Finalised Project Proposal

University College London

Department of Electronic & Electrical Engineering

Performance characterisation of 8-bit RISC and OISC architectures

Author:
Minduagas Jarmolovicius
zceemja@ucl.ac.uk

Supervisor:
Prof. Robert Killey
r.killey@ucl.ac.uk

October 13, 2019

1 About

In the past decades some processor architectures become more dominant than others in specific fields. One of noticeable cases is ARM RISC (Reduced Instruction Set Computer) architecture being used in mobile devices instead of more popular and robust x86 CISC (Complex Instruction Set Computer) architecture in favour of simplicity, cost and lower power consumption [5, 6]. This project will take a step back and investigate benefits of OISC[1] (One Instruction Set Computer) comparing to RISC architecture.

2 Aims and Objectives

The aim is to compare similar characteristic RISC and OISC architectures to determinate advantages and trade-offs following points:

- Which processor is easier to implement and expand;
- Which processor requires less resources to implement;
- Which processor performs on common benchmark;

Possible application of both architectures could be use inside of microcontroller or SoC (System on a chip) systems similar to 8bit Atmel AVR or Mirochip PIC microcontrollers, therefore processors must be capable of controlling and communicating with external modules such as UART[2] and GPIO (General Purpose Input/Output).

2.1 Objectives

In order to achieve defined aims following objectives are required:

- Design RISC architecture;
- Design and implement RISC microarchitecture;
- Develop a compiler for assembly and higher level programming language for RISC;
- Design OISC architecture;
- Design and implement OISC microarchitecture;
- Develop a compiler for assembly and higher level programming language for OISC;
- Develop a benchmark to test real life performance;
- Compile and run benchmark on both processors using written compiler;
- Compare power consumption of both systems;

[1] Also known as URISC (Ultimate Reduced Instruction Set Computer)
[2] Universal asynchronous receiver-transmitter

3 Preliminary Approach to Achieving Objectives

In order to compare RISC and OISC architectures the first both architectures need to be designed and implemented. Design will be done using SystemVerilog HDL (Hardware Description Language) and further simulated & tested using ModelSim. Further on both architectures will be implemented on Terasic DE0-Nano board which is based on Altera Cyclone IV E FPGA (Field Programmable Gate Array).

Finding which architecture requires less resources can be done by comparing number of LULs (LookUp Tables) in a FPGA, and measuring power consumption of FPGA while running benchmark across different clock frequencies.

3.1 RISC Design

Table 1 represents desired RISC processor instruction set. Whole instruction takes 8 bits, 4 most significant are for instruction, 2 bits after that is source/destination register address (rd) and 2 least significant bits are for source register address (rs). In addition, minor changes may be done to this instruction set during optimisation in order to achieve better performance or easier design.

Instr.	Description	Operation
COPY	Copies value from register/immediate to register	*if rs=rd then [rd] = imm else [rd] = [rs]*
ADD	Arithmetic Addition	*[rd] = [rd] + [rs]*
SUB	Arithmetic Subtraction	*[rd] = [rd] - [rs]*
AND	Bitwise AND	*[rd] = [rd] AND [rs]*
OR	Bitwise OR	*[rd] = [rd] OR [rs]*
XOR	Bitwise XOR	*[rd] = [rd] XOR [rs]*
GT	Greater Than	*[rd] = [rd] > [rs]*
EX	Extended instruction to perform arithmetic and logical shift and sign/unsigned integer conversions	*[rd] = func([rd], [rs]{0,2});* func defined by *[rs]{7,3}*
LW	Load word from coprocessor	*[rd] = CP([rs])*
SW	Store word to coprocessor	*CP([rs]) = [rd]*
JEQ	Jump on condition	*if [rd]=0h then CP = [rs]*
JUMP	Unconditional jump	*CP = [rs]*
SMP	Set memory page (allows to extend memory address to 16bit)	*MP = [rd]*
SCO	Set coprocessor (allows communication to external modules)	*CP = [rd]*
PUSH	Push register on top of stack	*SP + 1; mem(sp) =[rd]; [rd] = 0h*
POP	Pop register from top of stack	*[rd] = mem(sp); SP - 1*

Table 1: Instruction set for RISC processor

In table 1 operation *[x]* indicates value in address x, *x{a,b}* indicates bits from a to b. Other notations

CP - coprocessor
MP - memory page
SP - stack pointer
PC - program pointer
mem - system memory

Coprocessor is a general term for anything that process data outside main processor e.g. floating point calculation module, communication block for UART, etc.

The microarchitecture is shown in figure 1. This diagram based on MIPS processor described in [2]. This is only simplified diagram that does not include pipelining, hazard unit and other more complicated structures that may be implemented into microarchitecture.

Figure 1: RISC processor microarchitecture block diagram

3.2 OISC Design

There are multiple implementations of OISC architecture, in this project MOVE variant will be designed. This processor would have separate address and data buses that connect all blocks such as memory, ALU (Arithmetic Logic Unit) or memory, highly simplifying design, see figure 2. Instruction timing hazards could be prevented using basic instruction look-ahead checks or implemented into compiler. The simplicity should allow implement more

Figure 2: OISC processor microarchitecture basic block diagram

straightforward pipelining and/or allow multiple instruction execution at the same clock cycle which is predicted to have better performance than RISC implementation. Design of OISC would be further developed after RISC implementation.

3.3 Benchmark

Further research will be carried out in order to make suitable benchmark. Benchmark is required to reliably simulate real world scenarios where such processors could be applied. Some of these test may include calculating prime numbers, software cryptography, processing data from/to UART.

4 Preliminary Assessment of Safety Risks

4.1 Safety Risks

This project will be mainly carried out in a laboratory and a computer room. Table 2 represents relevant risks.

Task	Hazard	Risk	Control
HDL Design	Working in a lab Working on a computer	RSI - **Low** Eye strain - **Low** Bad posture - **Medium**	Properly use mouse/keyboard Make constant breaks Make sure sit properly
Circuit test	Working with hand tools Accident short-circuits Incorrectly connecting cables	Skin cuts - **Medium** Burns from hot wires/chips - **Medium**	Use proper procedure while handling tools Use current limit on power supplies

Table 2: Risk Assessment Grid

Overall risk level is very low / trivial.

4.2 Failure Risks

Below are possible risks that this project might fail and mitigations to prevent that:

- **Not having enough time** - ensure project schedule is followed, possibly use already developed publicly available RISC or OISC architecture if deadlines is not met;
- **Damaging FPGA board** - have planned access to other FPGA board;
- **Computer data loss** - use version control (git) with remote server;

5 Expected Schedule

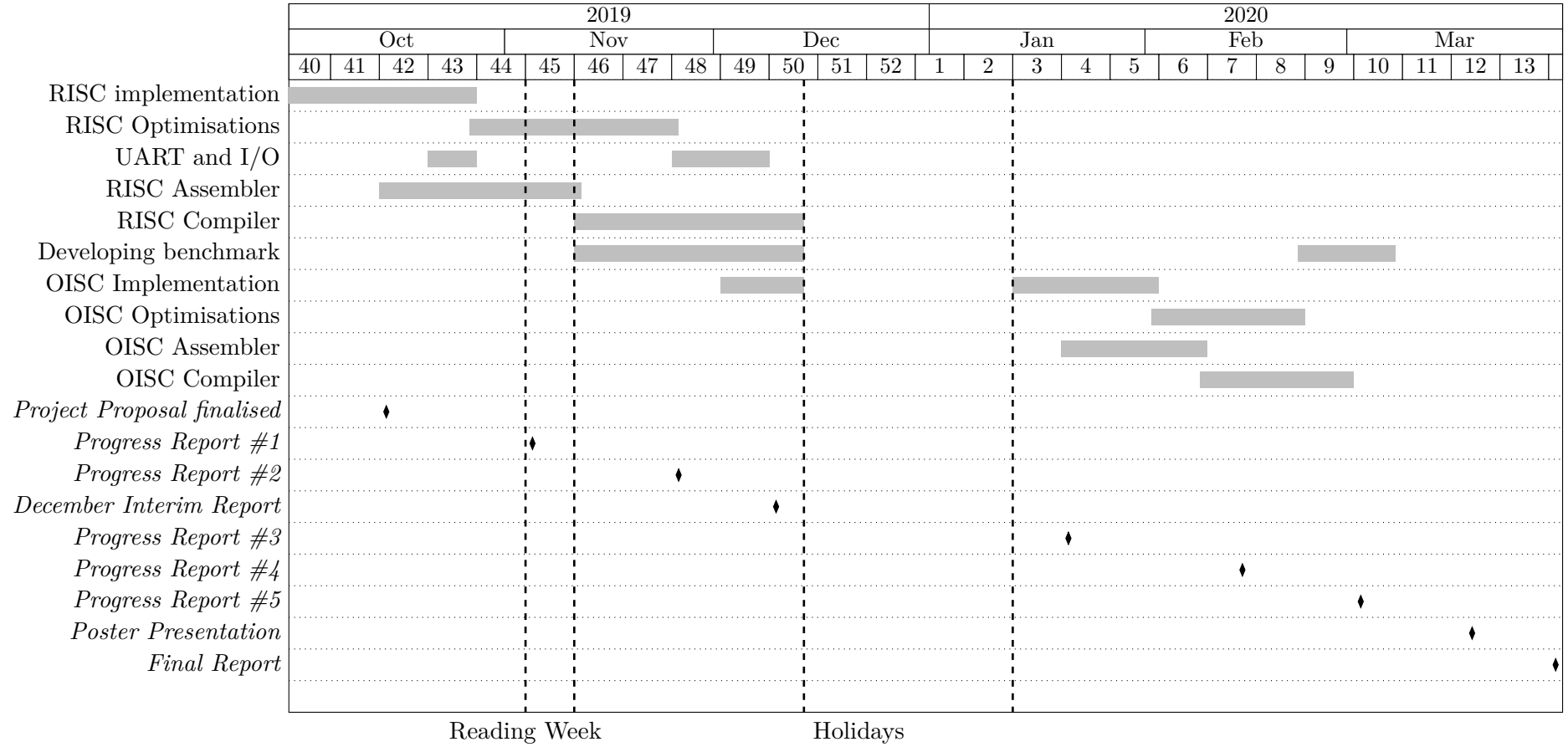

Table 3: Project expected schedule Grantt chart

6 Bibliography of Relevant Literature

List below describes relevant bibliography for this project and why is it used.

[1] W. Gilreath and P. Laplante, *Computer architecture: a minimalist perspective.* [Boston]: Kluwer Academic Publishers, 2003:
Book that describes OISC architectures with minimalistic approach.

[2] D. Money Harris and S. Harris, *Digital design and computer architecture.* Amsterdam: Elsevier, 2013:
Book that explains assembly, architecture and in depth of MIPS microarchitecture.

[3] T. Ahmed, N. Sakamoto, J. Anderson and Y. Hara-Azumi, "Synthesizable-from-C Embedded Processor Based on MIPS-ISA and OISC", *2015 IEEE 13th International Conference on Embedded and Ubiquitous Computing, 2015.* Available: https://janders.eecg.utoronto.ca/pdfs/euc_2015.pdf. [Accessed 9 October 2019]:
A recent research paper that investigates MIPS tradeoffs by attaching SUBLEQ OISC coprocessor.

[4] J. Kong, L. Ang, K. Seng and A. Adejo, "Minimal Instruction Set FPGA AES Processor using Handel - C", *2010 International Conference on Computer Applications and Industrial Electronic (ICCAIE 2010)*, p. 340, 2010. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5735100. [Accessed 9 October 2019]:
Further expands on idea of OISC to implement MISC to use it for encryption.

[5] E. Blern, J. Menon and K. Sankaralingarn, "Power Struggles: Revisiting the RISC vs. CISC Debate on Contemporary ARM and x86 Architectures", 2013. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6522302. [Accessed 9 October 2019]:
This paper goes though comparison of RISC and CISC in common modern systems.

[6] T. Jamil, "RISC versus CISC", *IEEE Potentials*, vol. 14, no. 3, pp. 13-16, 1995. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=464688 [Accessed 9 October 2019]:
An article that goes though differences between RISC and CISC in abstract.