UNIVERSIDAD DE GUADALAJARA

CUCEI

Proyecto Final GUIA DE USUARIO

Integrantes

Alcaraz Suarez Gabriel Isaí

Guzman Marquez Alan Sabastian

Quintero González Diego Gerardo

Sanchez Agredano Kevin Manuel

19 de mayo del 2024

Arquitectura de Computadoras

D03, 2024A

Prof. Lopez Arce Delgado Jorge Ernesto

Fase 1

Introducción

Esta guía proporciona instrucciones para utilizar el DataPath implementado en Verilog para ejecutar instrucciones básicas tipo R en un procesador. El DataPath consta de varios módulos que se encargan de controlar el flujo de datos y realizar operaciones aritméticas y lógicas.

El Decodificador de Instrucciones MIPS es una herramienta que convierte instrucciones MIPS en formato ensamblador a su equivalente en código binario de máquina. Está diseñado para facilitar la conversión de programas escritos en lenguaje ensamblador MIPS a un formato más adecuado para la ejecución en un procesador MIPS.

Requisitos

- Archivos de Python (Decodificador.py, data.asm, data.asm, data.txt)
- Software de simulación Verilog (ModelSim)
- Archivos Verilog del proyecto (ALU.v, Memoria.v, BancoRegistros.v, UnidadDeControl.v, Multiplexor.v, Tipo_R, ALU_Control.v, DataPath_TipoR.v, tb_DataPath.v)

Funcionalidades Principales

- Abrir Archivo: Permite al usuario seleccionar un archivo en formato ASM (Assembly) que contiene instrucciones MIPS.
- Guardar Archivo: Permite al usuario seleccionar la ubicación y el nombre del archivo donde se guardará el resultado de la conversión.
- Decodificar y Guardar: Realiza la conversión de las instrucciones del archivo de entrada y guarda el resultado en el archivo de salida especificado.
- Salir: Cierra la aplicación.

Pasos para utilizar el decodificador

- Para escribir las instrucciones en ensamblador (.asm) para el decodificador, es necesario escribir cada parte de cada instrucción separada por comas, haciendo salto de línea, y respetando el orden que es el que se toman los bits para cada instrucción.
- Por ejemplo para las Tipo R tenemos: OP, rs, rt, rd, shamt y funct.

Los datos que debemos ingresar en lenguaje ensamblador para estas instrucciones son: OP, rs(registro 1), rt(registro 2) y rd (registro de destino), ya que shamt y funct se asignan automáticamente a los 32 bits.

- Estas instrucciones toman los datos que estan en el banco de registro y le dicen al procesador que hacer con ellos.
- Ejemplos:
- add,5,6,12
- sub,5,8,10
- slt,2,1,0
- or,1,1,11
- Abrir la aplicación.
- Seleccionar el archivo de entrada haciendo clic en el botón "Abrir".
- Especificar el archivo de salida haciendo clic en el botón "Guardar".
- Hacer clic en el botón "Decodificar y Guardar" para realizar la conversión.
- Una vez completada la conversión, se mostrará un mensaje indicando que el archivo se ha decodificado y guardado correctamente.
- Abre el archivo de salida data.txt para verificar que las instrucciones hayan sido decodificadas correctamente.

Pasos para utilizar el DataPath

Compilación de los archivos Verilog

- Abre el software de simulación Verilog.
- Crea un nuevo proyecto y agrega todos los archivos Verilog del proyecto.
- Compila los archivos Verilog para verificar que no haya errores de sintaxis.

Simulación del TestBench

- Abre el archivo TB_DPTR.v en el editor de texto del software de simulación.
- Simula el TestBench para verificar el funcionamiento del DataPath.
- Observa los resultados de la simulación para asegurarte de que las instrucciones se ejecuten correctamente.

Análisis de los resultados

- Analiza los resultados de la simulación para verificar que el DataPath esté ejecutando las instrucciones correctamente.
- Verifica que las señales de control (MemToReg, RegWrite, MemToWrite, ALUOp) se estén generando correctamente para cada instrucción.

Depuración de errores

- Si encuentras algún error durante la simulación, revisa el código Verilog de los módulos involucrados para identificar y corregir el problema.
- Vuelve a compilar y simular el TestBench para verificar que se haya corregido el error.

Conclusión

Podrás utilizar el DataPath implementado en Verilog para ejecutar instrucciones básicas tipo R en un procesador. Recuerda seguir los pasos cuidadosamente y realizar pruebas exhaustivas para asegurarte de que el DataPath funcione correctamente. Deberías poder utilizar el programa de Python para decodificar instrucciones de ensamblador MIPS. Si tienes alguna pregunta o necesitas ayuda adicional, no dudes en consultar la documentación de Python o en pedir ayuda a un experto en programación.

Fase 2

Introducción

Esta guía proporciona instrucciones para utilizar el DataPath implementado en Verilog para ejecutar instrucciones básicas tipo R en un procesador. El Single Cycle DataPath consta de varios módulos que se encargan de controlar el flujo de datos y realizar operaciones aritméticas y lógicas de tipo R por lo pronto.

El Decodificador de Instrucciones MIPS es una herramienta que convierte instrucciones MIPS en formato ensamblador a su equivalente en código binario de máquina. Está diseñado para la conversión de programas escritos en lenguaje ensamblador MIPS a un formato más adecuado para la ejecución en un procesador MIPS.

Requisitos

- Archivos de Python (Decodificador.py, data.asm, data.asm, data.txt)
- Software de simulación Verilog (ModelSim)
- Archivos Verilog del proyecto (ALU.v, Memoria.v, BancoRegistros.v, UnidadDeControl.v, Multiplexor.v, Tipo_R, ALU_Control.v, DataPath_TipoR.v, tb_DataPath.v, MemoriaDeInstrucciones.v, PC.v , Sign_Extend.v, Sumador.v)

Funcionalidades Principales

- Abrir Archivo: Permite al usuario seleccionar un archivo en formato ASM (Assembly) que contiene instrucciones MIPS.
- Modificar un archivo ya previamente guardado para agregar instrucciones en ensamblador
- Guardar Archivo: Permite al usuario seleccionar la ubicación y el nombre del archivo donde se guardará el resultado de la conversión.
- Decodificar y Guardar: Realiza la conversión de las instrucciones del archivo de entrada y guarda el resultado en el archivo de salida especificado.
- Salir: Cierra la aplicación.

Pasos para utilizar el decodificador

- Para escribir las instrucciones en ensamblador (.asm) para el decodificador, es necesario escribir cada parte de cada instrucción separada por comas, haciendo salto de línea, y respetando el orden que es el que se toman los bits para cada

instrucción.

- Por ejemplo para las Tipo R tenemos: OP, rs, rt, rd, shamt y funct. Los datos que debemos ingresar en lenguaje ensamblador para estas instrucciones son: OP, rs(registro 1), rt(registro 2) y rd (registro de destino), ya que shamt y funct se asignan automáticamente a los 32 bits.
- Estas instrucciones toman los datos que estan en el banco de registro y le dicen al procesador que hacer con ellos.
- Ejemplos:

  add,5,6,12

  sub,5,8,10

  slt,2,1,0

  or,1,1,11

- Abrir la aplicación.
- Seleccionar el archivo de entrada haciendo clic en el botón "Abrir".
- Especificar el archivo de salida haciendo clic en el botón "Guardar".
- Hacer clic en el botón "Decodificar y Guardar" para realizar la conversión.
- Una vez completada la conversión, se mostrará un mensaje indicando que el archivo se ha decodificado y guardado correctamente.
- Abre el archivo de salida data.txt para verificar que las instrucciones hayan sido decodificadas correctamente.

Pasos para utilizar el DataPath

- Compilación de los archivos Verilog
- Abre el software de simulación Verilog.
- Crea un nuevo proyecto y agrega todos los archivos Verilog del proyecto.
- Compila los archivos Verilog para verificar que no haya errores de sintaxis.
- Simulación del TestBench
- Abre el archivo TB_DaraPath.v en el editor de texto del software de simulación.
- Simula el TestBench para verificar el funcionamiento del DataPath inciando el ciclo de reloj en 0 para no ver errores al momento de correr el test.

- Observa los resultados de la simulación para asegurarte de que las instrucciones se ejecuten correctamente.
- Análisis de los resultados
- Analiza los resultados de la simulación para verificar que el DataPath esté ejecutando las instrucciones correctamente.
- Verifica que las señales de control (MemToReg, RegWrite, MemToWrite, ALUOp) se estén generando correctamente para cada instrucción.
- Depuración de errores
- Si encuentras algún error durante la simulación, revisa el código Verilog de los módulos involucrados para identificar y corregir el problema.
- Vuelve a compilar y simular el TestBench para verificar que se haya corregido el error.

Conclusión

Podrás utilizar el DataPath implementado en Verilog para ejecutar instrucciones básicas tipo R en un procesador. Es necesario cuidar el ciclo de reloj, que comienze en 0 cada vez que se vaya a ejecutar la simulación de Verilog.

Fase 3

Introducción

Esta guía proporciona instrucciones detalladas para utilizar el DataPath implementado en Verilog para ejecutar instrucciones básicas tipo R en un procesador. Además de las funcionalidades previamente existentes, se han agregado nuevas características al decodificador, incluida la capacidad de editar el archivo de entrada y una vista previa del mismo.

Requisitos

- Archivos de Python
- Software de simulación Verilog (ModelSim)
- Todos los archivos Verilog del proyecto

Funcionalidades Principales

- Abrir Archivo: Permite al usuario seleccionar un archivo en formato ASM (Assembly) que contiene instrucciones MIPS.
- Guardar Archivo: Permite al usuario seleccionar la ubicación y el nombre del archivo donde se guardará el resultado de la conversión.
- Decodificar y Guardar: Realiza la conversión de las instrucciones del archivo de entrada y guarda el resultado en el archivo de salida especificado.
- Editar Archivo (Nueva): Permite al usuario abrir una ventana para editar el código ensamblador del archivo de entrada seleccionado previamente.
- Vista Previa del Archivo (Nueva): Permite al usuario visualizar el contenido del archivo de entrada en una ventana dedicada para una mejor observación.
- Salir: Cierra la aplicación.
- Pasos para utilizar el decodificador
- Para utilizar la nueva funcionalidad de editar el archivo de entrada:
- Selecciona el archivo de entrada haciendo clic en el botón "Abrir".
- Haz clic en el botón "Editar Archivo" para abrir una ventana donde puedes modificar el código ensamblador.
- Realiza las ediciones necesarias y guarda los cambios.
- Para utilizar la nueva funcionalidad de vista previa del archivo de entrada:
- Selecciona el archivo de entrada haciendo clic en el botón "Abrir".

- Haz clic en el botón "Vista Previa del Archivo" para observar el contenido del archivo en una ventana dedicada.

Pasos para utilizar el DataPath

Compilación de los archivos Verilog

- Abre el software de simulación Verilog.
- Crea un nuevo proyecto y agrega todos los archivos Verilog del proyecto.
- Asegúrate de incluir todos los módulos necesarios, incluyendo ALU.v, Memoria.v, BancoRegistros.v, UnidadDeControl.v, Multiplexor.v, Tipo_R.v, ALU_Control.v, DataPath_TipoR.v, tb_DataPath.v, MemoriaDeInstrucciones.v, PC.v, Sign_Extend.v, Sumador.v, y Shift_left_2.v.
- Compila los archivos Verilog para verificar que no haya errores de sintaxis.

Simulación del TestBench

- Abre el archivo tb_DataPath.v en el editor de texto del software de simulación.
- Configura el testbench para iniciar el ciclo de reloj en 0. Esto evitará errores al momento de correr la simulación.
- Simula el TestBench para verificar el funcionamiento del DataPath.
- Observa los resultados de la simulación para asegurarte de que las instrucciones se ejecuten correctamente.

Análisis de los resultados

- Analiza los resultados de la simulación para verificar que el DataPath esté ejecutando las instrucciones correctamente.
- Verifica que las señales de control (MemToReg, RegWrite, MemToWrite, ALUOp) se estén generando correctamente para cada instrucción.
- Asegúrate de que los valores en los registros y la memoria coincidan con los resultados esperados de las instrucciones ejecutadas.

Depuración de errores

- Si encuentras algún error durante la simulación, revisa el código Verilog de los módulos involucrados para identificar y corregir el problema.

- Verifica que todas las conexiones entre los módulos estén correctas y que las señales estén correctamente asignadas.
- Vuelve a compilar y simular el TestBench para verificar que se haya corregido el error.

Conclusión

Ahora el decodificador cuenta con nuevas funcionalidades que mejoran su utilidad y experiencia de usuario. La capacidad de editar el archivo de entrada proporciona flexibilidad para realizar cambios directamente en el código ensamblador. Además, la vista previa del archivo ofrece una forma más conveniente de revisar el contenido antes de la conversión. Estas adiciones complementan las funcionalidades existentes y hacen que el proceso de decodificación sea más eficiente y versátil.