Praktikum zur Vorlesung Technische Grundlagen der Informatik

Prof. Dr. A. Koch
Thorsten Wink

TECHNISCHE UNIVERSITÄT DARMSTADT

Wintersemester 09/10
Praktikum

Abschnitt 1.1 Einleitung

Im Praktikum sollen Sie den Stoff der Vorlesung praktisch an umfangreicheren Aufgaben vertiefen. Das Praktikum ist in zwei Teile geteilt: eine Aufgabe in MIPS-Assembler und eine Aufgabe in Verilog. Die beiden Aufgaben können unabhängig voneinander bearbeitet werden. Auch ist es möglich, nur eine der Aufgaben zu lösen.

Abschnitt 1.2 Modalitäten

Durch die erfolgreiche Teilnahme an diesem Praktikum kann ein Bonus für die Prüfungen Technische Grundlagen der Informatik (PO 09) bzw. Technische Grundlagen der Informatik I (PO 07, 04, sonstige) und Technische Grundlagen der Informatik II (PO 07, 04, sonstige) erworben werden. Studierenden, die TGDI I und TGDI II besuchen, werden die Punkte auf beide Prüfungen angerechnet.
Insgesamt sind 60 Praktikumspunkte erreichbar. Diese werden in Klausurpunkte umgerechnet und auf die in der Klausur erreichten Punkte addiert. Für die Prüfung TGDI I entsprechen 4 Praktikumspunkte einem Klausurpunkt. Für die Prüfungen TGDI und TGDI II wird der Umrechnungsfaktor erst nach der zweiten Klausur festgelegt. Die Punkte aus dem Praktikum können auch zum Bestehen der Prüfung verhelfen.
Bei der Bearbeitung der Aufgaben ist Gruppenarbeit in Teams von 2 Studierenden gestattet. Um den Bonus zu erhalten müssen Sie sich als Gruppe bis zum 27.01.10 im Webreg anmelden. Zusammenarbeit in größeren Gruppen ist nicht gestattet, jedes Team muss eine individuelle Lösung einreichen. Wir werden dies überprüfen! Bei Täuschungsversuchen wird kein Bonus gewährt, in schweren Fällen kann die gesamte Veranstaltung als nicht bestanden gewertet werden.
Die Lösung ist bis zum 29.01.10, 23:59 per Mail an tgdi@esa.informatik.tu-darmstadt.de zu schicken. Der Betreff muss die Matrikelnummern und die Nachnamen der Teilnehmer enthalten. Es dürfen nur die selbst erstellten Quellcodedateien eingesendet werden, weiteres Material wird nicht verarbeitet.
In der Woche vom 01.02.10 - 05.02.10 finden in S202/E119 die Testate statt, hier müssen Sie den abgegebenen Code einem Tutor erklären, desweiteren wird hier der Test auf dem FPGA durchgeführt.
Hier noch einige Hinweise, die Sie beachten sollten:

- Verändern Sie die vorgegebenen Code-Teile nicht. Alle eigenen Entwicklungen müssen in den von Ihnen erstellten Dateien enthalten sein.
- Das Interface des Moduls darf nicht verändert werden.
- Kommentieren Sie Ihren Code! Unkommentierter Code wird NICHT bewertet! Auch die Kommentare werden bewertet!
- Ihr Code muss mit dem MARS-Simulator (MIPS) bzw. mit ISE (Verilog) simulierbar bzw. synthetisierbar sein. Code, der diese Kritierien nicht erfüllt, wird NICHT bewertet! Das heißt: Das mindeste, was wir von Ihnen erwarten, ist dass Ihre MIPS oder Verilog-Programme fehlerfrei durch den jeweiligen Simulator laufen. Ist das nicht der Fall, gibt es automatisch Null Punkte auf die entsprechende Teilaufgabe.
- Bei der Ermittlung der Punkte zählt nicht nur der abgegebene Code, sondern auch das Testat. Wenn Sie als Gruppe arbeiten, stellen Sie sicher dass beide Teilnehmer über alles Bescheid wissen.
- Während des Testats müssen Sie in der Lage sein, Ihren Code zu simulieren. Falls Sie nicht mit ISE arbeiten, stellen Sie sicher dass Sie die Bedienung von ISE kennen. ISE ist auf den Poolrechnern verfügbar.

- Während des Praktikums stehen Ihnen Tutoren mit Sprechstunden zur Verfügung, die Termine werden im Forum und auf der Web-Seite bekannt gegeben. Hier ist auch ein Test auf einem FPGA möglich.
- Lesen Sie während des Praktikums das Forum der Fachschaft unter d120.de. Dort werden evtl. Probleme diskutiert oder Hinweise bekannt gegeben. Veröffentlichen Sie dort (und auch nirgendwo anders) vor dem Abgabeschluss keinen Lösungs-Code, auch keine Fragmente! Dies kann als Täuschungsversuch gewertet werden!

Abschnitt 1.3 Teil 1: MIPS-Assembler (20 Punkte)

Schreiben Sie ein MIPS-Programm, welches die Determinante einer 3x3-Matrix berechnet. Verwenden Sie dazu den vorgegebenen Code-Rahmen der Datei `det.s`. Die Werte der Matrix sind zeilenweise im Word-Array `matrix` abgelegt. Das Ergebnis muss am Ende im Register $s6 gespeichert werden. Verwenden Sie die in der Vorlesung vorgestellten Konventionen für die Registerverwendung.
Als Erweiterung soll Ihr Programm auch Determinanten von 4x4-Matrizen berechnen können. Das vorherige Programm soll dabei als Unterprogramm verwendet werden. Die Dimension wird als Parameter angegeben, siehe dazu auch den Code-Rahmen.
Als Lösung soll die `det.s` mit dem Quellcode eingereicht werden. Sie muss mit dem MARS-Simulator simulierbar sein.

Abschnitt 1.4 Teil 2: Verilog (40 Punkte)

Abschnitt 1.4.1 Spezifikation

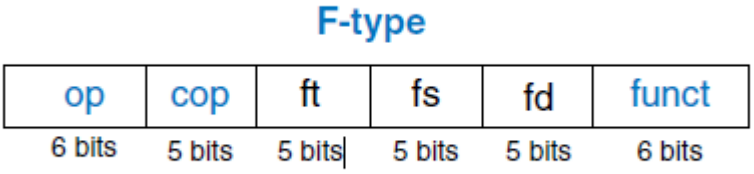
In der Vorlesung haben Sie in Kapitel 5 das Gleitkommaformat nach IEEE 754 kennengelernt. Im Praktikum werden Sie Module in Verilog beschreiben, die Gleitkommazahlen addieren, subtrahieren und multiplizieren können. Ihre Module werden in eine gegebene Verilog-Beschreibung des MIPS-Prozessors eingebunden, so dass Ihr Modul als Co-Prozessor dient (dies ist auch im realen MIPS-Prozessor so). Wir verwenden ein etwas verändertes Format der Gleitkommazahlen: 1 Bit Vorzeichen, 11 Bit Exponent, 20 Bit Mantisse.
Der bereitgestellte MIPS-Prozessor beherrscht die Operationen `ADD, SUB, AND, OR, SLT, LW, SW, BEQ, ADDI, J, lwc1, swc1`.

Abschnitt 1.4.2 Die Aufgabe - Überblick

Entwerfen Sie einen Co-Prozessor für den MIPS-Prozessor, der Gleikommazahlen nach der im vorigen Abschnitt definierten Spezifikation verarbeitet. Es sollen folgende Befehle unterstützt werden:

- add.s - Addition zweier (positiver) Gleikommazahlen
- sub.s - Subtraktion zweier (positiver) Gleitkommazahlen
- mul.s - Multiplikation zweier Gleitkommazahlen

Die Befehle sind alle ähnlich aufgebaut. Sie sind vom Typ F, der wie folgt aufgebaut ist:



Hierbei hat *op* den Wert 17, *cop* den Wert 16. *ft* und *fs* sind die Nummern der Quellregister aus dem Floating-Point-Registersatz, *fd* ist das Zielregister. *funct* gibt an, welche Operation ausgeführt werden soll. 0 steht für Addition, 1 für Subtraktion, 2 für Multiplikation. Nähere Infos finden Sie auch im Lehrbuch.
Vor Beginn einer Operation werden die Quellwerte aus dem Speicher in spezielle Register des Coprozessors geladen. Hierzu werden die Befehle `lwc1` und `swc1` unterstützt. Die Ansteuerung der Befehle ist bereits im Controller implementiert, nicht jedoch der Registersatz des Floating-Point-Prozessors.
Ihr Modul muss die folgende Schnittstelle haben:

```
module fpu(
  input        clk,              //Takt
  input        reset,            //Reset-Signal
  input[31:0]  instruction,      //aktueller Befehl
  input[31:0]  mem_readdata,     //Lesedaten vom Speicher zum Registersatz (fuer lwc1)
  output[31:0] mem_writedata,    //Schreibdaten vom Registersatz (Port 1) an Speicher (fuer swc1)

  input regdst,                  //Gibt an, welcher Teil des Befehls als Adresse fuer Read Port 1 des
                                 //FPU-Registersatzes verwendet wird
                                 //0: Zielregister aus dem FP-Befehl
                                 //1: instruction[20:16] (lwc1, swc1)
  input fpuregwritemux,          //Gibt an, woher die Daten am Write Port des Registersatzes stammen
                                 //0: Daten ist Ergebnis aus FPU
                                 //1: Daten aus Speicher

  input fpu_regwrite             //Write_enable des FPU-Registersatzes
);
```

Folgende Tabelle zeigt die Belegung der Signale:

Befehl	instruction	mem_readdata	mem_writedata	regdst	fpuregwritemux	fpu_regwrite
lwc1	Befehl	Lesedaten	X	1	1	1
swc1	Befehl	X	Schreibdaten	1	0	0
Floating-Point	Befehl	X	X	0	0	1

Desweiteren muss das Ergebnis von den Operationen innerhalb von einem Takt berechnet werden. Zum Testen wird als Ausgabe der Programm-Zähler auf die LEDs des Boards gelegt.

Abschnitt 1.4.3 Die Aufgabe - Implementierung

Folgende Dinge sind (grob gesagt) zu implementieren:

- Registersatz der FPU
- Floating-Point Addierer/Subtrahierer
- Floating-Point Multiplizierer
- Verschaltung der Module

Verändern Sie nur die Datei `fpu.v`. Sie können hierin beliebig viele Untermodule definieren. Das Interface der gegebenen `fpu.v` darf nicht verändert werden!

Abschnitt 1.4.4 Tests

Um Ihr Programm zu simulieren, müssen Sie Programmcode in den Programmspeicher laden. Dies geschieht über die Datei *memfile1.dat*. Jede Zeile entspricht einem Befehl. Die HEX-Darstellung können Sie z.B. mit dem MARS-Simulator ermitteln. Über die Datei *memfiledata.dat* können Sie Werte im Datenspeicher zu Beginn der Ausführung bereitstellen. Zur schnelleren Simulation ist in der Datei `divider.v`, die einen Taktteiler implementiert, ein `define`-Switch eingebaut. Zur Synthese muss Zeile 11 auskommentiert werden.
Es wird auf der Web-Seite ab dem 25.01.10 einen öffentlichen Testfall geben. Entwickeln Sie aber bereits vorher und auch danach eigene Testfälle! Im Testat werden weitere geheime Tests durchgeführt, die auch Dinge testen, die nicht im öffentlichen Test enthalten sind!
Im Testat wird Ihr Projekt synthetisiert und auf dem FPGA getestet. Eine fehlerfreie Synthese und ein lauffähiges Design wird ebenfalls überprüft und bewertet. In den Sprechstunden haben Sie die Möglichkeit, auf den im Testat verwendeten Rechnern zu arbeiten.

Abschnitt 1.5 Die verwendeten Tools

Sie sollten bereits in den Übungen den Verilog-Code der Übungsaufgaben simuliert oder synthetisiert haben. Hier noch einmal eine Anleitung zu den verwendeten Tools.
Zur Simulation und Synthese wird XILINX ISE 11.4 eingesetzt. Es ist auf den Poolrechnern der RBG installiert, aufzurufen mit dem Kommando `ise`. Deweiteren wird von XILINX die WebPack-Edition zur freien Nutzung zur Verfügung gestellt,

diese Version ist im Funktionsumfang eingeschränkt, reicht jedoch für das Praktikum aus. Sie ist sowohl für Linux als auch für Windows verfügbar, jedoch sehr umfachreich (4GB). Eine Anleitung zur Intallation steht auf unsere Web-Seite bereit.
Das FPGA ist ein Spartan 3E auf einem XILINX Starter-Kit Board. Hierzu ist eine umfangreiche Dokumentation auf der Web-Seite von XILINX verfübar.
Der MARS-Simulator kann unter http://courses.missouristate.edu/KenVollmar/MARS/ heruntergeladen werden. Er ist ebenfalls auf den Poolrechnern der RBG installiert.

Plagiarismus

Der Fachbereich Informatik misst der Einhaltung der Grundregeln der wissenschaftlichen Ethik großen Wert bei. Zu diesen gehört auch die strikte Verfolgung von Plagiarismus. Weitere Infos unter www.informatik.tu-darmstadt.de/plagiarism