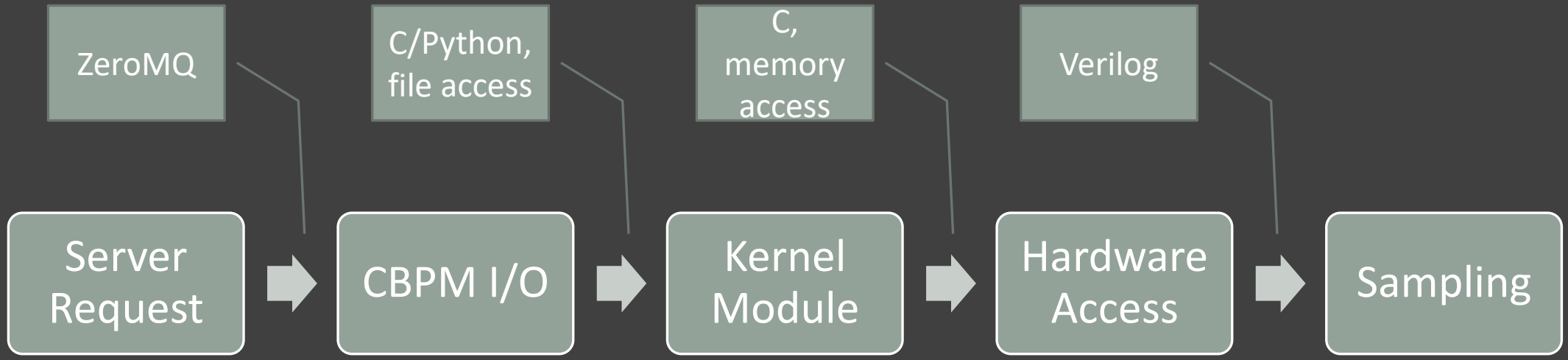
CBPM3 Status

MAY 17TH 2024

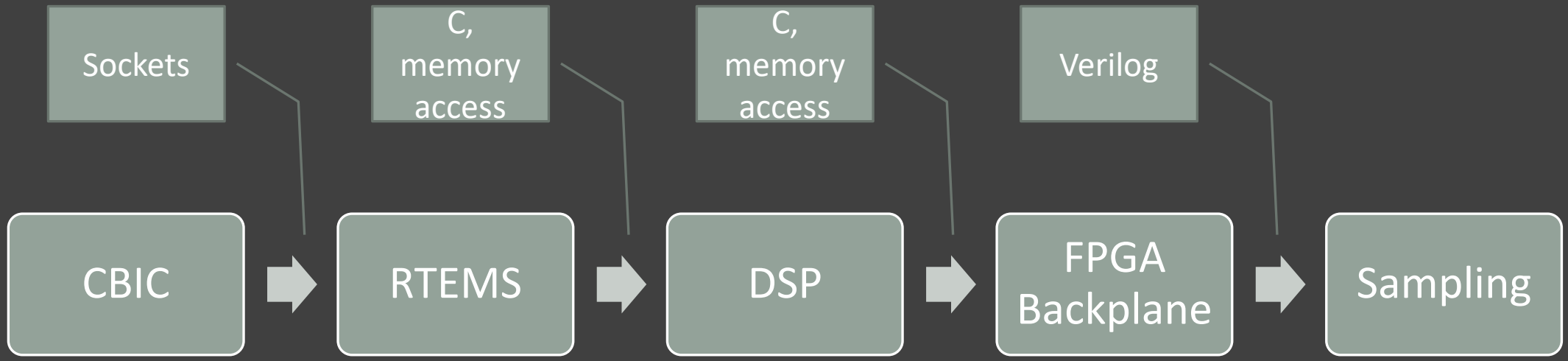
Overview

- System Level Summary
- Current Implementation
- Short Term Goals
- Long Term Goals
- Latest News

System Level Summary



CBPM2, For Comparison



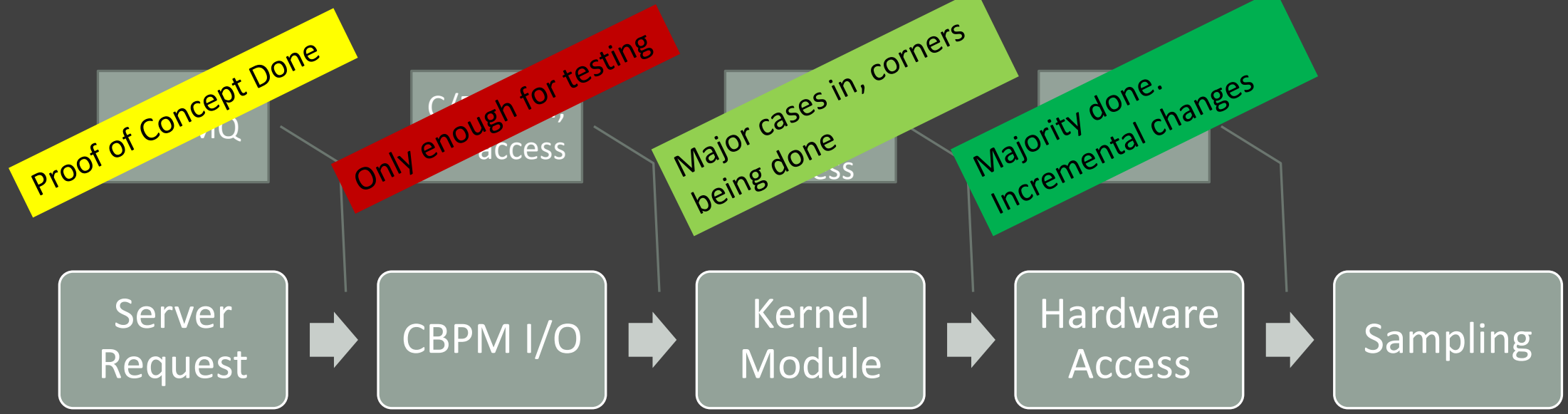
Digression – Taking Data

- User Requests data, including metadata – species, bunch, # of turns, etc.
- CBIC packs it up and sends it to the BPMs
- RTEMS handles the socket connection and hands off the metadata
- DSP sets up the parameters for sampling from the metadata
- When ready, enables sampling
- Backplane synchronizes between multiple triggers before sending an enable
- AFE's synchronize on backplane enable and turnmarker, take data per params

Digression – Taking Data

- Collects data from all bpms before formatting as needed, returns to user
- Reads DSP buffers and packs them before sending out over socket connection
- Reads out data from AFE's
- Processes if necessary
- Enable de-asserted
- Done with collection

Progress


Proof of Concept Done
Only enough for testing
Major cases in, corners being done
Majority done. Incremental changes
Server Request
CBPM I/O
Kernel Module
Hardware Access
Sampling

Current Implementation

- User process reads/writes using IOCTL and structs
- Kernel Module handles memory access
- Helper functions for debugging, logging
- Set up and trigger for data capture
- Return data when requested

Test Program - cbpmio.c

```
// Read AFE0 regs
ret_val = ioctl_get_afe0(file_desc, &afe0_regs);
if (ret_val)
    goto error;
print_afe_regs(&afe0_regs);

// Write AFE0 regs
afe0_regs.ch0_gain_control = 0x00FF;
afe0_regs.ch1_gain_control = 0x00FF;
afe0_regs.temp = 0xFFFF;
ret_val = ioctl_set_afe0(file_desc, &afe0_regs);
if (ret_val)
    goto error;
ret_val = ioctl_get_afe0(file_desc, &afe0_regs);
if (ret_val)
    goto error;
print_afe_regs(&afe0_regs);
```

Output

```
AFE Registers Dump:
Register            | Value
control             | 2
dcs_control         | 0
fpga_id             | 2f
ch0_gain_control    | 0
ch1_gain_control    | 100
adc_mem_first_1     | 0
adc_mem_first_2     | 0
turns_count_1       | 0
turns_count_2       | 0
next_mem_adr_1      | 0
next_mem_adr_2      | 0
temp                | 0
AFE Registers Dump:
Register            | Value
control             | 2
dcs_control         | 0
fpga_id             | 2f
ch0_gain_control    | 0
ch1_gain_control    | ff
adc_mem_first_1     | 0
adc_mem_first_2     | 0
turns_count_1       | 0
turns_count_2       | 0
next_mem_adr_1      | 0
next_mem_adr_2      | 0
temp                | 0
```

Bonus Points – See the problem?

Digression – I/O Options

- Currently using a quasi-object oriented approach
  - Struct defined with all registers, workflow is make-get-change-set
  - Common for mem mapped embedded stuff (timers, SPI configs, peripherals)
- Issues
  - AFE register map is a little weird
  - Timing board does not support reads so we have a virtual register for it
  - LOTS of transactions – bus is busy
- Other Options
  - Spent time trying to get arbitrary buffers of transactions working, eventually reverted
  - More custom ioctl functions for specific cases (in progress)

Short Term Goals

- Sweep mode ioctl
  - Will handle the timing steps automatically
  - Need to figure out the data handoff a little more
- Integrate generated headers for structs
  - Currently using rough structs put together for basic testing
- Get some believable data

Long Term Goals

- Prove my phase data alignment code works
- Fast sampling ioctl
  - See how quickly data can be pulled
  - Benchmark current arch, possibly plan changes to FPGA
- Get the Python interface polished
  - Ran into issues with how the buffers were packed, was easier to byte bang in C
- ZeroMQ interface, Time in code, phase measurement math, process arbitration in kernel module, self test code, error handling, logging, MAC address overlap, offline mode, channel comparisons, calibration options, status and heartbeat monitoring, FPGA interlocking, so on.

Latest News

- Odd register behavior prompting a second prototype box
- Not as problematic but not quite right
- Hardware issue? FPGA issue? Not a kernel mod issue, same result using other means
- Second prototype box mostly worked smoothly, a few oddities