18.145 Rechnerarchitekturen und Mikrosystemtechnik

Übung: 3 Architekturentwurf – Pipelining Prozessor A. Mäder

Name	Matrikelnummer

Bei diesem letzten Übungstermin sollen Sie den Datenpfad aus der vorherigen Übung so modifizieren und erweitern, dass ein kleiner 16-bit Rechner entsteht. Er soll über folgende Eigenschaften verfügen:

RISC-Architektur sehr einfacher Befehlssatz

Die eigentliche ISA soll auf den einfachen ALU-Befehlen aufbauen, mit Erweiterungen für

- unbedingte Sprünge
- bedingte Sprünge: `opA < 0`, `opA = 0`, `opA = opB`, `opA < opB` ...
- Lade- und Speicheroperationen: `Memory(...)` → `opA`
  `opA` → `Memory(...)`

Adressierungsarten der ALU-Befehle wären: direkte- und Registeradressierung.

Für die Lade- und Speicheroperationen sollen Register-, Basis- und Befehlszähler-relative Adressierung möglich sein.

Harvard-Architektur also getrennte Adress- und Datenspeicher

Pipelinestruktur ähnlich der in der Vorlesung vorgestellten MIPS-Architektur. Es sollte also in jedem Takt eine Instruktion begonnen und in der Pipeline weiterverarbeitet werden.

Vorgehensweise

Anders als bei den vorherigen Aufgaben, wird der Inhalt hier nur grob skizziert. Details der Implementation, insbesondere die genaue ISA und die zugrundeliegende Hardwarearchitektur werden nicht speziell vorgegeben, sondern sollen zu Beginn der Übung im Plenum erarbeitet werden.

In der schriftlichen Ausarbeitung wird der Befehlssatz spezifiziert und die Hardwarestruktur als Blockschaltbild dargestellt. Anschließend wwird der Entwurf in VHDL implementiert und durch Simulation validiert.

Tipps und Infos

- Reihenfolge:
  1. ISA festlegen
  2. Architektur skizzieren, dabei für jeden Befehl überlegen, durch welche Teile der Hardware er in den jeweiligen Stufen der Pipeline ausgeführt wird.
  3. Die Kontrollstrukturen für die zeitliche Abfolge (Steuerwerk, Pipelinekontrolle) überlegen und mit geeigneten Hilfsmitteln, beispielsweise Zustandsübergangsgraphen bei endlichen Automaten, darstellen.
  4. ... dann mit der VHDL-Codierung beginnen
- Als „Template“ zur VHDL-Codierung kann die Musterlösung der vorherigen Aufgabe benutzt werden. Je nach Art der ISA ist die ALU anzupassen.
- *Welche Hardwarekomponenten sind zu ergänzen?*
  - Pipelineregister
  - Flag-Register (nach der ALU): $= 0$, < 0
  - Programmzähler: ladbar, eigener Inkrementer $(+1)$
  - Instruktionregister, ggf. als Folge von Registern entsprechend den einzelnen Pipelinestufen.
  - Steuerkomponenten der Pipelinestufen
  - Instruktions- und Datenspeicher: **vorgegebene externe Komponenten**
  - ...
- Die Speicheranbindung erfolgt über ein SRAM-Interface, siehe dazu auch das *„VHDL Archiv“*.

```
entity sram64kx16 is
port (  nCS      : in    std_logic;        -- not chip select
        nWE      : in    std_logic;        -- not write  enable
        nOE      : in    std_logic;        -- not output enable
        Addr     : in    std_logic_vector(15 downto 0);
        Data     : inout std_logic_vector( 7 downto 0 ));
end entity sram64kx16;
```

Aufgaben

Die ersten drei Aufgabenteile sind schriftlich zu lösen, die anderen beiden Aufgabenteile ergeben sich bei der Arbeit am Rechner.

III-1 Entwickeln Sie die ISA des 16-bit „Minimalrechners". Geben Sie dabei Codierung der Befehle und die verwendeten Formate an.

III-2 Zeichnen Sie die zugehörige Hardwarestruktur auf Register-Transfer Ebene.

III-3 Schreiben Sie ein kleines Programm, das später auf der Hardware ablaufen soll.

Beispiele:

- Berechnung der Fibonacci-Folge: $f_n = f_{-1} + f_{n-2}, n \geq 2$ mit $f_0 = 0, f_1 = 1$
- Multiplikation von zwei (unsigned) Zahlen
- ...

III-4 Codieren Sie die Architektur in VHDL.

III-5 (*optional*) und simulieren Sie Ihr Programm aus Aufgabenteil 3.