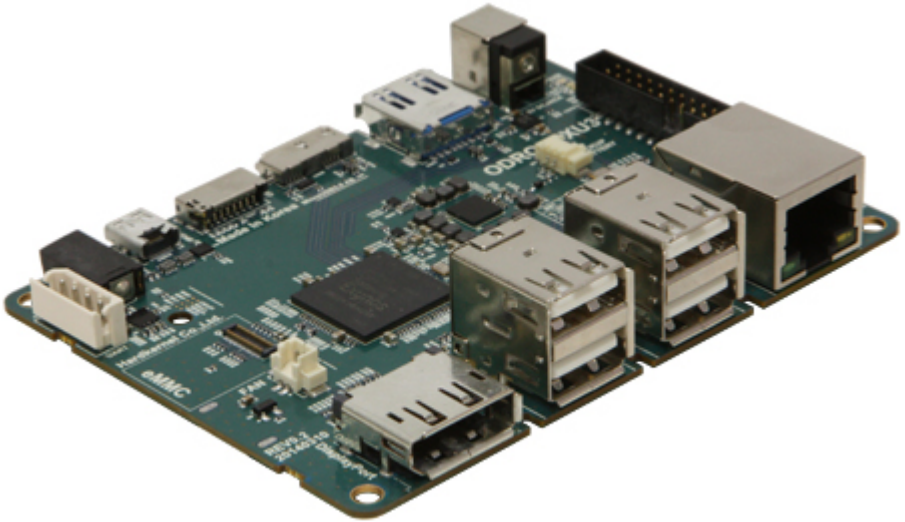
Benchmarking ARM big.LITTLE

CPE-534
Operating Systems
Spring 2020
William Pannell

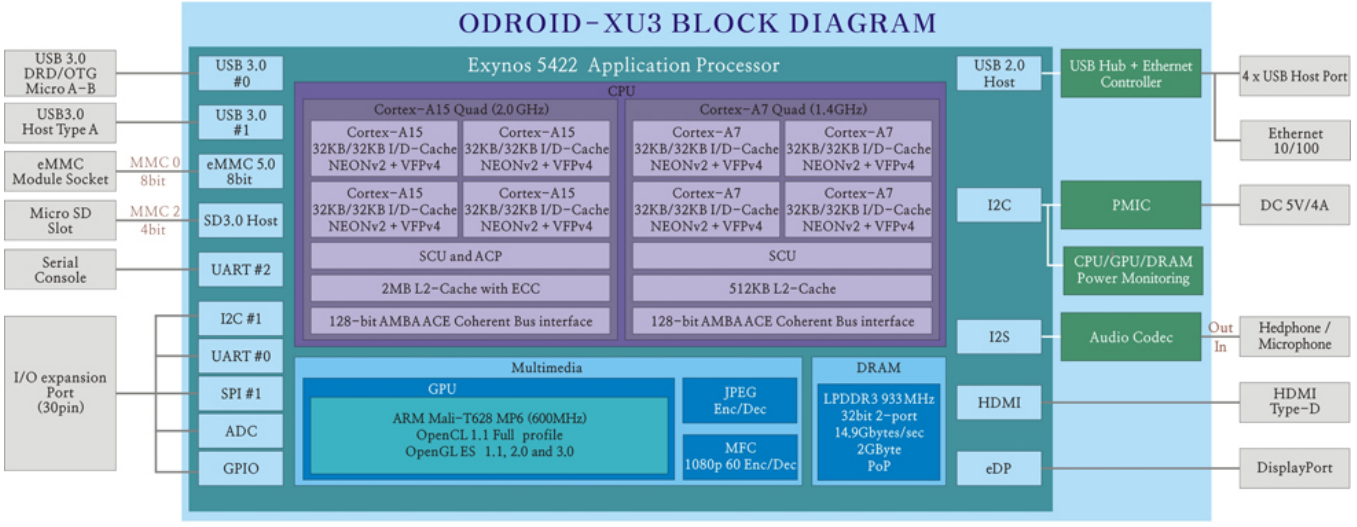
Unit Under Test

- The Odroid-XU3
  - Samsung Exynos 5422
    - 8 core big.LITTLE System on Chip
    - 4x ARM Cortex-A15 (2GHz)
    - 4x ARM Cortex-A7 (1.3GHz)
  - 2GB LPDDR3 (933MHz)
  - micro-HDMI (Mali T-628 MP6 iGPU)
  - 10/100 Mbps Ethernet
  - 4X USB 2.0, 1X USB 3.0
  - 30pin GPIO



Specs and images sourced from wiki.odroid.com

ARM big.LITTLE Architecture



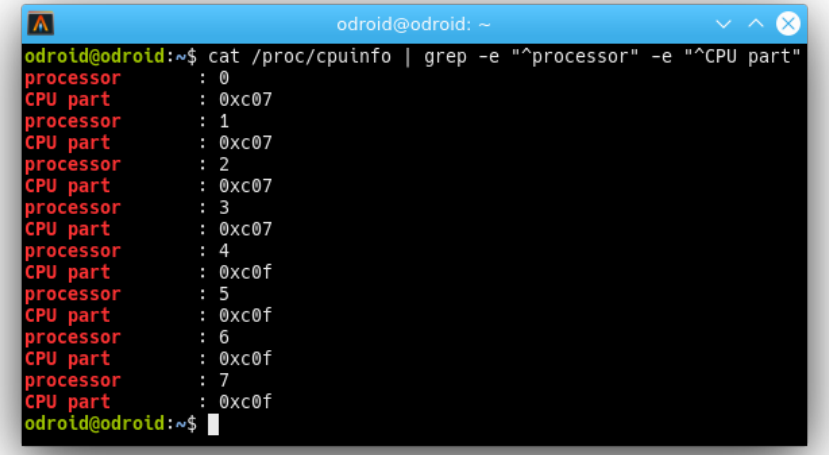
Block Diagram sourced from wiki.odroid.com

Highlights:

- “big” A15 Cores:
  - 32KB I/D cache per-core
  - Cores share 2MB L2 cache
- “LITTLE” A7 Cores
  - 32KB I/D cache per-core
  - Cores Share 512KB L2 Cache
- DRAM, GPU, and IO peripherals connected via 128-bit bus.

Identifying big.LITTLE cores

- Use /proc/cpuinfo to get information about the CPU
  - “processor” gives core enumeration
  - “CPU part” Identifies the core type
    - 0xC07 → Arm A7 → LITTLE Core
    - 0xC0F → Arm A15 → big Core
  - Cores 0-3 are LITTLE, 4-7 big



Benchmarking with 7ZIP

- 7Zip has builtin Benchmarking
- 7z b [N] [-mmt{N}] [-md{N}]
  - N runs
  - -mmt{N} Benchmark using N threads
  - -md{N} Benchmark with 2^N byte sample dict
    - -md22 yields 4MB test file
    - Tests start at 4MB and climb to -md{N} setting
    - 64MB test file crashes on this machine
- Each run consists of 1 Compression and 1 Decompression

Summarized from 7zip manual
https://sevenzip.osdn.jp/chm/cmdline/commands/bench.htm

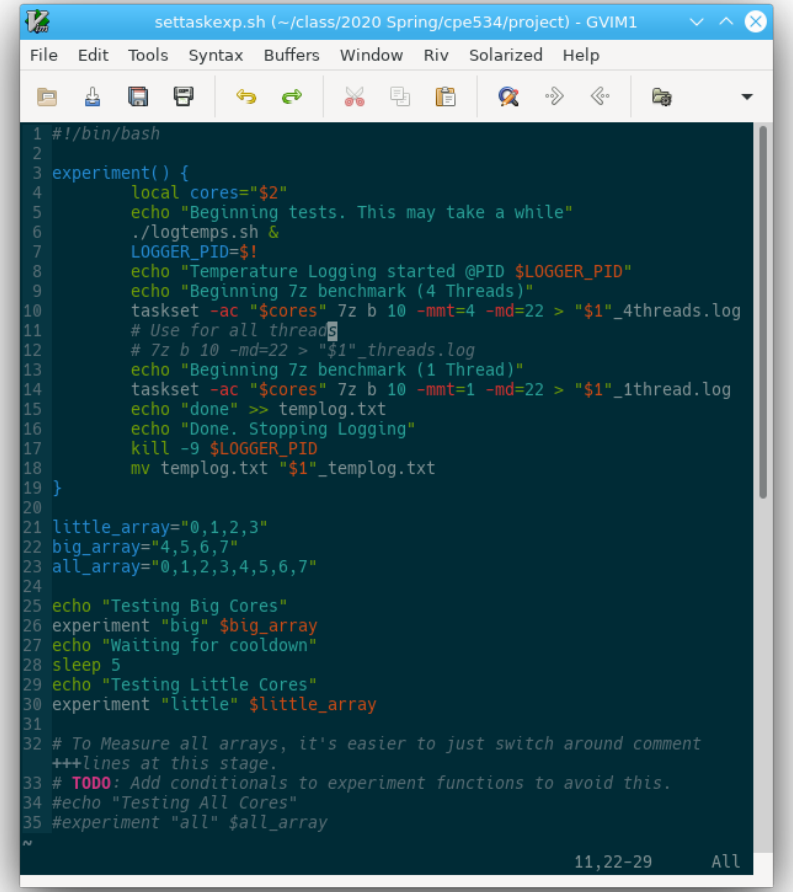
Benchmarking with 7ZIP

- Compression
  - Speed Depends upon:
    - RAM Latency
    - Data Cache Size/Speed (As opposed to Instruction Cache)
    - TLB hit/miss
  - Large Number of Random Accesses to RAM/Cache Memory
- Decompression
  - Speed Depends Upon:
    - 32bit integer operation latency (shift/mult/add)
    - Branch (Mis)Prediction Penalty
    - Conditional Execution
  - No Special Extension Instructions (SSE,NEON,etc.)

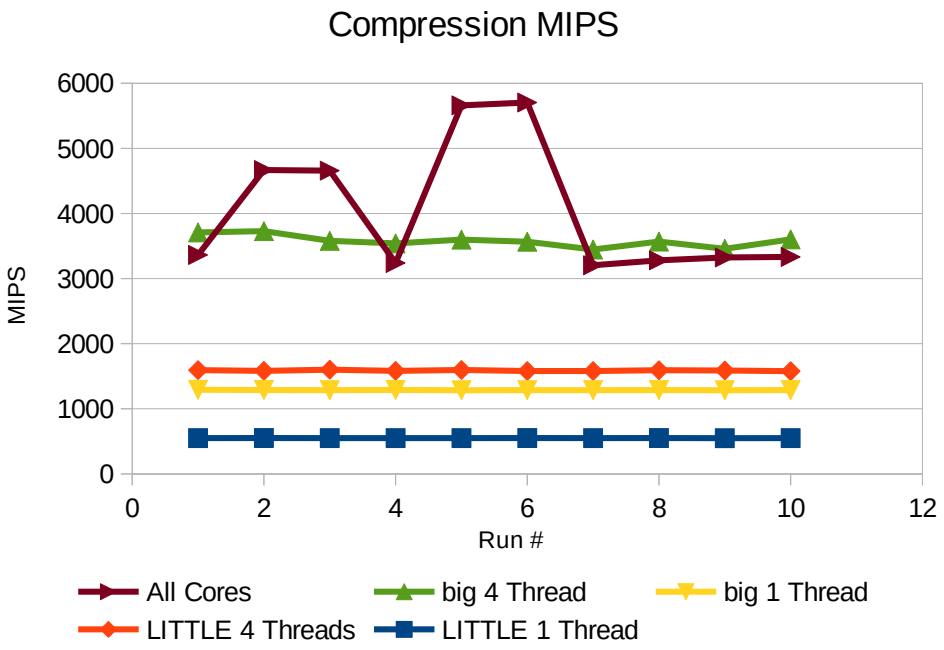
Summarized from 7zip manual
https://sevenzip.osdn.jp/chm/cmdline/commands/bench.htm

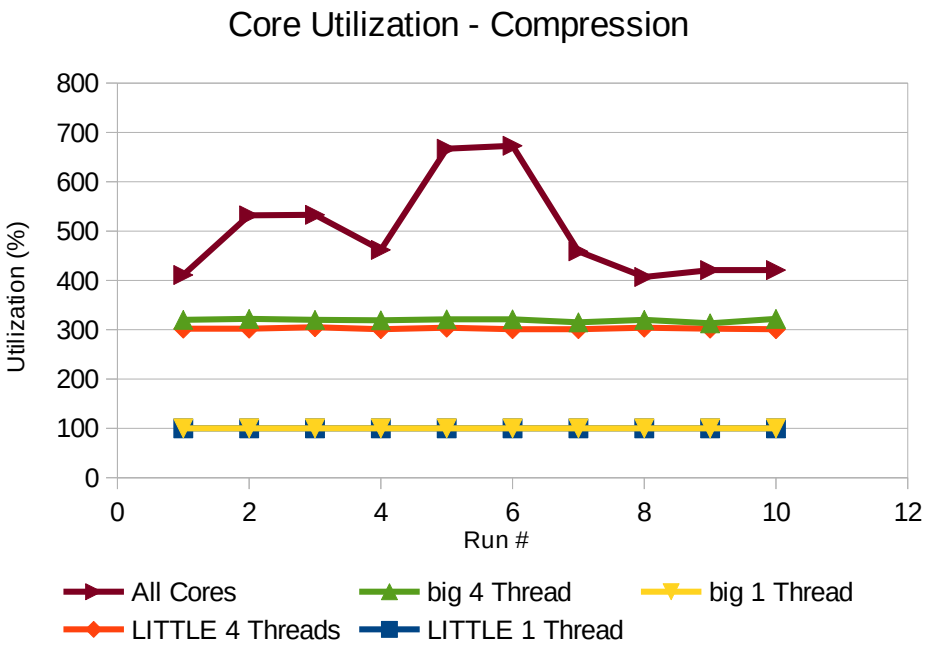
Methodology

- Use taskset to pin 7z to cores
- Perform 10 runs each using:
  - 4 big cores
  - 1 big core
  - 4 LITTLE cores
  - 1 LITTLE core
  - All cores
- Log temperature during tests



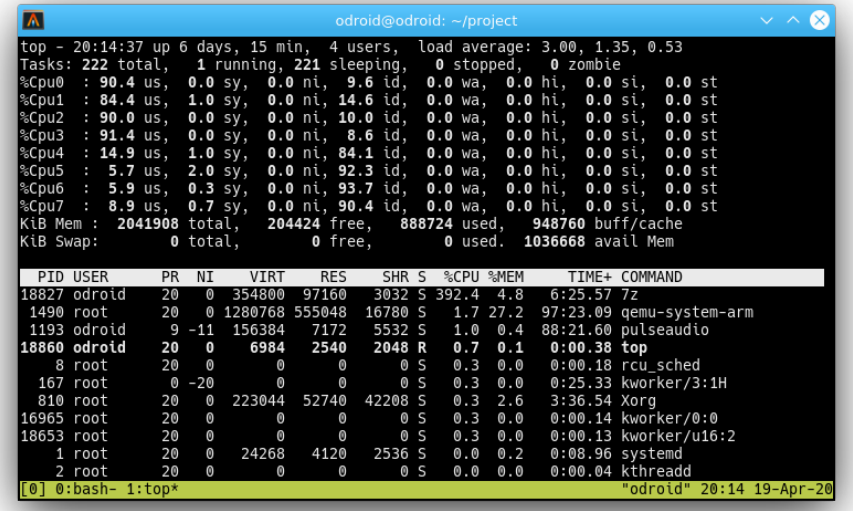
Compression Performance



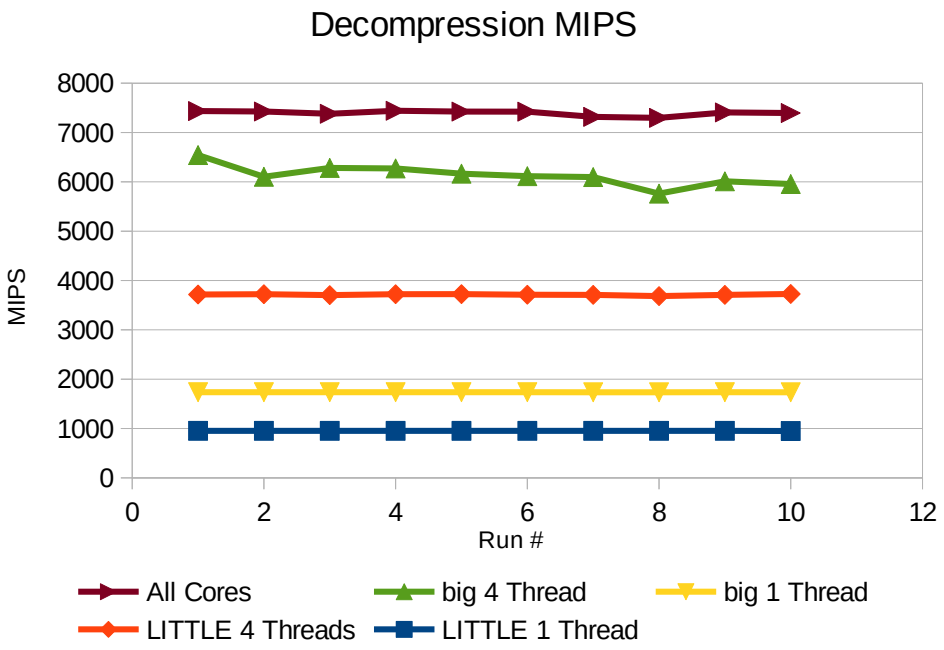


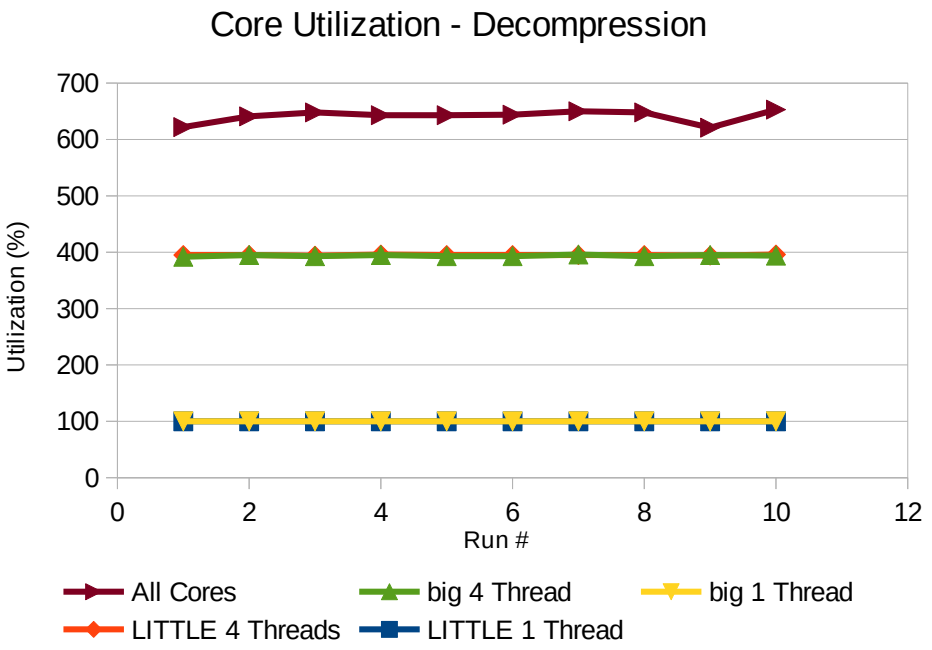
Compression – Core Utilization

- Screen shot of Top Running during All Cores test
- High Utilization on LITTLE cores, Low utilization of big cores
- High idle time on big cores
- Background processes not to blame



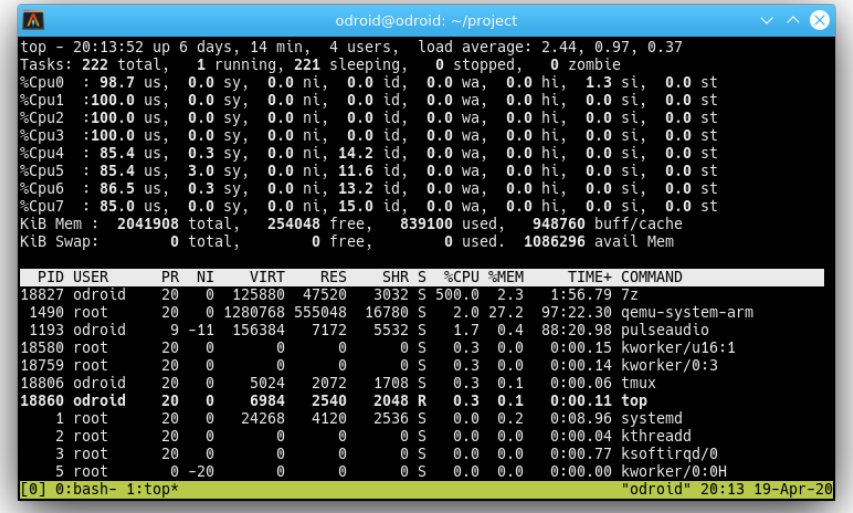
Decompression Performance





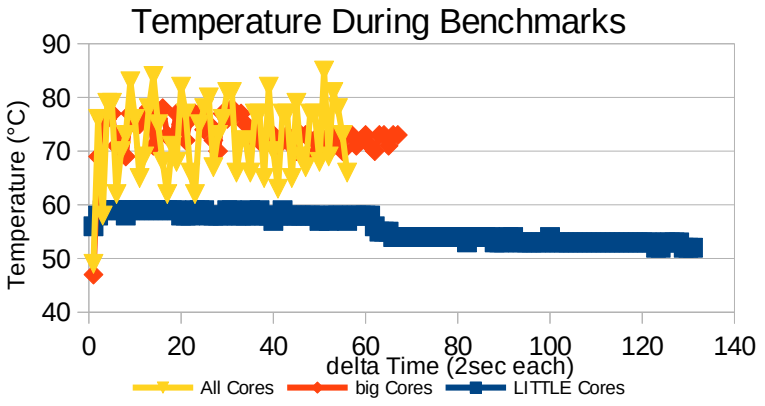
Decompression - Core Utilization

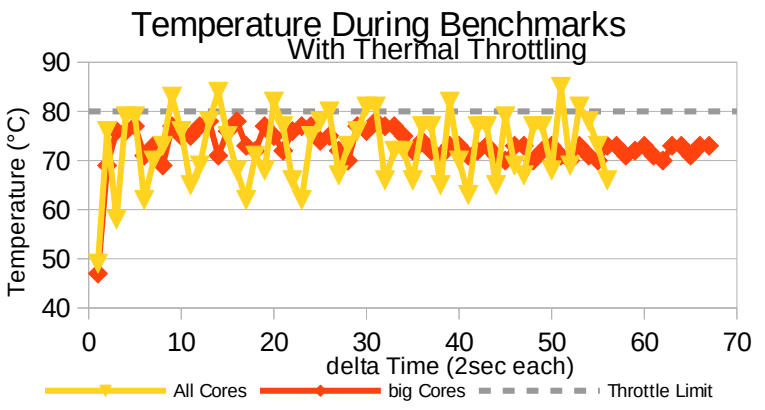
- Screen shot of Top Running during All Cores test
- Near 100% Utilization on all LITTLE cores
- 85% utilization of big cores
  - Averaging 13.5% idle time on each big core
- Background processes not to blame



Thermal Performance

- Temperature logged during both 1 and 4 threaded tests
  - 4 Threads First
  - 1 Thread test begins at Dip
- Evidence of Throttling During “All Threads” and “big cores 4 threads” runs
- No evidence of Throttling During LITTLE cores runs.





Performance Summary

- big ~2X faster than LITTLE
- Cache constraints hurts LITTLE Compression performance
- 4 Thread Compression Experiences Bottlenecks
- LITTLE punches above weight in integer-math-heavy workloads (decompression)
- 8 Core performance falls short of expectations due to thermal constraints.

Relative MIPS	Compression	Decompression
Big 4 Thread	100.0%	100.0%
Big 1 thread	35.9%	28.3%
LITTLE 4 thread	44.3%	60.6%
LITTLE 1 Thread	15.4%	15.5%
All cores	112.9%	120.6%

Further Experimentation

- Improve Datalogging
  - Monitor core frequencies to detect throttling
  - Instrument DRAM/Cache usage (perf in linux-tools)
  - Instrument Power Consumption
  - Log each run individually
  - Allow quantifying performance differences run-to-run
- Split Compression/Decompression phases into discrete tests
  - Provides better granularity of logging.
  - No built-in support, but 7z is open source
- Reduce Variables
  - Kill other processes (X11, PulseAudio, Virtualized Guests, etc.)
  - Migrate background tasks to other cores before starting tests
- Goals
  - Verify assumed bottlenecks
  - Identify cause of low utilization during 4/8 threaded compression tests.
  - Empower reasonable decisions (when to use big vs Little), in common workloads
  - Optimize All-Core Performance vs Power consumption

Acknowledgments

- UAH Faculty and Staff
  - Providing Odroid cluster for experiments
- Dr. Kulick
  - Providing Topic/Encouragement

Raw data, Formatted Data, Test Scripts, and Presentation can be found at:
https://github.com/wcpannell/armbiglittle