README

Este documento tiene como objetivo proporcionar una detallada información sobre cada uno de los módulos, mencionando sus funcionalidades y el cómo se integran entre sí para formar este un sistema funcional, así llevando a cabo un mejor uso del código. Se espera que sea digerible y comprensible para el usuario con objetivo que así conozca lo necesario para poder comprender y ejecutarlo sin preocupaciones.
es una especie de "cerebro" para una computadora muy simple, como una máquina que puede seguir instrucciones básicas, como sumar números o guardar información en la memoria.

Características

1. **Descripción de Módulos:** El proyecto consiste en varios módulos Verilog que representan componentes de un procesador, como la unidad de control, la ALU y la memoria RAM.
2. **Interconexión y Funcionalidad:** Los módulos están interconectados para formar un datapath, permitiendo el flujo de datos a través del procesador y ejecutando operaciones como señales de control, operaciones aritméticas y acceso a memoria.
3. **Organización:** Cada módulo desempeña un papel crucial en el funcionamiento del procesador, garantizando la ejecución adecuada de instrucciones. Están organizados de forma modular para facilitar su comprensión y mantenimiento.
4. **Visualización:** El código Verilog proporciona una abstracción de hardware, permitiendo visualizar cómo funcionan los componentes del procesador. Puede simularse y verificarse antes de la implementación física.
5. **Documentación y Comentarios:** Se incluye documentación para explicar la función de cada módulo, facilitando su comprensión y uso por parte de otros desarrolladores.

Descripción de los módulos

- UnidadControl: Genera señales de control según el Opcode de las instrucciones, dirigiendo el flujo de datos y operaciones en el procesador.
- SE (Sign-Extend): Extiende un valor de 16 bits a 32 bits, utilizado para operaciones tipo I que requieren valores inmediatos extendidos.
- SL32B (Shift Left 2 Bits): Desplaza 2 bits a la izquierda en un valor de 32 bits, útil para cálculos de direcciones de salto.
- RAM (Memoria de Acceso Aleatorio): Almacena y recupera datos de 32 bits, fundamental para operaciones de lectura y escritura.
- PC (Contador de Programa): Mantiene la dirección de la próxima instrucción a ejecutar y se actualiza en cada ciclo de reloj.
- Mux2a15B (Multiplexor 2 a 1 de 5 bits): Selecciona entre dos entradas de 5 bits según el valor de un selector, importante para seleccionar registros de destino.
- Mux2a1 (Multiplexor 2 a 1 de 32 bits): Similar al anterior, pero para señales de 32 bits, importante para seleccionar entre diferentes fuentes de datos.
- MemIns (Memoria de Instrucciones): Almacena instrucciones de 32 bits y las recupera según una dirección especificada.
- Registros: Implementa un banco de registros para leer y escribir datos de 32 bits, esencial para almacenamiento temporal y procesamiento de datos.
- AluControl (Controlador de ALU): Determina la operación que debe realizar la ALU según la función y el tipo de operación recibidos.
- ALU (Unidad Aritmético-Lógica): Realiza operaciones aritméticas y lógicas sobre dos operandos de 32 bits según la función especificada.
- Adder (Sumador): Suma dos números de 32 bits, comúnmente utilizado en operaciones aritméticas.
- Add4 (Sumador de 4): Agrega 4 al valor de entrada de 32 bits, útil para ciertas operaciones que requieren sumar un valor constante.
- Main (Módulo Principal): Conecta y organiza los componentes anteriores para formar el procesador completo.

¿Cómo lo uso?

1.- Instalación del software: Es necesario ModelSim así que asegurarse de tenerlo instalado es importante.

2.- Configuración del proyecto: Crear un nuevo proyecto en ModelSim e importar los archivos del código. Ve a “Project” -> “Add to Project” -> “Existing File...”.

3.- Compilación: Compilar el proyecto para verificar si este no contiene algún error y lleva integridad en este mismo.

4.- Cargar archivos de memoria: Se crean documentos, en este caso “instrucciones.txt” que contiene las instrucciones a cargar en el simulador y otro archivo “BR.txt” que contiene la configuración inicial del banco de registros.

5.- Simulación: Cargar los archivos de memoria y ejecutar la simulación en la parte de library.

6.- Visualización: Para poder ver la simulación visualizada necesitaras abrir la parte de y objects, copia todos los objects a la parte de wave y en la parte superior ejecutar “Run” tiene como icono un papel.

Agradecimientos:

Este proyecto fue posible gracias a todos sus integrantes y a la organización de equipo y por eso mismo, gracias por su valiosa contribución y apoyo a este proyecto

Juan José Rentería Haro

Javier Alberto Galindo Parra

Benjamin López Fletes

David Fernando Vázquez Mendoza