# Laboratorio 2: Lógica Combinacional y Aritmética

Ricardo Borbón Mena
Felipe Jiménez Ulate
Carlos Rodríguez Segura
CE3201-Taller de Diseño Digital
Instituto Tecnológico de Costa Rica

***Abstract*—The present laboratory used SystemVerilog to recreate a calculator using a parameterizable n-bit ALU structure, where later it will be possible to see how its implementation affects the propagation times, and the frequency at which it can be operate, thus provoking optimal ways of making this type of structures and some that present more drawbacks than others.**

***Palabras clave*—Unidad aritmética-lógica, circuitos combinacionales, operaciones lógicas, ruta crítica, tiempo de propagación y de contaminación**

## I. Introducción

A la hora de pensar en el procesamiento de datos de, por ejemplo, una computadora en la época actual, se pueden poseer diferentes preguntas sobre cómo es que esta los realiza. La respuesta a esta pregunta presenta muchos apartados importantes para estar completa, sin embargo a lo largo de este laboratorio podrá verse el funcionamiento de una de estas partes esenciales, una unidad lógico-aritmética.
Las sumas, restas, multiplicaciones y divisiones son operaciones presentes en todo tipo de procesamiento de datos, por esta razón es importante la ALU (véase su representación en la figura 1), la cual es un circuito digital encargado de realizar operaciones aritméticas y lógicas, es parte de la unidad central de procesamiento. Esta funciona de la siguiente manera [1]:

- La ALU posee una serie de redes de entrada y de salidas los cuales están conectados con los circuitos exteriores.
- Esta posee tres buses de datos paralelos, en los cuales se tienen dos operandos, llamados A y B y un resultado, que se toma como Y. También existe otro bus de datos el cual es el código de operación que define la operación aritmética que se va a realizar.
- Esta posee tres buses de datos paralelos, en los cuales se tienen dos operandos, llamados A y B y un resultado, que se toma como Y. También existe otro bus de datos el cual es el código de operación que define la operación aritmética que se va a realizar.

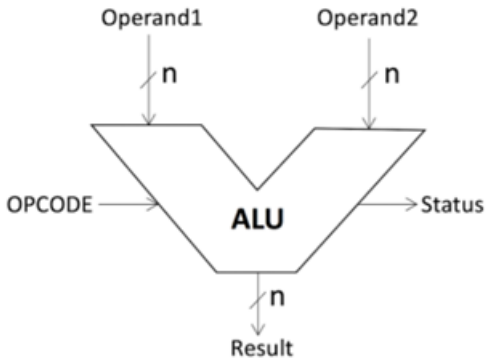


Fig. 1. Diagrama de una unidad aritmética-lógica. Recuperado de: https://unigal.mx/alu-unidad-logica-aritmetica-definicion-funcion-y-mas/

Tras tener el diagrama de una ALU en la figura 1 vienen las preguntas de qué posee esta unidad por dentro, cuáles son algunas operaciones o características que posee, por esta razón se van a ilustrar algunos diagramas de circuitos lógicos de las operaciones que se presentan.
En la figura 2 se puede apreciar el circuito lógico perteneciente a una semisuma lógica

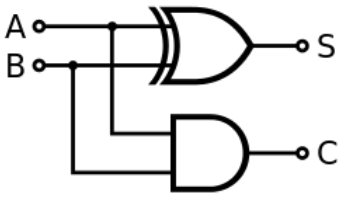


Fig. 2. Diagrama lógico de un semisumador. Recuperado de: https://es.wikipedia.org/wiki/Sumador

Y su tabla de verdad sería la de la tabla I

TABLE I
Tabla de verdad para 2.

| A | B | S | C |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

Teniendo este circuito se puede crear un full-adder, como se muestra en la figura 3

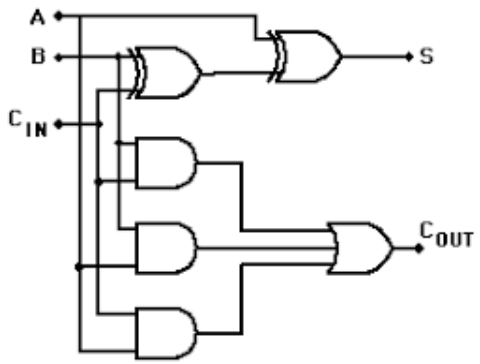


Fig. 3. Diagrama lógico de un sumador completo. Recuperado de: http://hyperphysics.phy-astr.gsu.edu/hbasees/Electronic/fulladd.html

La cual da como tabla de verdad la tabla II

TABLE II
TABLA DE VERDAD PARA 3.

| A | B | Cin | 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 |

Teniendo la suma falta otra operación básica, la resta, por esta razón a continuación se presenta el circuito lógico de la operación de un full-substractor en la figura 4

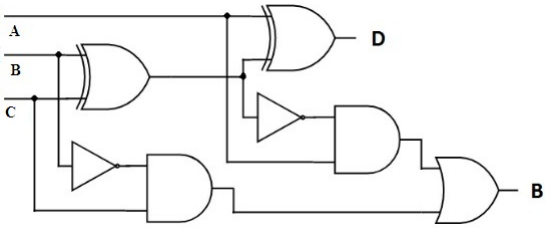


Fig. 4. Diagrama lógico de un restador completo. Recuperado de: https://es.fmuser.net/content/?13625.html

Y el resultado de la tabla de verdad sería el siguiente presente en la tabla III

TABLE III
TABLA DE VERDAD PARA 4.

| A | B | C | Diferencia | Presta |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

A partir de estas operaciones se pueden recrear otras que siguen siendo igual de esenciales para todo tipo de procesos, como lo son la multiplicación y la división, las cuales aplican un conjunto de sumas o de restas dependiendo el caso, estos serán presentados más adelante en el informe.

Así como se tienen estas operaciones aritméticas, también la unidad aritmético-lógica presenta otras operaciones como lo son el shift-left y shift-right, las cuales son operaciones que se encargan de mover un solo bit, ya sea hacia la izquierda o hacia la derecha, dependiendo de cuál se esté aplicando.

A su vez, una unidad aritmética-lógica posee diferentes salidas de estado, o banderas de estado, hay diferentes arquitecturas en las cuales se presentan algunas banderas más para diferentes indicadores, sin embargo, algunos ejemplos de estas serían[2]:
C: Carry, la operación realizada posee acarreo, esta flag se presenta en operaciones sin signo.
Z: Zero, todos los bits lógicos de Y son cero.
N: Negative, el resultado de la operación es negativo.
V: Overflow, expresa que la operación ha excedido el rango de Y, se presenta en operaciones con signo.

A la hora de realizar un circuito combinacional en este se debe tomar en cuenta diferentes aspectos como lo son dos parámetros que generan un retraso en estos circuitos. El primero se llama tiempo de propagación, el cual se refiere al tiempo máximo desde que una entrada cambia hasta que la o las salidas alcanzan su valor final. El segundo se conoce como tiempo de contaminación, y este se describe como el tiempo mínimo desde que una entrada cambia hasta que una de las salidas empieza a cambiar su valor[3].
Otra consideración que se debe tener es la de la ruta crítica, la cual presenta el camino más largo desde que se da una entrada hasta que se tiene la salida, por lo tanto, esta presenta el mayor tiempo de espera que se puede tener. Su importancia puede apreciarse fácilmente en un ejemplo como los procesadores con pipeline, donde al dividir la ejecución en etapas, cada una de estas etapas va a aumentar su tiempo de propagación total conforme la ruta crítica esté menos optimizada, por esto va a causar un rendimiento bajo en el procesador y retrasos aumentados. Otro punto clave de su importancia, es el hecho de que al durar más en darse el tiempo de propagación en la ruta crítica va a disminuir la frecuencia máxima de operación de este circuito [3].

## II. MÉTODO

### A. *Problema 1*

Para el primer problema se tiene que replicar una calculadora parametrizable, en este caso con las operaciones aritméticas de la suma, resta y se escogió como tercera operación la división entera. Por lo tanto la ALU a crear en este problema se va a subdividir en estas tres operaciones.

1) Suma: para este caso, se va a dividir el problema en diferentes módulos, los cuales van a simplificar el problema. Se van a aplicar sumadores completos de 1 bit, los cuales van a ir aplicándose en cadena hasta que se complete la suma correspondiente, que en este caso es de n bits y puede variar. La lógica de esto es

que primeramente se toma el primer dígito del primer número, y se suma con el primero del segundo número, Para este funcionamiento se estará haciendo uso del diagrama lógico presentado en la figura 3 donde se puede apreciar el uso de compuertas como el XOR, con su tabla de verdad presente en la tabla IV, y de compuertas AND, las cuales dan un valor de 1 en caso de que ambas entradas sean un 1.

TABLE IV
TABLA DE VERDAD COMPUERTA XOR.

| A | B | Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Las compuertas AND se utilizarán solo para obtener el bit de acarreo, mientras que la compuerta XOR como es apreciable en el caso de que ambas entradas sean 1, este va a tener un resultado de 0, esto se debe a que aquí se estaría activando la entrada de carry para el siguiente sumador, indicando que lleva un número de acarreo que debe sumarle al siguiente dígito. Esto se va a repetir hasta la cantidad de bits que presente el número mediante un ciclo for. El valor de la suma total va a ser presentado en un siete segmentos, siguiendo los lineamientos de los mapas de Karnaugh para cada valor a representar.

2) Resta: aplicando una lógica similar a la suma, la resta va a utilizar diferentes funciones de la operación pasada, solo que en este caso el segundo número tendría que ir negativo. Esto se va a lograr mediante el uso del complemento a 2. Esta es una forma en la que se puede representar un número binario en negativo, por lo que al realizar la suma del primero número positivo, y el segundo en negativo, se estaría realizando una resta del segundo al primero.
Esta forma de pasar un número binario a negativo se puede realizar encontrando primero el complemento a uno del número, intercambinando un 0 por un 1 y viceversa, donde a este resultado se le va a sumar un 1 para así dar el resultado del número en negativo. En la ecuación 1 se ve representado este procedimiento [4]

$$C_2 = C_1 + 1 \tag{1}$$

Al ya tener el número en negativo, se realiza lo mencionado anteriormente, se llama a la suma de n bits con este nuevo número para así realizar la resta. En este punto cabe mencionar que se hizo uso de las flags o banderas de estado de una ALU común, en el cual se tienen: Zero (el resultado da 0), Negative (en caso de que el segundo número restando sea mayor que el primero) y el Carry (cuando una suma lleva acarreo). Al igual que la suma, su valor resultante va a estar presentado en los siete segmentos que convengan para su representación.

3) División entera: a la hora de realizar esta operación, se realiza un procedimiento similar a la resta, en la cual se hacía uso de la función de suma. Para esta operación aritmética, se va a llamar repetidas veces la suma, la cantidad de veces que tenga que llamarse el valor del divisor el cual va a ser una entrada, para que el resultado con la resta al dividendo esté más cercano a 0, sería el valor del cosciente, el cual va a representarse en el siete segmentos como el resultado de la división.

### B. *Problema 2*

El segundo problema en este caso, aplica los resultados del primer problema, donde se va a tener que utilizar la ALU creada para que con esta se prueben diferentes tamaños de bits y aquí se pueda apreciar en la ruta crítica, el tiempo de propagación y con este obtener la frecuencia máxima de operación para cada caso de bits en la ALU.

Para poder realizar esto se va a seguir un diagrama propuesto el cual se muestra en la figura 5

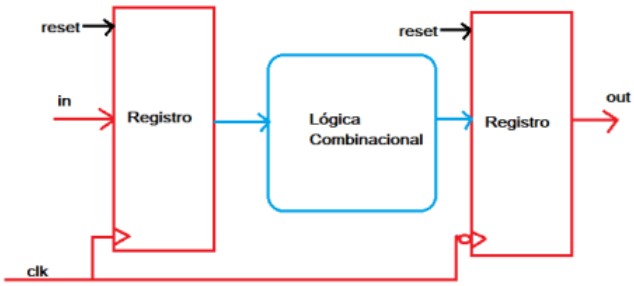


Fig. 5. Diagrama a diseñar para medir frecuencia máxima de operación de la ALU.

Todo esto será comprobado utilizando una herramienta perteneciente al software en el que se trabaja el cual es QuartusPrime, esta es el TimeQuest de Altera, donde se va a poder obtener la frecuencia buscada para cada caso de bits en la ALU. A partir de esto se realizará un gráfico y una tabla comparando los compartamientos que se presentan.

## III. RESULTADOS

Para los resultados obtenidos se tiene una representación de cada una de las pruebas realizadas para el problema 1 en el caso de 4 bits únicamente, se presentan los resultados primeramente en un testbench para apreciar que todo funcione correctamente, y luego estos resultados se presentan en las figuras siguientes donde se muestra cada uno de los valores obtenidos en siete segmentos tras hacer las operaciones mencionadas anteriormente.

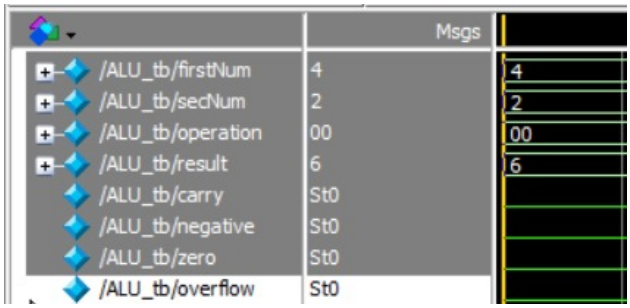


Fig. 6. Testbench de una suma de 4 bits (A, B, Operador, S, Flag C, Flag N, Flag Z, Flag V). Elaboración propia.

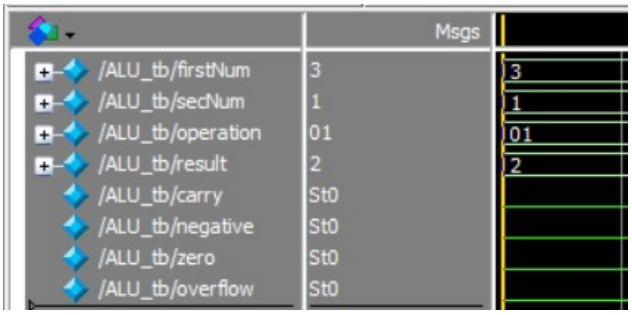


Fig. 7. Testbench de una resta de 4 bits (A, B, Operador, Resultado, Flag C, Flag N, Flag Z, Flag V). Elaboración propia.

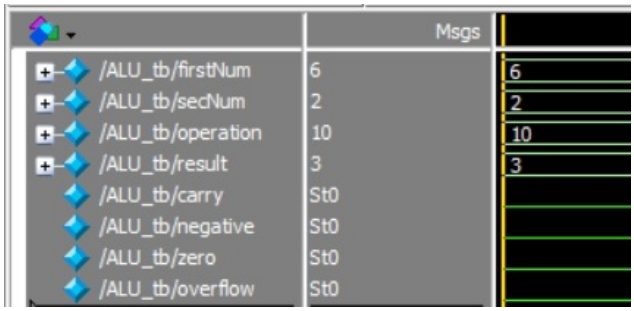


Fig. 8. Testbench de una división de 4 bits (A, B, Operador, Resultado, S, Flag C, Flag N, Flag Z, Flag V). Elaboración propia.

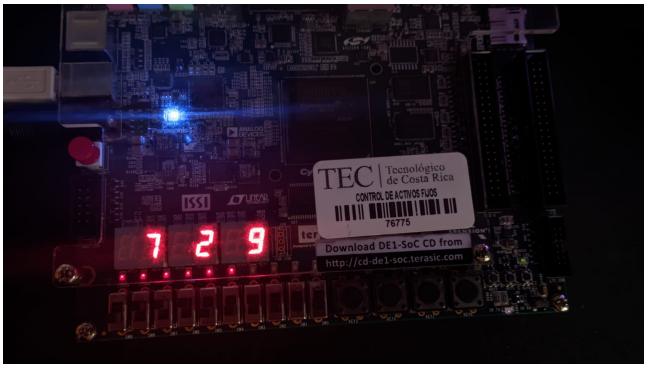


Fig. 9. Siete segmentos en una suma de 4 bits, sumando un valor de 7 y de 2. Elaboración propia.

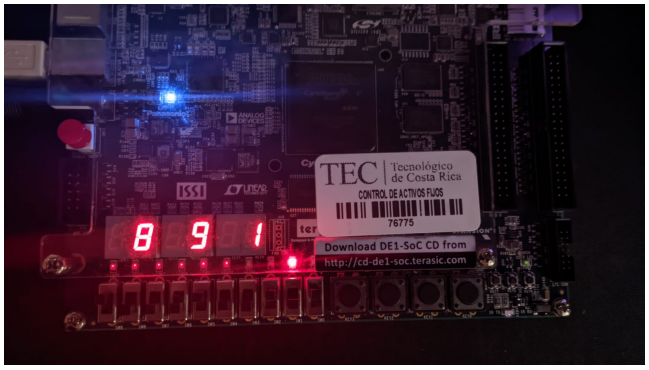


Fig. 10. Siete segmentos en una resta de 4 bits, restando un valor de 9 a 8 (Flag Negative encendida). Elaboración propia.

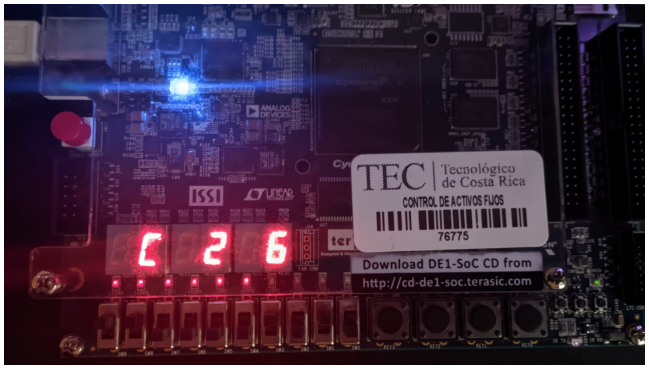


Fig. 11. Siete segmentos en una división de 4 bits, dividiendo un valor de 2 a C. Elaboración propia.

En el caso del problema 2, se presenta la siguiente tabla V mostrando frecuencias obtenidas y los casos a los que cada una de estas pertenece en donde lo que varía en estos es el número de bits que puede manipular la ALU.

TABLE V
VALORES DE FRECUENCIA OBTENIDO PARA CADA CASO.

| Caso | Número de bits | Frecuencia obtenida [MHz] |
|---|---|---|
| 1 | 2 | 726,85 |
| 2 | 4 | 428,34 |
| 3 | 8 | 203,67 |
| 4 | 16 | 148,59 |
| 5 | 32 | 91,19 |

## IV. DISCUSIÓN

Como puede mostrarse en la figuras 6, 7 y 8, cada uno de los valores que se aprecian en forma de números son los respectivos bits de cada una de las entradas, las cuales vendrían a ser el primer número, el segundo número, el operador, y el resultado. En el caso del operador, se tomó la asignación de 00 para la suma, el 01 para la resta, y el 10 para la división entera.

Como es apreciable las testbench no están compuestar únicamente de número, sino de gráficas, estas demuestran ya sea el resultado obtenido en cada flag (la primera siendo la flag de Carry, la segunda de Negative, la tercera Zero y la cuarta de Overflow). Para esto ejemplos propuestos en esas figuras,

no se dio el caso en que alguna flag estuviera encendida, sin embargo en caso de que alguna prueba presentara alguno de esos estados la bandera saldría como un 1, estando la gráfica arriba en lugar de abajo como está en los ejemplos. Esto funcionaría a la perfección en la opción de autochequeo realizada previamente, y al ponerse a prueba dio como resultado en la FPGA los valores presentados en las figuras 9, 10 y 11, donde los primeros 2 siete segmentos muestran los valores de los números a realizarles la operación presentados en sistema hexadecimal, y el siete segmentos de la derecha representaría el valor obtenido tras realizar la operación. Finalmente los leds que se presentan abajo a la derecha del último siete segmentos representarían las flags pertinentes, encendidas en el caso que sea necesario, el primer led es el de la flag Overflow, el segundo de flag Zero, el tercero representa la flag Negative y el último led es el flag Carry.

Para finalizar, en la tabla V se presentan las diferentes frecuencias a las que opera el circuito dependiendo de cuántos bits se estén teniendo, a partir de esta tabla se puede generar la siguiente gráfica en la figura 12.

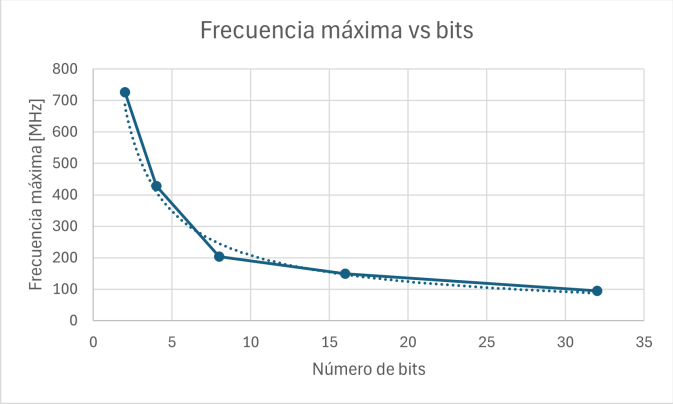


Fig. 12. Gráfica de frecuencia máxima contra bits. Elaboración propia.

Esta gráfica nos demuestra cómo se tiene una caída de frecuencia máxima a la que se puede operar dependiendo de la cantidad de bits que se estén utilizando en el circuito, esto se debe a que al poseer más bits con los que trabajar, va a tomar más tiempo a que se pueda realizar la lógica propuesta debido a la cantidad de veces que se tienen que realizar cada una de las operaciones para lograr dar un resultado. Esto puede disminuirse presentan una mejor optimización del circuito lógico donde este vaya a actuar más rápido al poseer más bits, sin embargo, este comportamiento se va a presentar la mayoría de veces.

## V. Conclusiones

Se logró replicar el funcionamiento de una ALU mediante el uso de operaciones básicas recreadas utilizando los circuitos lógicos pertenecientes a cada una de estas, y se entendió la importancia de esta unidad en cualquier procesador, la cual es que estas son las encargadas de realizar una tarea necesaria como lo son las operaciones. Se entendió la importancia de la presencia de diferentes funciones básicas en una unidad aritmética-lógica como lo son la presencia de operaciones básicas como resta, suma, multiplicación, división, shift-left y right, así como las banderas de estado, ya que sin estas se imposibilitaría la creación de sistemas más complejos. Se apreció la diferencia de frecuencias máximas a las que puede operar un circuito lógico dependiendo de la cantidad de bits a los que se someta, dando como resultado frecuencias más bajas ante una mayor cantidad de bits.

## References

[1] "ALU (unidad lógica aritmética): definición, función y más — UNIGAL". UNIGAL. Accedido el 27 de febrero de 2024. [En línea]. Disponible: https://unigal.mx/alu-unidad-logica-aritmetica-definicion-funcion-y-mas/

[2] J. Bramley. "Condition Codes 1: Condition flags and codes". Arm Community. Accedido el 28 de febrero de 2024. [En línea]. Disponible: https://community.arm.com/arm-community-blogs/b/architectures-and-processors-blog/posts/condition-codes-1-condition-flags-and-codes

[3] H. David y S. Harris, Digital Design and Computer Architecture: ARM Edition. Elsevier Sci. Technol. Books, 2015.

[4] Complemento a 2 — Electrónica digital. (s.f.). Repositorio de TecnoVilladiego. https://angelmicelti.github.io/4ESO/EDI/complemento_a_2.html