Thesis description

Carl-Johannes Johnsen (grc421)

17th January 2017

Background

Synchronous Message Exchange (SME) [1] is a programming model, which is similar to the Communicating Sequential Processes (CSP) model [2]. The key differences are that SME is globally synchronous, has broadcasting channels and a hidden clock. As such, SME is more suitable for devoloping hardware models than CSP, and is also an easier approach for programmers to generating hardware models.

MIPS (Microprocessor without Interlocked Pipeline Stages) is a reduced instruction set computer (RISC) instruction set architecture (ISA) [3]. The reason for selecting the MIPS instruction set, is due it being the architecture taught in the Machine Architecture class (ARK) at DIKU.

As taught in ARK, the basic MIPS processor consists of 5 stages: Instruction Fetch (IF), where the instruction is fetched from memory pointed to by an instruction pointer. Instruction Decode (ID), where the instruction is decoded, and where the register file is accessed. Execute (EX), where the actual computation is made. Memory (MEM), where data is read or written to memory. Finally, Write Back (WB), which takes the result, which is either from the EX or the MEM stage, and writes it back to the register file, if needed.

In the basic processor, one instruction is executed per clock cycle, and as such, the clock speed is determined by the amount of time needed to execute one instruction. By introducing a pipeline, the clock speed can be increased, as each instruction is divided into chunks.

By introducing a pipeline into the circuit, additional problems arise: Data hazards, where instructions depend on the result of a previous instruction, whose result is not ready in the registers. Branch prediction, since we need to compute the branch condition before branching, branch prediction is needed in order to reduce flushing the pipeline too much.

For making a hardware prototype, the circuit is written in VHDL (Very high speed integrated circuit Hardware Description Language), which is then written onto an FPGA (Field Programmable Gate Array).

Motivation and problem description

The ARK class only looks at the theory of hardware, i.e. how the logic works, how the different units are connected, and how the hardware affects the software. Therefore there is a gap in the education on the challenges of generating actual hardware.

This project aims to develop teaching material on how to design actual processor hardware by using SME. It will follow the same approach as the ARK course, and should be suitable for people with a computer science background such as myself.

Learning objectives

By the end of this project I expect to have obtained the ability to:

1. reason about differences in a software and a hardware program
2. describe an integrated circuit using SME
3. describe the evaluation of the correctness of an integrated circuit
4. communicate hardware design to software people

Major tasks to be performed along with time scheduling

Each step in this schedule will follow the same procedure: Studying the needed subject, implementing the feature/unit and finally documenting the process/writing report. The time division of the procedure will be 10% reading, 45% implementation and 45% documentation.

My plan of subjects are as follows:

Boolean circuits - This is meant to be an introduction to SME and hardware design. I will try to implement a few logic gates, implement communication amongst these and finally validate the correctness of the implementation.

Full adder - Now that we have some basic circuits, we can combine these to make a full adder. Although this is not needed with SME, it will give an insight on constructing arithmetic circuitry.

Single cycle MIPS CPU - With the adder in place, I can begin constructing the basic CPU. As taught in ARK, I start by constructing the single cycle CPU, i.e. in one cycle exactly one instruction is executed. As such, I need to implement more of the processor units, e.g. instruction decoding, the register file, the ALU. When I have each step covered, connecting them into the basic CPU should be straightforward.

Pipelined MIPS CPU - As mentioned in background, the single cycle CPU is not very efficient. So finally, I will introduce pipelines in the CPU, in order to increase the performance. Again, as mentioned, I will also need to handle the problems introduced by pipelining.

Week number	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Boolean circuits	▨	▨																	
Full adder			▨	▨															
Single cycle MIPS CPU					▨	▨	▨	▨	▨	▨	▨								
Pipelined MIPS CPU												▨	▨	▨	▨	▨	▨	▨	▨

References

[1] Brian Vinter & Kenneth Skovhede. *Synchronous Message Exchange for Hardware Designs.* © The authors and Open Channel Publishing Ltd. 2014.

[2] C.A.R. Hoare. *Communicating Sequential Processes.* © ACM 1978

[3] David A. Patterson & John L. Hennessy. *Computer Organization and Design - The Hardware/Software Interface (Revised 4th edition).* © Elsevier 2012