Rechnerarchitektur
Sommersemester 25

ertis embedded real-time systems
ulm university institute

Laborprojekt – Versuch 7

Dieser Versuch umfasst die Umsetzung bedingter Sprünge. Dazu erweitern Sie die vorhandene Architektur um einen zusätzlichen Addierer zur Bestimmung der Sprungadresse. Die Entscheidung, ob gesprungen wird, erfolgt durch die ALU mithilfe eines Zero-Flags sowie einer zusätzlichen Schaltung. Außerdem müssen bei einem ausgeführten bedingten Sprung bereits geladene Befehle gelöscht werden (Flushing). Dies wird durch Multiplexer realisiert.

Ziele des 7. Versuchs

- Erweiterung der ALU um ein Zero-Flag
- Konzeption, Umsetzung und Validierung bedingter Sprungbefehle
- Konzeption, Umsetzung und Validierung des Flushings

Vorbereitung

Spezifikation RISC-V (R/I/U/B-Befehle)

Erarbeiten Sie sich den Ablauf, die Phasen und die Steuerung zur Ausführung bedingter Sprungbefehle auf Grundlage der Vorlesung und der Referenzdokumentation (RV32I Base Integer Instruction Set, Version 2.0). Verstehen Sie die Befehlscodierung und die Steuerung so gut, dass Sie diese für jede Phase anhand der Referenzkarten erläutern können. Entwickeln Sie ein Konzept für die Umsetzung der RISC-V-Sprungbefehle und analysieren Sie die Herausforderungen bei bedingten Sprüngen.

Flushing im Pipeline-Prozessor

In einer gepipelten Prozessorarchitektur werden mehrere Instruktionen gleichzeitig in unterschiedlichen Phasen verarbeitet. Bei bedingten Sprüngen (z. B. `BEQ`, `BNE`) kann es jedoch passieren, dass bereits geladene oder begonnene Instruktionen im Pipelineverlauf fehlspeku-

lativ sind – also nicht mehr ausgeführt werden sollen, da der Programmzähler (PC) durch den Sprung verändert wurde.

Das sogenannte **Flushing** beschreibt den Prozess, bei dem diese fehlspekulativen Instruktionen gezielt aus der Pipeline entfernt werden. Dies verhindert, dass sie fälschlicherweise ausgeführt werden und das Ergebnis verfälschen.

Technisch wird Flushing häufig wie folgt umgesetzt:

- Durch das Einfügen von `NOP`-Instruktionen (No Operation) in die betroffenen Pipeline-Register
- Über Multiplexer, die bei Bedarf alle Steuer- und Datensignale in den folgenden Stufen auf einen definierten Zustand (z. B. null) setzen
- Alternativ durch einen gezielten Reset einzelner Register oder Pipeline-Stufen

In der Praxis wird Flushing ausgelöst, sobald ein Sprung als tatsächlich auszuführen erkannt wurde (z. B. in der MEM-Phase). Ab diesem Zeitpunkt müssen alle sich noch in der Pipeline befindlichen Instruktionen verworfen werden, die auf einer falschen Programmausführung basieren.

Flushing ist ein essenzieller Bestandteil moderner CPU-Designs mit Sprungvorhersage und Pipeline-Architektur und trägt zur korrekten Programmausführung bei.

GHDL-Standards

Lesen Sie sich in die GHDL-Optionen ein – insbesondere, wie sich Sprachstandards festlegen lassen. Ab sofort werden alle Komponenten und Testbenches ausschließlich mit VHDL 2008 genutzt.

Aufgaben

Aufgabe 1: Anpassung der ALU

- Erweitern Sie Ihre ALU um einen zusätzlichen Ausgang *po_zero*, der angibt, ob das Ergebnis der Operation null ist (*po_zero = '1'*).
- Testen Sie Ihre Erweiterung vollständig mit der bereitgestellten Testbench *my_alu_tb.vhdl*.

Aufgabe 2: Bedingte Sprünge

Die Umsetzung erfolgt durch:

- einen zusätzlichen Multiplexer zur Auswahl zwischen PC und Sprungziel
- einen Volladdierer zur Berechnung des Sprungziels aus dem PC und dem B-Immediate
- eine Logik zur Entscheidung, ob ein Sprung erfolgt

Für die Steuerung verwenden Sie das Signal *IS_BRANCH*. Die Befehle *BEQ* und *BNE* sind komplementär und lassen sich mithilfe eines Signals *CMP_RESULT* unterscheiden. Eine mögliche Bedingung für einen Sprung wäre:

```
is_branch AND (po_zero XOR cmp_result)
```

Beispiel: Bei *BEQ* und *po_zero = true* erfolgt der Sprung mit *CMP_RESULT = false*.

- Erstellen Sie eine Kopie der Datei *riu_only_RISC_V.vhdl* und benennen Sie sie in *riub_only_RISC_V.vhdl* um.
- Erweitern Sie den Decoder um die bedingten Sprünge und setzen Sie Ihr Konzept nachvollziehbar um.
- Ergänzen Sie die EX-Phase um einen Volladdierer zur Berechnung der Sprungadresse und ein entsprechendes Pipelineregister für die MEM-Phase.
- Integrieren Sie eine Logik zur Berechnung von *B_SEL* aus *CMP_RESULT*, *IS_BRANCH* und dem LSB des ALU-Ergebnisses. Halten Sie *B_SEL* bis zur MEM-Phase vor.
- Fügen Sie vor dem PC-Register einen Multiplexer mit *B_SEL* als Steuerungssignal ein. Die Eingänge sind: ursprünglicher PC-Eingang und Sprungziel aus der MEM-Phase.
- Ergänzen Sie ggf. den OPCODE der B-Befehle nach dem Sign-Extender.
- Testen Sie Ihr System mit der Testbench *riub_only_RISC_V_tb.vhdl*.
- Passen Sie ggf. das Bash-Skript im Verzeichnis *<Projektordner>/Skripte/RISCV* an und führen Sie es erfolgreich aus.

Aufgabe 3: Flushing

Bei der Ausführung eines (bedingten) Sprungs müssen spekulativ geladene Befehle gelöscht werden, bevor sie die MEM- bzw. WB-Phase erreichen. Dies kann durch:

- Multiplexer vor den Pipelineregistern
- oder mithilfe des vorhandenen Resets

realisiert werden.

- Setzen Sie ein Flushing-Konzept um und testen Sie Ihre Lösung mit der Testbench *riub_only_RISC_V_tb.vhdl*. Setzen Sie dazu in der Testbench die Konstante `WITH_FLUSH` auf $'1'$.
- Ergänzen Sie die Beschreibung Ihres Flushing-Konzepts in Form eines aussagekräftigen Kommentars.
- Passen Sie ggf. das Bash-Skript im Ordner *<Projektordner>/Skripte/RISCV* an und führen Sie es erfolgreich aus.

Aufgabe 4: Abgabe

Laden Sie die vollständige Ordnerstruktur inklusive aller Dateien als ZIP-Archiv unter folgendem Namen auf Moodle hoch:

Name_Vorname_Versuch7.zip