Hardware Design Lab

Report Group 6
Lukas Boland, Sven Ströher, Ana Carolina Ferreira, Julian Käuser
Supervisor: Sarath Kundumattathil Mohanan, M.Sc.
Begin: 07/31/2017 | Submission: 08/16/2017
Institute of Computer Engineering | Integrated Electronic Systems Lab | Prof. Dr.-Ing. Klaus Hofmann

TECHNISCHE
UNIVERSITÄT
DARMSTADT

1 Introduction

This semester's task in the HDL Lab is to design a general purpose processor implementing the ARM-Thumb2 architecture, and synthesize it with the Synopsis Design Compiler for the TSMC 45nm Standard Cell library. Further requirements are the RTL design with Verilog, the introduction of a pipelined design and the use of a given 1-Port memory. All these requirements have been met with our design, as pointed out in this report.
The paper is structured as following. At first, the different tasks are reviewed, and their distribution among the team members is outlined. An overview on our apporach is given. Then, the processor design and its submodules are reviewed. Furthermore, the RTL verification steps are shown. After a section on the synthesis process, the gate level verification results are shown. Finally, an evaluation of our design with regards to the task is done.

1.1 Tasks and Work Distribution

The task of the HDL Lab can be split into two major parts: micro-architecture design and rtl coding, and standard cell synthesis. Since the second part requires results of the first one, we started with the rtl design.

First, the necessary submodules in the processor were detected, and each team member was assigned one or more modules to work on. Creating and running a testbench for each designed module must always be performed in time with the module itself, so we assigned that task to everyone designing a module. Along with the module design, requirements regarding the Thumb2-architecture had to be checked and built in.

In the next step on the rtl level, finished and tested modules were integrated to verify their function. If necessary, a step back to the rtl design was taken in order to fix malfunctions in single modules or to adapt to changing requirements (e.g. changing communication between modules, new necessary signals).

The second major part, standard cell synthesis, was performed from the point where some modules were finished with the rtl design. Of course, for every change in the rtl design, the synthesis had to be re-done. With regard to the synthesis results for every module, some minor adaptions in the verilog code had to be done to guarantee the synthesizeability.

While the synthesis processes could only be started three days into the lab, they lasted until the final day. The RTL design also lasted until the end, since we tried to improve the design as far as possible. The approximate work distribution among the team members is shown below.

- Lukas Boland: RTL design (decoder) and top level verification
- Sven Ströher: RTL design (ALU, Stack), synthesis, gate level verification
- Ana Carolina Ferreira: RTL design (ALU, Fetch, controller), synthesis, gate level verification
- Julian Käuser: RTL design (memory interface, register file, controller), short report

2 Implementation

In this chapter, the designed micro-architecture are be presented. Achievements from verification (both RTL and gate levelare shown. Additionally, a view on the synthesis process and results is taken.

Our design implements the Thumb2-architecture almost completely. It is capable of executing all instructions except the `CMPZ` and `CMPNZ`, and `HINT` and `STATE CHANGE` (the last two types are explicitly not required and at least `STATE CHANGE` acnnot be implemented without an ARM-capable micro-architecture). We focused on implementing all instructions, since we interpreted the lab manual in this way. With these instructions, our processor is capable of executing both the `count32` and the `memcp46` benchmark applications.

2.1 Design

Our design is built of relatively large submodules, so that not many additional elements are necessary in the top module. The structure can be seen in Fig. 2.1. The main modules are the instruction decoder, the register file, the ALU, the memory interface and the inctruction fetch module. These modules will be described in detail in the following.

In the block structure, the modules are connected with several connections, but only the address and load inputs for the memory are busses with more than one input. All other connections can only be driven by one module, which simplifies the overall structure.

2.1.1 Processor Control

The control of the execution and memory access is performed by a state machine in the decoder module. Initially, a dedicated controller for the memory access and pipeline stalls was planned; while integrating and testing the decoder, the function of the controller was integrated into the decoder. It yet had many control functions in order to perform multi-cycle instruction like `MULTI-PUSH/-POP`.

In general, all control signals are outputs of the decoder. These include the ALU opcode, register selects, the flag updates and memory access signals. As the memory access must be arbitrated, the decoder controls the multiplexors assigning the read request and address signals from itself and the instruction fetch module. When the current instruction accesses the memory, a stall signal to the instruction fetch can be set by the decoder. On the other hand, the instruction fetch may stall the decoder (which then stalls the rest of the processor) if the next instruction is not loaded yet. With this mechanisms, we tried to keep the control of the processor as simple as possible.

2.1.2 Pipeline Stages

Our processor is comprised of two pipeline stages. One stage includes the instruction fetch and decoder, the other one does the execution and writeback tasks. These two stages have been chosen because of the structure of the benchmark applications and the 1-port memory. The applications are both relatively memory-access-loaded. The memory allows only one access per cycle; additionally, it is halfword-aligned, while the architecture uses byte-aligned addresses. The benchmarks both frequently access whole words

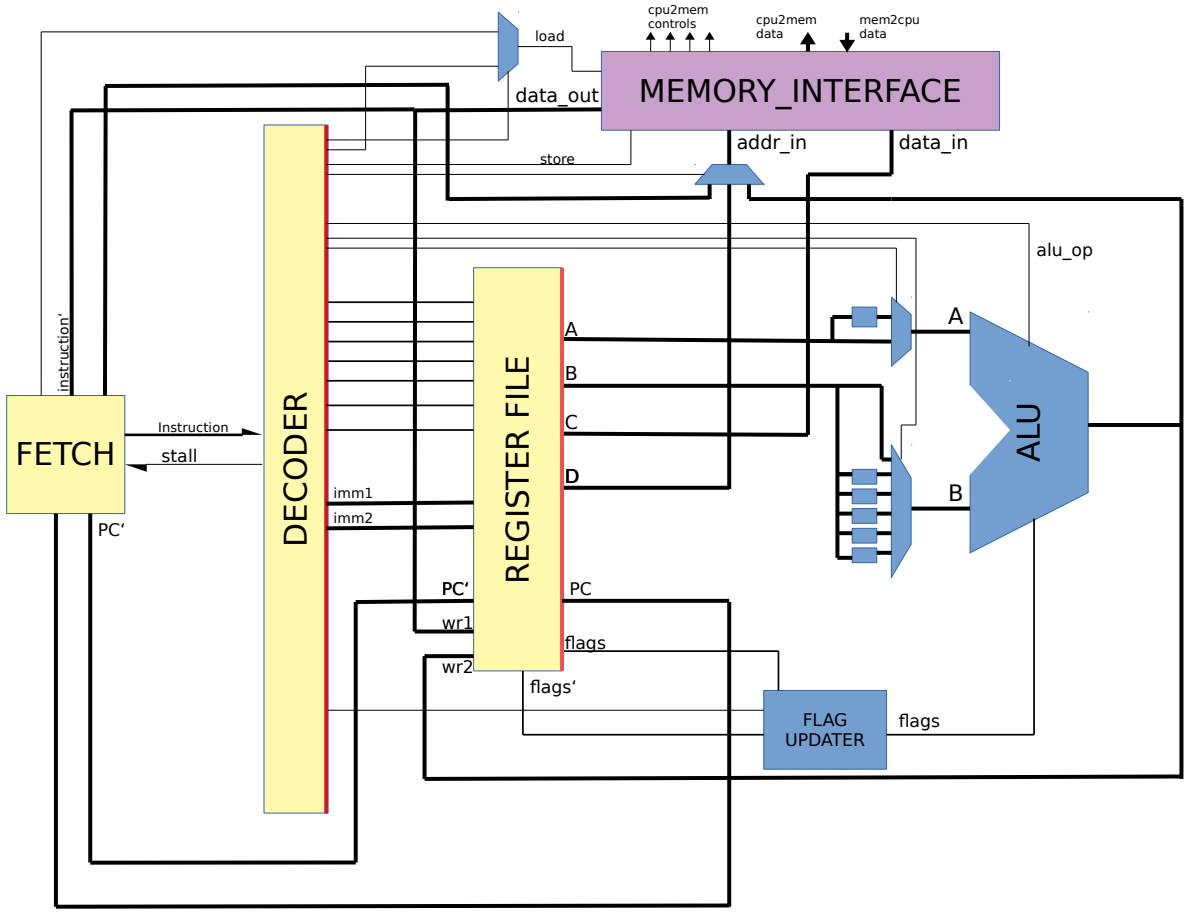

Abbildung 2.1: The block structure of the processor design. Register stages are marked red.

in the memory, so that in many cycles, the memory would be the bottleneck in the execution. With only two pipeline stages, few stalls would occur.

2.2 RTL Verification

2.3 Synthesis

During RTL coding, special attention was given to good coding practice for synthesis, in order to ensure that the synthesized hardware would behave exactly as specified, without any glitches or timing issues. Therefore, each module was separately and iteratively synthesized to make the debugging process more clear and modular. The main concern and reason for iterations was the need to avoid using latches in our design, which was solved by carefully assigning all outputs for all input conditions in case or if-else statements. The main reason to avoid the usage of latches is that they are asynchronous storage elements, which would potentially cause timing issues and racing in a fully synchronous design.

Once the final top module was completed, the synthesis iterations were started in order to obtain the fastest hardware possible based on the informations given in the timing report. Starting with a period of 1ns as a constraint, the value was continuously reduced until it was no longer possible to obtain a

slack greater than or equal to zero. A negative slack in the timing report means that it is not possible to generate a hardware to the given design that do not violate any timing request with the given period, while a positive or zero slack means that the synthesized hardware could be further optimized. The value used to synthesize the final hardware was 0.7 ns.

After the synthesis of the final hardware, an analysis of the schematic and the critical path was made in order to compare with the expected from the designed architecture. According to the timing report, the critical path consists of the instruction decoder reading a value from the register file, which is then processed by the ALU and written back to the register file. This path is expected due to the huge amount of combinational logic involved in the process, and the corresponding delay of the critical path is 0.69 ns.

As the guideline was to obtain a processor with the highest possible maximum frequency, there was no concern in optimizing the area or the energy consumption of the final hardware, nor to balance the three performance indicators. As a result, a processor with an area of 22453.12 μm^2 and a power of 6,9367 mW was obtained.

2.4 Gate Level Verification

After the synthesis, a gate level simulation is required to check if the behaviour of the final hardware corresponds to the logic simulation performed in the RTL coding stage, as well as to determine the maximum clock frequency under which the processor can run without any timing violations.

The two timing violations that must be avoided are the setup and hold time violations. A setup violation occurs when a signal does not arrive at a flip-flop with sufficient time, the so called setup time, before the next rising edge of the clock cycle. This means that the circuit is too slow for the critical path. On the other hand, the hold time of a flip-flop is the time that the input signal must remain there after the rising edge of the clock for the output to be safely switched. Therefore, a hold time violation occurs in the shortest paths and means that the circuit is too fast in that region.

With these concepts in mind, a testbench was run for each of the benchmark programs provided for the validation of the final synthesized processor design. Both testbenches were repeatedly run for increasing values of clock speed until any of them presented any timing violations. With this method, the lowest reached clock period under which the processor ran without any errors was 1.26 ns, which corresponds to a maximum frequency of 793 MHz. The final value of the clock period is way different than the one used as a timing constraint during the synthesis process, which was 0.7 ns.

2.5 Other

3 Evaluation

here some stuff on how good this processor is