**Horizontal Power Comparison of Three Architectures Across Different Work Loads**

This project consists of two aspects:

1) A custom-designed power measuring platform

2) ZephyrOS-based computation tests

Our goal is to characterize power performance of ZephyrOS based computational tasks on three different boards with different architectures. The boards are the Arduino Nano 33 BLE Sense (ARM M4), ESP32C3 (RISCV), and Raspberry Pi Pico (ARM M0+).

Custom Power Measuring Platform

The custom hardware platform uses a current sense resistor in series with the USB bus voltage that powers the platform to test. The goal is to measure currents from 100’s of nA to 100’s of mA. The current becomes a voltage with a gain of 0.1V/A that is amplified by an instrumentation amplifier with a gain of 100. The output voltage is then sampled by a 32-bit ADC at 20 SPS. A schematic of this is shown in Fig. 1.

Diagram

Description automatically generated

Fig. 1. Hardware Schematic

In order to minimize thermal noise, traces are kept short and resistances are kept small. The purpose of R2 is to provide a DC path to ground for the input of the amplifier [1]. R3 and C3 provide a low pass filter to reduce aliasing to the ADC, but the ADC has a built-in filter bank [2]. In our case, it is configured as an FIR filter.

Initial problems with the board were that the amplifier was configured with single-ended supply. A charge pump was added to produce a second negative supply. Analog functionality was verified with an LTSpice simulation with the schematic shown in Fig. 2. The complete board CAD model is shown in Fig. 3 and an image of it is shown in Fig. 4.

Diagram, schematic

Description automatically generated

Fig. 2 LTSpice Simulation Schematic

A screenshot of a computer

Description automatically generated with low confidence

Fig. 3 Complete Board CAD

A circuit board on a table

Description automatically generated with medium confidence

Fig. 4 Complete Board

The LTSpice schematic files is found in hardware/LTSpice and the Altium design files for the board are in hardware/Altium.

The next order of business is to write the Arduino code that reads the ADC samples and does the offset calculation. The base of the code is from [3], but this code had many issues that had to be rectified. The driver given is modified so that the proper channel is used, along with 20SPS sampling and an FIR filter. According to [2], this would give an effective number of bits of 24.6. At a reference voltage of 2.5V, this would mean the ADC resolution would be

The voltage that the ADC measures corresponds to the voltage produced after the current passes through the resistor and then amplified by the instrumentation amplifier:

Thus the voltage measured corresponds to ten times the current.

The calibration steps start upon startup of the Arduino code, and start with discarding the first 100 samples to reach steady state, and then using the next 100 samples to find an average value to subtract off. Upon doing so, the measurements under no load indicate values hovering around 0 and less than .

The samples are written to the serial port, which are then read with PuTTy and written into a file where they are then pasted into excel to find the average. The Arduino code is in src/Arduino.

For the current measurements, 250 samples are averaged.

ZephyrOs-based Computation Tests

There are several tests that are used to test the power consumption.

1. FIR (10 tap, 64 pts)
2. FFT (32 pt)
3. LZ4 Compression ("Lorem ipsum dolor sit amet, consectetur…”)
4. SHA256 Encryption (“abc”)
5. TFlite Sine Estimation Model
6. BLE Broadcast (For BLE devices)

The first two tests are written using code from online libraries [4, 5]. The rest of the tests are modified versions of example code found in the zephyrproject directory (zephyrproject/zephyr/samples/). The last test (broadcast) only applies to the BLE and ESP since they have radios.

Except for the broadcast test, the tests are further modified to find execution timing of the code (using a Zephyr library). 100 runs of the code that is running in a loop for the power test are timed and printed to the serial port. 5 of these for a total of 500 runs are averaged together to find an execution time per run. For the BLE and Pico, additional configuration is needed to print to the serial port. The Zephyr applications are found in src/Zephyr. There are two flavors, one is src/Zephyr/power and one is src/Zephyr/timing/nano\_pi and src/Zephyr/timing/esp.

The measurement data can be found in measurement/.

Building and Flashing

To build and flash these applications, you can use the commands written in the text files in src/. The lines above the west are done once. The BLE should be boot in bootloader mode first before flash (double press the RST button), and the Pico should be connected via USB while holding the RST button.

References

[1] ads8221.pdf

[2] ads1262.pdf

[3] <https://github.com/Protocentral/ProtoCentral_ads1262>

[4] https://lloydrochester.com/post/c/convolution/

[5] https://rosettacode.org/wiki/Fast\_Fourier\_transform#C++