Rechnerarchitektur
Sommersemester 24

ertis embedded real-time systems ulm university institute

Laborprojekt – Versuch 4

In Ihrem letzten Versuch haben Sie eine auf Registerbefehle beschränkte RISC-V -Architektur mit Pipelining umgesetzt. In diesem Versuch erweitern Sie die ALU um die noch fehlenden Vergleichsoperatoren und ergänzen die Architektur so, dass Immediate-Befehle verwendet werden können.

Ziele des 4. Versuchs

- Konzeptionierung, Umsetzung und Validierung der Vergleichsoperatoren in der ALU
- Konzeptionierung, Umsetzung und Validierung einer RISC-V beschränkt auf Register- und Immediate-Befehle

Vorbereitung

Spezifikation RISC-V (R/I-Befehle)

Eignen Sie sich den Ablauf, Phasen und Ansteuerung für die Ausführung von Register-Befehlen basierend auf der Vorlesung und der Referenz (RV32I Base Integer Instruction Set, Version 2.0) an. Eignen Sie sich ein Grundverständnis so an, dass Sie die Befehlscodierung und die Ansteuerung mithilfe der Referenzkarten für jede Phase wiedergeben und erklären können. Erarbeiten Sie sich ein Konzept zur Umsetzung der RISC-V für Register- und Immediate-Befehle. Eignen Sie sich die Funktionsweise der Vergleichsoperatoren der Alu sowie die Ansteuerung der Operatoren mit Immediates für die Umsetzung der RISC-V an.

GHDL-Standards

Lesen Sie sich in die GHDL-Optionen, insbesondere wie Sie Standards der Sprache einstellen, ein. Ab sofort werden alle Komponenten und Testbenches nur noch mit VHDL 2008 genutzt.

Aufgaben

Aufgabe 1: Änderung der Konstanten

In dem Package für Konstanten hat sich ein didaktischer Fehler eingeschlichen. Die Werte *ADD_OP_INS* und *ADDI_OP_INS*, welche den OP-Code der R- bzw. I-Befehle vorhalten, werden im aktualisierten Package nun sinnvoller mit *R_OP_INS* und *I_OP_INS* bezeichnet.

- Ändern Sie die Konstanten entsprechend in Ihrem Code und Testen Sie diese Änderungen erfolgreich.

Aufgabe 2: Erweitern der ALU

- Erweitern Sie die Entity *my_alu* in ihrer Datei *my_alu.vhdl* um die noch fehlenden Vergleichsoperatoren der R-Befehle, setzen Sie dazu das in der Spezifikation beschriebene Verhalten um.
- Testen Sie Ihre Änderung erfolgreich mithilfe der zur Verfügung gestellten Testbench *my_alu_tb.vhdl*.

Aufgabe 3: Erweitern des Decoders

- Erweitern Sie die Entity *decoder* in ihrer Datei *decoder.vhdl* um den Fall, dass es sich um ein I-Format handel. In diesem Fall wird das *I_IMM_SEL* auf 1 gesetzt und für die Ansteuerung des Multiplexers vor der Alu genutzt.
- Testen Sie Ihre Änderung erfolgreich mithilfe der zur Verfügung gestellten Testbench *decoder_tb.vhdl*.

Aufgabe 4: RISC-V für I-Befehle

Erweitern Sie im folgenden mithilfe der Komponenten Multiplexer, Sign-Extension und den generischen Registern Ihre Umsetzung der RISC-V für I-Befehle.

- Kopieren Sie sich den Inhalt ihrer Datei *r_only_RISC_V_tb.vhdl* in die Datei *r_i_only_RISC_V_tb.vhdl*.
- Erweitern Sie die Datei *r_i_only_RISC_V_tb.vhdl* so, dass Sie in der ID-Phase die Immediates erzeugen und über ein Pipeline-Register speichern. Anschließend soll über einen Multiplexer, dessen Selektor über das Signal *I_IMM_SEL* des Steuerworts aus dem Decoder entschieden werden, ob an der ALU der Registerfile-Ausgang oder das Immediate anliegt.

- Entfernen Sie in Ihrem Registerfile die Initialisierung des ersten und zweiten Registers.
- Erweitern Sie die Instruktions ihrer Testbench so, dass diese nun über die I-Befehle das erste Register mit einer 9 und das zweite Register mit einer 8 überschreibt (z. B. Addition mit 0) und testen Sie diese dann erfolgreich gegen ihre angepasste Testbench.
- Erstellen Sie ein Bash-Skript im Ordner *<Projektordner>/Testbenches/RISCV*, mit dem Sie die Simulation der RISC-V für Register-Befehle inklusive der Analyse und Elaboration automatisch und erfolgreich ausführen können.

Abnahme *Funktion des RISC-V für Register-Befehle*, Programmierkonventionen, RISC-V Spezifikation der Registerbefehle.

Aufgabe 5: Abgabe

Laden Sie die erstellte Ordnerstruktur mit den dazugehörigen Dateien als Zip-Datei unter dem Namen

Name_Vorname_Versuch4.zip in Moodle hoch.