phoeniX RISC-V Core V0.1

Arvin Delavari – Faraz Ghoreishy

arvin7807@gmail.com – farazghoreishy@gmail.com

Summer 2023

Iran University of Science and Technology

Electronics Research Center – Digital Design Laboratory

This page was initially left blank.

Preface:

phoeniX is a 5 stage pipelined 32-bit RISC-V processor written in Verilog HDL. This project was started in summer of 2023 in Digital Design Laboratory of Electronics Research Center at Iran University of Science and Technology. This document contains technical specifications and user manual for version 0.1 of phoeniX processor. This version of the processor is able to execute RV32I (integer operations extension) instructions of official RISC-V instruction set manual [1]. Other extensions will be supported by the core in the future updates.

phoeniX core is capable of execution of assembly and C codes using iverilog simulator. Guidance is included in this document for the flow of code execution in this processor. For execution of C codes, the processor will use RISC-V GCC compiler toolchain and for assembly codes, you can use GCC compiler or Venus simulator which is a free extension for the recognized code editor, Microsoft Visual Studio Code. Further descriptions are included to relative sections of this document.

This project is an open source CPU under the GNU V3.0 license [2] and is free to use. You can find source codes and documentations in the following GitHub repository:

<https://github.com/ArvinDelavari/PHOENIX-CORE>

We would like to express our sincere gratitude to the following individuals for their invaluable guidance and remarkable insights, which played a pivotal role in the completion of this project:

* Professor Sattar Mirzakuchaki PhD

Iran University of Science and Technology

* Professor Hadi Shahriar Shahhoseini PhD

Iran University of Science and Technology

* Nima Amirafshar MSc

Iran University of Science and Technology

Arvin Delavari – Faraz Ghoreishy

Iran University of Science and Technology – Summer 2023

Contents:

1. Introduction
   1. phoeniX core structure
   2. phoeniX software interface
   3. Required software
2. Building Blocks (Modules)
   1. Fetch Unit
   2. Instruction Decoder
   3. Immediate Generator
   4. Address Generator
   5. Arithmetic Logic Unit
   6. Register File
   7. Hazard Detection and Forwarding Unit
   8. Jump and Branch Unit
   9. Load Store Unit
3. Memory Interface Logic
4. Code Executant Software
   1. Code execution and simulation flow
   2. Windows (Venus Simulator)
   3. Linux (RISC-V GCC toolchain)
5. Synthesis Result
   1. Qflow toolchain
   2. TSMC 180nm PDK
   3. phoeniX ASIC result
6. References
7. Introduction

The phoeniX project was initiated in summer of 2023 in Electronics Research Center at Iran University of Science and Technology [3]. This RISC-V processor was designed in order to be an original base processor with acceptable performance and specification which aims to lay a solid groundwork and foundation for future computer architecture research and development endeavors.

The processor uses RISC-V open source Instruction Set Architecture (ISA) with a custom designed microarchitecture. The core is written in Verilog HDL and it is synthesizable for both FPGA and ASIC targets. Building blocks of this core were all written in separate modules which would help developers to add features and test computer architecture techniques on the base core. This modular design leads to a simpler way of debugging and also extending the core.

In the following version (V0.1), phoeniX is a 5 stage pipelined RV32I processor which supports “I-Extension” of RISC-V ISA. Other extensions will be added to the core in the upcoming updates soon.

phoeniX stands out with its user-friendly and straightforward flow for simulating and executing C and assembly codes on the processor. Unlike many other open-source processors, phoeniX offers a simplified approach that is easier to follow. The repository of this project provides comprehensive documentation, offering step-by-step guidance on executing codes on the core. This resource is highly accessible and designed to facilitate a seamless development experience in computer architecture and digital design field.

* 1. phoeniX core structure

In the beginning, one of the most specialties of this processor which should be mentioned is not having a single control unit. This processor uses a “Distributed Control Logic” which leads to elimination of control unit from the building blocks. By DCL it means that after decoding the instructions, relative fields such as opcodes, funct3, funct7 and other fields are straightly directed to the linked modules, and the process of control signals generation will be taken place in the target module hardware. For example, in a basic processor decoded fields are sent to control unit and inside this module, signals will be generated to define ALU operation, ALU multiplexers (for selection of registers bus or immediate value), bypassing and forwarding multiplexers and etc. In phoeniX processor, all of these control signals are determined inside the target modules, by sending the relative decoded fields directly to the target. In fact, modules are somehow known to be self-controllable which in this case we call the general system, Distributed Control Logic.

phoeniX is a 5 stage pipelined processor which has the classic machine cycle stages: Fetch, Decode, Execute, Memory Interface and Write back. Building blocks of the processor are mostly designed in separated modules which gives developers freedom for changes and the mentionable benefit of modularity. Registers used for latching data in different stages of the core are not designed as modules are completely written inside the main module named “phoeniX”. Detailed descriptions about the hardware design of the core and its modules are included Building Blocks (Modules) section of this documents.

Memory interface of the core is completely written and described in testbench file because of the limitations of memory simulation using Hardware Description Languages. Further information about the interface logic is inside Memory Interface section of this document.

The phoeniX core also has a hazard detection unit which gives the core, the ability of performing data-forwarding and bypassing techniques in order to skip stalls in the code including dependencies and hazards.

The maximum delay time (critical path) of phoeniX processor analyzed using Yosys [4] open-source synthesis tool and Vesta [5] static time analysis software, in different stages is something about 3800-3900 picoseconds which leads us to clock cycle width of 4 nanoseconds. This means the phoeniX processor will perform in a frequency rate of nearly 250MHz which is a remarkable number in benchmarking with other processor. Most of the commercial and industrial embedded processors designed by ARM with Cortex M0, M3 and M4 [6] microarchitecture will not reach this frequency rate. These processors find extensive integration with microcontroller series such as STM32 (by ST) or LPC (by NXP), making them highly prevalent and widely employed in the field of embedded systems.

Even other open-source RISC-V embedded processors such as PicoRV32 by Claire Wolf [7] won’t reach a higher frequency than the mentioned. PicoRV32, is one of the most recognized open-source processors and even used in fabricated microcontroller chips has a frequency rate ranged in 50 to 100MHz depending to the technology. Another well-known RISC-V processor is named ibex core by ETH Zurich [8] university which also has a frequency range between 100MHz and 300MHz in highest performance.

The phoeniX processor is a reliable and user-friendly solution, offering modularity and easy extensibility for developers and computer architecture researchers. It features a simplified design that is easy to understand, while still delivering competitive performance compared to other processors.

* 1. phoeniX software interface

The phoeniX processor uses standard GCC compiler toolchain [9] which is officially verified by RISC-V organization, in order to run C codes and assembly codes on the processor. This codes are turned into a firmware file including hex format of instructions and addresses by the compiler toolchain. The generated firmware file is given to the testbench of phoeniX CPU as the instruction memory. The processor can simulate the executed code using iverilog [10] version 12 tool. There is a shell script included in the repository which helps users to install all software requirements for simulating and coding on phoeniX processor.

The simulation and execution flow is very simplified by phoeniX code executants which will be described briefly in the upcoming sections of this document. There are solutions to simulate assembly and C codes both Windows and Linux systems. Linux systems can use the makefile in the main directory which will do the complete process of compiling, assembling, generating firmware file and executing on the phoeniX core without any complexity and user interference.

References:

[1] RISC-V ISA

[2] GNU LICENSE

[3] IUST ERC WEBSITE

[4] Yosys

[5] Vesta

[6] ARM Cortex M

[7] PicoRV32

[8] ibex

[9] GCC compiler toolchain

[10] iverilog