


Low-latency I/O RISC-V CPU core in FPGA fabric - Atharva Kashalkar


BeagleBoard.org
Jun 09, 2024

Table of contents

Chapter 1

Introduction

Implementation of PRU subsystem on BeagleV-Fire's FPGA fabric, resulting in a real-time microcontroller system working alongside the main CPU, providing low-latency access to I/O .

1.1 Summary links

- **Contributor:** Atharva Kashalkar
- **Mentors:** Jason Kridner, Cyril Jean
- **Code Repository:**
- – *Upstream Repository:* BeagleV-Fire gateware
- – *Daily Code Check-in Repository:* Fork for BeagleV-Fire gateware
- **Weekly/biweekly Updates Forums Thread:** Progress Reports
- **Blog:** TechTales

1.2 Status

This project is selected for GSOC 2024.

1.3 Proposal

Accounts created across OpenBeagle, Discord and Beagle Forum
PR was created for the cross-compilation task.

1.4 About

- **Resume:** Find my resume here
- **Forum:** u/roger18 (Atharva Kashalkar)
- **OpenBeagle:** Roger18 (Atharva Kashalkar)
- **Github:** RapidRoger18 (Atharva Kashalkar)
- **School:** Veermata Jijabai Technological Institute (VJTI)

- **Country:** India
- **Primary language:** English Hindi
- **Typical work hours:** 9 AM-11 PM Indian Standard Time
- **Previous GSoC participation:** First Time Applicant

Chapter 2

Project

Project name: Low-latency I/O RISC-V CPU core in FPGA fabric.

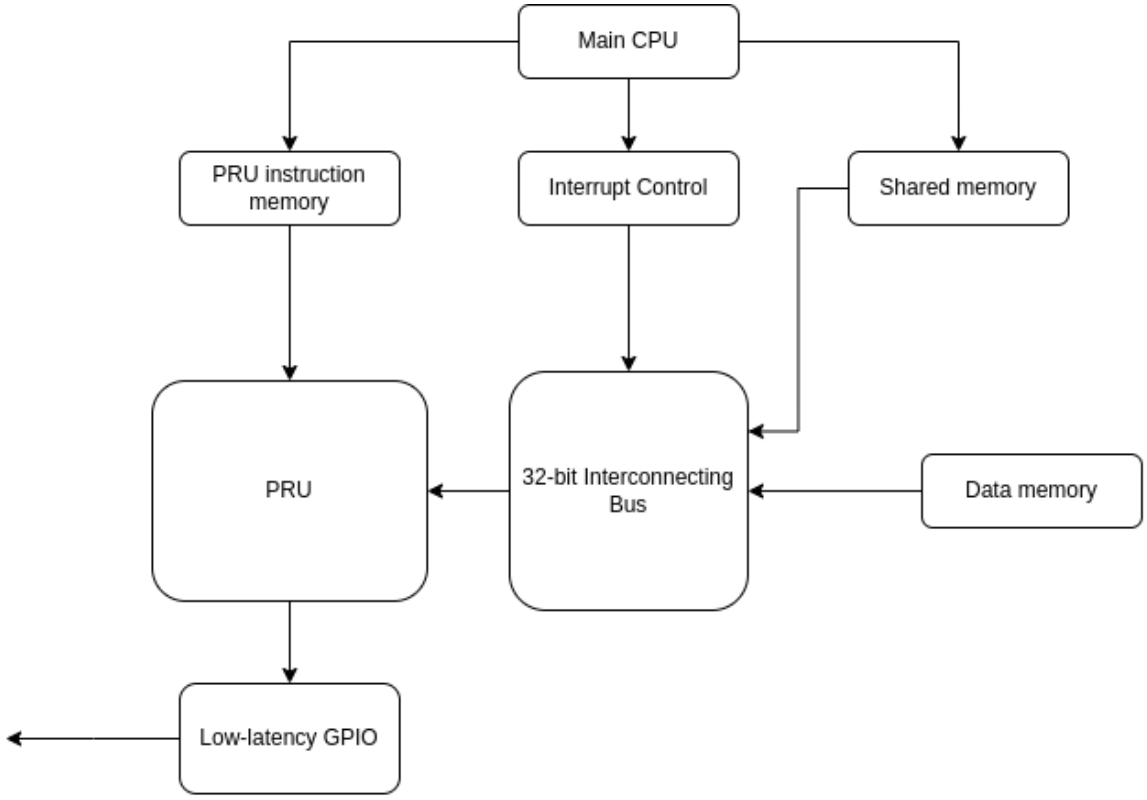
2.1 Description

To provide the capability of a Programmable Real-time Unit Industrial Control SubSystem (PRU-ICSS), which is present on several BeagleBone boards, I propose to deploy an existing RISC-V 32IM core with a customized Instruction Set on FPGA Fabric present on BeagleV-Fire. The goal of this deployment is to provide high bandwidth between the CPU and I/O, resulting in a on-board microcontroller.

2.2 Goals and Objectives

The main aim of this project is to deploy a soft core subsystem on BeagleV-Fire's FPGA fabric, functionally equivalent to PRU subsystem on BeagleBone Black. The core will feature RISC-V ISA customised to perform ultra low-latency I/O operations, i.e., single-cycle execution. This deployment will provide high-bandwidth data transfer in main CPU and I/O and also ensure high speed data processing similar to a microcontroller.

The programmable nature of the PRU, along with its access to pins, events, and all SoC resources, provides flexibility in implementing fast real-time responses, specialised data handling operations, custom peripheral interfaces, and in offloading tasks from the main processor cores of the system-on-chip (SoC).



Source:- Inspired from https://inst.eecs.berkeley.edu/~ee192/sp20/files/am335x-pru.pdf

Based on the requirements of the project, it is most efficient to use PicoRV (an open source RISC-V based processor) as a base and modify it to perform high speed I/O operations. The PicoRV possesses excellent compiler support and diverse instruction set but lacks I/O support and single-cycle execution for some instructions.

This problem will be resolved within the first stage of the project, which will focus on making the core I/O compatible and modifying its execution flow to ensure single-cycle execution for all instructions. Some present soft processor IPs like AMD's Microblaze used in Vivado Design Suite and Microchip's Mi-V used in Libero Design suite can provide good insights on how a functioning soft core IP will look like. As the Stage-1 of this project concludes with deployment of the RV core, Stage-2 will focus on establishing a communication medium between the PRU and the main CPU. This will ensure the 'on-the-fly' programming for the PRU and high bandwidth data transfer from I/O to the main CPU.

riscv64-unknown-elf-gcc compiler will compile the C program into bare-metal RISC-V based binary instructions within the linux booted on the main CPU. The communication between main CPU and PRU will be used to send these instructions into the program memory of PRU without needing to flash the FPGA each time.

This connection can be established in multiple ways:

1. The Program Memory of the PRU can be written into SPI flash that contains FPGA logic, and the data transfer will take place through 32-bit interconnecting AXI bus.
2. Shared memory between PRU and Main CPU can be used of PRU memory and Data Transfer.
3. 32-bit interconnecting AXI bus can be used to write instructions into Program memory of PRU which will be printed on FPGA logic, the same AXI bus will be used for other Data Transfers.

The 32-bit interconnecting AXI bus is more suitable for burst write data to RAMs, thus using shared memory or SPI flash makes suitable method to serve as a communication medium between PRU and CPU to ensure efficient and smooth processing.

At the conclusion of the project, BeagleV-Fire will host a fully functional PRU system that can be controlled and programmed through the main CPU.

2.3 Software

- Verilog HDL.
- Verilator.
- Libero SoC suite.
- Microchip Softconsole
- ModelSim ME.
- Linux.
- OpenBeagle CI.

2.4 Hardware

Ability to program BeagleV-Fire using serial port and set up JTAG for effective debugging.

Chapter 3

Timeline

3.1 Timeline summary

Date	Activity
April	Understand detailed use cases of existing cores and shortlist them based on requirements
May 1	Start bonding - Discussing implementation methods with Mentors
May 15	College Examinations
June 1	Start coding and introductory video
June 3	*Milestone #1, Introductory YouTube video (June 3rd)*
June 10	*Milestone #2, Modifiying RV core of I/O(June 10th)*
June 17	*Milestone #3, Modifying RV core for single-cycle execution(June 17th)*
June 24	*Milestone #4, BeagleV-Fire setup (June 24th)*
July 1	*Milestone #5, Verification of the core (July 1st)*
July 8	Submit midterm evaluations
July 15	*Milestone #6, Establishing communication (July 15th)*
July 22	*Milestone #7, Establishing communication (July 22nd)*
July 29	*Milestone #8, Setup Access (July 29th)*
August 5	*Milestone #9, Testing and Verification(Aug 5th)*
August 12	*Milestone #10, Documentation and Tutorial(Aug 12th)*
August 19	Submit final project video, submit final work to GSoC site, and complete final mentor evaluation

3.2 Timeline detailed

3.2.1 Community Bonding Period (May 1st - May 15th)

- Get to know mentors and discuss project implementation.
- read documentation, and get up to speed to begin working on the projects
- shortlisting pre-existing cores based on initial assessment, by reading available documentation.

3.2.2 Coding begins (May 27th)

3.2.3 Milestone #1, Introductory YouTube video (June 3rd)

- Make an introductory video
- Researching through other soft core IPs to understand the I/O interfacing used in the core.
- Setting up remote access on BeagleV-Fire and completing the LED-blink tutorial given in the documentation.

3.2.4 Milestone #2, Modifiying RV core of I/O(June 10th)

- Modification of the PicoRV core to interface GPIOs and memory.
- Removing unnecessary extensions to reduce size and complexity, without changing its efficiency.

3.2.5 Milestone #3, Modifying RV core for single-cycle execution(June 17th)

- Modification of the PicoRV core for single-cycle execution of all instructions.
- This is to make sure the PRU functions as required after modifications.

3.2.6 Milestone #4, BeagleV-Fire setup (June 24th)

- Integration of PRU core with BeagleV-Fire gateware to ensure PRU deployment through gateware.
- Temporary setup the program memory on FPGA logic for testing and verification of the core.

3.2.7 Milestone #5, Verification of the core (July 1st)

- Verification of all modifications to ensure correct instruction execution. This will be done using predefined verification methods.
- Ensuring stable PRU workflow.

3.2.8 Submit midterm evaluations (July 8th)

- Complete pending Stage 1 tasks, if any.

Important: July 12 - 18:00 UTC: Midterm evaluation deadline (standard coding period)

3.2.9 Milestone #6, Establishing communication (July 15th)

- Setup the program memory of the PRU within SPI flash as FPGA.
- Setting AXI bus access for data transfer.

3.2.10 Milestone #7, Establishing communication (July 22nd)

- Using Shared memory for data transfer and comparing the results with using AXI bus and selecting faster and reliable option.

3.2.11 Milestone #8, Setup Access (July 29th)

- Setting up a easy build script to compile the C program and send it to PRU's Program Memory.
- Necessary changes to TCL scripts and Device Tree.

3.2.12 Milestone #9, Testing and Verification(Aug 5th)

- Setting up gitlab CI/CD to include necessary files in the archives.
- Testing smooth and easy workflow from C program to execution in PRU.
- Testing the GPIO bandwidth and CPU resources with and without the use of PRU.

3.2.13 Milestone #10, Documentation and Tutorial(Aug 12th)

- Documenting the project and ways to access PRU on docs.beagleboard.org.
- Having an LED Blink tutorial for users to familiarize themselves with the PRU.

3.2.14 Final YouTube video (Aug 19th)

Submit final project video, submit final work to GSoC site, and complete final mentor evaluation

3.2.15 Final Submission (Aug 24th)

Important: August 19 - 26 - 18:00 UTC: Final week: GSoC contributors submit their final work product and their final mentor evaluation (standard coding period)

August 26 - September 2 - 18:00 UTC: Mentors submit final GSoC contributor evaluations (standard coding period)

3.2.16 Initial results (September 3)

Important: September 3 - November 4: GSoC contributors with extended timelines continue coding

November 4 - 18:00 UTC: Final date for all GSoC contributors to submit their final work product and final evaluation

November 11 - 18:00 UTC: Final date for mentors to submit evaluations for GSoC contributor projects with an extended deadline

---

Chapter 4

Experience

This project will require prior knowledge of Risc-V ISA, FPGA programming, Assembly, and Verilog.

I have previously worked on a project Risc-V 32IM core where I contributed towards the synthesis of a Risc-V core using Vivado Design suite and flashed it on UPduino FPGA using Yosys framework. This CPU was successfully able to run operations like the Fibonacci series and factorial of a number.

I was a Team Leader of the Semi-finalist Team in E-Yantra Robotics Competition (eYRC), IIT Bombay on Theme “AstroTinker Bot”, where we had to develop a single cycle Risc-V 32I core that was capable of implementing Dijktra Algorithm written in C code and cross-compiled to Risc-V binary instructions. This Competition exposed me to various debugging methods using JTAG connection through Quartus Prime Lite software. DE0-NANO development board with Cyclone IV FPGA was used to control the bot. The bot used PID-based Line Following and Electromagnet to pick and place blocks through an arena representing a Space Station. Verilog HDL codes for the project can be found on E-yantra_Astrotinker-bot github repo.

I have prior experience with ESP32 microcontrollers for small projects.

Chapter 5

Approach

As a starting point, I have forked the BeagleV-Fire gateware Repository and completed the Blinky Tutorial as specified on BeagleBoard Docs

Continuing on this I have set up Libero SoC suite and Softconsole on my Local machine that will provide an easy debugging method using JTAG. I have tested this setup by implementing a simple UART receiver and simulating it in ModelSim ME using a simple testbench.

The concept of having a microprocessor and a microcontroller on a single SoC is amazing. I can see a variety of applications where a SoC like BeagleV-Fire would be ideal. This inspires me to take on this project and give my contribution to the open-source community. I have my full commitment to this project and am willing to put in complete efforts to complete the project within the allocated time.

I would also like to keep improving with the project after GSoC.

5.1 Contingency

If I get stuck on my project and my mentor isn't around, I will use the following resources:-

- BeagleV-Fire
- PRU-Documentation
- Picorv32 Docs
- PRU cookbook
- TI PRU documentation

Moreover, the BeagleBoard community is very helpful in resolving doubts, I will use OpenBeagle forums to clarify any doubts left after referring to the above resources.

5.2 Benefit

This project will not only improve the use cases of BeagleV-Fire but also provide a very fast real-time subsystem that can be used as a microcontroller. It will provide the following functionalities: -

- Reduce memory and I/O resource consumption on the main CPU by performing basic computations in PRU itself thus providing more resources to perform complex tasks.
- Provide a real-time interface for fast, deterministic operations.
- Reprogammability of FPGA will allow to deployment of PRU only when necessary, thus providing additional logical elements when PRU is not being used.

- The configurable nature of FPGA will allow multiple levels of customizations and configurations on PRU, enabling the user to efficiently meet their requirements in the lowest possible resources.

The success of this project will result in an all-in-one SoC meeting all the requirements of BeagleBoard.org community.

5.3 Misc

PR request for cross-compilation task #182