DEPARTMENT OF
Electrical and Computer Engineering

11-203 Donadeo Innovation Centre for Engineering
9211-116 Street NW
University of Alberta
Edmonton, Alberta
Canada T6G 1H9

Preliminary Design: FPGA-Based RISC-V CPU and Fysh Programming Language and Compiler (System Name: Fysh-Fyve)

Revisions

Revision	Author	Changes	Date
001	Charles Ancheta, Yahya Al-Shamali, Kyle Prince	Initial Release	2024-02-16

Table of Contents

Acronyms

Acronym	Full Description
ISA	Instruction Set Architecture
RISC	Reduced Instruction Set Computer
FPGA	Field Programmable Gate Array
PL	Programmable Logic, the FPGA part of the Zybo development Board
AST	Abstract Syntax Tree
ECE	Electrical and Computer Engineering
CPU	Central Processing Unit

References

[1] S. Knudsen, “RISC-v fpga-based cpu and language.” Dec. 2023. Available: https://fysh-fyve.github.io/ECE492_RISCV_PP.pdf

1 Purpose

This document describes the preliminary design for the RISC-V FPGA-based CPU and Language [1].

2 Concept of Operation

The high-level operation of the RISC-V FPGA-based CPU and Language is illustrated by the following user stories and use cases.

2.1 User Stories

2.2 Use Cases

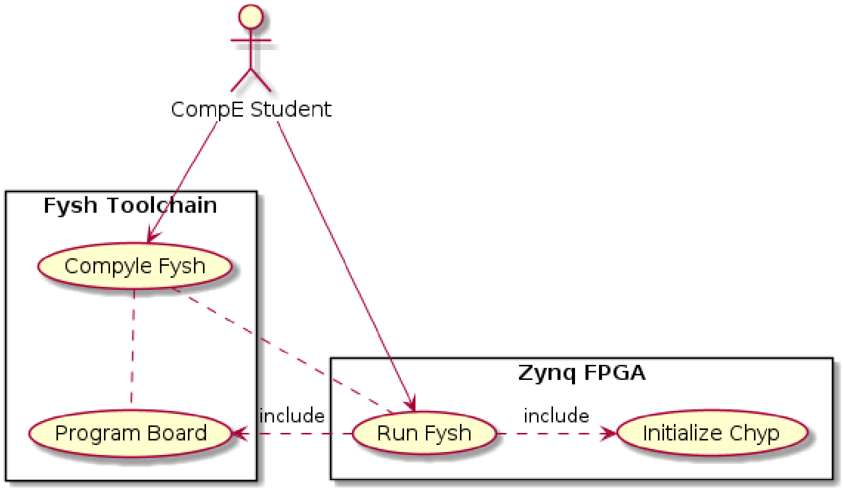

Figure 1: Use cases for the Fysh Toolchain

Use Case Description and Details	
Number	UC-001
Name (action)	Initialize Chyp
System	Zynq FPGA
Actor	CompE Student
Use Case Goal	Initialize Zynq FPGA with the Fysh-Fyve CPU.
Primary Actor	CompE Student
Preconditions	CompE Student has a development board.
Postconditions	FPGA is programmed with the RISC-V CPU.
Basic Flow	1. CompE student initializes the Fysh-Fyve project in Vivado. 2. CompE student runs the hardware synthesis and implementation. 3. CompE student generates a bitstream and programs the FPGA.
Alternate Flows	None

Table 1: UC-001 - Initialize Chyp

Use Case Description and Details	
Number	UC-002
Name (action)	Compyle Fysh
System	Fysh Toolchain
Actor	CompE Student
Use Case Goal	Compile Fysh source code into an executable format.
Primary Actor	CompE Student
Preconditions	Fysh source file exists.
Postconditions	RV32I machine code describing the Fysh source code is outputted.
Basic Flow	1. CompE student runs FyshSea, the Fysh Compyler, giving the Fysh source code as input. 2. FyshSea program exits and a binary file with RV32I format is outputted.
Alternate Flows	A. Fysh source code has a syntax or type error. 1. FyshSea outputs an error message describing the error. 2. FyshSea exits.

Table 2: UC-002 - Compyle Fysh

Use Case Description and Details	
Number	UC-003
Name (action)	Program Board
System	Fysh Toolchain
Actor	CompE Student
Use Case Goal	Download Fysh fyrmware into the Fysh-Fyve chyp.
Primary Actor	CompE Student
Preconditions	Fysh program is successully compiled into RV32I machine code.
Postconditions	The Zynq FPGA has the firmware loaded into the Fysh-Fyve CPU.
Basic Flow	1. CompE student transforms the firmware binary into a format that is usable by the Fysh-Fyve CPU. 2. CompE student downloads the FPGA-usable format into the FPGA.
Alternate Flows	A. Firmware size is too large. 1. FyshDude outputs an error message describing the error. 1. FyshDude exits.

Table 3: UC-003 - Program Board

Use Case Description and Details	
Number	UC-004
Name (action)	Run Fysh
System	Zynq FPGA
Actors	CompE Student
Use Case Goal	Execute program written in Fysh.
Primary Actor	CompE Student
Preconditions	1. FPGA is initialized with Fysh-Fyve CPU. 2. Fysh program is compiled into a binary. 3. Fysh program is programmed into the development board.
Postconditions	Fysh program is running on a device.
Basic Flow	1. CompE student depresses the reset button on the development board. 2. CPU runs the instructions one by one.
Alternate Flows	A. CPU encounters an exception. 1. CPU halts execution. 2. Error LED lights up.

Table 4: UC-004 - Run Fysh

3 Functional and Performance Requirements

In the table below, the “CPU” refers to the Fysh-Fyve CPU, the FPGA-Based RISC-V CPU for the Zybo development board. “Fysh” refers to the Fysh programming language, the custom-made esoteric programming language for this project. “FyshSea” refers to the Fysh Compyler, the compiler that targets RV32I with native support for the custom RISC-V instruction.

FR #	Functional Requirement Description
FR-01	The CPU Shall be compliant with the RV32I instruction set
FR-02	The CPU shall have general-purpose input/output pins
FR-03	The CPU shall have access to memory
FR-04	The CPU shall have a custom instruction to generate a 32-bit integer
FR-05	Fysh shall support basic integer operations
FR-06	Fysh shall support bit manipulation
FR-07	Fysh shall support memory addressing
FR-08	Fysh shall have a programming construct for random number generation
FR-09	Fysh shall be statically typed
FR-10	FyshSea shall convert Fysh source code to RV32I instructions
FR-11	FyshSea shall support the basic features of Fysh

Table 5: Functional Requirements

PR #	Performance Requirement Description	Related FRs
PR-01	The CPU shall process 1 instruction per 2 clock cycles	FR-06
PR-02	The Zybo development board shall have 128kB of RAM and 128kB or ROM	FR-04

Table 6: Performance Requirements

4 System Design

The overall system will have two parts as shown in figure 2, the Fysh toolchain in the development machine, and the RISC-V CPU in the Zybo board.

4.1 System Architecture

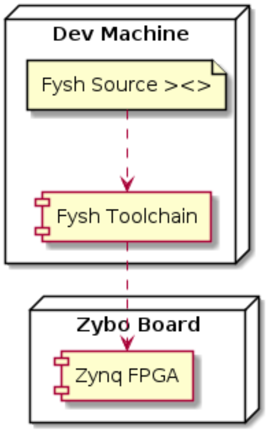

Figure 2: Deployment diagram for Fysh firmware.

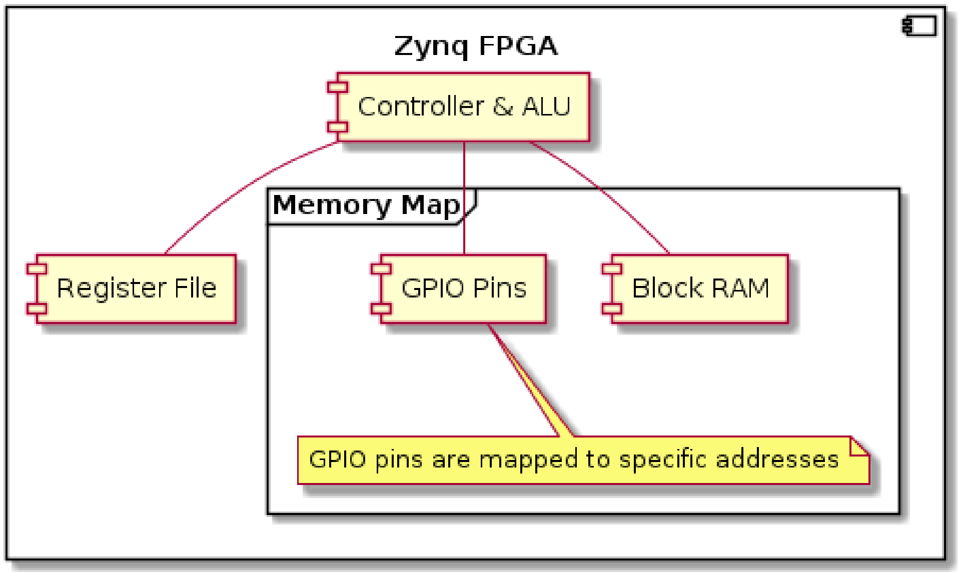

Figure 3: Simplified Fysh-Fyve Architecture.

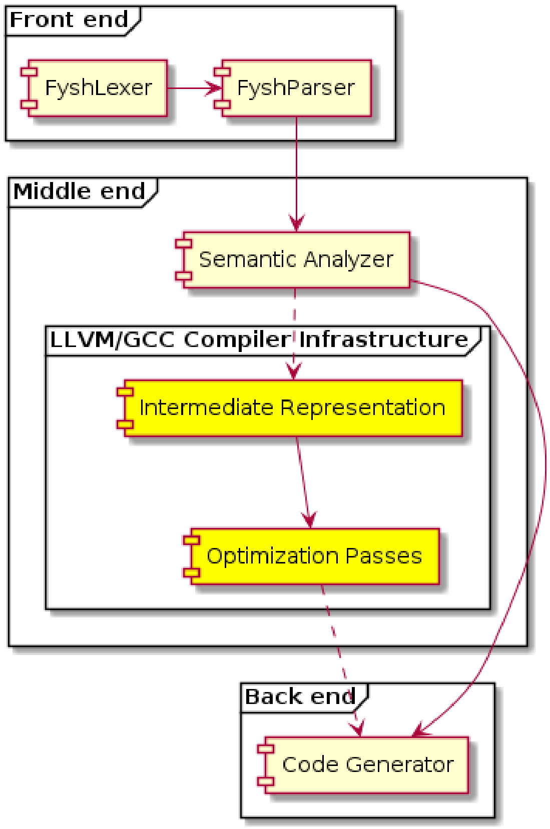

Figure 4: FyshSea Architecture. Components in yellow are optional.

4.1.1 Hardware Components

As stated in Section 2, the project shall be available to all ECE students, so minimal hardware is required. One only needs the Zybo development board to get started in creating projects.

4.1.2 Firmware Components

4.1.3 Software Components

5 System Requirements

SR #	System Requirement Desc	FR #	PR #	Notes
SR-NN				
SR-NN				
SR-NN				

Table 7: System Requirements

6 Minimum Design

In this section a minimum design, the "walking skeleton" is described. The purpose is to define the functionality to be implemented in the first development iteration. The outcome is reported to the client providing an opportunity for early feedback.

7 High-Level Hardware Design

8 High-Level Software/Firmware Design

The most abstract explanation of the software aspect of the project would be that a fysh program is compiled by the custom fysh compiler and executed on our hardware. More in depth, the compiler consists of multiple components which break down the fysh program so that it can be understood and executed using C++. The first of these components is the lexer, which takes in fysh code as input and turns it into a series of tokens, i.e tokenizes. Following the lexer is the parser, which takes a sequence of tokens from the lexer and produces an AST of the program while verifying the syntax.

9 Prototype Budget

Component	Mfr P/N	Mfr	Qty	Unit Price	Extended Price
FPGA Development Board	Zybo Z-7010	Xilinx/AMD	1	$299.00	$299.00
64x64 RGB LED Matrix	RGB-Matrix-P2.5-64x64	Waveshare	2	$36.10	$72.20
128x32 Monochromatic OLED Display	410-222	Digilent	1	$14.99	$14.99
				Total Cost	$386.19

Table 8: System Budget (ROM)