SAARLAND UNIVERSITY
COMPUTER SCIENCE

Prof. Dr. Jan Reineke
Dr. Andreas Abel
Lucas Biehl

Systemarchitektur SS 2021

Aufgabenblatt 7

Sie können Ihre Lösungen bis **Mittwoch, dem 09.06.2021, um 10:00 Uhr** im CMS abgeben.
Geben Sie auf Ihrer Lösung Ihr Tutorium sowie die Namen und Matrikelnummern aller Gruppenmitglieder an.

Aufgabe 7.1: Verilog: Tri-Port-RAM

Eine MIPS-Instruktion liest bis zu zwei Register-Operanden und beschreibt maximal ein Register pro Takt. Das sogenannte Register File, ein RAM-Baustein der die Werte der Register hält, benötigt also insgesamt drei Ports: zwei zum Lesen und einen zum Schreiben.

Ein Tri-Port-RAM hat also folgende Ein-/Ausgänge:

- `clock`, das Taktsignal,
- `Aad, Bad, Xad`, die Adressen der zu lesenden (A, B) bzw. des zu schreibenden (X) Registers,
- `Aout, Bout`, die Werte der Register an Stelle `Aad`/`Bad`,
- `Xin`, die Eingabe für Register an Stelle `Xad`, sowie
- `we`, ob Register X überhaupt geschrieben werden soll.

Implementieren Sie in Verilog ein Modul `TriPortRAM`, welches Werte asynchron liest (d.h. sobald sich die Registeradresse ändert, ändert sich auch die Ausgabe) und synchron schreibt (d.h. bei steigender Taktflanke die aktuelle Eingabe übernimmt). Das Register File soll 32 32-Bit Register umfassen.

Schreiben Sie einen Testbench, der ein Register zunächst beschreibt und anschließend daraus liest. Verwenden Sie Icarus Verilog und GTKwave um Ihr Ergebnis zu visualisieren. Reichen Sie einen Screenshot Ihres Simulationsergebnis ein.

Hinweis: Verilog kennt Arrays. `reg[2:0] a[3:0];` *definiert ein Array mit Namen a, welches vier 3-Bit Einträge hat. Mit* `a[i]` *greifen Sie auf den i-ten Eintrag zu.*

Aufgabe 7.2: Dekodierer

Betrachten Sie den Dekodierer auf Folie 20/21 (Foliensatz 9).

(a) In der Vorlesung wurden rekursive und nicht-rekursive Formeln für die Kosten und die Tiefe hergeleitet. Beweisen Sie die Korrektheit der nicht-rekursiven Formeln bezüglich der rekursiven Formeln.

(b) Diese Dekodiererschaltung hat asymptotisch lineare Tiefe. Gibt es einen Dekodierer mit logarithmischer Tiefe? Wenn ja, geben Sie eine rekursive Schaltungsdefinition sowie rekursive Formeln für Kosten und Tiefe an. Wenn nicht, belegen Sie Ihre Behauptung.

Aufgabe 7.3: Befehlssatzarchitektur - MIPS

In dieser Vorlesung verwenden wir MIPS als Beispiel für eine Befehlssatzarchitektur. Unter `https://www.mips.com/products/architectures/mips32-2/` finden Sie aktuelle Informationen rund um die neuste Version MIPS32. Insbesondere das Dokument mit der detaillierten Beschreibung der einzelnen Instruktionen[1]

[1] `https://www.mips.com/?do-download=the-mips32-instruction-set-v6-06`

wird sehr nützlich sein. Wir beschränken uns in der Vorlesung auf eine Teilmenge der dort beschriebenen Instruktionen. Machen Sie sich mit dem Aufbau des obigen Dokuments zur Befehlssatzarchitektur vertraut, d.h. vollziehen Sie für die Instruktionen aus der Vorlesung deren Bedeutung sowie deren Kodierung nach. Sie werden dieses Dokument auch während des ersten Projekts benötigen.

- Dekodieren Sie folgende Instruktion (höchstes Bit links)
  `00110100101001000000000001100100`
- Kodieren Sie folgende MIPS-Instruktionen
  ```
  add R1, R2, R3
  ori R25, R12, 3637
  ```
- Betrachten Sie das folgende Programm. Welchen Wert hat Register 3 nach Ausführung des Programms?
  ```
  lui R2, 0x0102
  ori R2, R2, 0x0304
  sw R2, 4(R0)
  lb R3, 4(R0)
  ```

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Prof. Dr. Jan Reineke
Dr. Andreas Abel
Lucas Biehl

System Architecture SS 2021

Assignment 7

You may submit your solutions via the CMS until **10:00 a.m. on Wednesday, June 9, 2021**.
Please state on your solutions your tutorial, and the names and matriculation numbers of all team members.

Problem 7.1: Verilog: Tri-Port RAM

A MIPS instruction reads up to two register operands and writes at most one register per clock cycle. The so-called register file, a RAM block that stores the values of the registers, therefore requires three ports: two for reading and one for writing.

A tri-port RAM thus has the following inputs/outputs:

- `clock`: the clock signal
- `Aad, Bad, Xad`: the addresses of the registers to be read (A, B) or written (X)
- `Aout, Bout`: the values of the registers at the addresses `Aad`/`Bad`
- `Xin`: the input for the register at address `Xad`
- `we`: whether register X should actually be written

Implement in Verilog a module `TriPortRAM` which reads values asynchronously (i.e., as soon as the register address changes, the output also changes) and writes synchronously (i.e., it stores the current input on a rising clock edge). The register file shall contain 32 32-bit registers.

Write a testbench that first writes to a register and then reads from it. Use Icarus Verilog and GTKwave to visualize your result. Submit a screenshot of your simulation result.

Note: Verilog supports arrays. `reg[2:0] a[3:0];` *defines an array named* `a` *which has four 3-bit entries. Use* `a[i]` *to access the i-th entry.*

Problem 7.2: Decoder

Consider the decoder on slide 20/21 (slide deck 9).

(a) In the lecture, we derived recursive and non-recursive formulas for the cost and the depth. Prove the correctness of the non-recursive formulas with respect to the recursive formulas.

(b) This decoder circuit has an asymptotically linear depth. Is there a decoder with a logarithmic depth? If so, give a recursive circuit definition and recursive formulas for the cost and the depth. If not, prove your claim.

Problem 7.3: Instruction Set Architecture - MIPS

In this course, we use MIPS as an example of an instruction set architecture. At `https://www.mips.com/products/architectures/mips32-2/` you can find up-to-date information on the version MIPS32. In particular, the document that contains detailed descriptions of each instruction[2] will be very useful. We will limit ourselves in this course to a subset of the instructions described in that document. Familiarize yourself with the structure of the document, i.e., for the instructions from the lecture, look up their meaning as well as their encoding. You will also need this document for the first project.

[2] `https://www.mips.com/?do-download=the-mips32-instruction-set-v6-06`

- Decode the following instruction (the most significant bit is on the left)
  `00110100101001000000000001100100`
- Encode the following MIPS instructions
  ```
  add R1, R2, R3
  ori R25, R12, 3637
  ```
- Consider the following program. What is the value of register `R3` after executing the program?
  ```
  lui R2, 0x0102
  ori R2, R2, 0x0304
  sw R2, 4(R0)
  lb R3, 4(R0)
  ```