Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías

Ingeniería en Computación

Act.6 (Jericalla-Evolución) BR + ALU + MemDatos

Reporte de Trabajo Colaborativo presentado por
Humberto de Jesus Peña Dueñas
Karla Rebeca Hernández Elizarrarás
Elizabeth Arroyo Moreno

alumnos de Cuarto Semestre ICOM

Arquitectura de Computadoras

Mtro. Jorge Ernesto Lopez Arce Delgado

Guadalajara, Jal. a 24 de Marzo de 2025

Introducción

La **Unidad Lógico-Aritmética (ALU)** es un componente fundamental dentro de una CPU, encargada de realizar operaciones aritméticas como suma, resta, multiplicación y división, así como operaciones lógicas como AND, OR, NOT y XOR. Además, desempeña un papel crucial en la ejecución de instrucciones secuenciales, garantizando precisión en el procesamiento de datos.

En esta actividad, se diseñaron e implementaron diversos módulos, cada uno con una función específica dentro del sistema, como el **banco de registros, memoria de datos, unidad de control y ALU**. Estos módulos fueron desarrollados utilizando conocimientos adquiridos en clases anteriores, incluyendo la codificación y funcionamiento de memorias **RAM y ROM**, lo que permitió una mayor comprensión del procesamiento de datos y almacenamiento de información en un sistema digital.

El **marco teórico** de esta práctica abarca conceptos clave en el diseño de hardware digital, incluyendo la arquitectura de procesadores, sistemas secuenciales, control de flujo de datos y estructuras de memoria. Se exploran también diseños similares utilizados en procesadores reales, junto con la teoría que respalda su funcionamiento, como la lógica combinacional y secuencial. En algunos casos, se pueden incluir **fórmulas matemáticas** para describir operaciones dentro de la ALU, como la suma binaria y las representaciones de números en complemento a dos.

Objetivos

Desarrollar un módulo de ALU en Verilog con dos entradas de 32 bits y una salida por operación, implementando de forma comportamental las operaciones **AND, OR, suma (ADD), resta (SUB), set on less than (SLT) y NOR**. Se utilizará un selector basado en una tabla de control para elegir la operación correspondiente, enviando ceros en caso de una opción no definida. Esta práctica permitirá reforzar el conocimiento sobre diseño digital y modelado de hardware en Verilog, aplicando principios fundamentales de las ALUs en procesadores.

Desarrollo

Con esta práctica realizamos 7 módulos cada uno con su respectivo papel:

ALU (Unidad Lógico aritmética)

- Realiza y toma en cuenta las operaciones lógicas entre 2 operadores como la suma, resta, AND, OR, comparación SLT(set or less than) y NOR.
- a y b operan, op define qué operación realizar con 4 bits.

Banco de Registros

- Simula un conjunto de registros de 32 bits de ancho y 32 registros de profundidad.
- Permite la lectura de dos registros (rd1 y rd2) simultáneamente y la escritura de un registro (wa) si la señal we está activa.
- La escritura ocurre en el flanco positivo de clk.

Módulo de Control

- Decodifica la instrucción en función del opcode de 4 bits.
- Genera las señales de control necesarias para el procesador:
    - aluOp: Indica la operación que debe realizar la ALU.
    - regWrite: Habilita la escritura en el banco de registros.
    - memWrite: Habilita la escritura en la memoria de datos.

DEMUX (Demultiplexor)

- Es un demultiplexor que separa la entrada in en out1 y out2 dependiendo de la señal sel:
    - Si sel = 1, dirige la entrada a out1 y pone out2 en cero.
    - Si sel = 0, dirige la entrada a out2 y pone out1 en cero.

Buffers

- Es un módulo simple que transmite la entrada in a la salida out sin modificarla.
- Se usa para almacenar temporalmente los datos antes de ser usados por otros módulos.

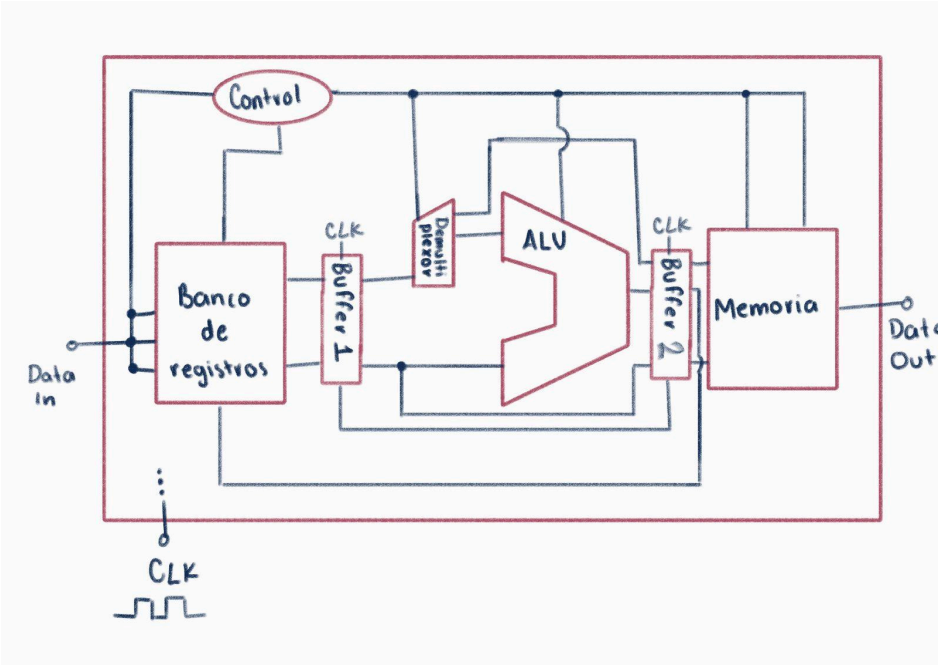
Memoria de Datos

- Simula una memoria de datos con 32 posiciones de 32 bits.
- Permite leer (rd) y escribir (wd) datos en una dirección de memoria (addr).
- La escritura ocurre en el flanco positivo de CLK si la señal we está activada.

Procesador (JericallaEvolucion)

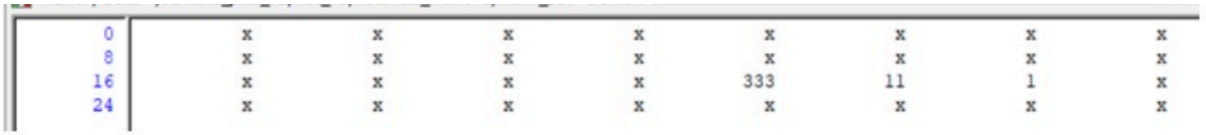
- Es la integración de todos los módulos anteriores para simular un procesador básico.
- Flujo de operación:
    1. Extrae datos del **Banco de Registros** (rd1, rd2).
    2. Pasa rd1 por un **Buffer** y luego por el **DEMUX** para separar datos si se va a escribir en memoria.
    3. La **ALU** procesa rd1 y rd2 con la operación indicada por el **Control.**
    4. La salida de la ALU pasa por otro **Buffer** y puede ser escrita en memoria o en los registros.
    5. Si la operación requiere memoria, la **Memoria de Datos** almacena o recupera datos.
    6. La salida final (dataOut) es el valor leído de la memoria.

Usando el siguiente diagrama: donde tiene de entrada y de salida 32 bits



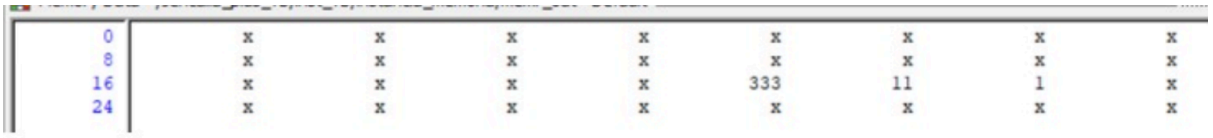
Los resultados obtenidos de la simulación al realizar las operaciones fueron los siguientes:

Antes de la ejecución



0	x	x	x	x	x	x	x	x
8	x	x	x	x	x	x	x	x
16	x	x	x	x	333	11	1	x
24	x	x	x	x	x	x	x	x

Después de la ejecución



0	x	x	x	x	x	x	x	x
8	x	x	x	x	x	x	x	x
16	x	x	x	x	333	11	1	x
24	x	x	x	x	x	x	x	x

Conclusión

En esta práctica fuimos capaces de implementar la ALU incorporando diversos módulos que incluyen: el banco de registros, la memoria de datos, el buffer, el demultiplexor y la unidad de control. Cada módulo cumplió con una función específica para poder funcionar correctamente.

En clases anteriores habíamos trabajado creando modelos de memoria y ALU lo cual nos preparó para comprender y diseñar este mini-procesador.

Al realizar dividir el código para poder llevarlo a cabo de forma organizada observamos las distintas funciones que son fundamentales para el manejo de la memoria un claro ejemplo es la memoria de datos donde mediante un archivo de texto obtuvimos los datos de entrada que se usaron para leer y escribir los valores con direcciones de memoria específicas y en la lectura de memoria se buscó no depender del reloj para que se garantizara el acceso inmediato de los datos.

Esta práctica ayuda a reforzar el funcionamiento de la ALU y la manera de direccionar datos usando un banco de registros y una memoria, procesando datos y haciendo pruebas, sin olvidar la importancia de los módulos en el módulo principal, analizando su eficiencia en las tareas que le asignamos.

Bibliografía

Stalling, W. (2005). Organización y Arquitectura de Computadores (7a ed.) Pearson-Prentice-Hall.

Patterson, D. A. (2007). Computer organization and design: The hardware/software interface. Morgan Kaufmann.

Abd-El-Barr, M., & El-Rewini, H. (2004). Fundamentals of computer organization and architecture (Vol. 38). John Wiley & Sons.

https://github.com/humbertoPena7/ACTIVIDAD-06-JERICALLA-evolucion.git