Technion - Israel institute of technology
Electrical & Computer Engineering Faculty
VLSI Labs

GPC_4T

GENERAL PURPOSE COMPUTE UNIT WITH 4 HW THREADS

Authors:
Adi Levy
Saar Kadosh

Advisor:
Amichai Ben-David

Semester of registration : Spring 2021
Submission date : January 2nd, 2022

GPC_4T:
4 Thread RISCV 32-bit Core & Memories

GPC_4T PROJECT

Adi Levy (adi.levy@campus.technion.ac.il)

Saar Kadosh (skadosh@campus.technion.ac.il)

CONTENTS

Revision History

Rev. No.	Who	Description	Rev. Date
0.3	*Saar Kadosh*	*Initial GPC_4T project*	*06 October 2021*
0.4	*Adi Levy*	*spit to 5 documents*	*06 December 2021*
0.5	*Adi Levy*	*References*	*02 January 2, 2022*

Related Documents

Name	Path	Description
riscv_isa_spec.pdf	RISCV_Spec	*The full RISCV Unprivileged Specification file. Including the RV32I Baseline ISA*

Glossary

Term	Description
GPC_4T	Hardware embedded 4 thread RISC-V core.
ISA	Instruction Set Architecture. (such as X86, ARM, RISC-V etc.)
IO	Input & output.
IP	Intellectual Property. In this case, RTL building block that can be consumed.
HAS	High Level Architecture Specifications.
MAS	Micro Architecture Specifications.
I_MEM	Instruction memory – where the program is loaded and ready for execution.
D_MEM	Data Memory – where the LOAD & STORE instructions read/write Data.
Pipeline	Common Way to parallel and utilize Hardware https://en.wikipedia.org/wiki/Instruction_pipelining
RISC	Reduce Instruction Set Computer. (Unlike CISC -Complex Instruction Set Computer) https://en.wikipedia.org/wiki/Reduced_instruction_set_computer
Thread	A "hardware thread" is a physical CPU or core that can run a program.
RISC-V	A relatively new open and free ISA. (comparable to intel X86, ARM) https://en.wikipedia.org/wiki/RISC-V
RV32I	“RISC-V 32-bit Integer” The RISC-V baseline compatible ISA (no extensions M/A/F etc.)
Standard interface	Functional characteristics to allow the exchange of information between two systems
Word	32-bits of data - 4 Bytes. The size of an integer in RV32I ISA.
Hazard	Potential source of harm. in this document when reading Outdated data, or wrongly executing Instruction.
Strap	Tie signals to constant value (1’b0 or 1’b1)
MSFF	Main-Secondary Flip Flop. (AKA Master-Slave Flip Flop)
Clock Gating	Logic that allows to condition the MSFF clock. Functionality and power reasons.
Polling	Actively sampling the status of an external device.

Table 1 – Glossary

1 INTRODUCTION

1.1 ABOUT THE PROJECT

The **GPC_4T** Project is a Computer Engineering Students project created by Adi Levy and Saar Kadosh as a part of BSc degree demands of the Faculty of Electrical and Computer Engineering at the Technion institute.
GPC_4T is an implementation of a **RISC-V** 32I base integer **GENERAL PURPOSE COMPUTE UNIT CORE** with 4 hardware Threads (Thus the name GPC_4T) and two **Memory modules** – Data memory and Instruction memory – working with the core.

The motivation of the project was to research and develop the abilities of the RISC-V instruction set architecture, which is an open-source ISA for CPUs and a very "hot name" in the industry.
Another motive was also to experience with architectural design, logic design and validation of a CPU.

In this design, there will be a single core with 4 hardware controlled main threads with 32 registers for each thread. In addition, the design will include memory modules – for the instruction and for the data.

The GPC_4T has an integration ability to another Faculty student's project known as "Ring Controller". As two components – GPC_4T, RC – , they can communicate each other.

When integrated, they constitute a "LOTR Tile" – a piece of the "LOTR Project" – A "Fabric" ring architecture design, and another project, led by this projects mentor Amichai Ben-David. The LOTR will implemented in the future.

The project contains many aspects such as architectural and logic design, HDL programming , low level programming & scripting and validation.

The entire RTL code of the project, documentations, applications and validation & verification tools and guides to operate them are all can be found in the LOTR GitHub repository.

1.2 ABOUT THE AUTHORS

Adi Levy , BSc Student , Studies Computer & Software Engineering with Advance Programming as areas of expertise at Technion – Israeli Institute of Technology. Currently employed at Intel as a Software Engineer.

Saar Kadosh , BSc Student , Studies Computer & Software Engineering Integrated Circuit and VLSI as areas of expertise at Technion – Israeli Institute of Technology. Currently employed at Apple inc. as an Electronic Engineer.

1.3 ABOUT THE ADVISOR/MENTOR

Amichai Ben-David , Electrical Engineer. Amichai was a massive contributor to the project, and it couldn't be done without his help. He is currently employed in Intel as design engineer and among other things research RISC-V there. Amichai contributes to the project from his free time.

2 STRUCTURE OF THE DOCUMENTATION

The project documentation was split into four standalone documents, each can be read separately. Each of the documents focus on different aspect of the project.
For better understanding, it is recommended to read all documents in the below order.

2.1 HIGH-LEVEL-SPECIFICATIONS (HAS)

High-Level-Specifications *or* HAS is one of the two Specifications methods used to describe this project. in this method we describe the project "from above" and without the implementation. HAS is easier to understand and its purpose is to briefly describe the architecture in general lines.

2.2 MICRO-ARCHITECTURE-SPECIFICATIONS (MAS)

Micro-Specifications *or* MAS is one of the two Specifications methods used to describe this project. in this method we describe the project "from inside" and expand details on the implementation. MAS is the logic design of the architectural demands from the HAS document.

2.3 VALIDATION PLAN

In GPC_4T project Validation was critical to ensure the correctness of GPC_4T core & Memory functionality, before continuing to future project like FPGA synthesis.
Because we don't have a physical GPC_4T unit, we checked GPC_4T functionality with ModelSim simulations on a Testbench.

2.4 DEVELOPMENT ENVIRONMENT AND METHODOLOGIES

Many tools and Development Environment were used in this project for Programming, simulations, automation and many more. All of these are detailed in is section.

3 FUTURE PLANS

For the upcoming future, Some of the future plans briefly:

- Project B - Implementation of an LOTR RISC-V Based System on Chip (SOC) on an FPGA – On our next project, that will be a continuous project to this one, we will design the complete LOTR project and synthesize it on FPGA.
- Optimize architecture of the pipe
- Support privileged CSR RISC V externals to allow Operating system
- Use windows-based Toolchain GCC (not WSL)
- Move all WSL\PS scripts to simple bash script
- Enable official RISC-V “computability test”
- Test the design on FPGA:
  - Synthesizable and timing conversion
- Enable more complex multi thread tests
- Support exception, error detections and interrupts .
- Add more features to GPC_4T

4 REFERENCES

- RISCV official ISA specifications.
  https://riscv.org/technical/specifications/

- RISCV Official organization's website.
  https://riscv.org/

- Wikipedia, The open online encyclopedia.
  https://en.wikipedia.org/wiki

- Prof. Freddy Gabbay's Lectures on Multithreading from Computer Architecture 046267 Course at The Faculty of EE & Computers, Technion.
  https://moodle.technion.ac.il/pluginfile.php/1609728/mod_resource/content/1/8-9-ca-multithreading.pdf

- Website of Digital Systems and Computer structure Course at The Faculty of EE & Computers, Technion.
  https://moodle.technion.ac.il/enrol/index.php?id=6123

- AnandTech, Tech blog, on Ring Architecture.
  https://www.anandtech.com/show/3922/intels-sandy-bridge-architecture-exposed/4

- Official GNU, about linker scripts
  https://ftp.gnu.org/old-gnu/Manuals/ld-2.9.1/html_chapter/ld_3.html

- RISC-V Toolchain Install guide and user guide repository by John Winans
  https://github.com/johnwinans/riscv-toolchain-install-guide

- GNU GCC Compiler guide
  https://gcc.gnu.org/onlinedocs/gcc/Link-Options.html

- Github – Getting started with Git & Github
  https://docs.github.com/en/get-started/quickstart/hello-world

- Git official, git documentations
  https://git-scm.com/docs/gittutorial

- System Verilog tutorial
  https://verificationguide.com/systemverilog/systemverilog-tutorial/

- Modelsim Edition download and documentation
  https://www.intel.com/content/www/us/en/software/programmable/quartus-prime/model-sim.html