BeagleBoard/GSoC/2021 Proposal/bela on bbai

< BeagleBoard | GSoC

Contents

Proposal-Bela support for BBAI

Youtube Video (https://www.youtube.com/watch?v=aVLRUyPBBJk)

About

Student: Dhruva Gole (https://elinux.org/User:DhruvaG2000)

Mentors: Giulio Moro (https://github.com/giuliomoro), Stephen Arnold and Robert Manzke

Code: https://github.com/BelaPlatform/Bela

Wiki: https://forum.beagleboard.org/t/bela-support-for-bbai-later-ti-chips/29257/7

GSoC: GSoC entry (https://summerofcode.withgoogle.com/projects/#5697403266531328)

Status

This project proposal has been accepted as part of GSoC 2021.

Proposal

Bela is a hardware and software system for creating beautiful interaction with sensors and sound. This project proposes to provide restructuring and improvement of existing Bela Software Code to allow for compatibility and easier transition to newer TI chips

(like in the BBAI).

About you

IRC: dhruvag2000
Github: https://github.com/DhruvaG2000
School: Veermata Jijabai Technological Inst.
Country: India
Primary language : English, Marathi, Hindi
Typical work hours: 10AM - 7PM Indian Standard Time
Previous GSoC participation: I find embedded and IOT pretty interesting, given I have experience with ESP32, SIMCOM, ESP8266, Atmega, and many other microcontroller devices and I think I will be able to excel in this project. This is the first time i am participating in GSoC.

About your project

Project name: Bela support for the BeagleBone AI
My Blog: My blog (https://dhruvag2000.github.io/bbai-notes/#Bela) related to this project and my findings.
Logs : I maintain weekly progress updates here (https://dhruvag2000.github.io/bbai-notes/logs.html).

Description

What is Bela?

As given on the official website (https://learn.bela.io/get-started-guide/say-hello-to-bela/#what-is-bela), Bela is a hardware and software system for creating beautiful interaction with sensors and sound. Bela consists of a Bela cape on top of a BeagleBone Black computer (uptil now). Bela has a lot of analog and digital inputs and outputs for hooking up sensors and controlling other devices, and most importantly Bela has **stereo audio i/o** allowing you to interact with the world of sound.
Both Bela systems use the same Bela software. It uses a customised Debian distribution which - most notably - uses a **Xenomai kernel** instead of a stock kernel. *Xenomai* is *co-kernel* for Linux which allows to achieve hard **real-time performance** on Linux machines (http://xenomai.org/). It thus takes advantage of features of the BeagleBone computers and can achieve extremely fast audio and sensor processing times. Although the proposal Title mentions support for AI, I will try to develop a standardized setup that allows an easy jump across all TI chips.

Applications of Bela

Bela is ideal for creating anything interactive that uses sensors and sound. So far, Bela has been used to create:

1. musical instruments and audio effects
2. kinetic sculptures
3. wearable devices
4. interactive sound installations

and many more applications that are listed here (https://learn.bela.io/get-started-guide/say-hello-to-bela/#what-is-bela)

Why add support for BBAI/newer TI chips?

The Beagle Black and PocketBeagle are getting outdated by the day and also it would be better to have a more standardised setup that allows to more easily jump across TI chips, soon newer boards with different and more efficient chips like the AM5X and the TI C66x digital-signal-processor (DSP) cores in the BBAI are coming up that will need compatiblity with the Bela Software and Hardware. As part of the process we will try to offload peripheral initialisation to the mcasp and spi driver to in order to make it more portable for porting to other boards in the family, at present or future.

Programming languages and tools to be used

C, C++, PRU, dtb, uboot, GNU Make, ARM Assembly

Proposed method of Implementation

The Bela cape is normally used in combination with the TI AM3358 chip on the BeagleBone Black. The hardware was partially working on the BBAI only using ALSA and the SPI driver [1] (https://github.com/giuliomoro/beaglebone-ai-bela), so the pinout and pin settings are somewhat okay, but the Bela real-time code on ARM and PRU is not running on BBAI yet. During this project I will:

1. Create a device tree overlay using Cape Compatibility layer for the BB-BONE-AUDI (https://github.com/beagleboard/bb.org-overlays/blob/master/src/arm/BB-BONE-AUDI-02-00A0.dts) which worked but with a few frequency issues.
2. create a device tree overlay for the BELA Cape to work on the BBAI using the Cape Compatibility layer
3. update the Bela code to use the McASP, GPIO and McSPI on the AM5729 SoC of the BBAI
4. adapt the Bela PRU and ARM code and workflow to use the PRU via remoteproc instead of uio_pruss
5. install a Xenomai patched kernel ✓ and run the full Bela stack.

The project will involve dealing with pinmuxing, uboot, PRU assembly, C++ for Linux user space applications, the Technical Reference Manual for the Sitara family of SoCs. (ref. here (https://elinux.org/BeagleBoard/GSoC/Ideas-2021#Beagle_Bone_Audio_Applications))

Syntax Analysis

The places within the Bela core code that will require intervention are:

1. in the Makefile, update the workflow to build the PRU code for remoteproc
2. in core/PRU.cpp, use remoteproc instead of the libprussdrv API
3. in pru/pru_rtaudio.p, the hardcoded McASP constants should be replaced with board-dependent ones.

All these changes will be made so that the same code base can run on all supported boards (e.g.: BBAI, BBB) with build- or runtime- checks. To explain in short how I aim to establish a common code base for supporting all boards, I will start off by enumerating different boards once we detect which board it is. Then in the libraries, the constants (like pin numbers and their symlinks) will be set accordingly and the rest of the code base will become much easier to use without the need to hard code anything.

PRU

1. The current Bela core code uses *pasm* to build the PRU assembly *pru/pru_rtaudio.p* and uses libprussdrv, which binds to the uio_pruss kernel driver to load the firmware to the PRU and handle access to the PRU RAM. Both pasm and libprussdrv are now deprecated, replaced by the clpru toolchain and remoteproc driver respectively.

2. The PRU firmware contains hardcoded values for the address of the McASP, McSPI and GPIO peripherals. These addresses will change for the BBAI, so these constants need to become conditional. ✓(yet to be tested, but base address for the McASP of the bbai has been updated)
3. As the Bela PRU firmware is written in assembly for pasm, instead of rewriting or updating it in such a way that it will stop working on the current Bela images, we will use the workflow detailed below.

Workflow for building the existing pasm PRU code with clpru

The workflow below works as a proof of concept, but needs further testing.

1. Build the .p file as is with pasm with *-V2 -b*. This generates a .bin file that contains the assembled program
2. I will be using the disassembler Giulio Moro put together hacking the one that was inside prudebug. (Find it here (https://github.com/giuliomoro/prudebug/tree/disassembler)(built with *gcc da.c -o prudis*) . (**Note:** *A disassembler* is a computer program that translates *machine language* into *assembly language*)
3. Process the bin through the disassembelr and make it ready to be included inside an *__asm__* directive (i.e.: add quotes and prepend a space at the beginning of each line):

```
  prudis file.bin | sed 's/^\(.*\)$/" \1\\n"/' > included_assembly.h
```

4. have the following in main.c:

```
void main()
{
    __asm__ __volatile__
    (
#include "included_assembly.h"
    );
}
```

5. build main.c with the regular clpru toolchain

6. I will also need to change these MCASP addresses in pru/pru_rtaudio.p :

```
#define MCASP0_BASE 0x48038000
#define MCASP1_BASE 0x4803C000
```

```
#define MCASP0_DATAPORT 0x46000000
#define MCASP1_DATAPORT 0x46400000
```

Additionally, I will need to compare the *McASP* and *McSPI* sections of the AM5729 and AM3358 manuals to verify that all the registers of the McASP and McSPI peripherals kept the same meaning and offsets between the two chips, or adjust the code accordingly.

PINMUXING

I am yet to figure out pinmuxing to make sure the relevant pins are set to the correct function (they are NOT by default). I will see here https://elinux.org/BeagleBoard/GSoC/2020_Projects/Cape_Compatibility and/or ask on the BB slack about for the best way to go about it. I have so far created a basic AUDIO overlay (named BBAI-AUDI-02-00A0.dts) that has been merged into the beagleboard / BeagleBoard-DeviceTrees repository (https://github.com/beagleboard/BeagleBoard-DeviceTrees/blob/v4.19.x-ti-overlays/src/arm/overlays/BBAI-AUDI-02-00A0.dts).

PRU->ARM INTERRUPTS

Bela can work with a PRU->ARM interrupt, which is default these days, but requires an rtdm driver, which is another layer of complications. As an intermediate step to avoid further complications, I will try to run it without the PRU->ARM interrupt by adding *BELA_USE_DEFINE=BELA_USE_POLL* to my *make* command line. After building the Xenomai kernel, I will build the rtdm_ kernel and revert to using PRU->ARM interrupts.

PRU transitioning from libprussdrv to rproc

The Remote Processor Framework (https://www.kernel.org/doc/Documentation/remoteproc.txt) (rproc) allows different platforms/architectures to control (power on, load firmware, power off) those remote processors while abstracting the hardware differences, so the entire driver doesn't need to be duplicated. I will need to change the initialization code in PRU.cpp (https://github.com/BelaPlatform/Bela/blob/master/core/PRU.cpp#L18) that is currently relying on lib *prussdrv* and move to *rproc* . Not sure if rproc provides some functionalities to access the PRU's RAM the way *prussdrv_map_prumem()* used to, but that essentially gives access to a previously mmap'ed area of memory , so it should be easy to replace, as long as I am able to find the correct addresses in the TRM. On the latest Bela code there's a *Mmap* class which can make this somehow simpler ref. here (http://docs.bela.io/classMmap.html).

XENOMAI kernel

Xenomai kernel (v4.19.94-ti-xenomai-r64) has been built and tested for the BBAI. I have installed the xenomai kernel and libraries (https://dhruvag2000.github.io/Blog-GSoC21/xenomai/install.html) through the default procedure to update kernel. I have also managed to successfully build the entire Bela core code on the BBAI so far.

Hardware required

The hardware listed below will be necessary for testing if my code implementation works correctly on the hardware.

1. BeagleBone AI (https://www.digikey.com/product-detail/en/seeed-technology-co-ltd/102110362/1597-102110362-ND/10492208).
2. Bela cape: The original Bela board (https://shop.bela.io/bela).

Timeline

Mar 29	Applications open	Students register with GSoC, work on proposal with mentors.
Apr 13	Proposal complete	Submitted to https://summerofcode.withgoogle.com ✓
May 17	Proposal accepted or rejected	1. Proposal Accepted (https://summerofcode.withgoogle.com/projects/#5697403266531328) ✓ 2. Community Bonding Period and discussion on the project and resources available. 3. Learn about embedded linux structure ✓ 4. Setting up beaglebone-ai i.e flashing up to date Linux image (https://beagleboard.org/latest-images) and connect to local area network (LAN) via either Ethernet or WiFi and try to run example codes from this repository (https://elinux.org/BeagleBoard/GSoC/2021_Proposal/bela_on_bbai) to test basic stuff is working.
Jun 07	Pre-work complete, Coding officially begins!	1. All the boards and BELA Cape will be available to me at this period of time and I should have set up my BeagleBone Black and Ai boards i.e flashing up to date Linux image. 2. Initial checks for hardware like audio ports and other peripheral devices will be completed. 3. A detailed spreadsheet will be created if not already available for cape pin mapping and referencing for further use during BeagleBone AI software development ease. (one exists here (https://docs.google.com/spreadsheets/d/1h-oUVhZXogOkKJkq73dM1JPOzcslBfcdpxTx4fZ-Cg0/edit#gid=227990209)) (some points above are ref. from here (https://elinux.org/index.php?title=BeagleBoard/GSoC/2020_Projects/Cape_Compatibility))
Jun 17	Milestone #1,	1. Introductory YouTube video ✓ 2. Setting pinmux values appropriately and fix the dtb to get the correct clock on the McASP MCLK pin. ✓ 3. Writing the BBAI-AUDI-02-00A0.dts (https://github.com/beagleboard/BeagleBoard-DeviceTrees/blob/v4.19.x-ti-overlays/src/arm/overlays/BBAI-AUDI-02-00A0.dts) overlay to port the old BB-BONE-AUDI overlay using the CCL to work on the AI. ✓ 4. verify that it works running with ALSA. ✓
June 24	Milestone #2	1. Writing the overlay for Bela Cape. 2. Study about the remote processor framework and rpmesg, and how I can integrate it into the PRU.cpp code 3. Test and debug the workflow for building the existing pasm PRU code with clpru, (or possibly reconsider/change it)
June 30	Milestone #3	1. Try to implement the PRU code using *rproc*. 2. Try to test and debug running Bela's PRU code on BBAI. 3. make READMEs for undocumented but pre-existing codes (for eg: this (https://github.com/BelaPlatform/rtdm_pruss_irq)) .
July 12 18:00 UTC	Milestone #4	1. Try to run Bela examples with Xenomai, but without rtdm(Real-time data monitoring) ie. generate (PRU->ARM interrupts)

		2. Start documenting everything done so far. 3. Make demo videos for proving whatever that has been done so far works on actual-hardware. Mentors and students can begin submitting Phase 1 evaluations
July 16 18:00 UTC	Phase 1 Evaluation deadline	
July 23	Milestone #5	1. run Bela with Xenomai, with (PRU->ARM interrupts) 2. Try to run the stock tutorials (https://learn.bela.io/tutorials/)
July 30	Milestone #6	1. Try to update the *.p files to suit the new gcc-pru supported syntax. 2. Validate existing work with some functional tests and examples (https://github.com/BelaPlatform/Bela/tree/master/examples/). 3. Test and debug if any errors occur while running the stock tutorials and functional tests.
Aug 06	Milestone #7	1. Create detailed Documentation about the above implementation. 2. Provide guides on how to provide support for upcoming TI processors into the existing code base.
August 10	Milestone #8	1. Writing the overall project summary and outcomes. 2. Start making the Completion YouTube video.
August 16 - 26 18:00 UTC	Final week: Students submit their final work product and their final mentor evaluation	
August 23 - 30 18:00 UTC	Mentors submit final student evaluations	

Experience and approach

1. I have used C++, C and Python programming languages over the past 3 years in a variety of projects involving embedded systems using the ESP32, Arduino UNO, ESP8266 and am also well-versed with freeRTOS.
2. I have an aptitude for writing good reports and blogs, and have written a small blog on how to use a debugger (https://dhruvag2000.github.io/gdb-for-noobs/).
3. I recently did a project (https://github.com/DhruvaG2000/ESP32_wifiDHT11) using ESP32, in which I used the DHT11 sensor to display humidity and temperature on a local HTML server . Other than that I have worked on developing hardware and making documentation for a 3 DOF arm (https://github.com/SRA-VJTI/ROS-Workshop-2.1) based on an ESP32 custom board.
4. I also interned at an embedded device startup where I

    1. Interfaced ADS1115 ADC with the ESP32 and used it to read battery voltage.
    2. Used UART for ESP32 and SIMCOM SIM 7600IE communication to gain LTE support.
    3. Published local sensor data to the cloud via LTE.
5. I actively contribute to open source (most recently, I contributed to the ADS1115 library for ESP32 on the unclerus repo and can be seen here (https://github.com/UncleRus/esp-idf-lib/pull/163)).
6. Currently I am working on designing a Development board (https://github.com/DhruvaG2000/RP2040_Dev_Board) for the Raspberry Pico (RP2040) using KiCAD.
7. I also do a lot of mini projects throughout the year, you can find my several more interesting projects at my github page (https://github.com/DhruvaG2000)

Contingency

I believe that if I get stuck on my project and my mentor isn't around, I will use the resources that are available to me. Some of those information portals are listed below.

1. Derek Molloy's beagle bone guide (http://derekmolloy.ie/beaglebone) provides all the information needed for getting up and running with my beagle.
2. remoteproc (https://www.kernel.org/doc/html/latest/staging/remoteproc.html)
3. BBAI vs BBB pin headers Google Sheet (https://docs.google.com/spreadsheets/d/1h-oUVhZXogOkKJkq73dM1JPOzcslBfcdpxTx4fZ-Cg0/edit?usp=sharing)
4. beaglebone pru gpio example (https://credentiality2.blogspot.com/2015/09/beaglebone-pru-gpio-example.html)
5. official BELA website (https://learn.bela.io/tutorials/)
6. Ask on the BeagleBoard.org forum (https://forum.beagleboard.org/t/bela-support-for-bbai-later-ti-chips/29257/4)

Benefit

If successfully completed, this project will add support for the Bela cape + Xenomai + PRU on the BeagleBone AI, and also the code will be easier to port to other Texas Instruments systems-on-chip.

By going through the steps needed to have the Bela environment running on BBAI, we will go through refactoring and rationalisation, using mainline drivers and APIs where possible. This will make Bela easier to maintain and to port to new platforms, benefiting the project's longevity and allowing it to expand its user base.
-Giulio Moro

Misc

Completed all the requirements listed on the ideas page (https://elinux.org/BeagleBoard/GSoC/Ideas-2021#General_requirements). The code for the cross-compilation task can be found here (https://github.com/DhruvaG2000/gsoc-application) submitted through pull request #149 (https://github.com/jadonk/gsoc-application/pull/149).