GsoC 2016 Application

About me

First name: Patryk
Surname: Mężydło
E-Mail: mezydlo.p@gmail.com
IRC nickname: pmezydlo
Programming Languages: C, C++, Python, Verilog, Assembler
Interests and hobbies: Electronic, Embedded systems, FPGA, Judo, Walking
Native language: Polish
Other languages: English
Timezone: CET
GitHub: https://github.com/pmezydlo
Work hours: 10:00-22:00 (CET)

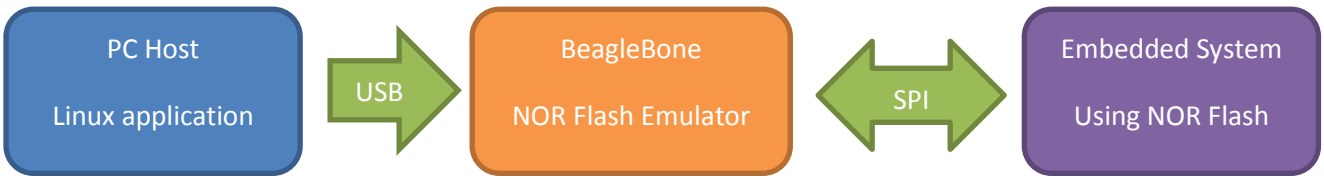
Currently I am in my second year of engineering studies at Gdańsk University of Technology in Poland. This is the first edition of gsoc I'm applying for, I would like to test my skills and contribute to the development of the BeagleBoard community, I want to gain additional experience.

About project

My organization is BeagleBoard.
The website is here: http://beagleboard.org/

SPI Flash Emulator

We encounter the non-volatile nor-flash memory in embedded systems more and more frequently. Unfortunately, it is not a good memory considering its time parameters. Writing and erasing it takes quite a long time even the use of high-speed programmer. My task will be to develop a Nor-flash emulator, which would shorten the time required to test the prototypes.



Implementation of the project

80% of the project is destined for BeagleBone and it is based on writing LKM (Loadable Kernel Module) for Linux operating system.

The main task of the driver during the initialization is to set the way McSPI hardware interface works. McSPI will be used for communication between the AM335x processor with peripherals through SPI.

For efficient exchange of data and time optimization I will use hardware DMA and interrupts.

Memory allocation through module will be held also during the initialization (size of the allocation given in parameter during the installation of the driver).

Two GPIO outputs will be used to indicate the driver's condition by LEDs.

Thanks to the use of kobjects I will get an easy acess to the allocated memory, intended to serve as emulator's memory (storing the current content of NOR Flash memory).

The remaining 20% of the project will be to create a communication bridge between the PC computer (Linux operating system) and BeagleBone, which will make memory programming possible.

After launching and providing the batch file's name in parameters, it will start communication. The communication's correctness will be validated by checksum CRC32.

Development timeline

I will devote a few hours each week to write the documentation.

0. Before the first week (26 May - 28 May)
- creating a GitHub repository,
- creating eLinux page,
- preparation of few sd card with a Firmware Images,
- installation of necessary software,

1. Week 1(23 May -29 May)
- skeleton LKM
- implementation GPIO
- create functions KObject
- allocation memory
- create debugging tool(reading logs kernel)

2. Week 2(30 May-4 June)
- configuration USB device
- create USB Driver
- write PC host application (LIBUSB)

3. Week 3(06 June -11 June)
- transfer data to beaglebone
- transmission of binary files

4. Week 4(13 June-18 June)
    - functions counting checksum (CRC32)
    - finally write PC host application (LIBUSB)

5. Week 5(20 June-25 June)
    - tool for observing the state driver
    - reading and changing registers
    - define McSPI register

6. Week 6(27 June -2 July)
    - set McSPI registers in slave mode
    - preparation of test device SPI
    - first tests (reading McSPI buffer)

7. Week 7(4 July-9 July)
    - create all interrupt
    - calculating an address
    - reading and writing to memory
    - tests without DMA

8. Week 8(11 July -16 July)
    - set dma registers
    - create fifo

9. Week 9(18 July-23 July)
    - interrupt for dma
    - rewriting with fifo to memory
    - tests with DMA

10. Week 10(25 July -5 August)
    - create a bash script to compile
    - more tests
    - clean up code

11. Week 11(6 August -20 August)
    - latest tests
    - documentation, tutorials and examples.

The completion of the project will help me

To successfully accomplish the project, I will do my best to thoroughly study the subject in order to understand it well.

More difficult parts of the project, such as, for instance, operating McSPI, can be made easier by good contact with the mentor, regular commits and familiarizing with documentation and ready solutions.

I have already designed pcb for cape, which adjusts voltages between emulator and embedded system. It enables to connect to emulator any embedded system.

Python script reading kernel logs, as well as digital oscilloscope capable of decoding SPI will help me in the process of debugging.

I spent the last four months familiarizing myself with BeagleBone.

During this time I was able to do the following:

- compile the kernel, bootloader and device tree
- submit compiled parts
- eMMC and SD flash
- write simple LKMs (kernel threads, interrupts, linux headers)
- LKMs mechanism, kobjects, gpio, memory allocation
- write assembler code for PRU and write DTS for GPIO

Thoughts on the project, misc

I hardly ever encounter negative opinions about the time required for nor flash programming during the tests of embedded systems, but this solution is rather expensive, as it requires the use of professional emulators, which price reaches up to $1000. With the use emulator based on BeagleBone, the costs could be reduced by several times.

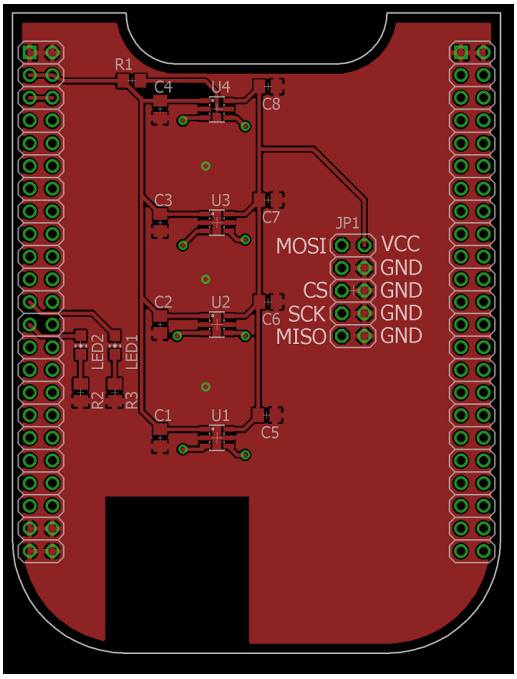
I think that reducing the time and costs will positively influence the community, because the emulator could be accessed by beginner programmers.

Each project gives valuable experience to the BeagleBone community in the form of code, that –well described – gives plenty of examples and ready solutions for other beginner programmers.

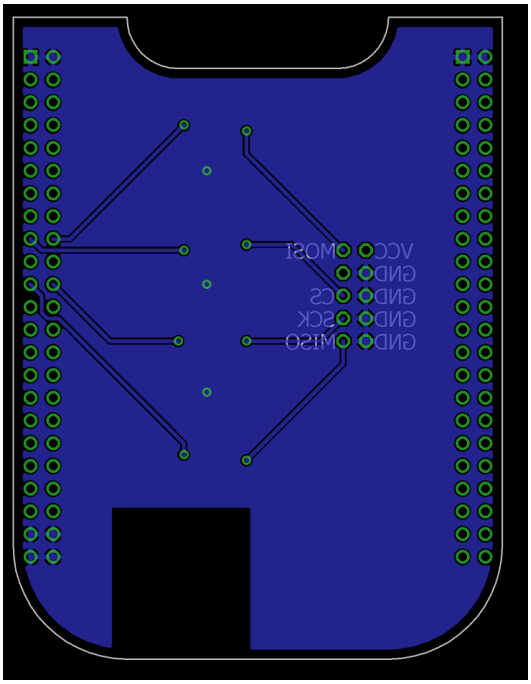
If I encounter a problem and my mentor would be unavailable at the moment, the first thing I would do is to grab a pencil and a piece of paper – apparently these are the best helpers for a programmer. Later I would try to find the reason behind this problem. Finally, I would try to do something else.

CAPE PCB Layout

TOP



BOTTOM



Scheme