EECS 151/251A - LB
RISC-V FPGA Project

Alejandro Sierra
Jakob Peter Karg
UC Berkeley

Spring 2017

1 Design objectives

The goal of this project was to implement a complete version of a RISC-V CPU with a three stage pipeline. The CPU was implemented using Verilog and targeted the VIRTEX 5 LX110T FPGA by Xilinx.

1.1 Pipeline

The CPU was implemented with a three stage pipeline. Because the memory is implemented using synchronous bram blocks, two of the three stages were fixed to be at the instruction- and data-memory. We decided to put the third stage in between the register file and the ALU.

1.2 Memory hierarchy

The instruction- and data-memory are implemented using synchronous bram blocks. There are three memories: The instruction memory, the data memory and the BIOS memory. The BIOS memory is a read-only memory which contains the BIOS, which is executed when the CPU gets reset. The BIOS can read instructions over the UART and write them to the instruction memory. This way, the processor can be programmed to execute a program over UART. We are using memory-mapped IO to connect to various peripherals.

We implemented a memory-read- and a memory-write-controller that map the addresses to the corresponding memory / IO device.

2 High-Level organization

The CPU is broken up into three pipeline stages as described in 1.1. The first stage contains the instruction memory, the register file and the control logic for the next pc. The second stage contains the ALU as well as the calculation of the branch address and the branch condition and a memory write controller. The third stage contains the data memory and the memory mapped IO, the memory read controller and the writeback to the register file. In parallel to all of that there is the control unit, which takes in the instruction in the first stage and calculates the control signals for all multiplexers in the CPU. It detects data hazards and forwards the data through the necessary multiplexers. It also detects control flow hazards and sets the multiplexer which controls the next pc and kills the instruction in the second stage if necessary. The block diagram can be seen in figure 1.

Figure 1: Block diagram of the three stage pipeline

3 Detailed description of modules

Stage 1: Instruction Fetch and Decode

In the first stage, the instruction is read out of the instruction memory. The BIOS memory and the instruction memory are in parallel and there is a multiplexer, which is controlled by bit 31 of the program counter, which decides if the instruction from the instruction memory or the BIOS memory should be executed. Because all the memories are synchronous, it is important that the address input should be the address of the next instruction to execute, not the current one. So the address should be pc+4, not just pc.

After the instruction was read from the instruction memory, it gets decoded. This means that the registers specified by the instructions are read from the register files. These are asynchronous reads, so the data is available shortly after the instruction is available at the input of the register file.

3.1 Stage 2: Execute

In the second stage, the instruction gets executed. There are two multiplexers, one for each input of the ALU, which are controlled by the control unit (3.1). These multiplexers forward either the register output or the correct immediate, depending on the instruction, to the input of the ALU. They can also forward the output of the previous operation to the ALU if necessary. The ALU is controlled by the ALU controller, which takes in the instruction and passes an opcode to the ALU, which then executes the correct operation.

Also in the second stage, the branch address is calculated by a dedicated adder, which adds the pc and the immediate and the branch condition is calculated by the ALU.

Because the memories are synchronous, the memory write controller is also in the second stage. It takes in the data, which is either the output of the register file or the forwarded output of the previous operation if there is a data hazard and it creates write enables for the different memories and IO devices depending on the memory map. It also creates a data out signal, which contains the data that should be written to the memory. This is necessary

Stage 3: Memory and Writeback

The output of the ALU gets connected to the

Control unit