Google Summer of Code 2014
Project with BeagleBoard.org – Report

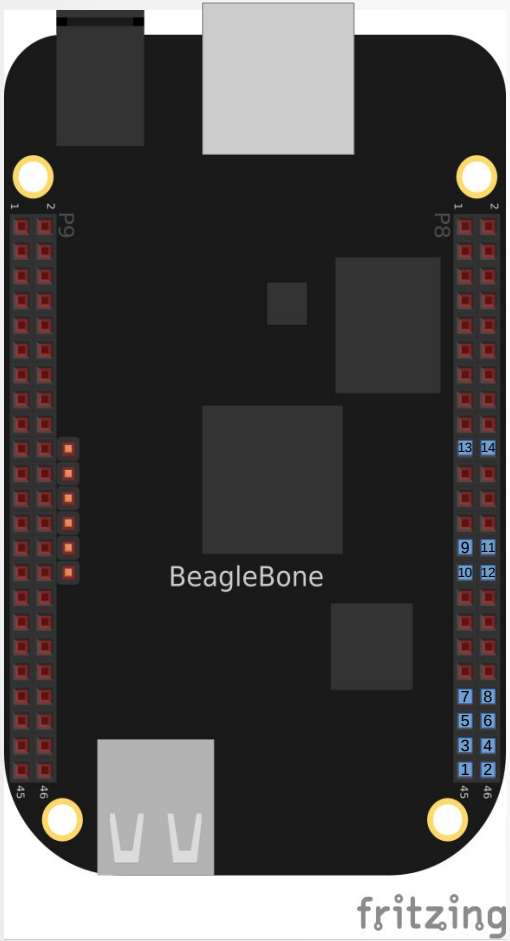
BeagleLogic

A logic analyzer on the BeagleBone Black

Kumar Abhishek
Indian Institute of Technology, Kharagpur

Mentors: Matt Ranostay, Hunyue Yau and Charles Steinkuehler

Your BeagleBone: Now a Logic Analyzer

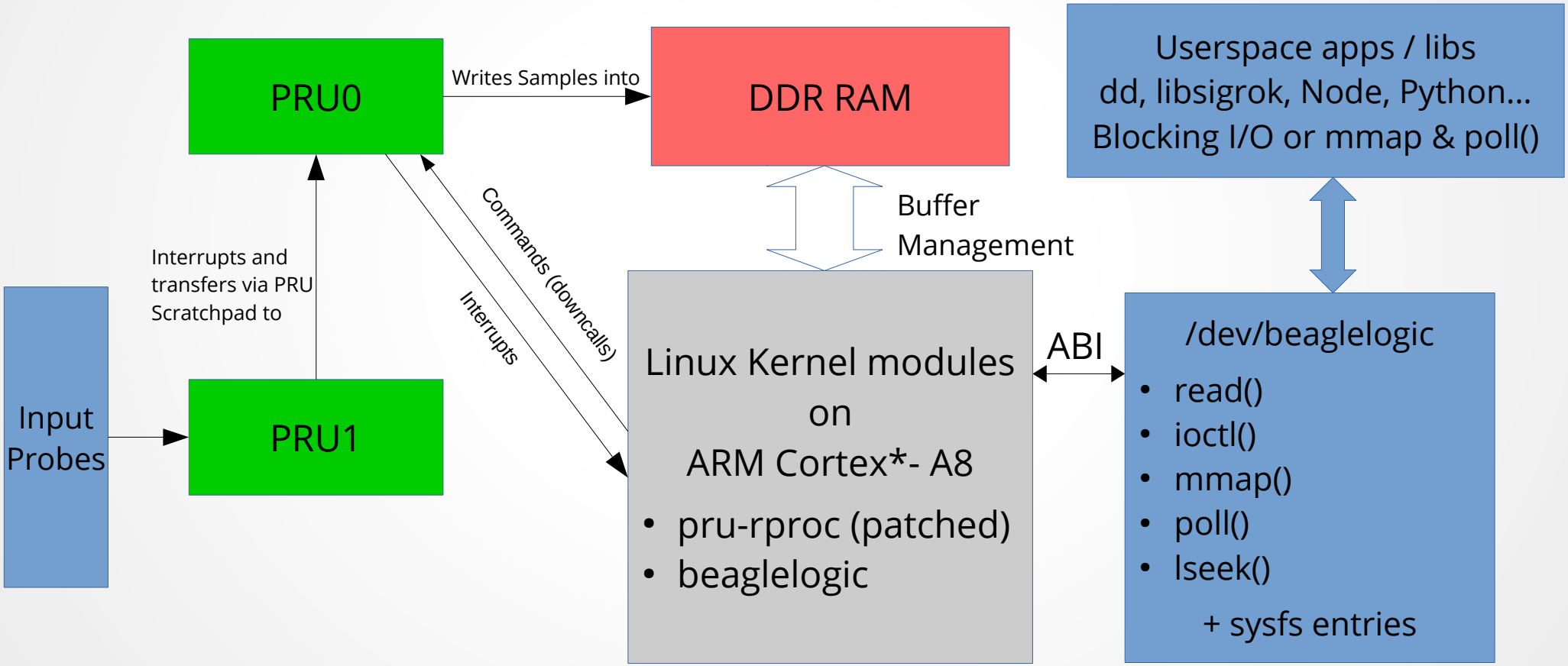


- Up to 14 channels (when booting from uSD)
- Max Sample Rate: 100 Msamples/sec

  Any (100 / n) [n = integer] MHz sample rate is supported
- Continuous (FIFO) and one-shot capture modes
- Maximum Buffer Depth: ~320 MB
- Capture and process *in situ* (with sigrok)
- A proof-of-concept web interface accessible at port 4000 for small capture sizes (upto 3K samples)

(** Requires disabling HDMI cape via uEnv.txt)

How it's made possible



BeagleLogic Firmware v1.0

- Week 1 & 2
- PASM + libprussdrv (UIO)
- Tried and validated approach of using ***both PRUs in tandem***
  PRU1 – Sample Loop, PRU0 – Writing to memory [like DMA]
- Uses the XIN and XOUT assembly instructions for broadside transfer between PRU1 and PRU0 in a single clock cycle
- **Advantages over Single-PRU approach:**
  - Sampling made independent of memory/bus latencies
  - 100MSamples/sec maximum achievable throughput [Single-PRU approach limited to max 50MSamples/sec]

Issues with the UIO driver [uio_pruss] & libprussdrv

- memcpy() from PRU shared memory too slow (~25 MB/s) compared to usual 200MB/s+ memcpy() speeds with normal malloc'd memory on the Bone
- Why?
  - *uio_pruss allocates memory with dma_alloc_coherent()*
  - *memory marked non-cached; always fetched directly from RAM. Slow memcpy.*
- uio_pruss crashed if instructed to allocate more than 8 MB of shared memory
- Therefore this approach led to:
  - Limited sample depth
  - Real time capturing difficult
- Could we do better?

The 'beaglelogic' kernel module

- Began as a part of the pru_rproc kernel module which was a remoteproc implementation of driving the PRUs [`tree/drivers/remoteproc/pru_rproc.c`]
- Separated from `pru_rproc.c` into `beaglelogic.c` once ready.
- Inspiration from the WS28xx Lighting example for PRU remoteproc by Matt Ranostay
- If I can allocate only max. 8 MB of memory chunk in one go, why not allocate a larger buffer in 8 MB chunks instead?
  - *Scatter-Gather I/O*
  - Allocate memory with *kmalloc( )* instead of *dma_alloc_coherent( )* - allow caching
  - Then write physical address of DDR Memory chunks into PRU0 Data RAM
  - PRU0 raises IRQ when done with one buffer chunk and moves to the other chunk on-the-fly filling the buffers one-by-one
  - Since we are in the kernel now, ensure cache coherency using dma_map_*( ) and dma_unmap_*( ) at the appropriate moments

/dev/beaglelogic

- The biggest advantage of the kernel module approach
- Expose the complete capture buffer as a character device node which can be read from just like a normal file.
- Turns “dd” into a Logic Analyzer client!
- `dd if=/dev/beaglelogic of=mycapture bs=256K count=32`
  is all you need to do on your BeagleBone to capture 8 MSamples :)
- While doing normal blocking I/O, the read( ) is automatically handled by the kernel module to return only when it has a full buffer, so the userspace application does not have to worry about IRQs and stuff.
- But, How to configure?

Configuration

- sysfs @ `/sys/devices/virtual/misc/beaglelogic` – write access is root only
- Key Attributes:
  - `samplerate` – from 10 to 100000000 (in Hz)
  - `triggerflags` – 0 : one shot; 1 : continuous [till close(fd)]
  - `sampleunit` – 0 : 16-bit samples (Little-Endian), 1: 8-bit
  - `memalloc` – Amount of memory to reserve (in bytes)

    Tested upto 320 MB

    More and there will be a kernel oops

    When the module is first loaded with modprobe/insmod, memory is not allocated, so memalloc must be done before first data capture through BeagleLogic
- Userspace applications accessing /dev/beaglelogic can also perform ioctl() for the same purpose on the open file handle (fd).
- For a complete reference, refer Wiki

Results

- Could reserve up to 320 MB of the system memory.
- copy_to_user( ) on buffers giving speeds up to 200+ MB/s
- Possible to do streaming capture if storage/network capabilities allow so. If doing compression, limited by CPU load percentage (considering memcpy's as well).
- Able to capture 1 GB of samples with LZO compression using simple *nix tools at 10 MSamples / s like this:

```
dd if=/dev/beaglelogic bs=1M count=1024 | lzop > dump.lzo
```

- This was actually a I^2S dump which was subsequently decoded to 100 seconds of Wave Audio using sigrok !

BeagleLogic Firmware v1.2 & Kernel Module

- Weeks 3 to 6 [Kernel Module – v1.0 & Firmware v1.2]
- Weeks 7 & 8 [Kernel Module – v1.1]
  - Added non-blocking I/O and made mmap( ) functional
- Ported the PASM Core code to the new PRU C Compiler
- Firmware now in C with cycle-critical code in ASM
- Single Device tree overlay that configures default settings (samplerate etc.) for BeagleLogic and loads the firmware into the PRUs [no libprussdrv]
- The module patches have been submitted to the BeagleBone kernel [maintained by Robert Nelson] and are **now present in kernels v3.8.13-bone60 and above**.
- "`modprobe beaglelogic`" & "`config-pin overlay BB-BEAGLELOGIC`" initializes BeagleLogic on the latest Debian images, provided HDMI cape is disabled on boot*

  * firmware files need to be manually copied to /lib/firmware if not already present

sigrok bindings for BeagleLogic

- Merged into upstream libsigrok on 22nd July
- Driver is named “beaglelogic”
- Channels named as: [P8_39 to P8_46]
- Sample command line with sigrok-cli:

```
sigrok-cli -d beaglelogic --samples 1M -c
samplerate=10M --channels P8_46,P8_45,P8_44 -o
output.sr
```

- Only 8 Channels [P8_39 to P8_46] are enabled by default
- To enable the 4 (or 6) extra channels:

```
-d beaglelogic →-d beaglelogic:numchannels=14
```

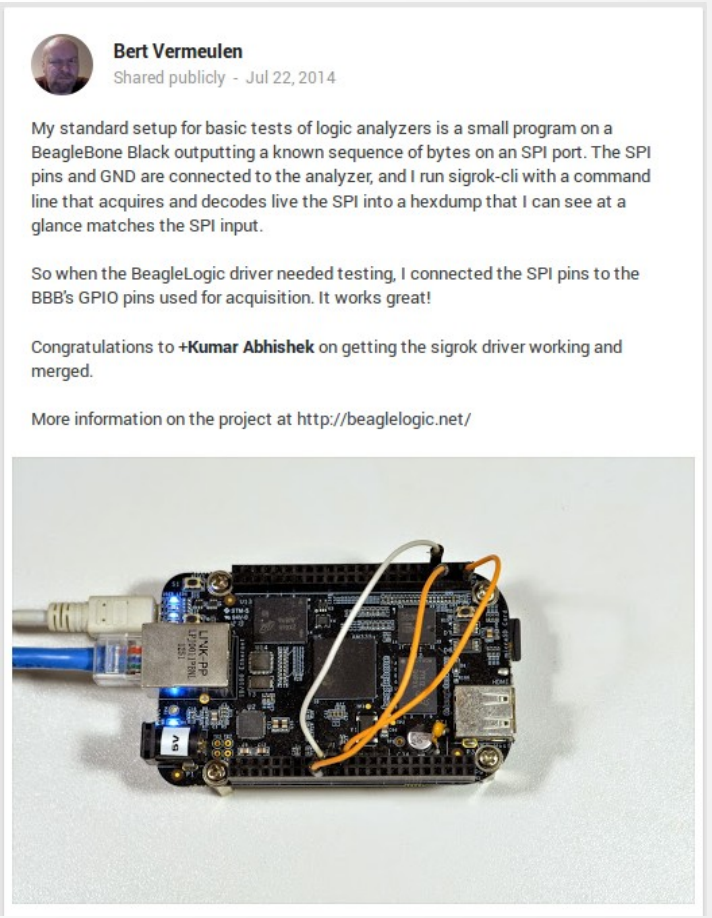
sigrok bindings for BeagleLogic

Internals:

- Configures the kernel module through ioctl( ) calls
- Accesses the kernel capture buffers directly through mmap [zero copy] – so data will not be copied unless required to be used.
- Uses polling (non-block I/O) with GPollFD.
  Since sigrok supports GPollFD for use with devices utilizing serial port, the binding code was simple.

BeagleLogic Featured

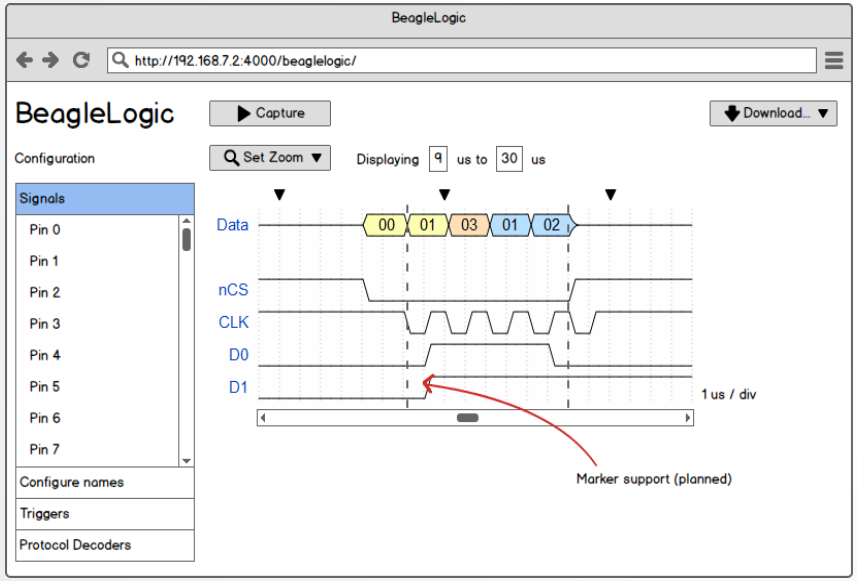
On HackADay

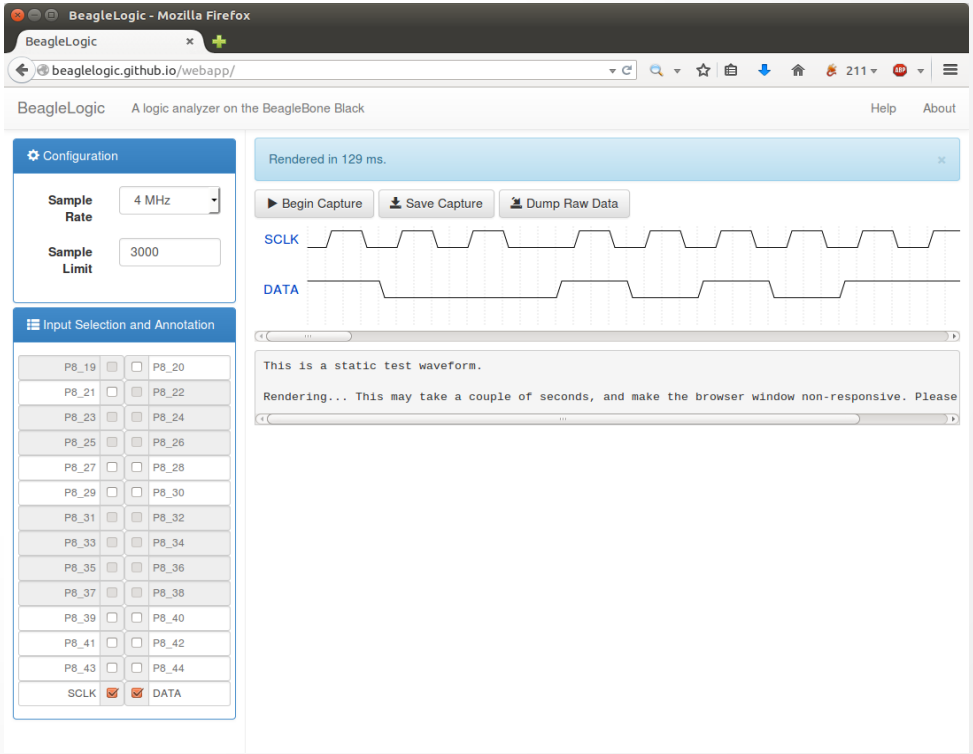


Web Interface

- Taking the Logic Analyzer interface to your web browser.
- The most interesting part of the BeagleLogic Project, which makes it a companion to Bone101

Mockup presented in the proposal





Web Interface Internals

Frontend built on:

- Bootstrap [UI]
- WaveDrom [The rendering of the waveforms]
- jQuery
- Socket.IO-Client [WebSocket between client and backend]

Backend:

- Node – tree/beaglelogic-server/app.js
  - Socket.IO
  - Express Static File Server (express.static) at port 4000
  - Child process (launches sigrok-cli , receives and sends the output to client via Socket.IO)

Web Interface Internals

- Backend serves the web page (client) on port 4000. Can be usually accessed at http://192.168.7.2:4000
- Establishes a Socket.IO Link between the client and the backend
- The Client requests capture over a message. The server:
  - Executes “sigrok-cli” with the appropriate command line args
    ** This will be replaced with node-ffi bindings for libsigrok or a custom C application in the near future*
  - Captures the stdout & stderr of the child process and sends it over to the client
- The Client receives and parses the output to WaveJSON required by WaveDrom for rendering. It also assigns custom labels (annotations) to pins as given by the user to give more readable names e.g. P8_45 = SCLK, P8_46 = SDATA and so on.

Current Limitations of the Web Interface

- Captures limited to 3K samples as larger captured sets are not efficiently rendered by WaveDrom (can make your browser window non-responsive for over a minute with > 50K Samples).
  Long-term Workaround: A Custom Canvas based view component
- The link speed between the Bone and the PC (over RNDIS) is ~50 Mbps on average. But data transfer through Socket.IO happens only at 400 KB/s ~ 3 Mbps
  Long-term Workaround: WebSockets in C; compression with RLE/LZ4
- No Zooming into / out of the waveform. To save the captured waveform, you save the webpage itself.
- The web interface in its current form is usable and well-suited for learning logic protocols like I2C, SPI and 1-wire and as a quick debugging tool.

How to get BeagleLogic (up and running)

- If running the official BeagleBoard Debian System image and the kernel:

```
sudo /opt/scripts/tools/update-kernel.sh
sudo cp -v /opt/scripts/device/bone/capes/BB-BEAGLELOGIC/* /lib/firmware
```

  - BeagleBone kernels v3.8.13-bone60+ fully support BeagleLogic and sigrok (As of the time of writing the latest is bone63)
- git clone, compile and “make install” the latest version of libsigrok, libsigrokdecode and sigrok-cli on the BeagleBone.

  ** sigrok packages are available for Debian but the current version does not include BeagleLogic support*
- Disable HDMI cape in uEnv.txt and reboot
- Load the kernel module with “`modprobe beaglelogic`” and the device tree overlay with “`config-pin overlay BB-BEAGLELOGIC`” . Refer to kernel logs with “dmesg” for detailed status information
- Allocate sample buffers with “`echo 8388608 > /sys/devices/virtual/misc/beaglelogic/memalloc`”
- Ready to capture

The road ahead

- The official BeagleLogic cape
- BeagleLogic can be more than a logic analyzer – leveraging the high speed data capture interface for purposes beyond observing digital signals.
- USB Gadget kernel drivers for BeagleLogic
  - To be added to the RNDIS, CDC (Virtual Serial Port) and Mass Storage interfaces currently offered by the BeagleBone out-of-the-box
  - Can use with PC-based clients with a custom protocol / real time compression to utilize the full link bandwidth
- Improvements to the web interface
- Porting to latest kernels and mainlining the driver
- Suggestions from the community

Stay updated @

www.beaglelogic.net - All Documentation

Blog:

www.theembeddedkitchen.net/gsoc2014/

Code @GitHub:

www.github.com/abhishek-kakkar/BeagleLogic/

IRC (freenode): #beagle ; nick: Abhishek_

Thanks to

- Jason Kridner and the BeagleBoard.org community
- Project Mentors:
  - Matt Ranostay
  - Hunyue Yau
  - Charles Steinkuehler
- Pantelis Antoniou
- Bert Vermeulen and the sigrok community
- IRC Channels: #beagle-gsoc, #sigrok