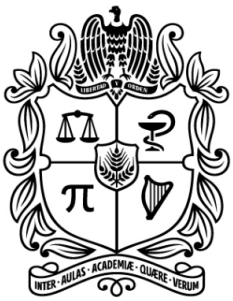

Sumador de 4 Bits usando *NAND*

Rojas Hernandez, Camilo Andrés

Documento: 1.007.751.072

Correo Institucional: carojashe@Unal.edu.co

25 de mayo de 2022

Resumen: El sumador de 4 bits es una herramienta muy útil para realizar cálculos aritméticos básicos en cuestión de microsegundos, la desventaja que tiene este es que siempre nos encontraremos con limitantes de bits a mostrar, en este caso 4 bits, pero el proceso de su desarrollo el ingenio que tiene este detrás, hace que este simple circuito lógico de lo mejor del ingenio humano. Veremos su implementación en un software especializado en circuitos lógicos.

Palabras clave: LOGISIM, NAND, Puerta Universal

1. Introducción

A lo largo de la historia el hombre ha buscado la necesidad de poder simplificar y facilitar el desarrollos de algunos cálculos, los cuales son base fundamental para la construcción y planeación; tras cientos de años de experimentación el hombre desarrolló las máquinas, las cuales han evolucionado con el pasar de los años, llegando a realizar operaciones aritméticas en cuestión de micro segundos, aquí entra nuestro sumador de 4 bits, el cual fue desarrollado en un comienzo para realizar el cálculo para el cual fue designado, sumar bits. En este documento usted encontrará cuál fue el proceso lógico para poder desarrollar este circuito, comenzando desde cómo sumar, la tabla de verdad, hasta su implementación en un software especializado en circuitos lógicos **LOGISIM**.

2. Metodología

El trabajo se desarrollo con de una manera estructuralmente correcta, la cual consistió primero en una vista de qué se buscaba con el trabajo, trabajando primero en cómo funciona una suma en binario, para luego, poder desarrollar una tabla de verdad, su mapa K respectivo, circuito lógico, entre otros recursos que son soporte para el procedimiento desarrollado, así como la investigación y búsqueda del funcionamiento del software **LOGISIM**.

3. Desarrollo y Resultados

Antes de entrar de lleno recordemos cómo se desarrollan las sumas en binario, la cual comenzaremos en primera instancia con un bit.

$$\begin{array}{cc} & 1 \\ + & 1 \\ \hline 1 & 0 \end{array} \qquad Acarreo$$

Sumamos con normalidad, lo cual tiene como resultado que, vamos a contar y llevamos un acarreo dependiendo del caso, en este caso tenemos ese cacarreo, por lo tanto en la siguiente casilla colocamos 1.

Ahora, al analizaremos la tabla de verdad para cuando hay que tener acarreo y cuánto es el resultado de la suma, por lo tanto organizamos las siguientes tablas.

A	B	Suma	Acarreo
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Analizando la tabla de verdad para la suma y el acarreo podemos generar dos K-Mapas, los cuales son los siguientes:

Formanos los grupos respectivos para la suma obtenemos la siguiente expresión

$$f = AB' + A'B$$

SUMA

A\B	0	1
0	0	1
1	1	0

pero como en este caso solo utilizaremos *NAND* entonces simplificaremos nuestra expresión hasta estar utilizando solo el operador *NAND*, el cual se define por la función $NAND = B' + A' = \overline{AB} = A \times B$.

Antes de continuar hay que tener en cuenta que NAND es una puerta Universal, por lo cual nos es posible el desarrollo de todos estos circuitos lógicos que se verán a continuación.

$$\begin{aligned} f &= AB' + A'B \\ &= AA' + AB' + A'B + BB' \\ &= A(A' + B') + B(A' + B') \\ &= \overline{\overline{A(A' + B') + B(A' + B')}} \\ &= \overline{\overline{A(A' + B')B(A' + B')}} \\ &= \overline{\overline{AB'} \cdot \overline{BA'}} \end{aligned}$$

Pero antes de continuar, miremos por un momento esta demostración $A\overline{AB} = AB'$, la cual será de ayuda para el último paso de nuestra demostración:

$$\begin{aligned} A\overline{AB} &= A(A' + B') \\ &= AA' + AB' \\ A\overline{AB} &= AB' \end{aligned}$$

Ahora continuando con la demostración, entonces tenemos que:

$$\begin{aligned} f &= AB' + A'B \\ &\vdots \\ &= \overline{\overline{AB'} \cdot \overline{BA'}} \\ &= \overline{\overline{A\overline{AB}} \cdot \overline{B\overline{AB}}} \\ &= \overline{A\overline{AB}} \times \overline{B\overline{AB}} \\ &= [A \times \overline{AB}] \times [B \times \overline{AB}] \\ &= [A \times (A \times B)] \times [B \times (A \times B)] \end{aligned}$$

De esta manera podemos organizar nuestra función en términos de la puerta lógica *NAND*.

A continuación trabajaremos con el acarreo, el cual en este caso es simple, ya que solo hay acarreo cuando cuando hay una suma de dos bits ambos con el valor de 1, así como ilustra el siguiente K-Mapa.

ACARREO

A\B	0	1
0	0	0
1	0	1

Tenemos como resultado la siguiente función

$$f = AB$$

pero usando nuevamente propiedades de la lógica obtenemos el siguiente resultado.

$$\begin{aligned} f &= \overline{\overline{AB}} \\ &= \overline{\overline{AB} \cdot \overline{AB}} \\ f &= [A \times B] \times [A \times B] \end{aligned}$$

Entonces aprovechando que tenemos las funciones que necesitabamos, una que lleve la suma y otra que lleve el acarreo, y además, ambas escrita de la forma en que solo se utilicen los operadores *NAND*, tan solo nos queda organizarlo dentro del Software **LOGISIM**, lo cual resulta en el siguiente diagrama:

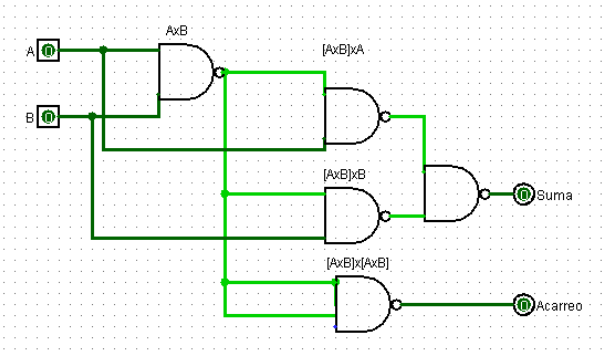

Figura 1: Sumador de 1 bit con acarreo **LOGISIM**.

Perfecto, ahora sabemos cómo funciona la suma en un bit, pero, este no es el resultado que esperamos queremos sumar 4 bits para obtener un resultado en 5 bits, pero, ahora debemos de centrarno en cómo funciona una suma binaria de **n** dígitos, a continuación veremos cómo funciona.

$$\begin{array}{rll} & 110 & Acarreo \\ & 011 & \\ + & 011 & \\ \hline & 110 & Resultado + Acarreo \end{array}$$

Ahora que nos damos cuenta, para poder hacer una suma de **n** bits, necesitamos primero un bit más, el cual será el acarreo, el cual siempre iniciará en cero (0), ahora bien, organicemos las tablas respectivas, para el resultado y el acarreo que lleva toda la suma, ya que también habrá un acarreo para el siguiente bit.

A	B	C	SUMA	ACARREO
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

Ahora si nos dedicamos a ver cual es el K-mapa que genera y la función que tiene tanto la suma como el acarreo, nos encontraremos con lo siguiente:

SUMA

A**BC	**00**	**01**	**11**	**10**
0	0	1	0	1
1	1	0	1	0

Lo cual nos genera la siguiente función:

$$f = ABC + A'B'C + A'BC' + AB'C'$$

Ahora, como bien sabemos, necesitamos tenerlo en términos de puertas *NAND*, por lo tanto procedemos utilizar propiedades de la logica para su desarrollo.

$$\begin{aligned}
f &= ABC + A'B'C + A'BC' + AB'C' \\
&= (AA' + AB + A'B' + BB')C + (A'B + AB')C' \\
&= [(A + B')(A' + B)]C + (A'B + AB')C' \\
&= [(A'' + B')(A' + B'')]C + (A'B + AB')C' \\
&= [\overline{A'B} \cdot \overline{AB'}]C + (A'B + AB')C' \\
&= [\overline{A'B + AB'}]C + (A'B + AB')C' \\
&= [\underbrace{\overline{A'B + AB'}}_{A'}]\underbrace{C}_{B} + (\underbrace{A'B + AB'}_{A})\underbrace{C'}_{B'} \\
&= [A \times (A \times B)] \times [B \times (A \times B)] \\
&\quad Considerando\ A = [A \times (A \times B)] \times [B \times (A \times B)] \\
&\quad Considerando\ B = C
\end{aligned}$$

Para facilitar la comprensión se dejará de esa manera la función, ahora veamos qué pasa cuando trabajamos con el acarreo.

Algo que se va a tener en cuenta es que debido a la complejidad de la escritura para convertir la función en términos de *NAND*, se pasarán algunos pasos, así como se mostró en la demostración anterior.

Si a continuación analizamos el acarreo y el K-mapa que este forma, nos encontraremos con el siguiente.

ACARREO

A**BC	**00**	**01**	**11**	**10**
0	0	0	1	0
1	0	1	1	1

Al formarse los grupos, entonces quedará la función

$$f = BC + AC + AB$$

, la cual para poder operar con las puertas lógicas *NAND* nuestra expresión queda reducida a la siguiente:

$$\begin{aligned}
f &= BC + AC + AB \\
&= AB + C(A + B) \\
&= AB + C\overline{\overline{(A + B)}} \\
&= AB + C(\overline{\overline{AB}}) \\
&= \overline{\overline{AB + C[(A \times B)(A \times B)]}} \\
&= \vdots \\
&= ([[([A \times B] \times B) \times ([A \times B] \times A)] \times C] \times C) \times [A \times B
\end{aligned}$$

Por lo cual, a la hora de implementarlo en **LOGISIM**, obtenemos el siguiente circuito.

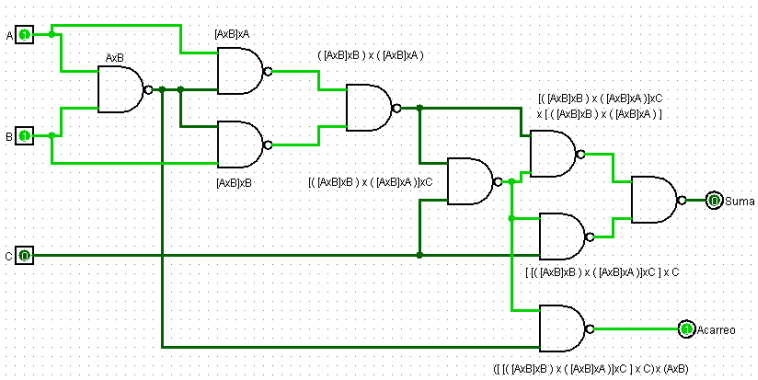

Figura 2: Sumador completo de 1 bit **LOGISIM**.

Por lo cual, ahora ya teniendo sumadores de un sol bit, los cuales pueden dar un acarreo de salida, y recibir un acarreo de entrada, podemos simular una suma

tal como la hariamos a mano, por lo cual, en nuestro caso que será de 4 bits, necesitaremos un total de 4 de estos circuitos, cada uno recibirá el enésimo bit, comenzando desde el 1 y terminando en el 8, así como se ilustrará a continuación.

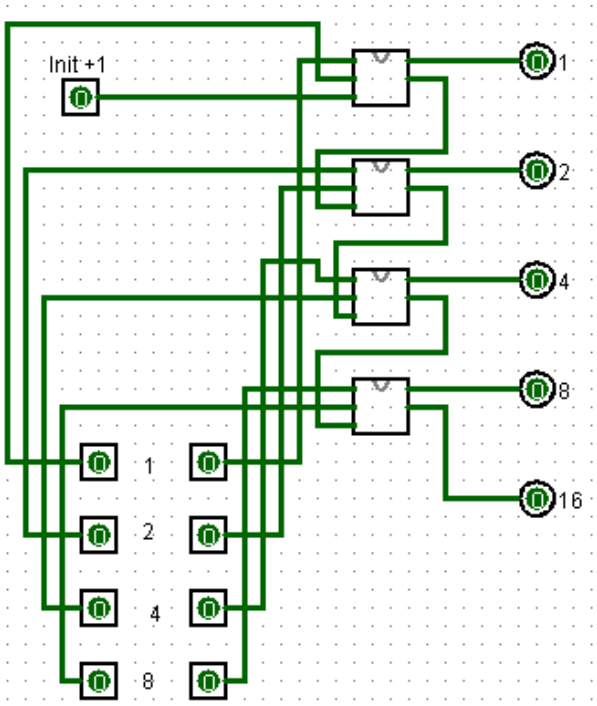

Figura 3: Sumador de 4 bits **LOGISIM**.

Este sumador de 4 bits no traduce automáticamente a decimal, pero como estamos tratando con un sistema de numeración binario, basta con pasar la base de binario a decimal.

4. Conclusión

La puerta NAND es universal, lo cual nos permitió realizar el circuito anterior, reemplazando el clásico XOR para la realización de las operaciones sumas, los AND y los OR, pero a consecuencia de esto, nuestros circuitos tienden a ser muy largos, y si nos detenemos a ver el aspecto teórico incluso desde mi punto de vista pierden un poco el sentido, se vuelven largos y redundantes, lo cual hace que la elaboración de este sistema sea algo más costoso, esto, debido a la gran cantidad de puertas lógicas NAND que se utilizaron.