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

1 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
RISC	Reduced Instruction Set Computer
FPGA	Field Programmable Gate Array
PL	Programmable Logic, the FPGA part of the Zybo development Board

References

[1] D. Murph and L. Farrer, "Why gitlab uses the term all-remote to describe its 100," *GitLab*. Feb. 2020. Available: `https://about.gitlab.com/company/culture/all-remote/terminology/`

[2] J. MacFarlane, "Creating a pdf | pandoc user's guide," *Pandoc*. Aug. 2022. Available: `https://pandoc.org/MANUAL.html#creating-a-pdf`

[3] G. L. McDowell, *Cracking the coding interview: 189 programming questions and solutions*. CareerCup, LLC, 2021.

[4] C. Ancheta, Y. Al-Shamali, and K. Prince, "Proposal response: RISC-v fpga-based cpu and language." Jan. 2024. Available: `https://fysh-fyve.github.io/main/ECE492_RISCV_PPR_V01.pdf`

[5] S. Knudsen, "RISC-v fpga-based cpu and language." Dec. 2023. Available: `https://fysh-fyve.github.io/main/ECE492_RISCV_PP.pdf`

References [1] must [2] be [3] used or else they don't show up in the bibliography [4] [5].

2 Purpose

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

3 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.

3.1 User Stories

<Have at most 3 paragraphs per user story, preferably one>

3.2 Use Cases

<Use UML diagrams and some annotation to present use cases. See this tutorial. Add descriptive text for each use case. Keep the number of use cases below 15; ≤ 10 is best. Figure 1 is an example use case diagram, and Table 1 is a Use Case description table. If a table is complicated, the use case is probably too low-level>

```
left to right direction
actor "ECE Student" as fc
rectangle Restaurant {
  usecase "Eat Food" as UC1
  usecase "Pay for Food" as UC2
  usecase "Drink" as UC3
}
fc --> UC1
fc --> UC2
fc --> UC3
```

Use Case Description and Details	
Number	UC-001
Name (action)	<same as the name of the use case in the diagram>
System	<same as the system boundary name>
Actors	1. <all actors associated with the system> 2. 3.
Use Case Goal	
Primary Actor	
Preconditions	
Postconditions	
Basic Flow	1. 2. 3.
Alternate Flows	A. 1. 2. 3. B. C.

Table 1: Use Case – <use case name, same as the descriptor in the UML oval>

4 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. "Fyshc" 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	Fysh shall support basic integer arithmetic
FR-04	Fysh shall support memory addressing
FR-05	Fysh shall have a programming construct for random number generation
FR-06	Fyshc shall convert Fysh source code to RV32I instructions
FR-07	Fyshc shall support all features of Fysh

Table 2: Functional Requirements

<Define any performance requirements and match to associated FRs, if applicable. Keep below 20, but there can be more because there may be multiple performance requirements for one FR>

PR #	Performance Requirement Description	Related FRs
PR-01		<zero or more>
PR-02	Example Description	01

Table 3: Performance Requirements

5 System Design

<Define the system architecture that will meet the FRs and PRs. Think of deployment first (are there separate physical parts, like a smartphone and remote sensor?), and then for each deployment element, define the main HW, SW, and FW components and their relationship. A UML deployment diagram with component annotations is appropriate, or separate deployment and component diagrams.

Here, one introductory paragraph is required and expected. It should describe the system at a very high level. E.g., “The Toddler Monitor system comprises two main subsystem, the user’s smartphone and the monitor. Major nodes and components in each subsystem are identified below”.>
The overall system will have two parts as shown in figure 1, the Fysh cross-compiler (Fyshc) in the development machine, and the RISC-V CPU (Fysh-Fyve) in the Zybo board.

5.1 System Architecture

<One or two UML diagrams. Either a UML deployment diagram with component annotations is appropriate, or separate deployment and component diagrams.

One or two pages should be needed including a brief text explanation in which the figure(s) are referenced>
<In the subsections below, list the major HW, FW, SW subsystems and components in each. For example, if the design relies on a smartphone, that is both a deployment node and a HW component. Note any special considerations, such as specific component or node requirements; e.g., a simple inertial motion unit may

be sufficient for a component that measures motion, or maybe something with 9 degrees of freedom is needed...>

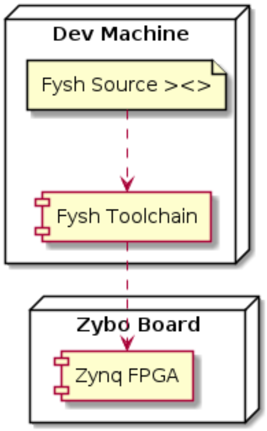

Figure 1: Deployment diagram for Fysh firmware

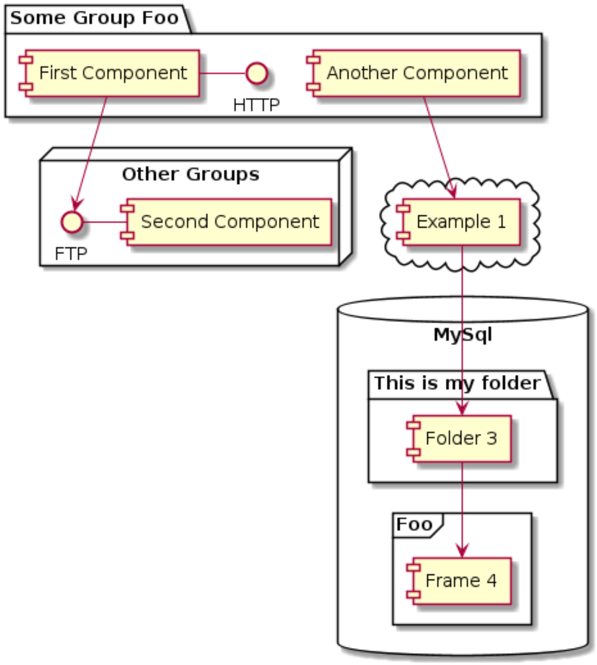

Figure 2: System Architecture <an example component diagram https://plantuml.com/component-diagram>

5.1.1 Hardware Components

<Describe the main HW components in each deployment note; refer to appropriate figure. 1 page max>

5.1.2 Firmware Components

<Describe the main FW components in each deployment note; refer to appropriate figure. May combine with software depending on project. 1 page max>

5.1.3 Software Components

<Describe the main SW components in each deployment note; refer to appropriate figure. May combine with firmware depending on project. 1 page max>

6 System Requirements

<Based on the deployment and component diagrams, identify the main system requirements for each note and component. Keep them very high level and concrete; should refer to tangible things like Microcontrollers, sensors, interfaces, size, weight, power, buttons, keypads, standards, software libraries, APIs, modules, OSes, versions, standards, ... Try to cross-reference with FRs and PRs. 1 page max>

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

Table 4: System Requirements

7 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.

<Define the key features/functions to be implemented first and show how they support an end-to-end functional system. The purpose is to implement the minimum needed to prove that each major components works. 1 page max>

8 High-Level Hardware Design

<For each hardware node/component in the architecture, describe the physical HW and how it is connected to other components. 1 page max>

9 High-Level Software/Firmware Design

<For each software/firmware component in the architecture, describe its implementation and how it is related to other components. 1 page max>

10 Prototype Budget

<Provide a rough-order-of-magnitude, ROM, protoype budget based on quantity 1 prices. Include only the main components. Don't include labour, development tools, and other supports. 1 page max. Note, the extended price will be the unit price if only 1 component is needed. If more than one, say Qty = N, then the extended price will be N*unit price>

Component	Mfr P/N	Mfr	Qty	Unit Price	Extended Price
FPGA Development Board	Zybo Z-7010	Xilinx/AMD	1	$299.00	$299.00
64x64 LED Matrix	RGB-Matrix-P2.5-64x64	Waveshare	2	$36.10	$72.20
				Total Cost	$371.20

Table 5: System Budget (ROM)