

Práctica 2

Nombre	Carné	Participación
Ramiro A. Chacón Castañeda	2915963551903	100%
Jeysson E. Godoy Torres	3429393512210	100%
Mario R. Palma Villeda	3384151142007	100%
Erick E. Pineda Palma	2472451942001	100%
Eduardo R. Cruz Sánchez	2472597802001	100%

Universidad de San Carlos de Guatemala

964: Laboratorio de Organización Computacional

Aux. Javier Alejandro Gutiérrez de León

Jueves 20 de junio 2024

Contenido

Introducción.....	3
Objetivos.....	4
Descripción del Problema.....	5
Funciones Booleanas y Mapas K.....	6
Diagramas.....	10
Equipo Utilizado.....	13
Presupuesto.....	14
Aporte Por Integrante.....	15
Conclusiones.....	17
Anexos.....	18
Video.....	18
Fotografías.....	19

Introducción

En el ámbito de la electrónica digital, los bloques MSI (Medium Scale Integration, por sus siglas en inglés) desempeñan un papel crucial en la construcción de circuitos complejos. Dentro de esta categoría, los bloques MSI tipo aritmético son especialmente importantes para realizar operaciones aritméticas básicas, como sumas, restas y multiplicaciones, de manera eficiente y confiable.

Los bloques digitales combinacionales MSI tipo aritmético están diseñados para realizar operaciones matemáticas utilizando compuertas lógicas, registros y otros componentes digitales. Estos bloques están configurados para recibir dos o más operandos y generar una salida que representa el resultado de la operación.

Objetivos

Objetivo General

Construir una Unidad Aritmética Lógica Básica (ALU).

Objetivos Específicos

1. Aprender el funcionamiento de Multiplexores, Demultiplexores, Comparadores y Decodificadores.
2. Construir un diseño óptimo, logrando utilizar la menor cantidad de dispositivos.
3. Aprender el funcionamiento de Operaciones Lógicas, Aritméticas y Comparativas con números binarios


Descripción del Problema

Como estudiantes del curso Organización Computacional, han sido contratados por Intel Corporation para desarrollar un prototipo de calculadora llamado "LogicCalc". Intel busca una solución óptima basada en lógica combinatorial que sea capaz de realizar cálculos aritméticos y lógicos. Para cumplir con estos requisitos, Intel ha proporcionado las especificaciones únicas para una Unidad Aritmética Lógica Básica (ALU).

Funciones Booleanas y Mapas K

Funcion logica XNOR

Símbolo



Función lógica

$$F_2 = (A' \cdot B + A \cdot B')' = (A \oplus B)'$$

Tabla de verdad

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	1

Para la comprobación de igualdad, entre N número de bits, únicamente prescindimos de la utilización de una compuerta lógica XNOR que a efectos prácticos fue realizada con una compuerta XOR y una NOT.

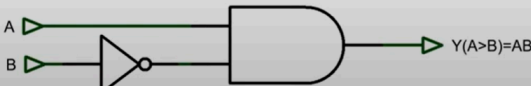
Comparador Mayor o Menor

$y(A > B) = A \cdot B'$

B	A	A > B
0	0	NO
0	1	SI
1	0	NO
1	1	NO

→

B'	A	Y
1	0	0
1	1	1
0	0	0
0	1	0




$y(A < B) = A' \cdot B$

B	A	A < B
0	0	NO
0	1	NO
1	0	SI
1	1	NO

→

B	A'	Y
0	1	0
0	0	0
1	1	1
1	0	0



Para lograr que se realice la comparación de 2 bits y determinar si uno es mayor que el otro precisamos de colocarlos como entradas en un compuerta AND, donde uno de los casos tiene como entradas al primer bit y al inverso del segundo, siendo similar al segundo caso con la única diferencia que se invierten los papeles de los bits involucrados, es decir, el bit 2 irá al inversor.

Ecuaciones generales para el comparador de 4 bits

Ecuación que describe la igualdad general entre los 4 bits

$$(A=B) = Y_3 \cdot Y_2 \cdot Y_1 \cdot Y_0$$

Ecuaciones de Mayor que y Menor que

$$(A < B) = A_3' \cdot B_3 + Y_3 \cdot A_2' \cdot B_2 + Y_3 \cdot Y_2 \cdot A_1' \cdot B_1 + Y_3 \cdot Y_2 \cdot Y_1 \cdot A_0' \cdot B_0$$

$$(A > B) = A_3 \cdot B_3' + Y_3 \cdot A_2 \cdot B_2' + Y_3 \cdot Y_2 \cdot A_1 \cdot B_1' + Y_3 \cdot Y_2 \cdot Y_1 \cdot A_0 \cdot B_0'$$

Aclaración: El and con los términos definidos por “y” viene dado por el principio de buscar el descarte de igualdad en cada uno de los términos.

Sumador de 4 bits

Se abordó el diseño de un sumador/restador de 4 bits con un enfoque meticuloso desde la etapa combinacional. Iniciamos identificando y diseñando las expresiones booleanas que describen las operaciones de suma y haciendo una tabla de funcionamiento del mismo.

En la etapa combinacional, desarrollamos circuitos que manejan adecuadamente el acarreo y el complemento a dos, elementos esenciales para el correcto funcionamiento del sumador/restador. Conectamos varios sumadores completos de 1 bit en cascada para formar un sumador de 4 bits. Cada sumador de 1 bit se conecta secuencialmente, donde el acarreo de salida de uno se convierte en la entrada de acarreo del siguiente, asegurando así que la suma se realice correctamente a lo largo de los 4 bits.

Para la funcionalidad de resta, implementamos la técnica de complemento a dos. Esto implica convertir el sustraendo en su complemento a dos y agregarlo al minuendo utilizando el mismo sumador de 4 bits diseñado para la suma. El acarreo inicial en la operación de resta se establece en '1' para manejar adecuadamente los casos de desbordamiento negativo.

En la práctica, la integración de la lógica de sumador y restador en el mismo circuito optimizó el diseño y maximizó la eficiencia del hardware utilizado. Esto se logró mediante la configuración apropiada de las entradas y el control de los acarros para cumplir con los requisitos de suma y resta en un único sistema combinacional.

En resumen, nuestro enfoque meticuloso en la etapa combinacional y la integración secuencial de sumadores de 1 bit nos permitió implementar con éxito un sumador/restador de 4 bits

funcional y eficiente, capaz de realizar operaciones binarias precisas y confiables en cualquier combinación deseada.

Multiplicación

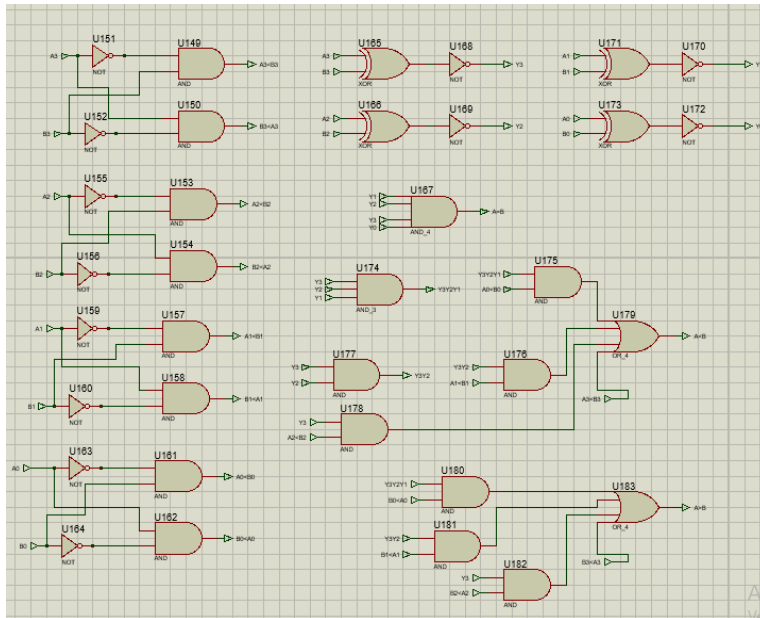
Cada bit del multiplicando se multiplica por cada bit del multiplicador, generando productos parciales que son desplazados según su posición correspondiente. Luego, estos productos parciales se suman para obtener el resultado final, que puede tener hasta 8 bits. Este proceso se puede implementar mediante circuitos lógicos combinacionales utilizando puertas AND para la generación de productos parciales y sumadores binarios para la acumulación de estos productos.

Potenciación

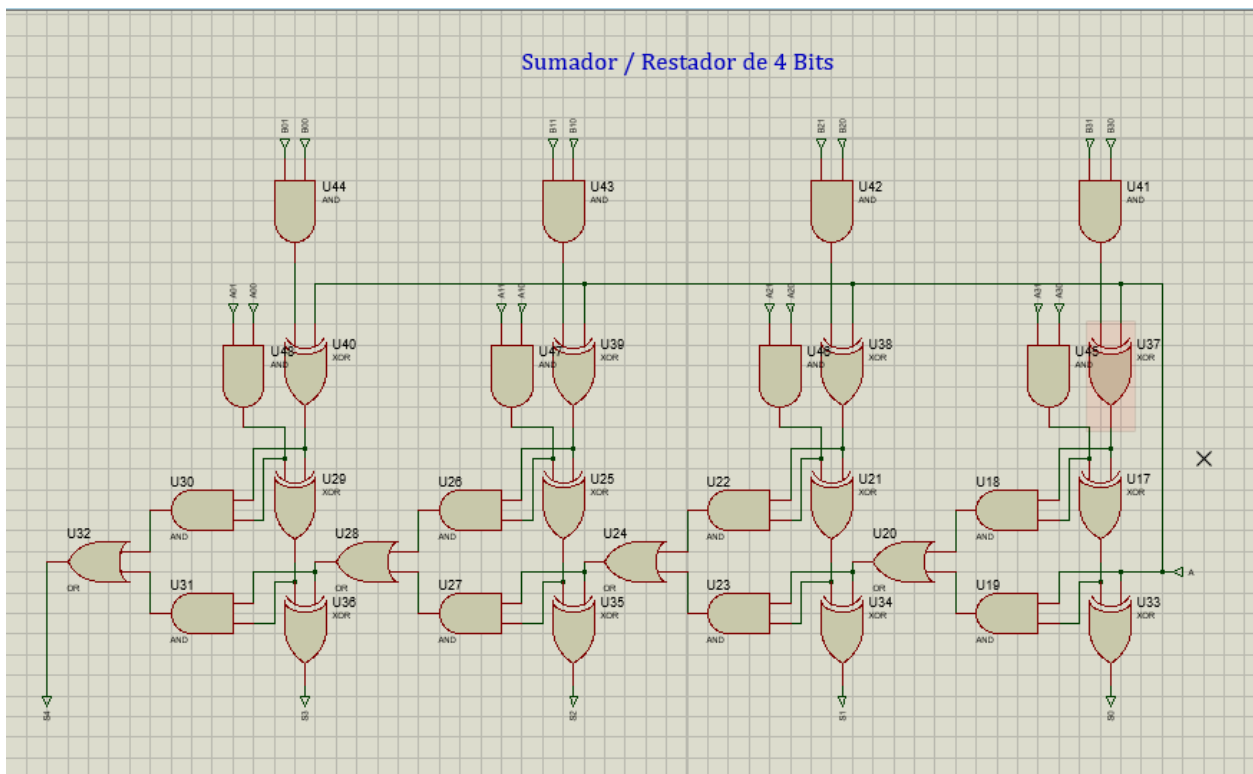
Un circuito para calcular una potencia de un número de 4 bits elevado a la 2 o a la 3, utilizando ADN y sumadores completos, se basa en realizar multiplicaciones sucesivas. Para elevar un número de 4 bits AA al cuadrado (A^2), el circuito multiplica AA por sí mismo utilizando un multiplicador de 4 bits por 4 bits, como se explicó anteriormente. Para elevar AA al cubo (A^3), primero se calcula A^2 y luego se multiplica el resultado nuevamente por AA utilizando otro multiplicador de 4 bits. Los productos parciales se generan con puertas AND y se suman con sumadores completos (full adders) para obtener el resultado final, que puede ser de hasta 12 bits para A^3 debido a la acumulación de bits en las multiplicaciones sucesivas.

Diagramas

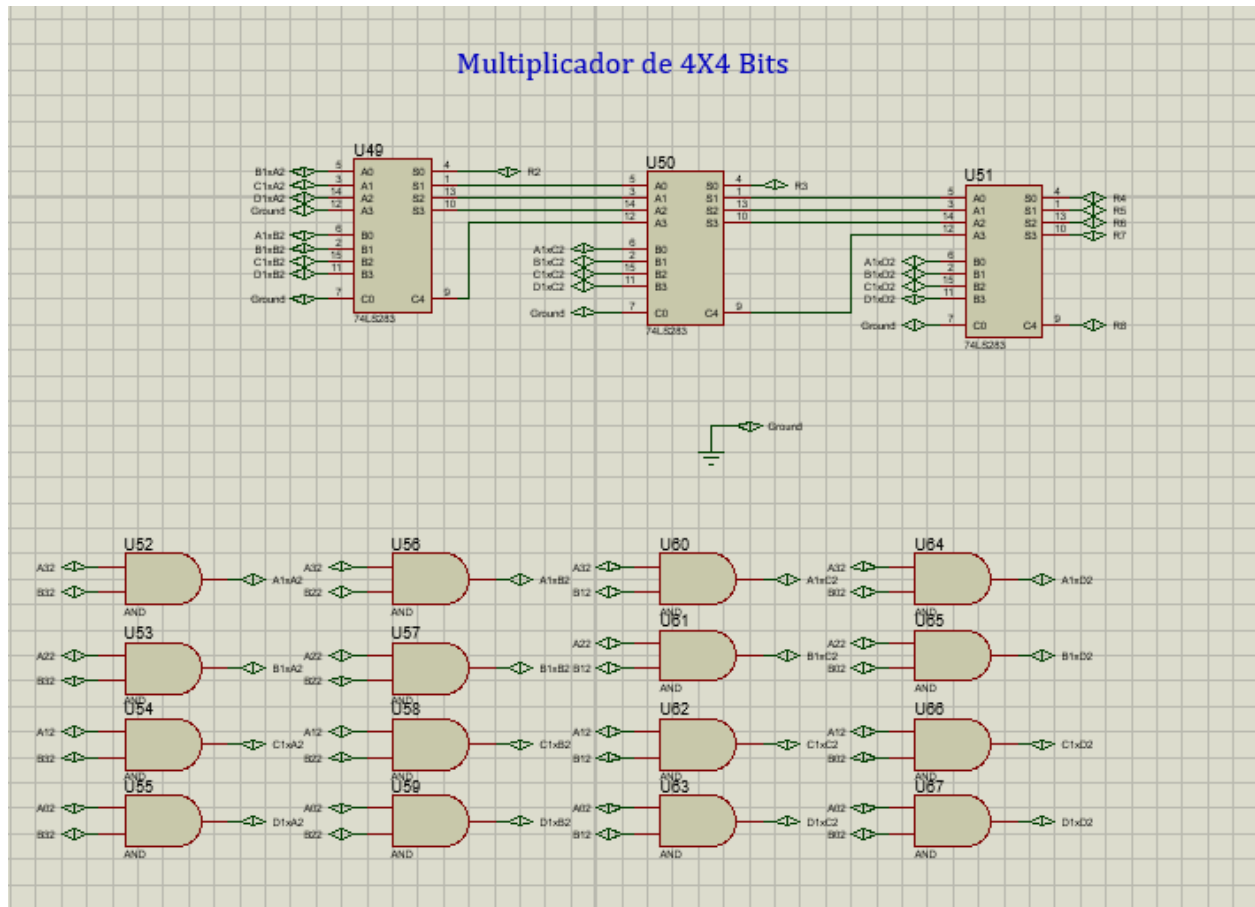
Bloque comparador



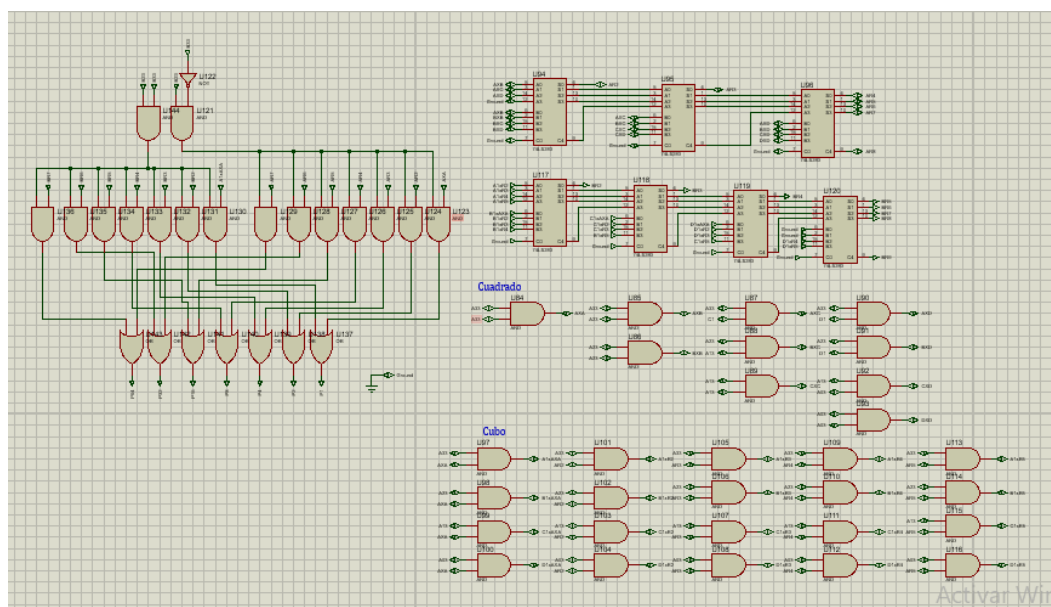
Sumador restador



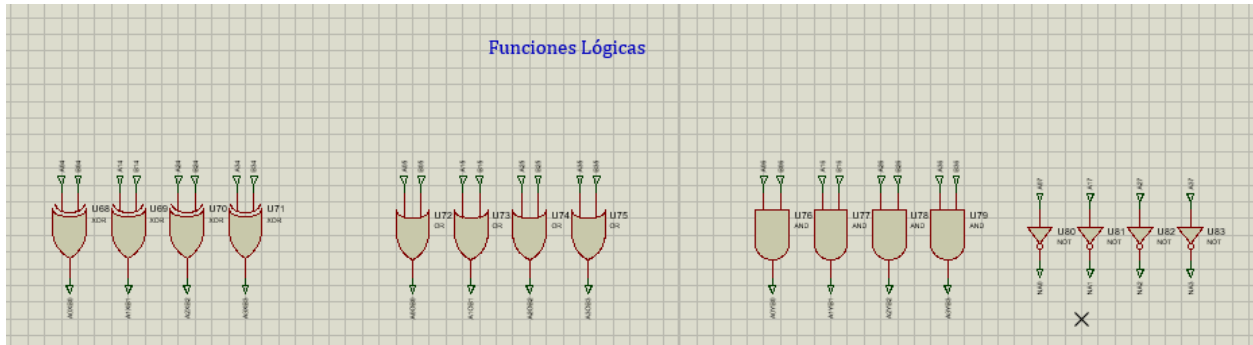
Multiplicador



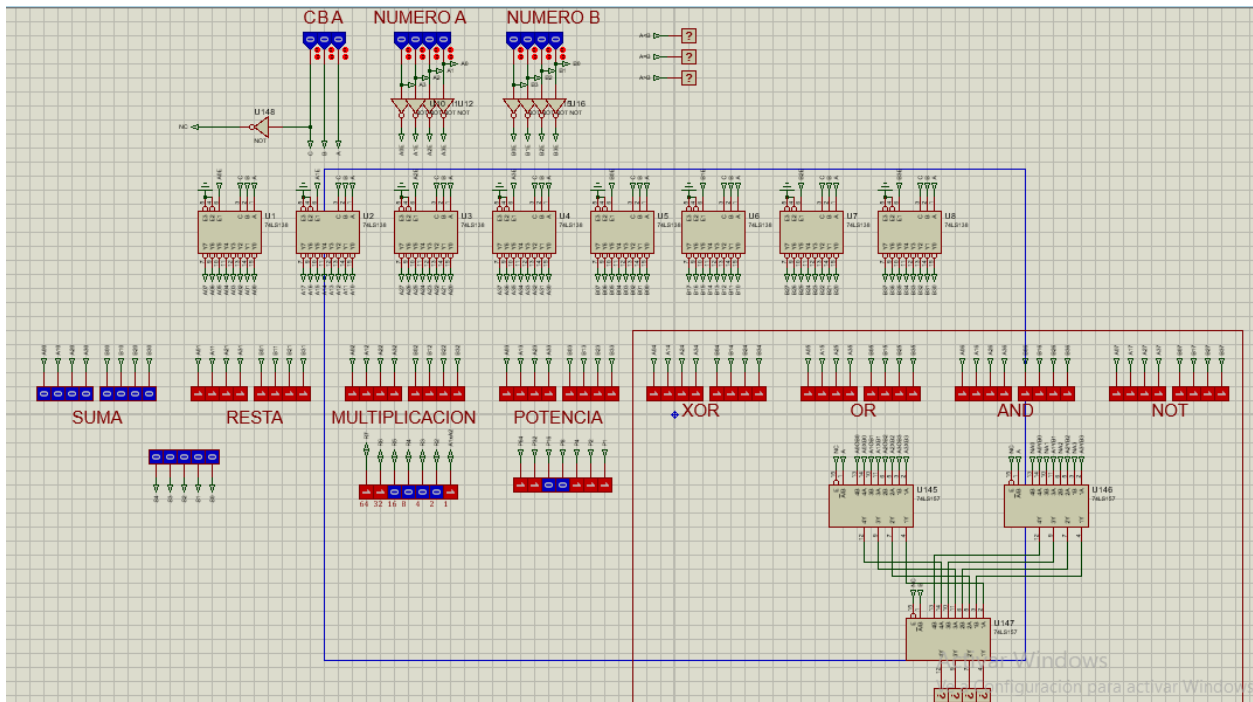
Potencias 2 y 3



Funciones lógicas



Controlador de entrada y salida de bits



Equipo Utilizado

- Protoboard
- Cable para protoboard rojo
- Cable para protoboard negro
- Diodos led
- Pulsadores de 4 pines
- Arduino UNO
- Jumpers
- Resistencias 220 ohm
- Resistencias 330 ohm
- Pinzas
- Corta cables
- Integrados:
 - AND
 - OR
 - NOT
 - XOR
 - Demux 74LS138
 - Mux 74LS157
 - Sumador 74LS283

Presupuesto

Item	Cantidad	Precio (Q)	Total (Q)
Integrado 74ls185	1	45.00	45.00
Protoboard	8	39.00	312.00
Integrado AND 74LS08	7	5.00	35.00
Integrado OR 74LS32	6	5.00	30.00
Integrado NOT 74LS04	4	5	20
Resistencias 220 Ohm	2	1.00	2.00
Resistencias 330 Ohm	11	1.00	11.00
Diodos Led	4	1.00	4.00
Cable para Protoboard Negro mt	3	2.00	6.00
Cable para Protoboard Rojo mt	3	2.00	6.00
Jumpers kit	2	48.00	96.00
Pulsador de 4 Patas	4	4.75	19.00
Arduino UNO	3	150.00	450.00
Fuente de alimentación de 9V	1	10.00	10.00
Cutter	1	14.00	14.00
Pinzas	1	13.00	13.00
Cortacables	1	59.00	59.00
Integrado 74LS138	8	7.00	56.00
Integrado 74LS157	7	7.00	49.00

Aporte Por Integrante**Mario Palma**

Realización de el redireccionamiento de los operaciones lógicas y aritméticas utilizando demultiplexores en Proteus de igual manera en físico en las protoboard, también hice el redireccionamiento de los resultados del bloque lógico y bloque aritmético para que solo muestre el resultado de la operación elegida, ayude a conectar a los demux todas las operaciones tanto del bloque lógico y aritmético y también hice las compuertas lógicas todo lo mencionado tanto como en proteus y en físico, también tuve un aporte en la documentación.

Erick Pineda

Realización de la lógica del comparador de 4 bits en proteus y realización en físico junto en colaboración al compañero Jeysson, colaboración en la realización de la documentación, colaboración en el testeo de los diferentes sistemas para ayudar a encontrar errores y apoyo moral a mi compañeros.

Eduardo Cruz

Realización de las funciones de multiplicación y potenciación. De igual manera, la realización del circuito de forma virtual en Proteus e igualmente en formato físico.

Jeysson Godoy

Realización del código en Python y su conexión con arduino tanto para el arduino físico como para el digital, colaboración en el comparador de 4 bits, ayuda general al momento de unir todas las partes de la práctica y colaboración en la elaboración de la documentación.

Ramiro Chacón

Realización del Redireccionamiento (DEMUX) Diseño del sumador/restador en proteus, realización de sumador/restador en físico. Unión de todos los circuitos en uno solo. Diseño del MUX para las operaciones lógicas, Diseño del MUX para las operaciones aritméticas. Testeo del circuito en físico y solución de errores.

Conclusiones

- Logramos construir una ALU básica que cumple con las funciones necesarias para realizar operaciones aritméticas y lógicas esenciales, demostrando su funcionalidad y eficiencia en distintas pruebas.
- Aprendimos detalladamente cómo funcionan los multiplexores, demultiplexores, comparadores y decodificadores, y aplicamos este conocimiento en el diseño y construcción de la ALU, lo que nos permitió manejar señales y datos de manera efectiva.
- Conseguimos hacer un diseño eficiente, utilizando la menor cantidad de dispositivos posible sin comprometer el rendimiento. Esto no solo optimizó el uso de recursos, sino que también simplificó el montaje y redujo costos.
- Aprendimos en profundidad cómo realizar operaciones lógicas, aritméticas y comparativas con números binarios. Este conocimiento fue fundamental para programar y probar la ALU, asegurando su correcto funcionamiento en diversas operaciones.

Anexos

Video

<https://1drv.ms/f/s!Ak7O0SrSpIUHgZowBlUPzWZzeJNSKw>

Fotografías

