Adept

addressing energy in parallel technologies

Benchmarking for Power Consumption Monitoring

Michèle Weiland & Nick Johnson

The Adept Project, EPCC

Nick.Johnson@ed.ac.uk

Agenda

- Adept Project
- Motivation – system characterization
- Description of Benchmarks
- Description of the ODROID platform
- Results
  - AXPY
  - Dot-Product
  - Scaling Ratio
- Insights, conclusions + future work

The Adept Project

- EU Funded
- 3 years duration
  - Oct 2013 – Oct 2016
- 5 Partners from academia & industry plus an IAB.
  - EPCC, Uppsala, Ghent, Ericsson, Alpha-Data
  - Michèle Weiland from EPCC is the Project Co-ordinator.

@adept_project

www.adept-project.eu

- The design space for modern and emerging systems is large and certainly heterogeneous.
- Users will have a large choice in the architecture they use, perhaps even with a single system.
- One key goal of the Adept Project is to influence architecture selection by providing energy usage and performance predictions for a range of different parallel platforms using a modelling tool.
  - Benchmarks are a key part and pre-cursor to this.
  - Essential to extract information about different architectures to build accurate models.
  - This includes all system components, characterizing the CPU is just one *small* part.

Systems Characterization

- The focus of the benchmarks is to characterize a system in as fine a detail as possible.
- This differs from the usual HPC benchmarking process whereby you try to coax as much performance from a given code as you can by tuning the system to the benchmark (or *vice versa*).
- Existing benchmarks and tools focus too narrowly on specific sub-systems **or** types of system
    - For example, HPL will tell you nothing about Disk I/O performance.
    - IOPerf tells you nothing about power consumption.
    - GPUbench doesn't work on FPGAs

- We want to see the effect of system-level (turbo states, HT, process placement), toolchain-level (compiler options) and application-level (data decomposition, programming model) choices on **both** the run-time **and** the energy consumption.
- We also need to include examples which map to common use cases of all scales of hardware, from SoCs to supercomputers.

The Adept Benchmarks

- The benchmarks are split into 3 levels:
- Micro-benchmarks – single purpose codes which exercise a single (or as close to single as possible) instruction.
  - For example, arithmetic addition or division, PCIe data transfer or IPC communication.
- Kernel-benchmarks – comprised from a low number of micro-benchmark operations which represent common kernels seen in real codes.
  - For example, FFTs, String searches or BLAS routines.
- Toy-applications – comprised from a low number of kernel-benchmark operations and residing somewhere between a single kernel and a complete case-study.
  - For example, a solver (including setup, tear-down, data generation and pre/post-processing) from the CP2K package.

Hardware

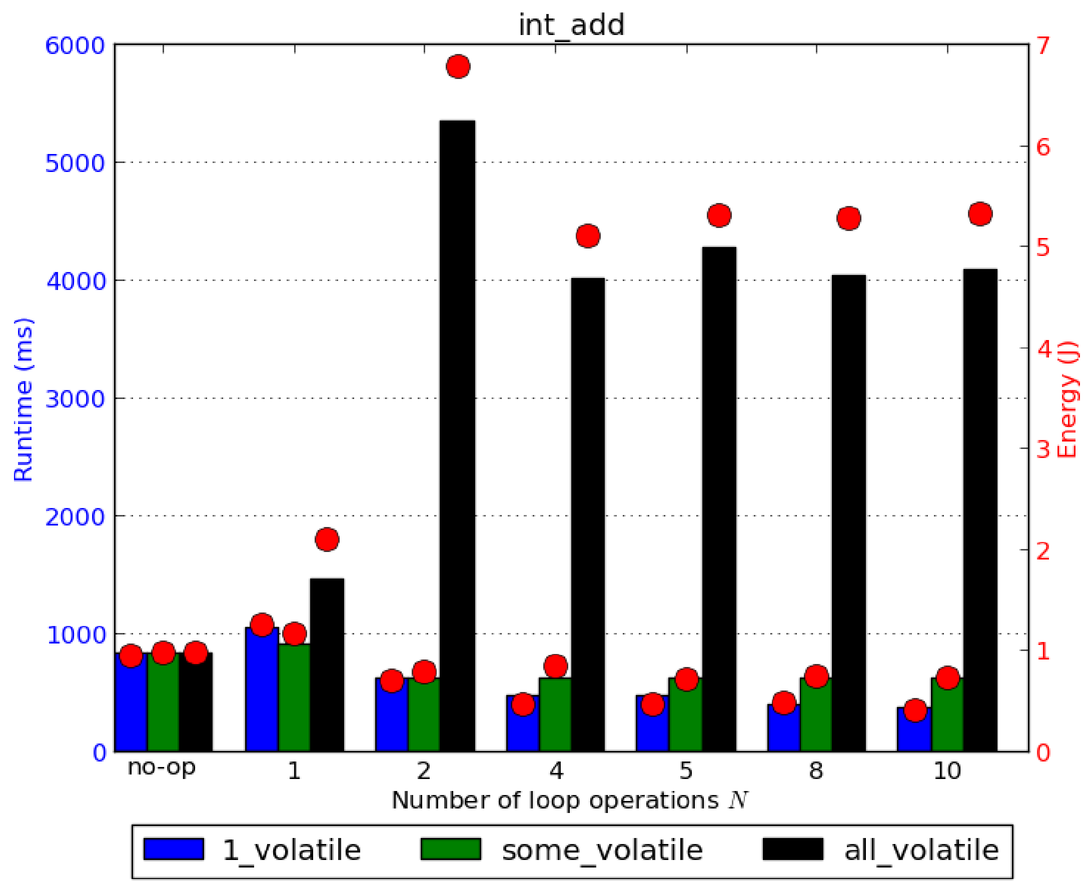
- All the following results are from an ODROID XU+E
- SoC based on Samsung Exynos5 Octa A15 + A7 CPUs
- Includes a PowerVR GPU (OpenGL + CL compatible)
- 2GBytes of RAM
- Integrated power sensors for Memory, CPUs, GPU along with per core temperature measurement.
- Allows pinning of jobs to CPUs

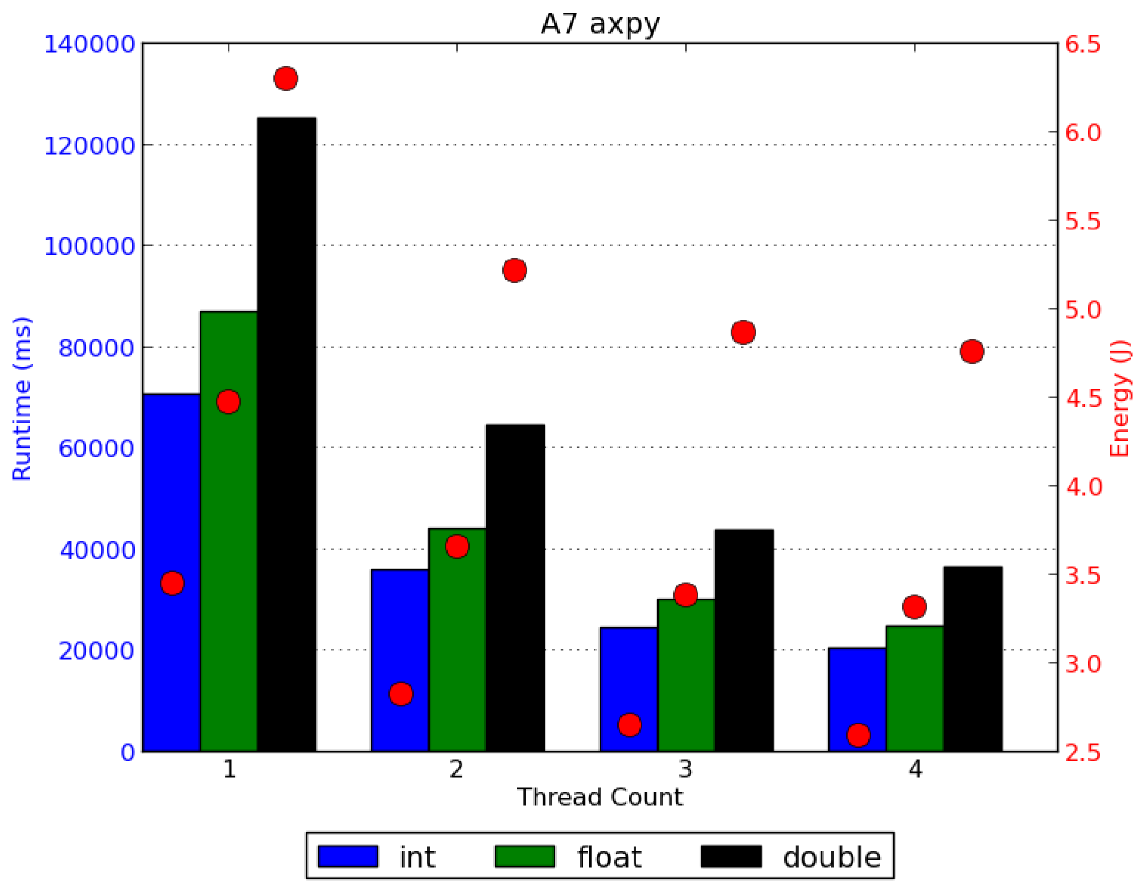

FAN
DisplayPort

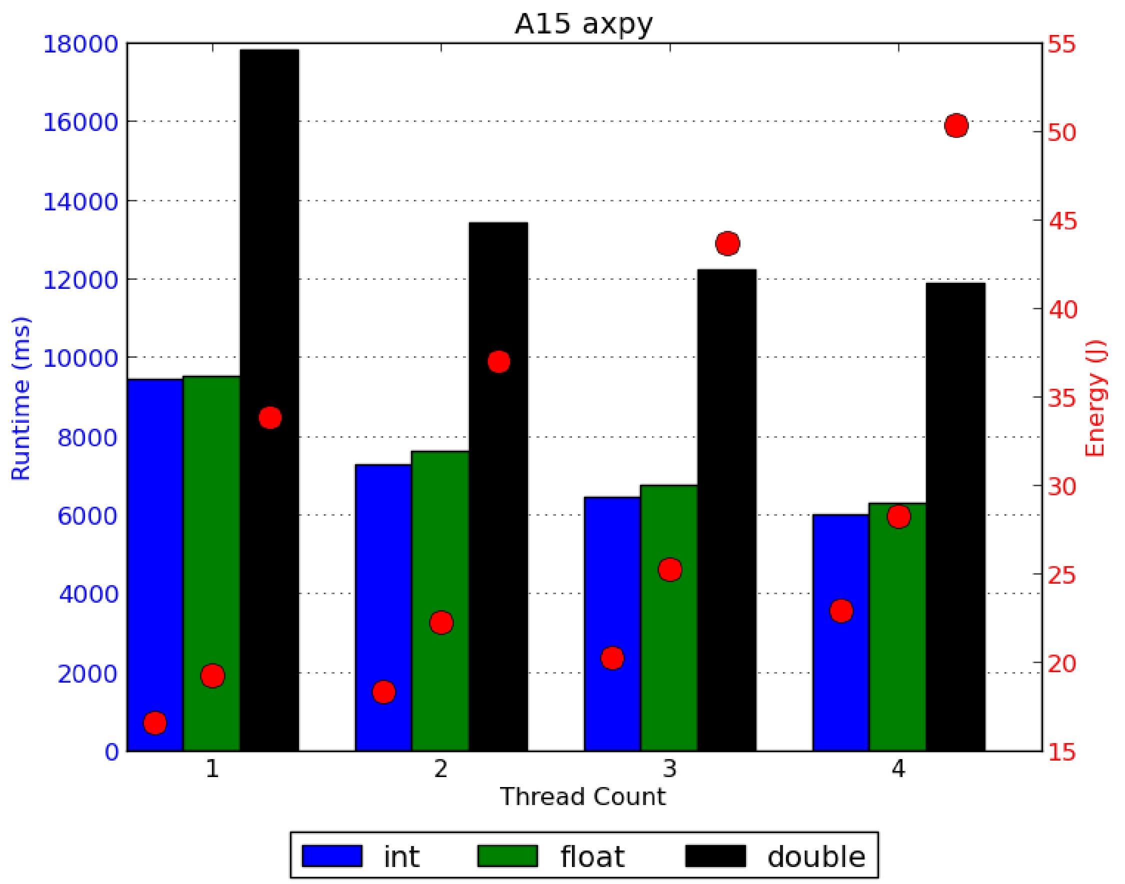
- Sensors are a TI INA231
- Both voltage and current are measured then multiplied (in the sensor) to give power readings.
- Presented to the application via files in the linux /proc filesystem
- Update frequency is ~5Hz

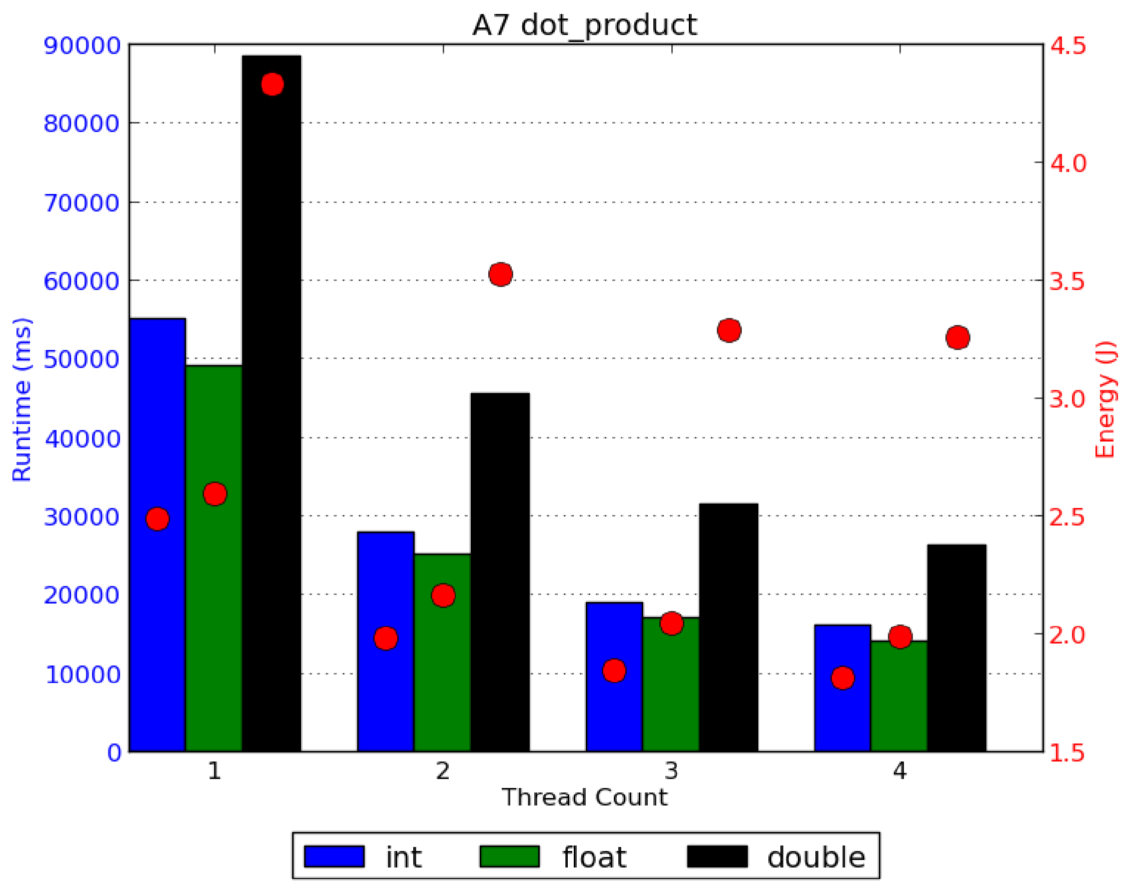
Results

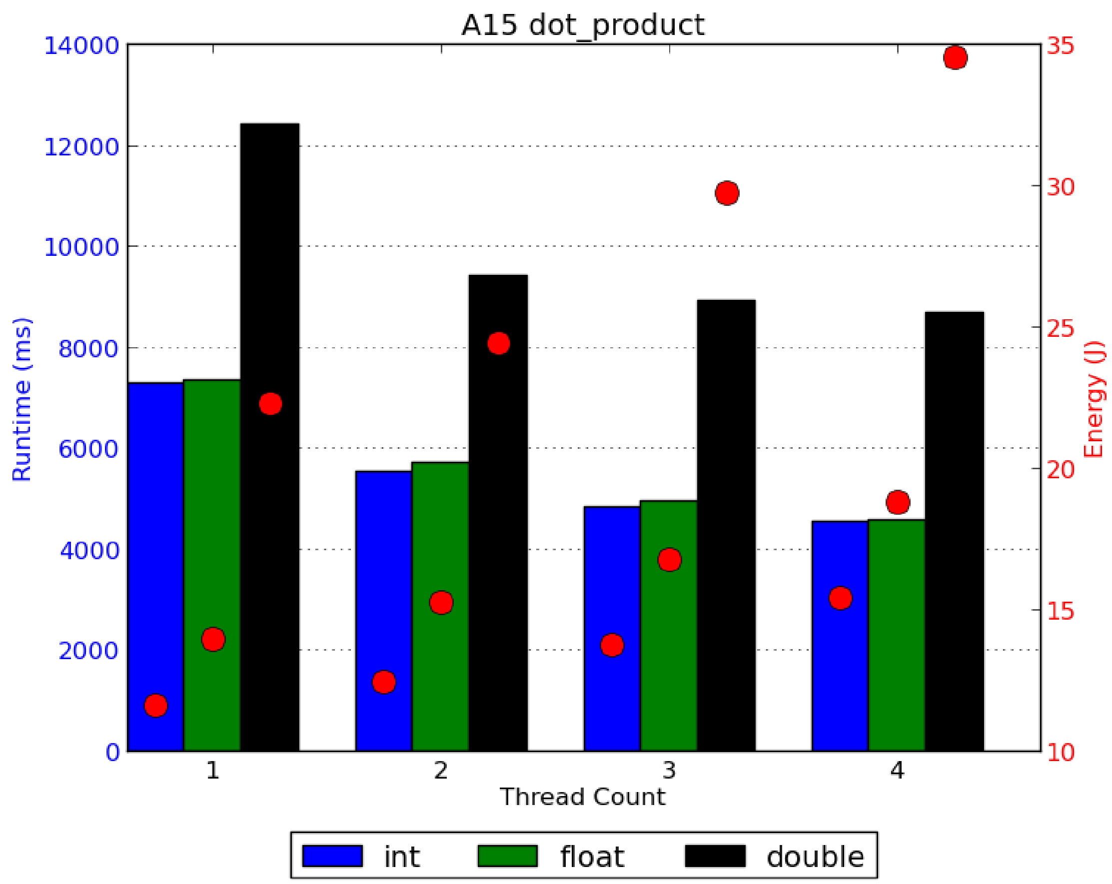
- All results are generated on the ODROID.
- The CPU is fixed to either A15 or A7 prior to running the benchmark.
- Aside from the benchmark (and OS), the only other code running is the sampling script.
- Each benchmark is run 10 times. The run with **minimum** runtime is used along with the power data for **that** run.
- In this system, `int` and `float` are 4 bytes, `double` is 8.


int_add
Runtime (ms)
Energy (J)
0
1000
2000
3000
4000
5000
6000
0
1
2
3
4
5
6
7
no-op
1
2
4
5
8
10
Number of loop operations N
1_volatile
some_volatile
all_volatile


A7 axpy
Runtime (ms)
140000
120000
100000
80000
60000
40000
20000
0
Energy (J)
6.5
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
1
2
3
4
Thread Count
int
float
double


A15 axpy
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
Runtime (ms)
55
50
45
40
35
30
25
20
15
Energy (J)
1
2
3
4
Thread Count
int
float
double


A7 dot_product
Runtime (ms)
90000
80000
70000
60000
50000
40000
30000
20000
10000
0
Energy (J)
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1
2
3
4
Thread Count
int
float
double

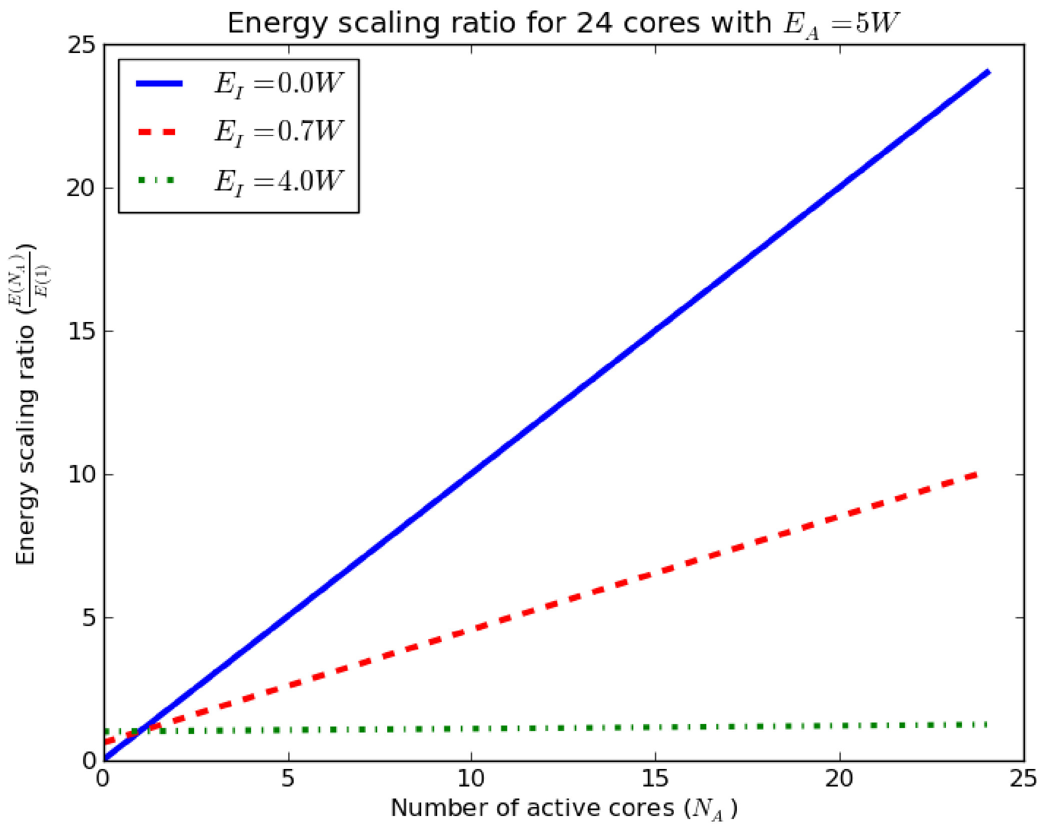

A15 dot_product
Runtime (ms)
14000
12000
10000
8000
6000
4000
2000
0
Energy (J)
35
30
25
20
15
10
1
2
3
4
Thread Count
int
float
double

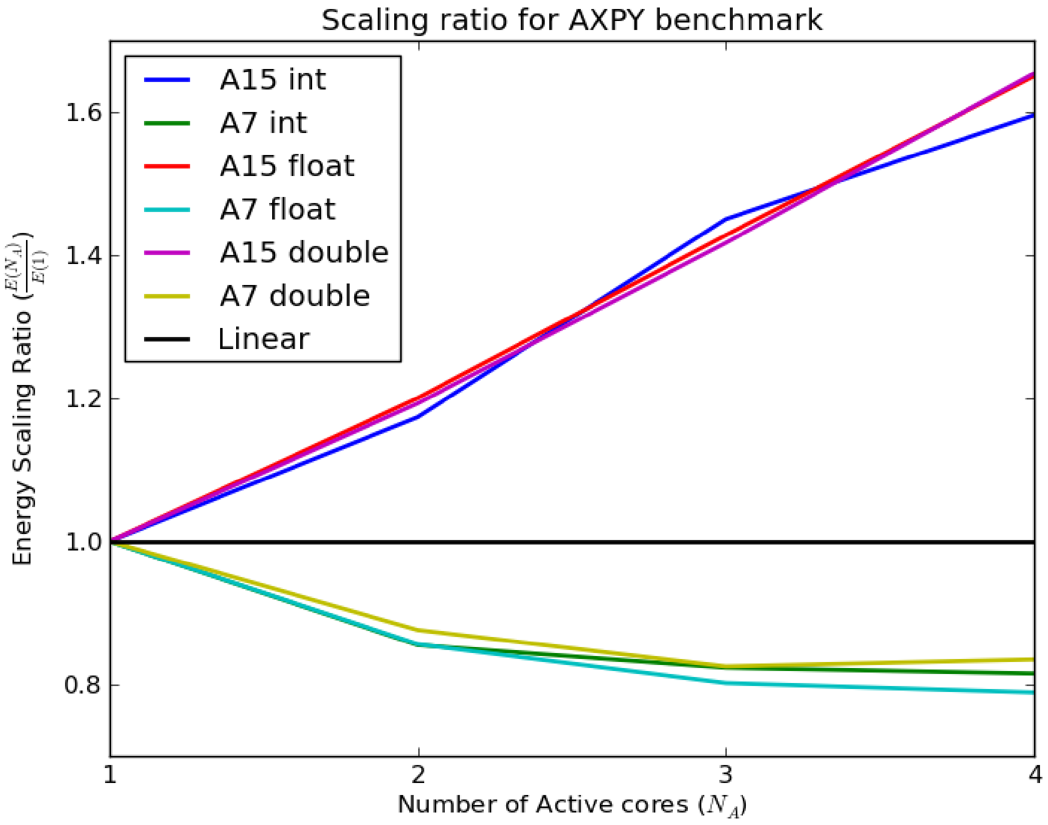
- It is obvious from the graphs that there is something quite different about the behaviours of the two processors.
- Naively we would expect overall energy consumption to drop as runtime drops.
- It is also useful the quantify in some way the power-performance of the processor as the active (used) thread count increases.

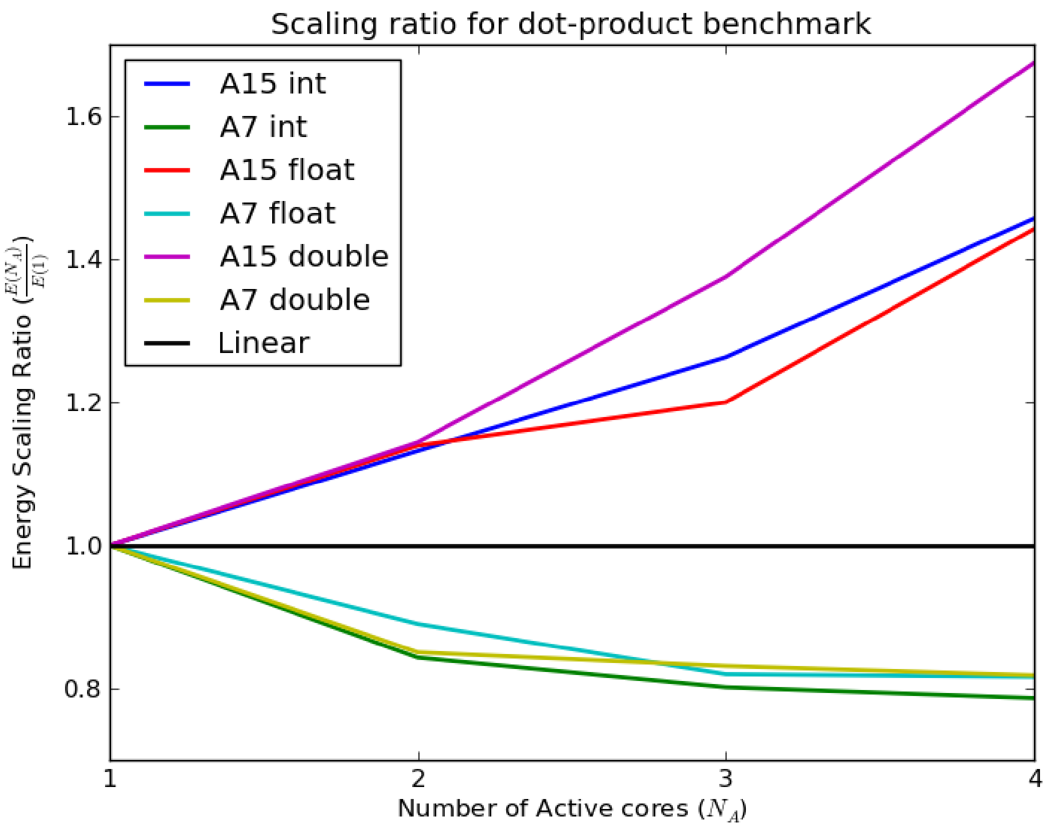
$$E(1) = E_A \times 1 + E_I \times (N_T - 1)$$

$$E(N_A) = E_A \times N_A + E_I \times (N_T - N_A)$$

$$\frac{E(N_A)}{E(1)} = \frac{E_A \times N_A + E_I \times (N_T - N_A)}{E_A + E_I \times (N_T - 1)}$$


Energy scaling ratio for 24 cores with $E_A = 5W$
$E_I = 0.0W$
$E_I = 0.7W$
$E_I = 4.0W$
Energy scaling ratio ($\frac{E(N_A)}{E(1)}$)
Number of active cores (N_A)
0
5
10
15
20
25


Scaling ratio for AXPY benchmark
A15 int
A7 int
A15 float
A7 float
A15 double
A7 double
Linear
Energy Scaling Ratio ($\frac{E(N_A)}{E(1)}$)
1.6
1.4
1.2
1.0
0.8
1
2
3
4
Number of Active cores (N_A)


Scaling ratio for dot-product benchmark
A15 int
A7 int
A15 float
A7 float
A15 double
A7 double
Linear
Energy Scaling Ratio ($\frac{E(N_A)}{E(1)}$)
1.6
1.4
1.2
1.0
0.8
1
2
3
4
Number of Active cores (N_A)

Scaling ratio - insights

- What insight does this give us?
  - The A7 is most (energy) efficient running with all cores active.
  - The A15 is the opposite. It is less efficient as more cores are active. The time to solution drops, but there is a penalty to pay in terms of extra energy required.
- Why should this be the case?
  - The difference between **Active** and **Idle** power for the two processors is different.
  - There is little difference in power consumption between Active and Idle state for the A7, therefore not using the extra cores does not save energy.
  - The opposite is true of the A15, the Active-Idle difference is large. Using fewer cores consumes less energy.
- Knowledge of these effects is important to build accurate models of this system.

Future Work

- This is, quite clearly, a small part of the puzzle.
- We will adapt the scaling metric to cover other parts of the system, GPU, Memory, Disk, NIC etc.
- We will investigate systems larger than SoCs
  - We have a measurement solution for x86 but no results to share as yet.
  - We will also consider heterogeneous systems where computation is split between, say, GPU & CPU.
- We will consider complete applications.
  - For example, CP2K or LUDWIG.

Adept

addressing energy in parallel technologies