Rechnerarchitektur
Sommersemester 25

ertis embedded real-time systems ulm university institute

Laborprojekt – Versuch 5

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

Ziele des 5. 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: Erweitern der ALU

- Erweitern Sie die Entity *my_alu* in Ihrer Datei *my_alu.vhdl* um die noch fehlenden Vergleichsoperatoren der R-Befehle (`SLT`, `SLTU`), 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 2: Erweitern des Decoders

- Erweitern Sie die Entity *decoder* in Ihrer Datei *decoder.vhdl* um den Fall, dass es sich um ein I-Format handelt. 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 3: Anpassung des Registerfiles

In dem aktuell spezifizierten Register-File (`register_file`) werden Register asynchron ausgelesen und synchron beschrieben. Dies führt zu zwei wesentlichen Problemen. Die Inhalte der Register können sich bei einem gleichzeitigen Lese- und Schreibzugriff ändern uns sind nicht mehr konsistent. Dies führt, wie im vorherigen Versuch gezeigt dazu das schon in der WB-Phase die Daten überschrieben werden und an den Ausgang gelegt werden. Das ist problematisch, wenn die alten Daten erwartet werden, und führt zu einem Read-After-Write Hazard.

- Passen Sie Ihr Register-File so an, dass die zu lesenden Registerwerte bei einer positiven Taktflanke in den Signale `s_read1` und `s_read2` zwischengespeichert werden, bevor das Registerfile mit den neuen Werten beschrieben wird.
- Verbinden Sie die erzeugten Signale, welche den zwischengespeicherten Wert vor dem Schreiben enthalten, mit dem jeweiligen Ausgang außerhalb des Takt-sensitiven Prozesses.
- Testen Sie Ihre Änderung erfolgreich mittels der Testbench *register_file_tb.vhdl* zur Vermeidung des Read-After-Write Hazard.

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 wie in der Abbildung 1 dargestellt.

- Kopieren Sie sich den Inhalt Ihrer Datei *r_only_RISC_V.vhdl* in die Datei *ri_only_RISC_V.vhdl*.
- Erweitern Sie die Datei *ri_only_RISC_V.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 in der EX-Phase anliegt.
- Entfernen Sie in Ihrem Registerfile die Initialisierung des ersten und zweiten Registers, da dies nun über I-Befehle erfolgen kann.
- Testen Sie Ihre Umsetzung erfolgreich gegen die zur Verfügung gestellte Testbench *ri_only_RISC_V_tb.vhdl*.
- Erstellen Sie ein Bash-Skript, mit dem Sie die Simulation der RISC-V für R- und I-Befehle inklusive der Analyse und Elaboration automatisch und erfolgreich ausführen können.

Abnahme *Funktion des RISC-V für R- und I-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_Versuch5.zip in Moodle hoch.

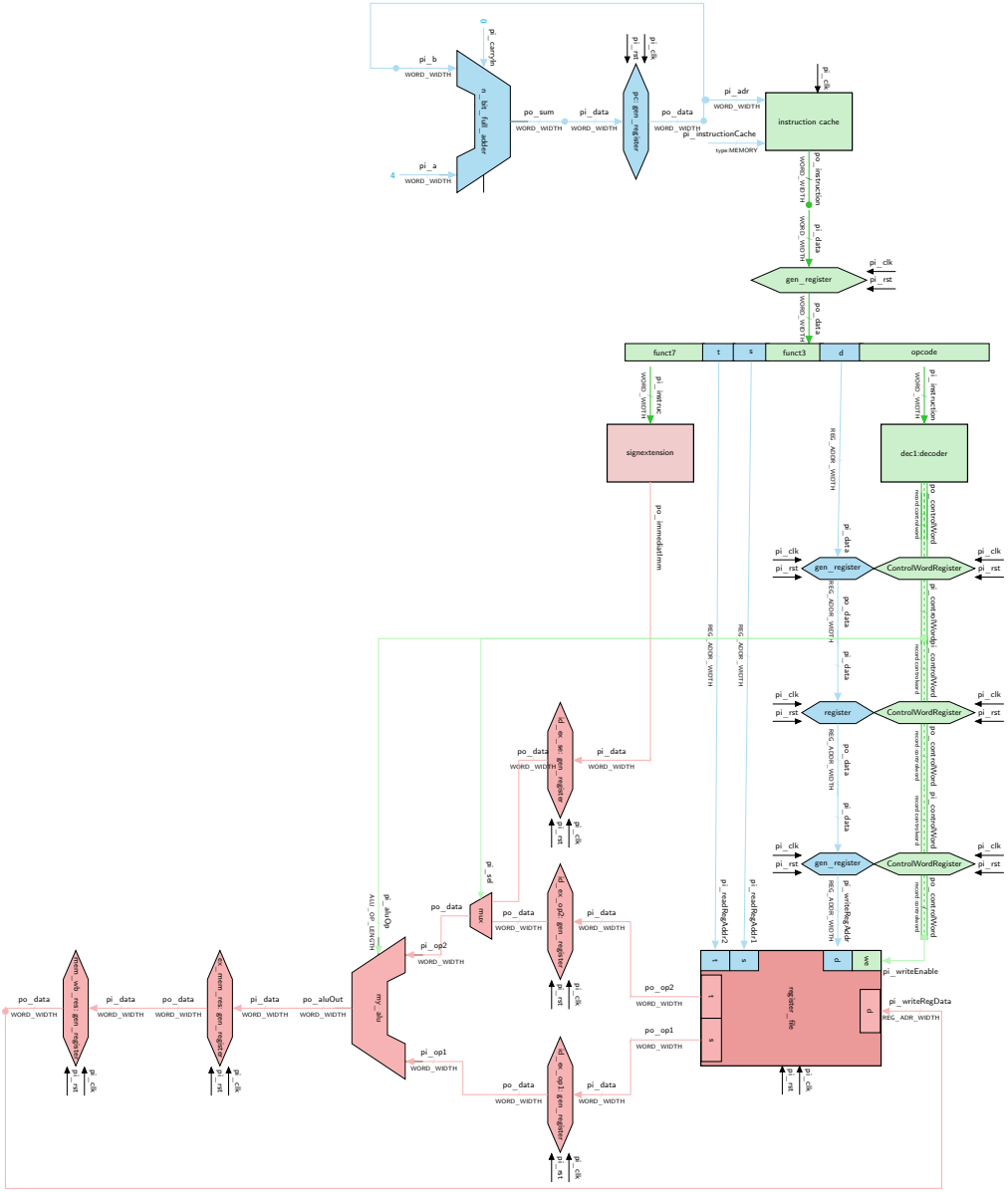

Abbildung 1: Mikroarchitektur der RISC-V beschränkt auf R- und I-Befehle