Laboratorio 3: Lógica combinacional y Aritmética I

Fiorella Delgado León Jonathan Guzmán Araya, Gerald Valverde McKenzie
Instituto Tecnológico de Costa Rica
Área Académica Ingeniería en Computadores
fiorelladelgado53@gmail.com jonathana1196@gmail.com gvmckenzie@mckode.com

***Abstract*—**

***Index Terms*—ALU, compuertas lógicas, flip-flops, microprocesadores**

I. Introducción

Los circuitos combinacionales son aquellos en los que las salidas solo dependen solamente de las entradas, y no de ningún tipo de sincronización con señales de reloj, esto hace que los sistemas combinacionales sean generalmente rápidos. En sistemas digitales complejos, como los microprocesadores, los circuitos de lógica combinacional desempeñan un papel fundamental. La arquitectura de un microprocesador es como la que se observa en la Figura 1.

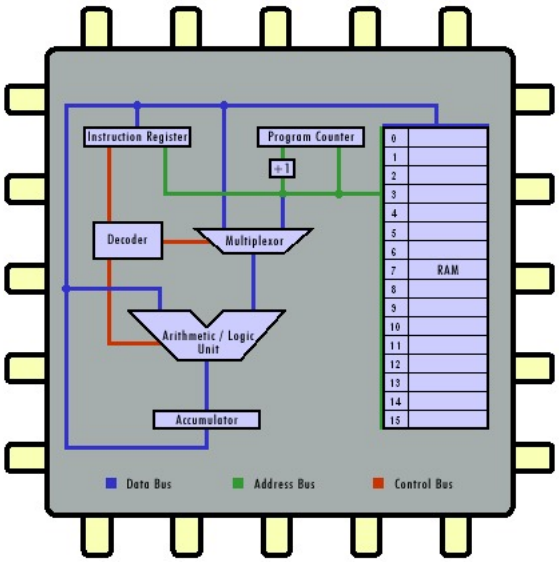

Fig. 1. Arquitectura básica de un microprocesador [1]

Una función esencial de muchas computadoras (microprocesadores) y calculadoras es la realización de operaciones lógicas y aritméticas. Estas operaciones se efectúan en la unidad aritmética-lógica de una computadora, donde se combinan compuertas lógicas y flip-flops de manera que puedan sumar, restar, multiplicar y dividir números binarios [2]. El símbolo de una ALU es como el de la Figura 2.

Además de los operadores lógicos, la ALU cuenta con una serie de registros para almacenar los datos y bits de información sobre los resultados, también llamados banderas [3], las banderas más comunes en una ALU son: Z, C, V, N.

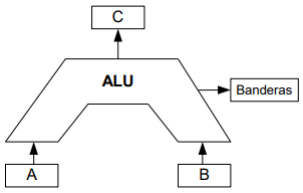

Fig. 2. Símbolo ALU [3]

Fig. 3. Compuerta NOT y su representación en álgebra de Boole [2]

- Zero flag: usada para determinar si dos valores son iguales, generalmente se usa una resta, y se revisa si el resultado es cero, si es así, esta bandera se enciende.
- Overflow flag: usada para determinar si el resultado de una operación (usualmente suma o multiplicación), tiene un tamaño superior al máximo representable, o inferior al mínimo representable en la ALU, es posible que existan como banderas separadas.
- Negative flag: usada para determinar si el resultado de una resta es menor a cero
- Carry/Borrow flag: Esta bandera indica cuando una operación resulta en un valor más largo del cual el acumulador puede representar o menor del cual el acumulador puede representar.

La ALU es simplemente un operador, es decir solo realiza operaciones, esta no toma decisiones, acepta datos binarios que están almacenados en la memoria y ejecuta operaciones con estos datos, de acuerdo a las instrucciones que vienen de la unidad de control.

Para realizar operaciones lógicas la ALU utiliza compuertas lógicas como:

- NOT: La Figura 3 muestra el símbolo para un circuito NOT, la operación NOT cambia de un nivel lógico al nivel lógico opuesto. Cuando la entrada está a nivel ALTO (1), la salida se pone a nivel BAJO (0). Cuando la entrada está a nivel BAJO, la salida se pone a nivel ALTO [4].
- AND: La Figura 4 muestra el símbolo lógico para una compuerta AND de dos entradas. La operación AND genera un nivel ALTO sólo cuando todas las entradas están a nivel ALTO, para el caso de dos entradas. Cuando

Q	Q'
0	1
1	0

TABLE I
TABLA DE VERDAD DE LA COMPUERTA NOT

una entrada está a nivel ALTO y la otra entrada está a nivel ALTO, la salida se pone a nivel ALTO. Cuando cualquiera de las entradas o todas ellas están a nivel BAJO, la salida se pone a nivel BAJO [4].

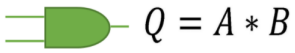

Fig. 4. Compuerta AND y su representación en álgebra de Boole [2]

A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

TABLE II
TABLA DE VERDAD DE LA COMPUERTA AND

- OR: La Figura 5 muestra el símbolo lógico para una compuerta OR de dos entradas. La operación OR genera un nivel ALTO cuando una o más entradas están a nivel ALTO, para el caso de dos entradas. Cuando una de las entradas está a nivel ALTO o ambas entradas están a nivel ALTO, la salida es un nivel ALTO. Cuando ambas entradas están a nivel BAJO, la salida será un nivel BAJO [4].

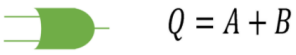

Fig. 5. Compuerta OR y su representación en álgebra de Boole [2]

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

TABLE III
TABLA DE VERDAD DE LA COMPUERTA OR

- XOR: La Figura 6 muestra el símbolo lógico para una compuerta XOR de dos entradas. En una compuerta OR-exclusiva, la salida X es un nivel ALTO si la entrada A está a nivel BAJO y la entrada B está a nivel ALTO; o si la entrada A está a nivel ALTO y la entrada B está a nivel BAJO; X es un nivel BAJO si tanto A como B están a nivel ALTO o BAJO [4].

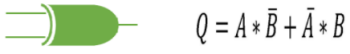

Fig. 6. Compuerta XOR y su representación en álgebra de Boole [2]

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

TABLE IV
TABLA DE VERDAD DE LA COMPUERTA XOR

Uno de los problemas más desafiantes en el diseño de circuitos es el tiempo, hacer que un circuito funcione rápido. Una salida necesita tiempo para cambiar en respuesta a un cambio en la entrada, como se observa en la Figura 7 [5].

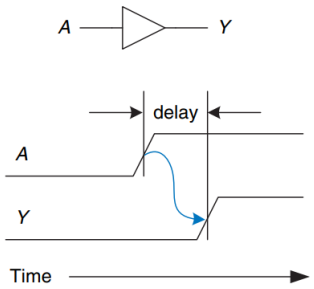

Fig. 7. Tiempo de respuesta de una señal [5]

En la lógica combinacional existen dos tipos de tiempos, tiempo de propagación y tiempo de contaminación. El tiempo de propagación t_{pd} es el tiempo máximo desde que la entrada cambia hasta que el o las salidas llegan a su valor final y el tiempo de contaminación t_{cd} es el tiempo mínimo que dura la entrada en cambiar hasta que alguna salida empiece a cambiar su valor [5].

Además de los tiempos de propagación y contaminación en la electrónica de circuitos existe algo llamado la ruta crítica, que es la ruta más larga que tiene una entrada, hablando de circuitos combinacionales, para llegar a la salida, por lo tanto, esta ruta también sería la más lenta. Ahora para describir a un circuito hay dos factores muy importantes: latencia y tasa de transferencia. La latencia funcionaria como la frecuencia del circuito, ya que es el tiempo que se necesita para que el cambio en la entrada haga un cambio en la salida; esta puede ser expresada en tiempo o, para circuitos sincrónicos, un cierto número de ciclos de reloj. Y la tasa de transferencia a la velocidad con la que la información puede ser procesada. Por ejemplo, en la Figura 8 se tienen tres rutas distintas desde una entrada hasta la salida, siendo la ruta azul la más lenta de estas.

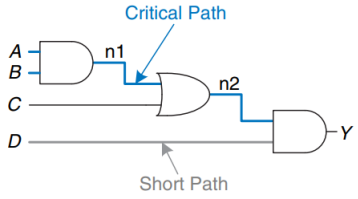

Fig. 8. Ruta crítica [5]

Así el tiempo de propagación de un circuito combinacional es la suma de los tiempos de propagación de cada elemento

que lo compone como se observa en la Ecuación 1 y el tiempo de contaminación es la suma de los tiempos de contaminación de la más corta del circuito combinacional que se puede observar en la Ecuación 2.

$$t_{pd} = 2t_{pdAND} + t_{pdOR} \tag{1}$$

$$t_{pd} = 2t_{pdAND} + t_{pdOR} \tag{2}$$

En circuitos más complejos como un procesador pipeline como el de la Figura 9, la velocidad de ciclos del reloj decrece haciendo retrasos en la ruta crítica, para esto se utiliza la técnica de Pipelining, el cual consiste en dividir el circuito en varias partes con un registro al final de cada una, de esta manera se divide la ruta crítica en pequeñas rutas, permitiendo que la velocidad del reloj aumente y consecuentemente también lo haga la tasa de transferencia. La segmentación consiste en descomponer la ejecución de cada instrucción en varias etapas para poder empezar a procesar una instrucción diferente en cada una de ellas y trabajar con varias a la vez. Las etapas de segmentación pueden ser: IF (búsqueda), ID (decodificación), EX (ejecución), MEM (memoria), WB (escritura), en una arquitectura relativamente sencilla.

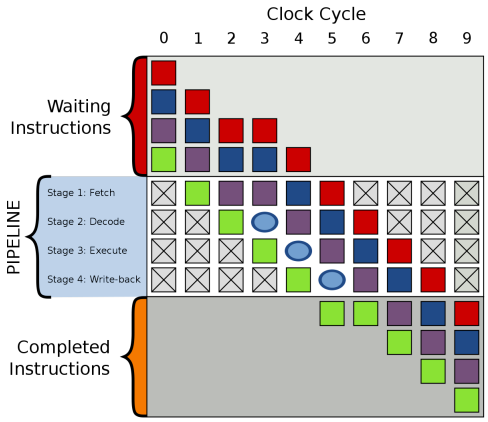

Fig. 9. Procesador Pipeline [6]

La frecuencia máxima de operación de un circuito está determinada por la ruta crítica del mismo, para esto se toma la ruta crítica (en un procesador se realiza tomando la instrucción que tarda más en ejecutarse por completo sin afectar a las otras) y se calcula el tiempo que tarda en ciclos de reloj T_c, como se observa en la Ecuación 3, esta frecuencia se mide en Hz.

$$f_c = \frac{1}{T_c} \tag{3}$$

II. Desarrollo

Para el desarrollo de este laboratorio, se realizó la implementación de dos soluciones a dos problemas por medio del lenguaje System Verilog.

A. Experimento 1

El primer experimento del laboratorio consiste en la implementación de una Unidad Lógica Aritmética. Dicha unidad debe realizar una serie de operaciones, siendo una calculadora parametrizable de manera que la cantidad de bits de entrada puedan ser modificados. En la Figura MOOOOOODIFICAR se observan las diversas operaciones y los valores de la entrada de selección para ejecutar dicha operación. Así mismo la ALU es implementada con banderas de estado tales como: Negativo (N), Cero (Z), Acarreo (C) y Desbordamiento (V).

TABLE V
Tabla de las operaciones con su respectivo selector.

Operación	Selector
Suma	0000
Resta	0001
Multiplicación	0010
División Q1	0011
Módulo Q3	0100
AND	0101
OR	0110
XOR	0111
Shift Left	1000
Shift Right	1001

La ALU fue implementada mediante el modelado estructural, en donde se utilizan módulos independientes para llegar a formar la ALU completamente.

En la Figura 10, se puede observar el diagrama de primer nivel de la calculadora parametrizable, en donde las entradas A y B, son los dos datos de entrada que van a ser operados, para este caso pueden ser de N bits; la entrada S, es la entrada de selección, la cual se utiliza para escoger la operación que se desea aplicar a los dos operandos ingresados (A y B), para este caso y con las operaciones que puede realizar esta calculadora.

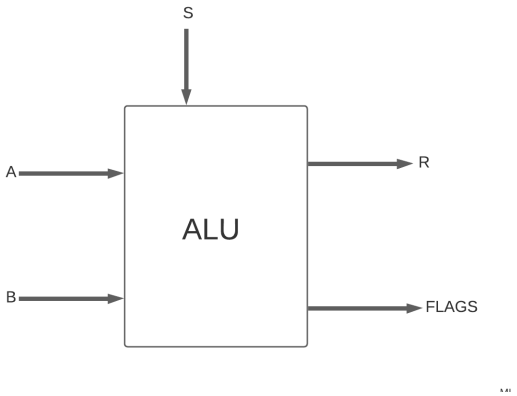

Fig. 10. Diagrama de primer nivel de la calculadora parametrizable

La salida Q es el resultado generado por la ALU al operar los datos ingresados, dicha salida es de 4 bits. Por otro lado, la salida FLAGS es un bit array que posee cada una de las banderas de estado implementadas por lo tanto es una salida de 4 bits.

1) Restricciones de diseño:

- La ALU implementada posee una salida Q de 4 bits por lo cuál cualquier dato mayor a este que se desee visualizar

a la salida de la calculadora, se verá demostrado por la flag V.

- Tanto el sumador como el restador son implementados por medio de un modelo estructural, por lo cuál sus entradas no son parametrizables, siendo estas de 4 bits cada una.
- La operación de resta sólo admite números positivos en sus operandos siendo el resultado $Q = A - B$.

2) Implementación: La estructura interna de la ALU consiste en una estructura modular en la que se compone de diversos módulos. En la Figura 11, se logra apreciar como los diferentes subsistemas interactúan entre sí. Por otro lado, en la Figura 12, se observa el diagrama de tercer nivel del dispositivo siendo este con mayor detalle, en donde hay 10 módulos que representan cada una de las operaciones, recibiendo de entrada A y B, dando como resultado una señal I_i. Las señales I_i, son los datos de entrada para el MUX encargado de filtrar el dato que se va mostrar a la salida por medio de los bits de entrada de la señal S.

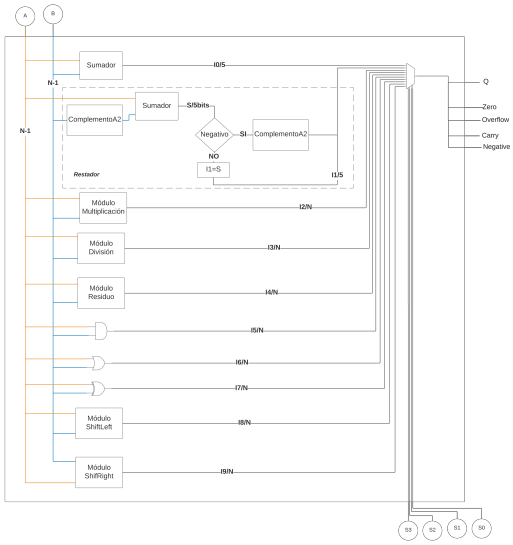

Fig. 11. Diagrama de segundo nivel de la ALU

Tal como se menciona en las restricciones del experimento 1, tanto el sumador como el restador han sido implementados por medio del modelo estructural. Por lo que el sumador de 4 bits ha sido desarrollado conectando 4 sumadores completos de 1 bit (véase la estructura interna de este en la Figura 13) en forma de cascada tal como se observa en la Figura 14, en donde los datos A_i representan cada uno de los bits de entrada del primer operando siendo A_0 el LSB, así mismo las señales B_i representan los datos de la segunda entrada. Las señales S_i representa el resultado del sumador y C_o es el acarreo de salida o Carry Output. Para fines de diseño se definió como quinto bit de la señal S.

En cuanto a la implementación del restador se le aplica el complemento A2 al segundo operando B, seguidamente se realiza la operación de $A + ComplementoA2(B)$, se procede analizando signo del resultado siendo este el MSB, si el signo de la operación es negativo se procede a realizar el complemento A2 al resultado, en el caso que el signo sea

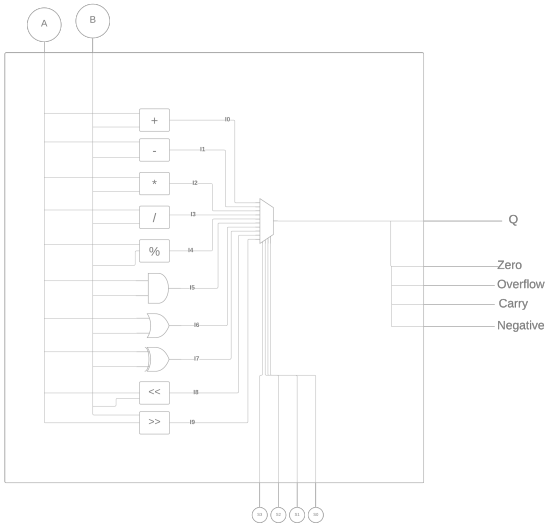

Fig. 12. Diagrama de tercer nivel de la implementación de la ALU

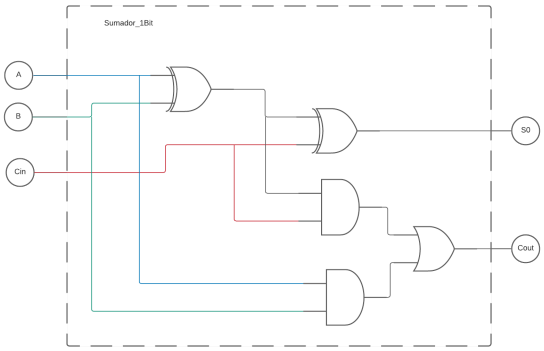

Fig. 13. Diagrama lógico de un sumador de 1 bit

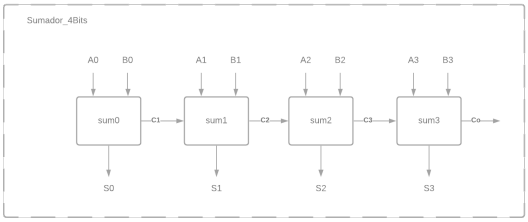

Fig. 14. Diagrama del modelo estructural de un sumador de 5 bits

positivo se mantiene sin cambios, tal como se muestra en la Figura 15.

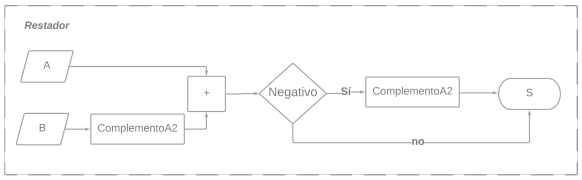

Fig. 15. Diagrama de flujo del restador de 4 bits

Finalmente, para la implementación en la FPGA se implementa el diagrama resultante de la Figura 16, el cuál consiste en la representación de los datos por medio de un displays. Siendo la salida de 4 bits el caso extremo a representar corresponde a la salida $Q = 1111$ siendo su representación en decimal de $Q = 15$, siendo este un número de dos dígitos (decenas y unidades) por lo que se decide la implementación de un display para cada dígito. Para realizarlo, se procede a calcular las unidades con la operación $Q\%10$, así mismo se procede con el cálculo de las decenas por medio de $Q//10$. Las salidas de dichas operaciones son enviadas a un decodificador el cuál se encarga de convertir el número binario que representa cada dígito decimal a un arreglo de 7 bits que corresponde a la elaboración de la representación en el display de 7 segmentos.

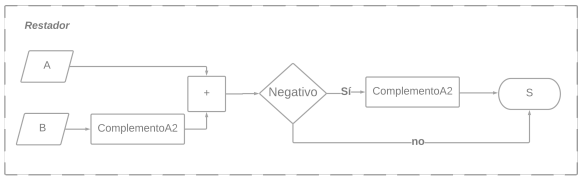

Fig. 16. Diagrama de la implementación de la ALU en la FPGA

B. Experimento 2

En este experimento, se requiere implementar dos registros de carga paralela entre la ALU; esto con el fin de determinar la frecuencia máxima a la que puede operar un reloj. En esta implementación se desarrollaron 5 test-bench en donde se comprueba el funcionamiento del ALU con 5, 32, 64, 128 y 256 bits.

En la Figura 17, se puede observar cómo el circuito electrónico parametrizable para la medición de frecuencia máxima de operación tiene 4 entradas que son ingresadas externamente mediante el uso de switches y botones; la entrada E0 (Dato de entrada) es de n bits, la entrada E1 (Reset1) es el reset para el primer registro es de 1 bit, la entrada E2 (Reset2) es el reset para el segundo registro es de 1 bit y la entrada E3 (Clock) es una entrada en común de reloj para ambos registros y es de 1 bit. Por otro lado, el mismo circuito sólo posee una salida la cual es el Resultado (S0) y es de n bits, este hace referencia al resultado de operar los dos números en el ALU.

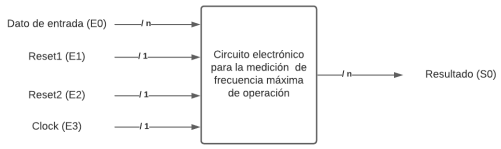

Fig. 17. Diagrama de primer nivel del circuito electrónico para la medición de frecuencia máxima de operación

En la Figura 18, se puede observar el diagrama de segundo nivel del circuito electrónico para la medición de frecuencia máxima de operación; dicho diagrama consta de 3 submódulos principales: el primero de ellos es el Registro 1, este consta de la entrada E0 (Dato de entrada) es de n bits, la entrada E1 (Reset1) es el reset para el primer registro es de 1 bit y la entrada E3 (Clock) es una entrada en común de reloj para ambos registros y es de 1 bit; así mismo tiene la salida de este módulo que es OutR1 y esta es ambos números introducidos y guardados en el Registro 1. Al ALU le entra esa salida del Registro 1 (OutR1) con los dos números a operar y genera una salida OutALU con el resultado de la operación. Y finalmente el Registro 2 posee la entrada E2 (Reset2) es el reset para el segundo registro es de 1 bit, la entrada E3 (Clock) es una entrada en común de reloj para ambos registros y es de 1 bit y la entrada OutALU, proveniente de la salida del ALU; su salida correspondiente es Resultado (S0) y es de n bits. Como se ha estudiado e investigado anteriormente, la función de los registros es almacenar un valor, cuando se activa una señal.

Por otro lado, el mismo circuito sólo posee una salida la cual es el Resultado (S0) y es de n bits, este hace referencia al resultado de operar los dos números en el ALU. la entrada E2 (Reset2) es el reset para el segundo registro es de 1 bit

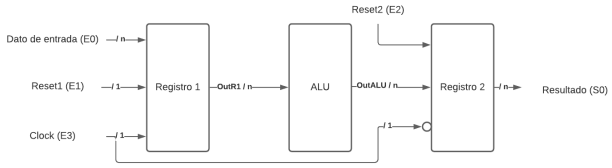

Fig. 18. Diagrama de segundo nivel del circuito electrónico para la medición de frecuencia máxima de operación

III. Resultados

A. Experimento 1

Al realizar el experimento 1 de este laboratorio, se realizaron testbench de autochequeo para cada uno de los módulos implementados en la ALU. En la Figura 19, se puede observar la salida de onda del tesbench para el módulo del sumador implementado en la ALU.

En la Figura 20, se puede observar la salida de onda del tesbench para el módulo del restador implementado en la ALU.

En la Figura 21, se puede observar la salida de onda del tesbench para el módulo de la multiplicación implementado en la ALU.

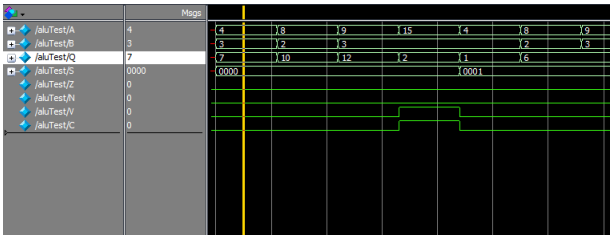

Fig. 19. Onda de salida del sumador en la ALU

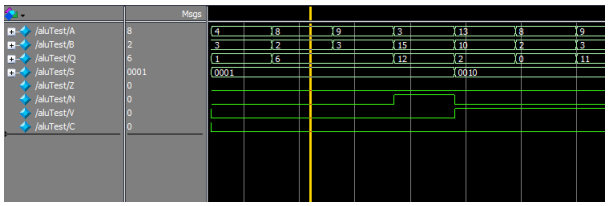

Fig. 20. Onda de salida del restador en la ALU

En la Figura 22, se puede observar la salida de onda del tesbench para el módulo de la división implementado en la ALU.

En la Figura 23, se puede observar la salida de onda del tesbench para el módulo del módulo o residuo implementado en la ALU.

En la Figura 24, se puede observar la salida de onda del tesbench para el módulo de la compuerta AND implementado en la ALU.

En la Figura 25, se puede observar la salida de onda del tesbench para el módulo de la compuerta OR implementado en la ALU.

En la Figura 26, se puede observar la salida de onda del tesbench para el módulo de la compuerta XOR implementado en la ALU.

En la Figura 27, se puede observar la salida de onda del tesbench para el módulo del shift left implementado en la ALU.

En la Figura 28, se puede observar la salida de onda del tesbench para el módulo del shift right implementado en la ALU.

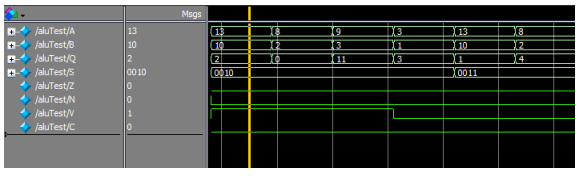

Fig. 21. Onda de salida de la multiplicación en la ALU

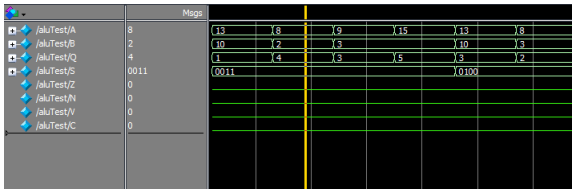

Fig. 22. Onda de salida de la división en la ALU

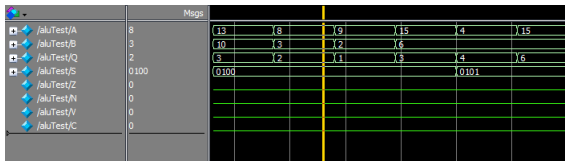

Fig. 23. Onda de salida del residuo o módulo en la ALU

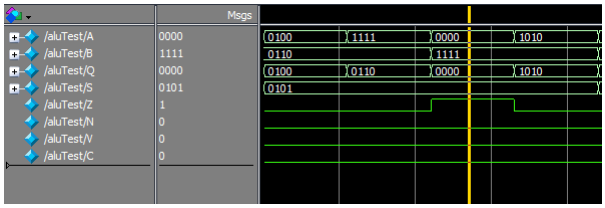

Fig. 24. Onda de salida de la compuerta AND en la ALU

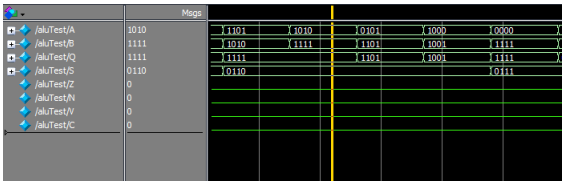

Fig. 25. Onda de salida de la compuerta OR en la ALU

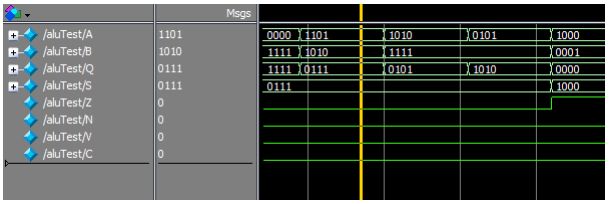

Fig. 26. Onda de salida de la compuerta XOR en la ALU

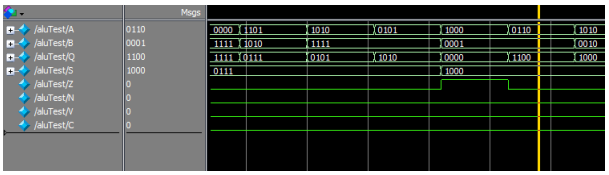

Fig. 27. Onda de salida del shift left en la ALU

Fig. 28. Onda de salida del shift right en la ALU

En la Figura 29, se puede observar el diagrama generado por Quartus de la unidad aritmética lógica (ALU).

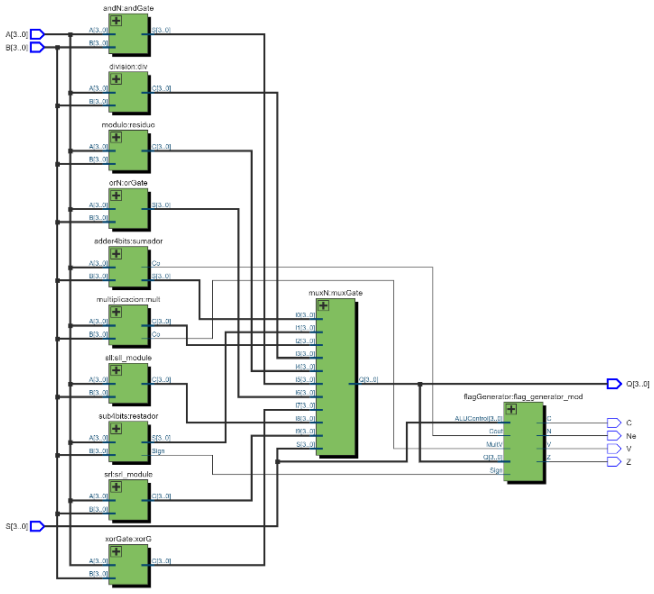

Fig. 29. Diagrama generado por Quartus del ALU

B. Experimento 2

Al realizar este experimento, se obtuvieron los datos que se encuentra en la Tabla VI, en la cual se puede apreciar la cantidad de ALMs, la cantidad de registros utilizados, la cantidad de bits de entrada y la frecuencia máxima.

TABLE VI
TABLA DE LAS OPERACIONES CON SU RESPECTIVO SELECTOR.

Bits de entrada	ALMs	Registros	Pines	Frecuencia (max)
10	4	15	22	380.23 MHz
32	13	48	66	369.28 MHz
64	25	96	130	375.94 MHz
128	49	192	258	374.53 MHz
256	193	384	514	—

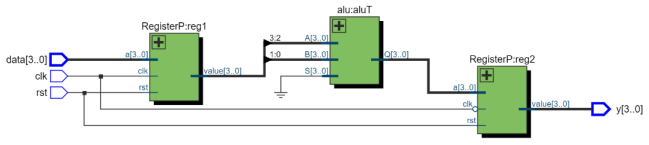

Fig. 30. Diagrama generado por Quartus del ALU, al utilizar dos registros

IV. ANÁLISIS DE RESULTADOS

A. Experimento 1

Tal como se puede observar en la Figura 19 como al sumar los números binarios: 0100 y 0011, su salida es operada correctamente por el módulo sumador que se encuentra en la ALU y su resultado es de 0111.

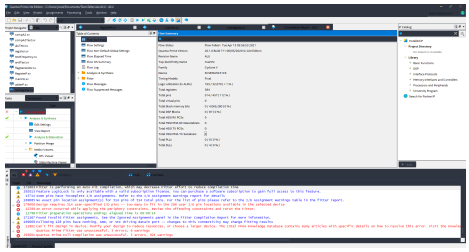

Fig. 31. Error generado al tratar de introducir 256 bits en el ALU, para implementarlo en la FPGA

En la Figura 20 como al restar los números binarios: 0011 y 1111, su salida es operada correctamente por el módulo restador que se encuentra en la ALU y su resultado es de 1100, además pone en alto la bandera del Negativo.

En la Figura 21 como al multiplicar los números binarios: 1001 y 0011, su salida es operada correctamente por el módulo de multiplicación que se encuentra en la ALU y su resultado es de 1011, además pone en alto la bandera de Overflow.

En la Figura 22 como al sacar la división entera de los números binarios: 1101 y 1010, su salida es operada correctamente por el módulo división que se encuentra en la ALU y su resultado es de 0001.

En la Figura 23 como al sacar el residuo de los números binarios: 1101 y 1010, su salida es operada correctamente por el módulo modulo que se encuentra en la ALU y su resultado es de 0011.

En la Figura 22 como al sacar la operación lógica AND de los números binarios: 0101 y 0100, su salida es operada correctamente por el módulo andN que se encuentra en la ALU y su resultado es de 0100.

En la Figura 22 como al sacar la operación lógica OR de los números binarios: 1101 y 1010, su salida es operada correctamente por el módulo orN que se encuentra en la ALU y su resultado es de 1111.

En la Figura 22 como al sacar la operación lógica XOR de los números binarios: 1101 y 1010, su salida es operada correctamente por el módulo xorN que se encuentra en la ALU y su resultado es de 0111.

En la Figura 27 como al sacar el shift left de los números binarios: 1000 y 0001, su salida es operada correctamente por el módulo shiftLeft que se encuentra en la ALU y su resultado es de 0000.

En la Figura 28 como al sacar el shift right de los números binarios: 1101 y 0010, su salida es operada correctamente por el módulo shiftRight que se encuentra en la ALU y su resultado es de 0011.

Con cada una de estas pruebas generadas en el testbench de auto chequeo de la unidad aritmética lógica, se puede comprobar el adecuado funcionamiento de cada uno de los módulos de la misma.

Como se puede observar en la Figura 29, el diagrama generado por Quartus corresponde y es equivalente al diagrama que fue generado en la sección de Desarrollo en donde sale cada uno de los módulos que contiene la ALU, con su respectiva entrada de selección, el MUX y el generador de banderas.

B. Experimento 2

Como se puede observar en la Tabla VI, al aumentar la cantidad de bits en la entrada del ALU, aumenta la cantidad de ALMs, la cantidad de Registros y de Pines que se utilizan en la FPGA; con esto se puede probar cómo al aumentar la cantidad de bits en la entrada, la FPGA consume más recursos.

Al querer implementar la ALU con 256 bits de entrada, se consumen todos los recursos de la FPGA y no existen pines suficientes, ya que la misma indica que se está consumiendo un 112 porciento, por lo cual es imposible utilizar 256 bits de entrada.

Con los resultados y datos obtenidos en la Tabla VI, se puede observar como la máxima frecuencia en la cual puede operar es de $389.23MHZ$, la cual hace referencia a los 10 bits de entrada, esto se debe a que la frecuencia es inversamente proporcional al tiempo y con 10 bits, la ejecución de las operaciones es menor que con una mayor cantidad de bits.

Como se puede observar en la Figura 30, el diagrama generado por Quartus corresponde y es equivalente al diagrama que fue generado en la sección de Desarrollo (Figura 18) en donde sale cada uno de los dos registros así como el ALU, con sus respectivas entradas de reset, y clock.

En la Figura 31, se puede observar como al tratar de introducir 256 bit, Quartus genera un error, porque no hay pines suficientes para poder implementar dicho programa.

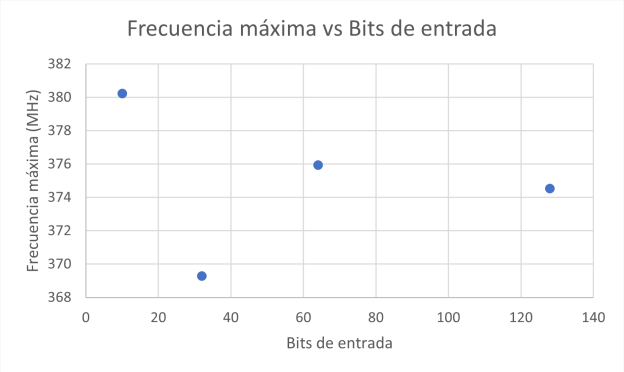

Fig. 32. Gráfico de frecuencia máxima vs bits de entrada

En la Figura 32, se observa la tendencia que tienen los puntos del gráfico a decrecer, esto se presenta porque la frecuencia delimita al tiempo de propagación y cada vez que se aumentan los bits de entrada, se consumen más recursos de la FPGA como ALMs, registros y pines, lo que provoca que el tiempo de respuesta sea más lento y más larga la ruta crítica que pueden tomar los datos.

V. Conclusiones

VI. Bibliografía

References

[1] A. J. F. González. (2017, Jul.) Arquitectura de microprocesadores desde el principio. beBee. [Online]. Available: https://www.bebee.com/producer/@http-alfrdo-j-feijoo-webnode-es-contacto/arquitectura-de-microprocesadores-desde-el-principio

[2] G. M. Tocci, Ronald J. Neal S. Widmer, *Sistemas digitales: principios y aplicaciones*, L. M. C. Castillo, Ed. Pearson Educación, 2007.

[3] D. A. G. García. Procesadores. Departemento de Mecatrónica. [Online]. Available: http://homepage.cem.itesm.mx/garcia.andres/PDF201411/ArquitecturaComputacional.pdf

[4] T. L. Floyd, *Fundamentos de Sistemas Digitales*, M. Martin-Romo, Ed. Pearson Educación, 2006.

[5] D. M. H. Sarah L. Harris, *Digital Design and Computer Architecture*. Morgan Kauffmann, 2010.

[6] Arquitectura en pipeline (informática). Wikipedia. [Online]. Available: https://es.wikipedia.org/wiki/Arquitectura_en_pipeline_(inform\unhbox\voidb@x\bgroup\let\unhbox\voidb@x\setbox\@tempboxa\hbox{a\global\mathchardef\accent@spacefactor\spacefactor}\let\begingroup\endgroup\relax\let\ignorespaces\relax\accent19a\egroup\spacefactor\accent@spacefactortica)