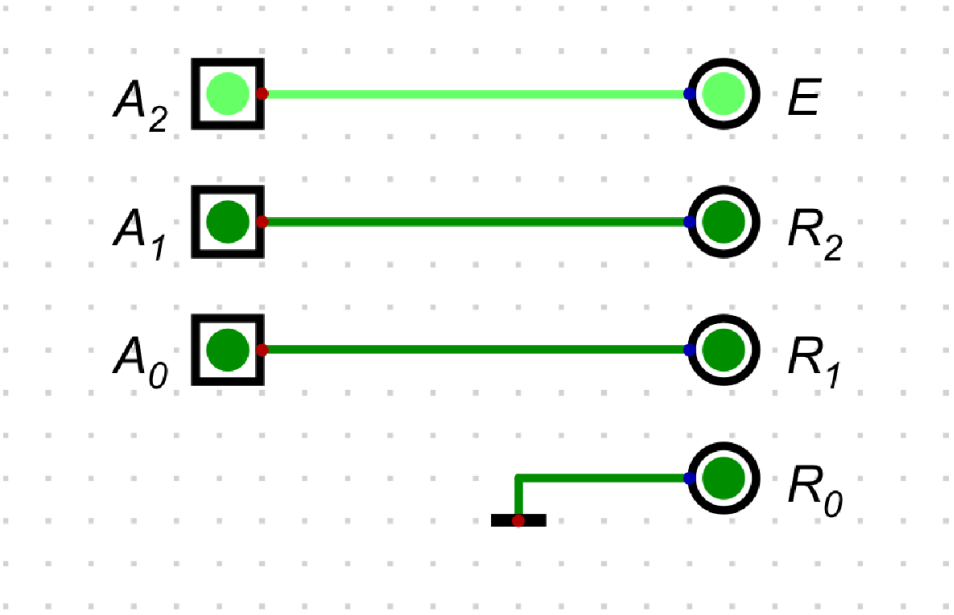


En la segunda prueba del módulo se ve como con los dígitos de A,001 y de B,111 resultan en 001, lo cual es correcto ya que indica que A es mayor a B, visto al pasar los valores a decimal siendo A 1 y B -1. No hay bit E porque no ser necesario.

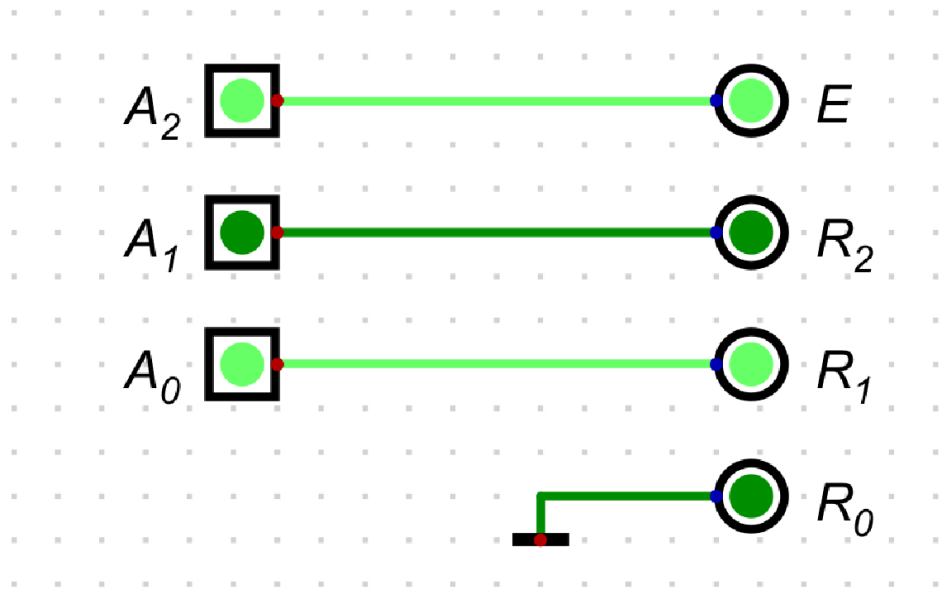


En la tercera prueba del cuarto módulo se ve como con los dígitos de A,101 y de B,001 resultan en 100, lo que es correcto ya que indica que A es menor a B, visto al pasar los valores a decimal siendo A –3 y B 1. No hay bit E por no ser necesario.

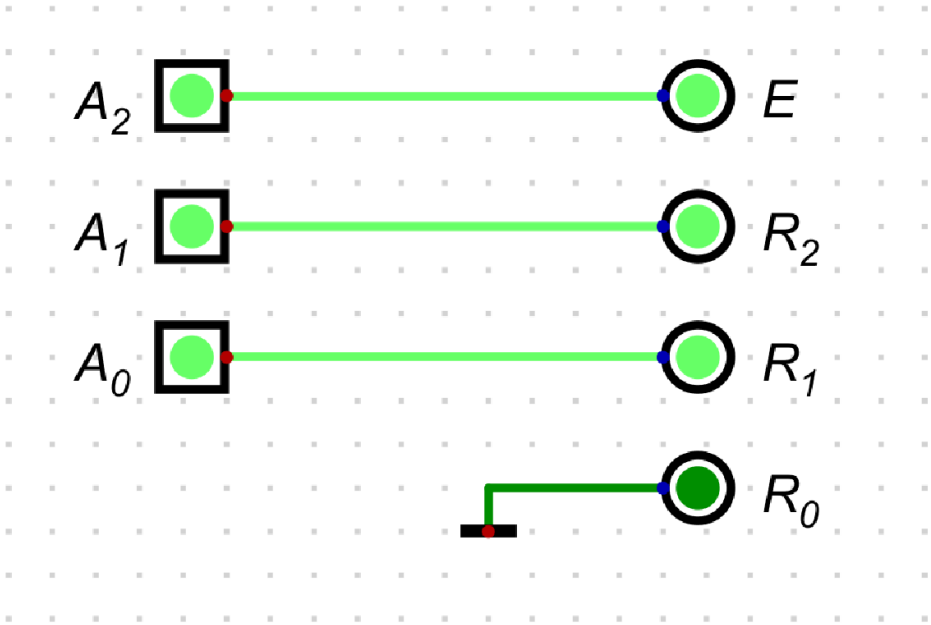
A x 2:



En la primera prueba del quinto módulo se ve como los dígitos 100 resultan en 1000, lo que es correcto ya que indica que el valor del doble de A que es 4, pasado a decimal es 8. El bit E se usa como nuevo bit más significativo para evitar el Overflow, que en este caso vale 1.

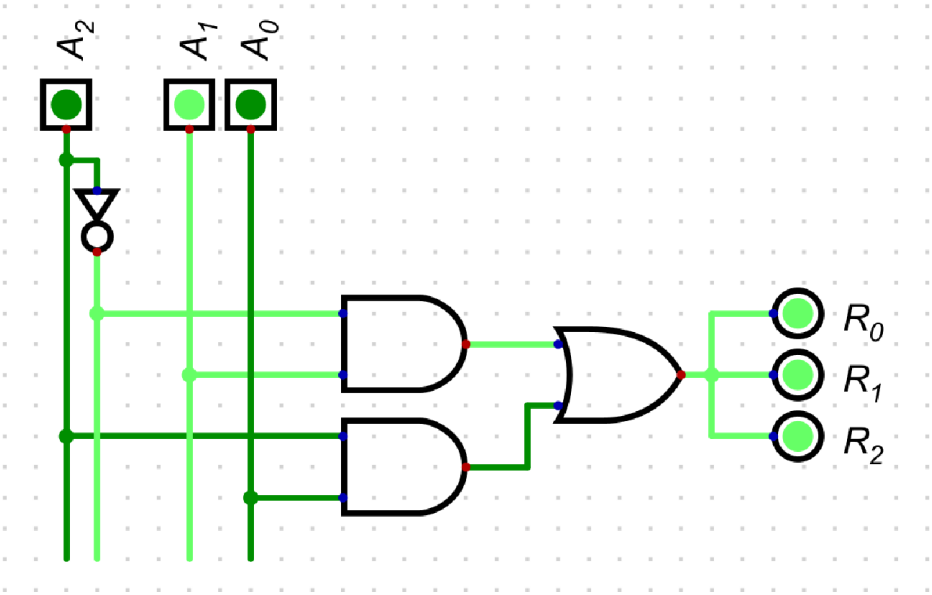


En la segunda prueba del módulo se ve como los dígitos 101 resultan en 1010, lo que es correcto ya que indica que el valor del doble de A que es 5, pasado a decimal es 10. El bit E se usa como nuevo bit más significativo para evitar el Overflow, que en este caso vale 1.

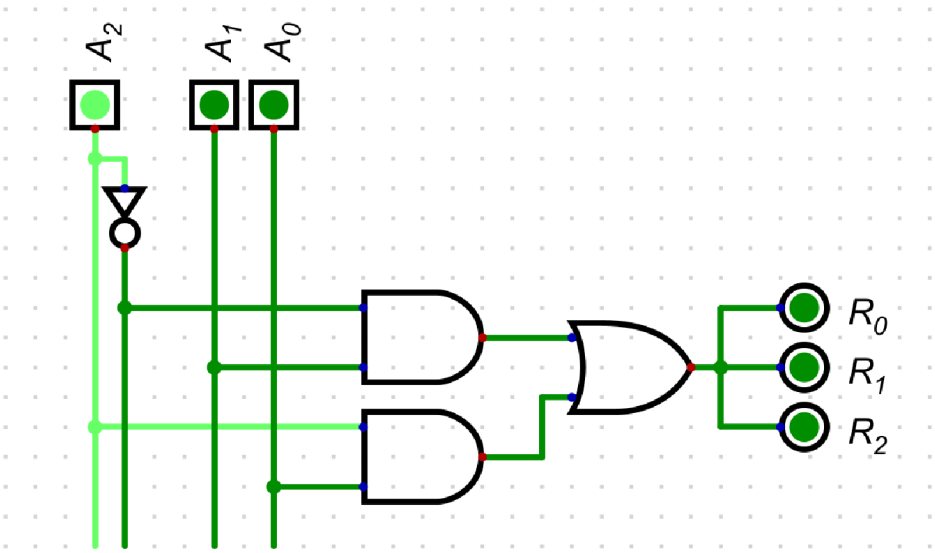


En la tercera prueba del quinto módulo se ve como los dígitos 111 resultan en 1110, lo que es correcto ya que indica que el valor del doble de A que es 7, pasado a decimal es 14. El bit E se usa como nuevo bit más significativo para evitar el overflow, que en este caso vale 1.

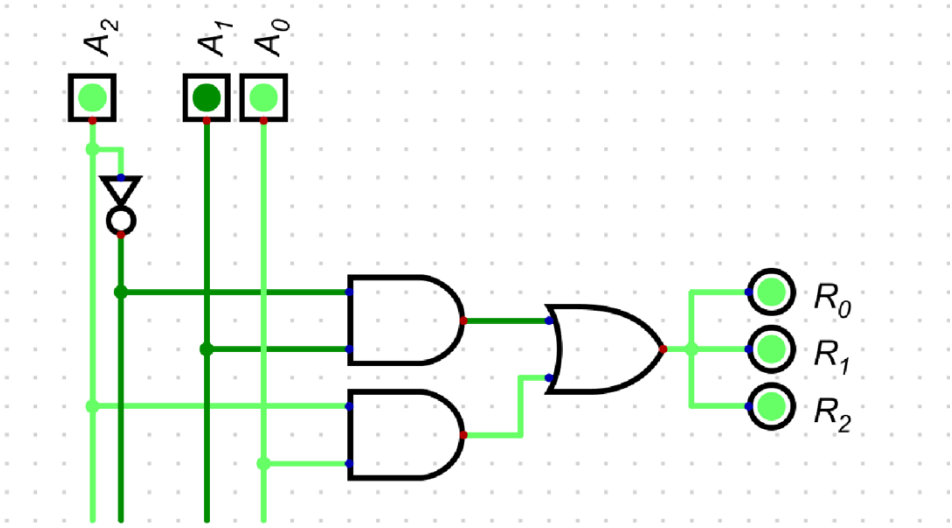
Número primo:



En la primera prueba del sexto módulo se ve como los dígitos 010 resultan en 111, lo que es correcto ya que indica que el valor de A es primo, viéndose en decimal que es 2. No hay bit E porque no era necesario en el módulo.



En la segunda prueba del módulo se ve como los dígitos 100 resultan en 000, lo que es correcto ya que indica que el valor de A no es primo, viéndose en decimal que es 4. No hay bit E por no ser necesario.



En la tercera prueba del módulo se ve como los dígitos 101 resultan en 111, lo que es correcto ya que indica que el valor de A es primo, viéndose en decimal que es 5. No hay bit E por no ser necesario.

CONCLUSIÓN:

Tras haber realizado la práctica entera, hemos podido ver las formas de aplicar los conocimientos y bases obtenidas en las clases anteriores, tanto de representación de valores e información digital con los tipos de Complemento a 2, Signo y magnitud y Binario natural sin signo, como de lógica secuencial para organizar los circuitos finales.

Ha sido bastante útil para llegar a entender este tipo de conceptos y obtener experiencia aplicada con la herramienta Digital, y así ver los resultados, tanto correctos como erróneos de forma directa. Es por esto también que nos hemos podido dedicar a la lógica y desarrollo de cada circuito por separado para juntarlo al final.

Respecto a los apartados o módulos realizados, concluimos en que la resta de A – B y la comparación de ambos han sido más costosos ya que no conllevaban el tiempo de realizar un Mapa de Karnaugh para cada salida, sino más bien entender el temario con lógica de la suma y los resultados con cálculos.

Para finalizar, destacamos la importancia de la lógica de los módulos y de la ALU general y los conocimientos teóricos y lógicos previos para así llegar a la construcción final de la práctica y su buen funcionamiento.