Computer Organization and Networks Practicals 2020

October 7, 2020

Contents

0 Introduction

This document describes the tasks for the course "Computer Organization and Networks Practicals" for the winter term 2020. In this course, we are going to study computer architectures and networking stacks. We will discuss how CPUs are designed and how their interaction with memory works. Furthermore, we look at the TCP/IP stack including IPv4 and IPv6. Before presenting the assignments, we cover all organizational parts:

- Registration
- Assignment sheet
- Communication channels
- Tutorial videos
- Toolchain
- Question hours
- Submissions
- Task interviews
- Grading
- Plagiarism

0.1 Registration

The registration in TUGRAZOnline for this course ends on 2020-10-09. Please register for one of the ten groups. If you don't have a TUGRAZOnline account yet, please contact us before 2020-10-09 via con@iaik.tugraz.at.

If you participate in any submission process, you are going to get a grade at the end of the semester.

0.2 Assignment sheet

The main purpose of the assignment sheet is to specify what you need to do to receive a positive grade at the end of the semester. The assignment sheet (you are reading right now) can be found online (a link is also provided on the course website) in the upstream git repository at

`https://extgit.iaik.tugraz.at/con/practicals-2020`

This repository provides the latest version of the assignment. Be aware that not all exercises are included at the beginning of the semester. When providing an update of the assignment, we will push a new version to this repository and will announce it in `#con` and the newsgroup.

0.3 Communication Channels

We provide the following communication channels:

CON Email. We provide the email address con@iaik.tugraz.at for personal requests. Use this email only if you have a question, which cannot be discussed publicly.

Newsgroup. The newsgroup `tu-graz.lv.con` is used for general questions for the practicals. It is the best way to share questions and answers publicly. Course instructors or Teaching Assistants (TAs) are going to answer your questions here. The newsgroup is also used to announce updates for the practicals. So please read it regularly!

Discord. Discord is used to handle question hours and task interviews online. You need to register an account on Discord and then join the "IAIK" server. It helps your TA to recognize you if you don't pick an arbitrary username, but one related to your civil name. You can use the following invitation link:

`https://discord.gg/mxuUnjP`

- `#con` is a generic channel for all CON participants to ask questions in textual form. Besides the newsgroup, it is another place to ask questions textually, but be aware that TAs do not *have* to answer questions here.
- `#con-tutor` is a prefix used for audio-only channels per TA. During their respective question hour, you can ask questions here and your TA will answer them.

0.4 Tutorial videos

Date	Content
2020-10-09	Git and Logisim
2020-10-23	SystemVerilog: syntax and examples
2020-11-06	SystemVerilog finite state machines
2020-11-20	RISC-V and calling conventions
2020-12-04	C and network basics
2020-12-18	Network protocols

Table 0.1: Tutorial session videos.

Tutorial videos are prerecorded videos (c.f. Table 0.1) to be published on the day of the deadline of the previous exercise. The link will be shared on `#con` and in the newsgroup. The main goal is to introduce students to the next task and show them the required toolchain.

0.5 Toolchain

The toolchain is introduced in tutorial videos (Section 0.4), but we still want to document it here once more. The entire software stack, occuring in these practicals, is given by:

- git for version control (and a GitLab server for submissions)
- Logisim (→ Logisim homepage)
- SystemVerilog (→ IEEE 1800-2017)
- Yosys for synthesis (→ Yosys Open SYnthesis Suite)
- GTKWave for debugging (→ GTK+ based wave viewer)
- RISC-V (→ RISC-V ISA specification)
- asmlib, a python library (also on PyPI), to simulate RISC-V execution cross-platform in python
- The C and C++ programming languages

In our build scripts, we provide `Makefile`s which can be run with GNU Make. bash scripts are also going to be used.

As we don't want to bother students with installing software, we provide a virtual machine for VirtualBox. The virtual machine has the entire software stack preinstalled and is based on Ubuntu 18.04:

`https://seafile.iaik.tugraz.at/f/93576e64fa0e46e8a331/`

0.6 Question hours

	Tuesday	**Wednesday**	**Thursday**
08:00	Richard, Michael	Nikolaus	Amir
09:00	Cagdas, Martin		
10:00			
11:00		Črt	
12:00		Lukas	
13:00			Ferdinand
14:00			Moritz

Table 0.2: TA weekly question hour times.

Question hours are specific for your TA. They take place every week and you can look up your TA's day and time in Table 0.2. In the `#con-tutor` channel of your TA, you can ask any questions about the tasks during the one hour question time.

0.7 Submissions

At the beginning of the semester, you will receive account credentials for some GitLab instance. Using git, you can submit your deliverables in your personal git repository. To submit, pay attention to the following aspects:

- You need to *tag* your commit. The tag name format is `submission-task-x`, where `x` corresponds to the respective task. For example, the git tag for the submission of Task 1 is `submission-task-1`.
- After tagging the commit, don't forget to push your tag!
- You can check the state of your git repository by visiting your git repository in GitLab.
- If you tagged the wrong commit, you can delete the tag and tag the correct commit.

The latest possible submission deadlines for the respective tasks are given in Table 0.3. These timestamps are hard deadlines. If you submit your solution after a deadline, you will lose 8 points (from your achieved points on the respective task) per started 24 hours.

Task	**Deadline**	**Max. points**
Task 1 (AddSub Machine)	Fr, 2020-10-23 23:59	max. 15 points
Task 2 (PicoRISC-V CPU in SystemVerilog)	Fr, 2020-11-06 23:59	max. 20 points
Task 3 (Multiplier and CPU Integration)	Fr, 2020-11-20 23:59	max. 20 points
Task 4 (XGCD in RISC-V)	Fr, 2020-12-04 23:59	max. 15 points
Task 5 (Ping me)	Fr, 2020-12-18 23:59	max. 15 points
Task 6 (Talk UDP to me)	Fr, 2021-01-22 23:59	max. 15 points

Table 0.3: Task submission deadlines and maximum achievable points.

0.8 Task Interviews

There are going to be two task interviews (for three tasks each). The dates for the interviews will be organized by your TA and he/she will inform you ahead of time.

The interviews cover general questions of each topic and also discuss your particular solution you submitted. For your task interview, please join channel `#con-waiting-room` on Discord ten minutes before the interview. This is also the appropriate place to test your microphone. Your TA is going to pick you up at your designated time slot. Then both will join the `#con-tutor` channel of your TA. In `#con-tutor`, the task interview will be held.

The goal of interviews is to verify you did the implementation yourself, understood the topic and collect feedback on both sides.

0.9 Grading

The maximum points per task are listed in Table 0.3. Depending on your achieved points, you will get the associated grade according to Table 0.4.

0–50	Points	→	Nicht genügend (5)
51–62	Points	→	Genügend (4)
63–75	Points	→	Befriedigend (3)
76–87	Points	→	Gut (2)
88–100	Points	→	Sehr gut (1)

Table 0.4: Points to grade mapping.

0.10 Plagiarism

We will regularly check all submissions using automated plagiarism checking tools. If we detect a case of plagiarism, all involved people (the source and all sinks) will receive the grade `U` (Ungültig/Täuschung). Please also refer to the lecture slides for further information. Cases of plagiarism are handled as soon they are detected.

To avoid getting into a situation of plagiarism follow the following rules:

- Don't share code!
- Don't tell/dictate your solution to others!
- Commit regularly to show activity!

1 Task 1: AddSub Machine in Logisim

In the first task, you build an AddSub machine with Logisim, which is capable of performing a 4-bit addition and subtraction operation. The inputs for the computation are stored in four registers. You have two selection signals sel_a and sel_b to select the corresponding inputs. The output of the computation is written back to one of the four registers, determined by the selection signal sel_d. Furthermore, the machine supports loading a new value from the data input signal input into one of the registers. The control signals op and load determine, which operation is performed: adding, subtracting, or loading a new value. The detailed behavior is given in the specification below.

In Figure 1.1, you see the top level interface of the AddSub machine. Your implementation must have the same interface.

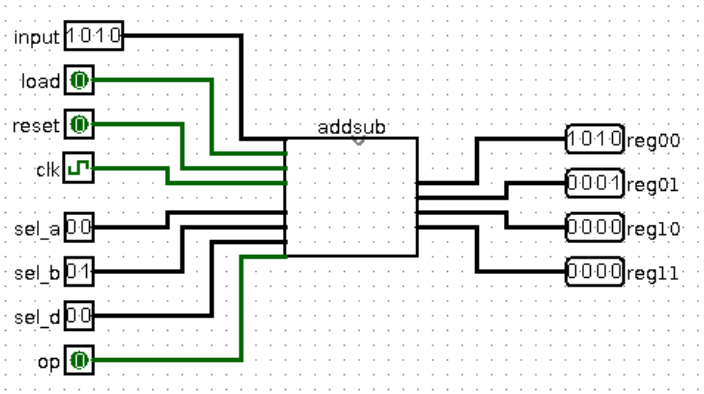

Figure 1.1: High-level view of the AddSub machine.

1.1 Specification

Your circuit must fulfill the following specification:

- You must not use building blocks from the "Arithmetic" section of Logisim.
- input is a 4-bit value, the selection signals are 2 bit values, and other signals are 1 bit values.
- On each positive edge of clk, the value in the register addressed by sel_d is updated.

- The destination register shall store the value input if load is 1 and store a computation result if load is 0.
- sel_a and sel_b select the inputs for the computation: sel_a defines from which register the operand a is read and sel_b defines from which register operand b is read.
- op selects whether an addition or subtraction is performed. If op is 0, the machine computes the overflowing 4-bit sum $a + b$. If op is 1, the machine computes the underflowing 4-bit difference $a - b$.
- If reset is set to 1, all registers are reset to 0.

In order to test your circuit, we provide you with two testcases:

- Testcase 1:
  1. Set input to 0001. Set load to 1. Set sel_d to 01.
  2. Create a positive edge. Set input to 1010. Set sel_d to 00.
  3. Create a positive edge. Set load to 0. Set sel_a to 00 and sel_b to 01.
  4. Verify that reg00 stores 1010 and reg01 stores 0001.
  5. Create a positive edge. Verify that reg00 stores 1011.
  6. Create a positive edge. Verify that reg00 stores 1100.
- Testcase 2:
  1. Set input to 1100. Set load to 1. Set sel_d to 11.
  2. Create a positive edge. Set input to 0010. Set sel_d to 10.
  3. Create a positive edge. Set load to 0. Set op to 1. Set sel_d to 11. Set sel_a to 11 and sel_b to 10.
  4. Create a positive edge. Verify that reg10 stores 0010 and reg11 stores 1010.
  5. Verify that with every consecutive positive edge, the value of reg11 cycles through 1000, 0110, 0100, 0010, 0000, 1110, 1100, and 1010.

1.2 Understanding Scientific Practice

Before you get started, check the documents we provide on plagiarism. It is part of Task 1 to read these documents and to confirm them.

- Understand what is plagiarism. You can find more information on this topic on plagiarism.org.

- Study the document "Guidelines on Safeguarding Good Scientific Practice".
- Understand the consequences described in Section 0.10 and in the lecture slides.

1.3 Deliverables

All files must be submitted in folder `task-1` of your repository.

1. After reading content according to Section 1.2, create a text file with the name `scientific_practice.txt` and write a statement that
   - you understood what plagiarism is,
   - you understood the consequences,
   - and that you won't submit a plagiarism.

   Add this file to your git repository.
2. The main circuit needs to follow the specification mentioned in Section 1.1. Name solution as file `addsub_machine.circ`. Add this file to your git repository.
3. Edit the `README.md` file and describe which parts of your submission are complete to give your TA an overview. Also add this file to your repository.

IMPORTANT Make sure to *commit* your files and *push* them to GitLab.

IMPORTANT Don't forget to create a tag and push the tags according to Section 0.7.

2 Errata

This chapter lists releases and changes of this file.

2020-10-07 Initial release.