Department of Electronic and Telecommunication Engineering

University of Moratuwa

EN 4202 - Project

Project Proposal

RISC-V Base ISA Processor

Group Members -

M.M.G.U.M Thilakasiri	120660P
D.M.Y.N Seneviratne	120602R
H.A.R.T Wijesekara	120715P
H.M.N Perera	120465X

Supervisor -

Dr. Ajith Pasquel

September 10, 2016

Abstract

Advent of Internet of Things, Systems on Chips and embedded/mobile computing have made processing and control elements ubiquitous in current high tech industry, demanding a wide range of processors with different power-area-performance balance. Though there are several successful processors in today's market, patents, closed sources and very high licensing fees have made these processors inaccessible to academia and small companies, stagnating research and stifling the growth in the field.

In response, UC Berkeley has developed an open source ISA called RISC-V, with the hope of rejuvenating processor architecture research and ultimately producing a free computing framework available to everybody. With the ISA itself being simpler and more efficient than current commercial architectures, a well designed implementation of RISC-V has the potential of attaining superior performance over other processors in the market. With already available open source compiler and toolchain support it is expected that a RISC-V implementation will have a significant success academically and commercially.

Hence this project plans to develop a performance-optimized implementation for the RISC-V ISA targeted towards FPGA and ASIC platforms. To deal with the limited development time, it is currently planned to support only the minimal basic ISA of RISC-V, onto which further extensions can be added later. By surveying the previous research work and current state of the art, this proposal shows that it is possible, practical and economically feasible to develop a highly optimized RISC-V processor and compare it with other commercial processors within the time frame of this project.

Contents

1 Introduction

The document proposes a project focusing on developing an open source processor implementing the RISC-V ISA developed by the University of California(UC), Berkeley.

The first section introduces the problem, develops a solution, states the goals of the project and discusses its scope and extent in solving the problem. A brief literature review on the current state of the art is presented in thereafter. Then, the planned methodology in accomplishing the project goals is discussed, along with alternative methods that could be taken and comparisons between these different strategies. The next section describes the development of the project. The architecture of the project, its block diagrams, required resources, various risks that have to be mitigated and proposed strategies to mitigate them is discussed in detail. After discussing the tentative timeline for different stages of the project and how tasks are delegated between the four group members, the proposal concludes with a section on expected outputs, benefits and beneficiaries of a successful project.

1.1 Problem Statement

Processors are an essential component in almost all high tech products in the market. For many decades it has been the fastest growing technology in the world. However, some trends have been developing in the processor industry in the past years that might cause this rate to slow down.

- Currently, a major share of PC and server processor industry is on the hands of a few giant cooperations such as Intel and AMD. Their most successful processor ISAs and microarchitectures are patent protected, closed source or trademarked, making them unavailable to the public. It has generally stifled innovation in processor architectures. Further rapid development of processor technologies can easily be gained through distributed open research among a large community, but only if freely accessible ISAs and microarchitectures were available publicly.

- Embedded systems is another field where processors are required heavily. Generally manufacturing companies of embedded products buy softcore processors from vendors like ARM and integrate them with their products. However these processors are very expensive due to their high licensing fees and is practically unaccessible to individual researchers, freelance coders, most of the academia and small/medium scale industries.

- Microprocessors such as MicroBlaze developed by Xilinx for Xilinx FPGAs and Nios II developed by Altera for Altera family of FPGAs are only capable of running free of license issues on their own hardware platforms. This limits the portability and range of applications we can use these processors with.

All these issues combined limits the design and implementation of processors to a few large companies, while completely excluding individual researchers, universities and small start-ups. Currently this has stagnated the processor development, with the major companies marketing same processor architectures again and again with minor improvements. This has stifled the development of other fields also, like embedded systems and IoT, since good processors are not available at an affordable cost to small companies.

Therefore need for a low cost, multi-platform, open source processor is being felt more and more. Our project will develop a processor core as a solution for above mentioned problems. We will use RISC V open source ISA for the processor because of its emerging popularity, its large research community and the extensive compilers and toolchains available.

1.2 Primary objectives of the project

The project aims to,

Design a high performance processor using RISC V ISA

After RISC-V's introduction many academics have used it to develop processors for different levels of application. Some currently available work are included in the literature review section. The project will be another such implementation, but concentrating on giving the maximum possible performance in FPGA and ASIC environments. The goals of achieving lower power and area are important but is secondary to the performance goal.

Create a portable, extensible and hardware independent soft processor core

The design is planned to be coded in Verilog HDL. It is to be organized in such a way to allow later extensions and upgrades (floating point, multiply divide) and to also allow easy instantiation within other HDL based designs.

Furthermore apart from using this on FPGA the processor has to be fabricatable on silicon without any extensive changes.

Achieve equivalent or superior performance to similar processors such as Xilinx MicroBlaze and Altera Nios II

One of the main objectives of the project is to produce a processor design which is capable of competing with currently available soft-core processors like ARM, Xilinx, Nios etc. To achieve this it is planned to research in the performance of those processors and their design, studying academic findings which can be included to increase performance and seeking guidance from our supervisor Dr. Ajith Pasquel.

1.3 Scope of the project

While pursuing above goals, we have limited ourselves to the following scope,

RISC-V Base ISA with only integer operations

The project currently concentrates only on implementing the integer pipeline of the processor. This is mainly due to the time limitation of the project. However the design will ensure that future extensions to floating point units will be possible.

Optional multiplication and division extensions (if time permits)

The base ISA includes only addition and subtraction as mathematical operations. The integer multiply and add instructions will be supported only if time is left after completing the base ISA pipeline.

Design for optimal performance, and for portability

The main objective of the design is performance, and costs like area and power will be considered secondary to that goal. In terms of portability, the design will be pluggable to other HDL systems and SoC type applications.

Implement in FPGA, but extendable to ASIC implementation

The design will be hardware independent as possible, but focusing mainly on FPGA implementation. Extensive floorplanning for ASIC would be done only if time permits.

Compare with existing designs and implementations

The processor will be compared with Xilinx MicroBlaze and Altera Nios II in their respective FPGA environments.

2 Literature Review

The RISC-V architecture specification was introduced by the Computer Science division of EECS Department of the University of California, Berkeley, with the goal of providing a open source framework of computer architecture research and education. The ISA was designed to be simple, realistic and micro-architecture independent, thus providing adaptability to multiple implementation domains such as embedded/IOT, desktop, mobile and server computing [1]. The architecture is presented in a modular manner, with a simple Base ISA complemented by a series of extensions (such as Single Precision Floating Point, Double Precision Floating Point, Integer Multiply-Divide etc.)[2]. The specifications for the base ISA to be used in this implementation is included at [2] and the draft for the ongoing supervisor level specification is in [3].

A number of text books is to be referenced to gain a basic understanding of the principles of computer micro architectural design needed in this project. The books [4] and [5] provide basic beginner level understanding of the concepts needed. The book [6] is a more advanced text providing more insight into the quantitative aspects of architectural design decisions. It is the main reference textbook for the project and provides extensive examples on real world architectures from Intel, AMD, MIPS and other processors and discusses the good and bad design decisions taken on their design. Additionally the online Computer Architecture lecture series by Dr. Onur Mutlu of Carnegie Mellon University (available at https://users.ece.cmu.edu/ omutlu/) provides useful insight to processor design.

Currently several other RISC-V processors are being developed by other research teams from around the world. The ISA developers have released open source design files for a number of different RISC-V implementations with different pipeline stages, controllers and interlocks[7]. But the designs are targeted towards ISA simulation and education purposes and not completely optimized for implementation. UC Berkeley also has done a number of practical implementations by using agile design approaches described in [8], some of which include [9], [10], [11]. Out of these, the Rocket Chip is the leading RISC-V implementation and is available open source at [12]. Pulpino, a RISC-V implementation developed by ETH, Zurich, focuses mainly on parallel computing and low power applications, with source code available at [13]. The SHAKTI processor initiative is another research project funded by Indian government to develop its own line of RISC-V processors [14].

In addition to researching other RISC-V implementations, it is also important to study other processor architectures. For most popular architectures, the ISA is available but the micro architectural details are usually hidden and is proprietary. Intel x86 architecture [15] is the most popular processor family in the desktop market. The architecture itself is a Complex Instruction Set (CISC) type, but the processors translate each instruction into several low complexity instructions and executes them in a RISC type core [6]. AMD also uses extensions of this ISA in most of their processors. For embedded and mobile applications the processor of choice is ARM. The ARM family of processors uses a RISC architecture [16], which is simpler, lower power and area efficient than x86 computers. The processors are marketed under 3 brands, Cortex-M, Cortex-R and Cortex-A which are optimized for power efficiency, high performance and balanced power/performance respectively [17]. For the FPGA market, Xilinx and Altera provides two soft processors, MicroBlaze [18] and Nios-II [19] respectively. Since in the project, the implementation is being simulated in a FPGA, these two processors hold the best choice for performance comparison and benchmarking.

3 Methodology

- Study the general principles and design strategies used in the processor micro architectural design. This can be done using standard text books, online lecture series and tutorials. This stage is aimed to bring the group members to the required basic knowledge level before attempting a new design.
- Study existing ISAs, microarchitectural designs and implementations available in the literature, specifically ARM, MicroBlaze and Nios II. By this, we can get a clear view on how designs and architectures differ from one another, implementation strategies, design strategies etc.
- Investigate other RISC-V implementations such as Rocket Chip and PULPino. Check whether we can adapt any parts of available processor designs when developing our own core. We can do improvements to available designs and try to find with better solutions.
- Develop the architecture of the processor and improve its functionality in order to optimize in means of performance. Through this process, we will try to achieve equivalent or superior performance to similar processors such as Xilinx MicroBlaze and Altera Nios II.

- Develop paper designs and implement it using Verilog.
- After getting a functional model of the processor, we plan to carry out several cycles of optimizations, adapting the processor more towards the hardware environment available in FPGAs.
- Testing the implemented designs and benchmarking. At each stage, a set of benchmarks will be run to accurately measure the improvement in performance. Modifications will be done to the design based on the results.

3.1 Required Tools and Resources

- Xilinx Vivado
- Altera Quartus II
- ModelSim
- RISC-V Toolchains and Compiler
- Altera and Xilinx Development Boards

3.2 Alternative Strategies and Key Design Decisions

These are some of the alternative strategies that could be chosen when designing the RISC-V processor. Although there are numerous design decisions to take at each stage of development, these have to be taken as early as possible since they decide to a large extent how the internals of the processor should be designed.

Pipelining depth of the processing core

The depth of the pipeline depends on how much the frequency of the processor should be. Higher frequencies need deeper pipelines so that less work is done between two pipeline registers. But deeper pipelines consume more resources, both due to additional registers and due to increased difficulty of managing control, data and structural hazards.

In order vs out of order execution in the processor core

In order machines executes instructions in the program order, while out of order machines shifts certain instructions up and down to improve performance. But this performance comes at the cost of a large increase in complexity since an improved controller is required to prevent errors due to such instruction shifting.

Number of levels in the cache subsystem

The number of levels of cache directly relates to its performance. Multiple level caches, with faster, smaller upper levels and slower larger lower levels, can be more easily optimized for different conditions, but are more complex and power-area consuming. Single level caches are simpler, though not as efficient.

Size of the caches

Size of the caches directly impact performance, power and area. Larger caches consume more power, but also provides considerable performance improvements. In most cases, the cache size will be determined by die size or resource availability.

Implementation for ASIC or FPGA

The optimizations that needs to be done is slightly different in FPGA and in ASIC implementations. With sufficient care in the paper design of the core, the HDL code can be written so that optimization is similar in both ASIC and FPGA. This way, the final product will be mostly independent of its implementation.

4 Project Structure and Development

For the proposed project, the tentative architecture is as seen in Figure 1. The processor basically consists of two main sections, the memory subsystem and the computing subsystem.

The memory subsystem consists of the cache architecture, external RAM, memory controller and external I/O devices (which are integrated into the same memory bus). The external RAM acts as the main memory of the system and contains both data and instructions required for the processor operation. External I/O is also mapped to the same memory bus using address partitioning. The cache architecture consists of two layers. The smaller and faster level 1 consists of separate memories for instructions and data. The slower unified level 2 cache stores a larger set of data. The memory controller manages the data shifting between each of the above components.

The computing subsystem consists of a prefetch unit (to bring in instructions into the system), a register file (to store temporary data), an execution unit (to do the computations) and a load store unit (to send and receive data from memory). All these units are controlled by a separate control unit. The control units have a large number of smaller submodules, the decoder (which translate instructions to signals), branch predictors (which predict branch instructions and improves efficiency), interlock managers (to stall the processor when necessary) and forwarding managers (to back-carry data along the pipeline) being the most important out of them.

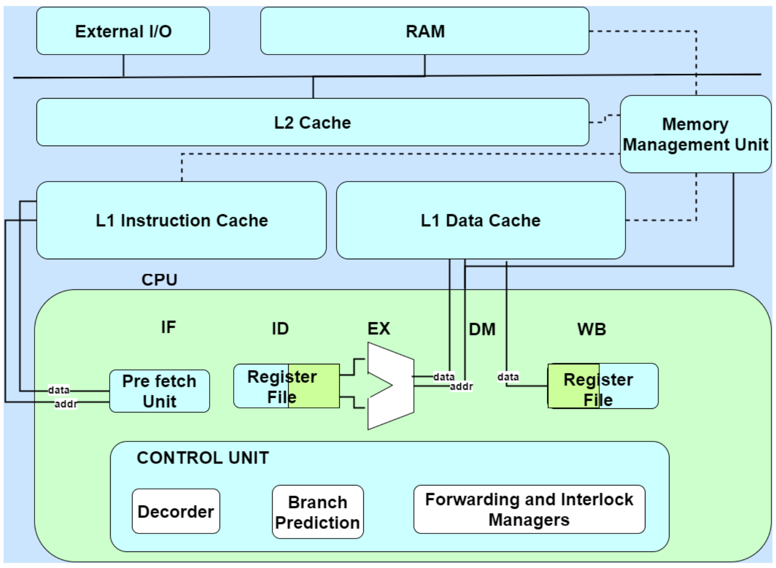


Figure 1: Basic architecture of the project

4.1 Budget

As per the scope of the project we do not need any external source of funding at this stage, because we will not have any significant expenditure. Resources which are needed for the project such as FPGA boards and licensed software is already available in the department.

4.2 Risks and risk management

According to the SWOT analysis conducted on our project we found out several kinds of risks that we need to address. We divided them in to two main categories, one is technical risk factors and the other one is sustainability risk factors.

Technical risk factors

- Can we compete with the major processor families like ARM and Intel?
  - Some of the major players in the field like Intel and ARM have the competitive advantage over us because of their expert designs and the sophisticate development process inside their organizations. They have already designed their own ISA and advanced architectures for their designs.
  - However we already have a slight advantage since the RISC-V ISA is widely considered to be superior to the fairly outdated ARM and x86 architectures. Therefore by including state-of-the-art and innovative designs in the implementation we have a chance to compete with the best of them.
- Can we exhaustively test the processor within the project period?

  - With the time constraints implementation and testing complete processor will not be realistic. Therefore initially the project focuses only on the RISC-V base ISA.
  - Thereafter if time permits, additional functionalities can be introduced to the project.

Sustainability risk factors

- Can we support the processor 5-6 years to the future?
  - A processor needs constant updates and support for as long as it is being used. Since our project ends within the next year, supporting it to an extended time is difficult.
  - The larger RISC-V development project (of which this project is part of) is initially planned to be carried out under a number of phases. After this project a number of final year projects will be carried out in future years to improve, upgrade and debug the processor core taped out here.
- Can the processor compete in the future market?
  - With the competition and the well-established competitors, the future market for the processor is uncertain.
  - To address this risk we need to use the most recent agile design and optimization strategies to make the project modular, parameterized and extendable.

5 Timeline and Task Delegation

Learning Phase (1st of May 2016 to 31st of August 2016)

Within this 18 weeks, we will refer papers, journals, books which are related to processor architectures, designing of processors etc. Principles of designing a processor will be studied comprehensively. More time will be allocated to study the advanced principles in the area. This phase also includes identifying the important benchmarks of Microblaze microprocessor developed by Xilinx as we will be benchmarking with it after the implementation of our processor. We will also be studying the RISC V Instruction Set as we use RISC V as our Instruction Set Architecture.

Planning and Design Phase (20th of June 2016 to 10th October 2016)

We will plan and design the hardware architecture of the processor including cache design and interface design. This phase is started after learning the basics relevant to designing a processor. In parallel with planning phase, we will continue the learning phase in order to learn new areas we come across when designing. Size of cache memories, number of cache levels, type of write strategy, cache line size, number of pipelining stages and other optimizations will be decided or parameterized. Cache architecture and pipelining architecture will be completely designed in this phase. Implementation of the design will be started after planning the basic hardware architecture.

Implementation Phase (18th of July 2016 to 30th November 2016)

Implementation, which runs for about 20 weeks takes the largest share of time allocated for the project. We will implement our design using Verilog language in Xilinx Vivado software. A Zynq-7000 board is the FPGA platform selected for implementation. L1 cache and L2 cache, pipelining architecture and control, memory interfaces, interrupts and other modules needed for the design will be implemented. As part of this phase will conduct in parallel with designing phase and testing phase, we will have to modify our design appropriately.

Testing Phase (12th of September 2016 to 8th of January 2017)

Testing is another critical phase where we can test functionality of our design. It is very important to ensure that it meets all the specifications we developed for the design during the planning phase. We will use softwares such as Matlab, ModelSim for testing purposes. After designing each module, we will test the module alone for functionality. After the module tests are completed, a few modules will be combined together and test the integrated unit for functionality. Those will also be tested for timing. If we could not achieve the required timing from the design, relevant modifications must be done to pass timing. At the end stages of testing process, design will be benched mark with processors such as Microblaze (if time is sufficient, we will also compare with Nios II). Testing and implementation are iterative processes in that it needs to be done until we meet the required functionality.

Closure Phase (1st of January 2017 to 31st of January 2017)

This is the final stage of the project. After going through several stages of testing, we will finalize the implementation for the processor. It is critically important to have a proper documentation for the design. It will be important for everyone who will go through our design for development purposes, research purposes, learning purposes etc. The design will be documented in detail including architecture, design strategies, implementation details, functional details etc. Finally a presentation will be made to introduce our processor to a panel of judges.

6 Expected Outputs

The primary objective of this project is to develop an innovative, unique, high performance RISC-V Base ISA processor which is practically implementable in FPGA or ASIC. Lowering the power consumption and die area is considered as secondary goals. A successfully completed design will produce a processor with the following benefits,

- An open source processor which could run a RISC-V program
- A processor compatible with a range of applications in embedded computing, IoT, embedded vision, and other low power high performance computing applications.
- A soft core processor with high level language programmability (C) using RISC-V toolchain.

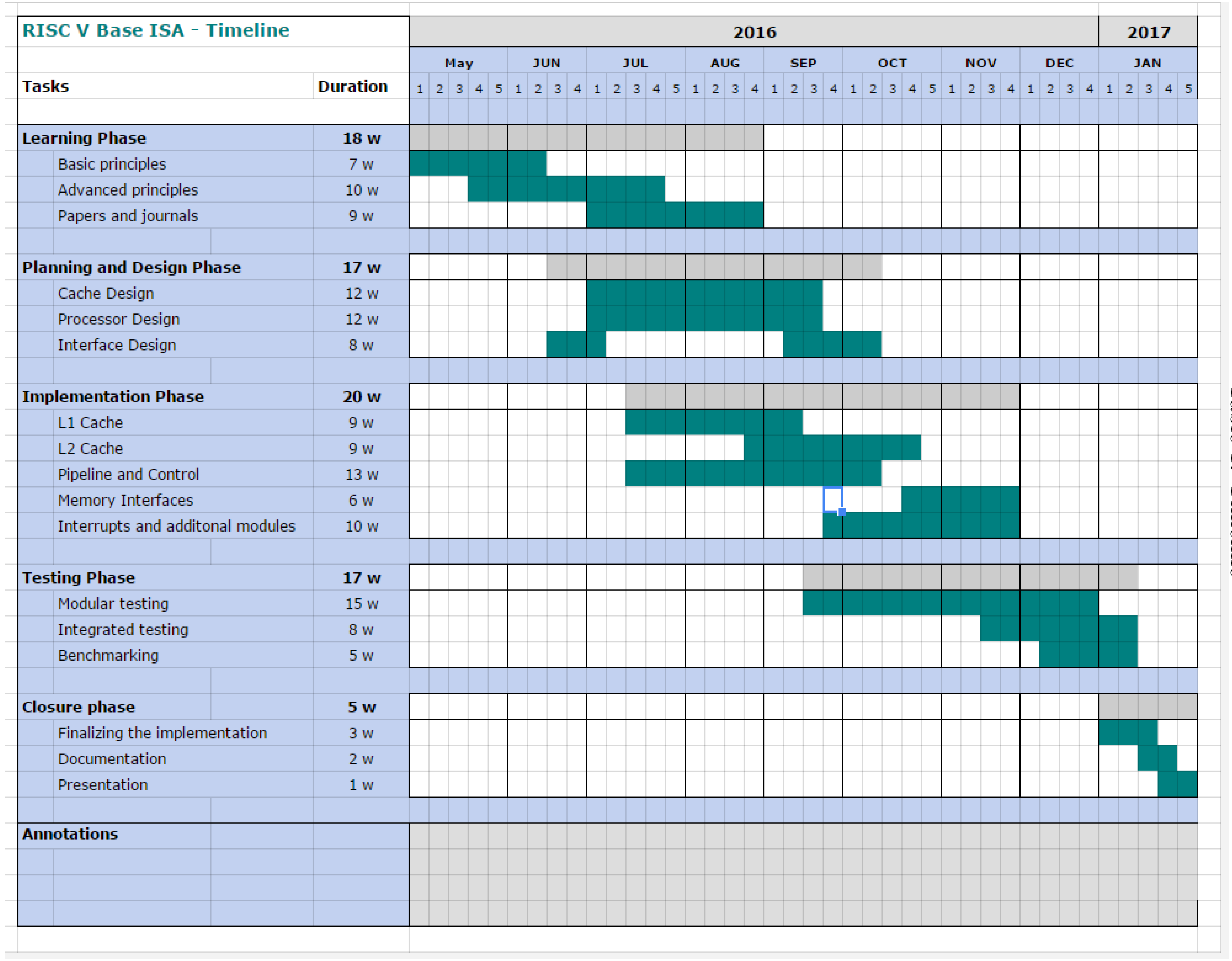
- A parameterizable processor architecture for different user requirements. Parameterizable features will be cache size, block size, number of pipeline stages, availability of external memory etc.
- A processor easily portable between both FPGA and ASIC platforms.

A successful project is expected to benefit the following groups,

- IoT, ASIC and other hardware developers who do not have access to expensive and licensed processor cores.
- Undergraduates and researchers who could use the core for prototyping, product development and educational purposes.
- The RISC-V community, who will get another implementation for their ISA
- University of Moratuwa, which could use and improve the processor in further projects, hopefully creating a core comparable with ARM or Intel processors
- Group members, who would get a good hands-on experience on hardware development to support their future career paths.

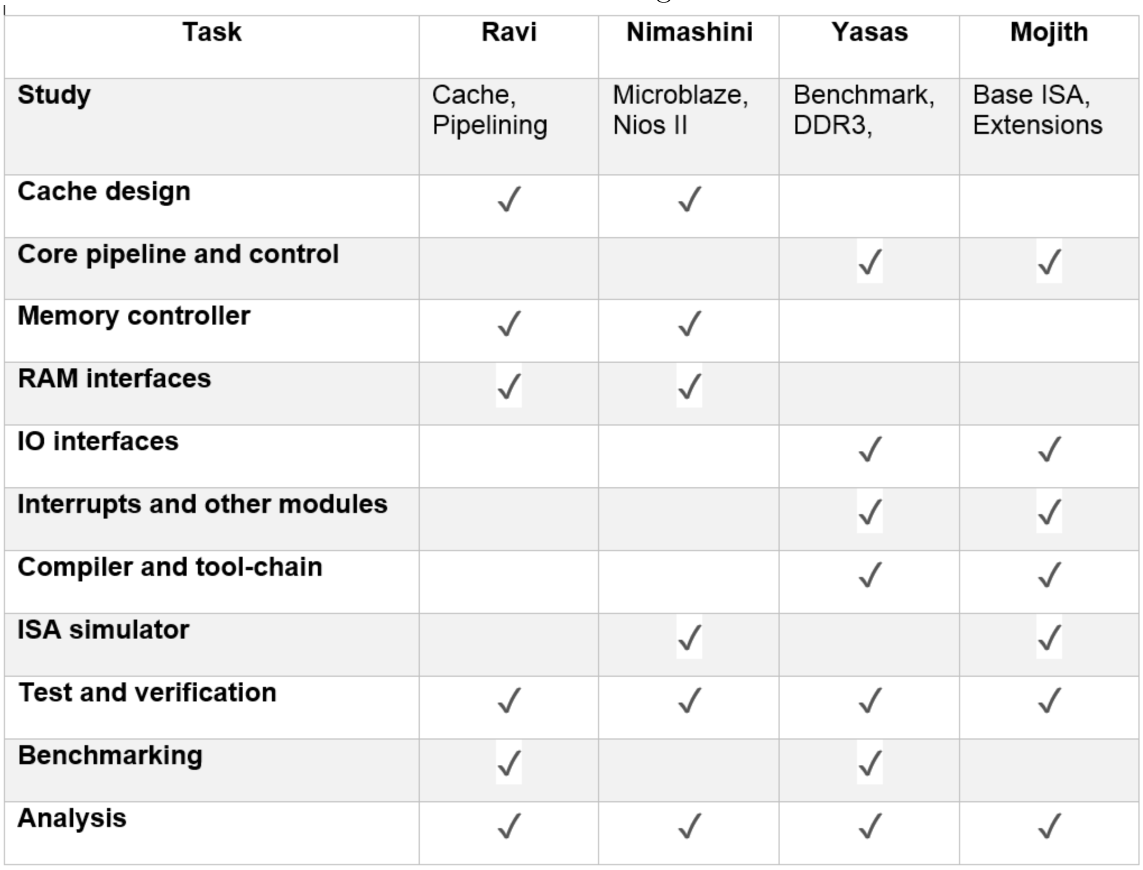
7 Appendix

Table 1: Timeline



RISC V Base ISA - Timeline		2016																																		2017				
		May					JUN				JUL					AUG				SEP				OCT					NOV				DEC				JAN				
Tasks	**Duration**	1	2	3	4	5	1	2	3	4	1	2	3	4	5	1	2	3	4	1	2	3	4	1	2	3	4	5	1	2	3	4	1	2	3	4	1	2	3	4	5
Learning Phase	**18 w**	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨																						
Basic principles	7 w	🟩	🟩	🟩	🟩	🟩	🟩	🟩																																	
Advanced principles	10 w				🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																											
Papers and journals	9 w										🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																						
Planning and Design Phase	**17 w**								▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨																
Cache Design	12 w										🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																			
Processor Design	12 w										🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																			
Interface Design	8 w								🟩	🟩	🟩										🟩	🟩	🟩	🟩	🟩																
Implementation Phase	**20 w**												▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨									
L1 Cache	9 w												🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																				
L2 Cache	9 w																		🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩														
Pipeline and Control	13 w												🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																
Memory Interfaces	6 w																										🟩	🟩	🟩	🟩	🟩	🟩									
Interrupts and additonal modules	10 w																						🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩									
Testing Phase	**17 w**																					▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨			
Modular testing	15 w																					🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩					
Integrated testing	8 w																														🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩			
Benchmarking	5 w																																	🟩	🟩	🟩	🟩	🟩			
Closure phase	**5 w**																																				▨	▨	▨	▨	▨
Finalizing the implementation	3 w																																				🟩	🟩	🟩		
Documentation	2 w																																						🟩	🟩	
Presentation	1 w																																							🟩	🟩
Annotations																																									

Table 2: Task delegation



Task	Ravi	Nimashini	Yasas	Mojith
Study	Cache, Pipelining	Microblaze, Nios II	Benchmark, DDR3,	Base ISA, Extensions
Cache design	✓	✓		
Core pipeline and control			✓	✓
Memory controller	✓	✓		
RAM interfaces	✓	✓		
IO interfaces			✓	✓
Interrupts and other modules			✓	✓
Compiler and tool-chain			✓	✓
ISA simulator		✓		✓
Test and verification	✓	✓	✓	✓
Benchmarking	✓		✓	
Analysis	✓	✓	✓	✓

References

[1] A. Waterman, Y. Lee, R. Avizienis, H. Cook, D. Patterson, and K. Asanovic, "The RISC-V instruction set," in *2013 IEEE Hot Chips 25 Symposium (HCS)*, Aug 2013, pp. 1–1.

[2] A. Waterman, Y. Lee, D. A. Patterson, and K. Asanovic, "The RISC-V instruction set manual, volume 1: User-level ISA, version 2.1," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-118, May 2016. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-118.html

[3] A. Waterman, Y. Lee, R. Avizienis, D. A. Patterson, and K. Asanovic, "The RISC-V instruction set manual volume 2: Privileged architecture version 1.9," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-129, Jul 2016. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-129.html

[4] A. S. Tanenbaum and J. R. Goodman, *Structured Computer Organization*, 4th ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998.

[5] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design, Fourth Edition, Fourth Edition: The Hardware/Software Interface (The Morgan Kaufmann Series in Computer Architecture and Design)*, 4th ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2008.

[6] J. L. Hennessy and D. A. Patterson, *Computer Architecture, Fifth Edition: A Quantitative Approach*, 5th ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011.

[7] C. Celio. "Educational microarchitectures for RISC-V ISA". University of California, Berkeley. [Online]. Available: https://github.com/ucb-bar/riscv-sodor

[8] Y. Lee, A. Waterman, H. Cook, B. Zimmer, B. Keller, A. Puggelli, J. Kwak, R. Jevtic, S. Bailey, M. Blagojevic, P. F. Chiu, R. Avizienis, B. Richards, J. Bachrach, D. Patterson, E. Alon, B. Nikolic, and K. Asanovic, "An agile approach to building RISC-V microprocessors," *IEEE Micro*, vol. 36, no. 2, pp. 8–20, Mar 2016.

[9] Y. Lee, A. Waterman, R. Avizienis, H. Cook, C. Sun, V. Stojanovi?, and K. Asanovi?, "A 45nm 1.3ghz 16.7 double-precision GFLOPS/W RISC-V processor with vector accelerators," in *European Solid State Circuits Conference (ESSCIRC), ESSCIRC 2014 - 40th*, Sept 2014, pp. 199–202.

[10] B. Zimmer, Y. Lee, A. Puggelli, J. Kwak, R. Jevtic, B. Keller, S. Bailey, M. Blagojevic, P. F. Chiu, H. P. Le, P. H. Chen, N. Sutardja, R. Avizienis, A. Waterman, B. Richards, P. Flatresse, E. Alon, K. Asanovi?, and B. Nikoli?, "A RISC-V vector processor with tightly-integrated switched-capacitor DC-DC converters in 28nm FDSOI," in *2015 Symposium on VLSI Circuits (VLSI Circuits)*, June 2015, pp. C316–C317.

[11] V. Patil, A. Raveendran, P. M. Sobha, A. D. Selvakumar, and D. Vivian, "Out of order floating point coprocessor for RISC V ISA," in *VLSI Design and Test (VDAT), 2015 19th International Symposium on*, June 2015, pp. 1–7.

[12] "Rocket chip generator". University of California, Berkeley. [Online]. Available: https://github.com/ucb-bar/rocket-chip

[13] "PULPino". ETH, Zurich. [Online]. Available: https://github.com/ucb-bar/rocket-chip

[14] N. Gala, A. Menon, R. Bodduna, G. S. Madhusudan, and V. Kamakoti, "SHAKTI processors: An open-source hardware initiative," in *2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID)*, Jan 2016, pp. 7–8.

[15] Intel, *Intel 64 and IA-32 Architectures Software Developer's Manual.* [Online]. Available: http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-manual-325462.pdf

[16] ARM-Holdings, *ARM Architecture Reference Manual*. [Online]. Available: https://www.scss.tcd.ie/~waldroj/3d1/arm_arm.pdf

[17] ——, *Cortex-M7 Technical Reference Manual*. [Online]. Available: http://infocenter.arm.com/help/topic/com.arm.doc.ddi0489b/DDI0489B_cortex_m7_trm.pdf

[18] Xilinx, *MicroBlaze Processor Reference Guide*. [Online]. Available: http://www.xilinx.com/support/documentation/sw_manuals/mb_ref_guide.pdf

[19] Altera, *Nios II Classic Processor Reference Guide*. [Online]. Available: https://www.altera.com/en_US/pdfs/literature/hb/nios2/n2cpu_nii5v1.pdf