Contents

References

[1] The Sniper Multi-Core Simulator

[2] O. Tange (2011): GNU Parallel - The Command-Line Power Tool

[3] Mark, Robert, and Bharat's office hours

[4] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh and A. Gupta, The SPLASH-2 Programs: Characterizaion and Methodological Considerations, Proceedings 22nd Annual International Symposium on Computer Architecture, Santa Margherita Ligure, Italy, 1995, pp. 24-36

[5] John L. Hennessy and David A. Patterson. 2017. Computer Architecture, Sixth Edition: A Quantitative Approach (6th. ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

1 Intro

The purpose of this experiement is to sweep different test applications (benchmarks) across different configurations of L2 cache sizes, in order to learn the `SniperSim` tool. We analyze the affects of L2 cache sizes on instructions per clock cycle (IPC) and energy consumption. It is observed that as L2 cache size increases, total energy consumption of a test application also increases slightly, but the L2 energy consumption increases more noticeably. L2 cache size also affects performance (measured by IPC here) differently depending on how much time a process spends on certain types of instructions; L2 cache size is likely to affect IPC if a benchmark substantially relied on memory access operations.

2 Experimental Setup

Nine simulations run for an x86 architecture simulator, Sniper 7.3 [1]. They are simulated with three `splash2` benchmarks: `fft_O2`, `ocean.cont`, and `radix`[4]. Each were configured with the same L1 cache size (32 KB) and configuration values set in the `gainestown.cfg`. Differing L2 cache size topologies (configurations) were tested with the three test applications: 512 KB, 1 MB, and 4 MB (See Fig. 9 of Lab 0 Specification). Input size used for all tests was preset `small`. Figure 1 visualizes the topology for simulations using 512 KB L2 cache; the configuration is the same across the other simulations, only differing in what is labeled for L2 size. To note: L2 cache associativity is 8 and cache block size is 64 bytes across all simulations, given by `gainestown`.

Simulations ran concurrently using bash script(s) and GNU `parallel` shell tool[2], and post processing of the data were handled with python (v2.7) and bash scripts (included separately). Simulations ran on a python virtual environment and in a detached `tmux` session, due to long duration of the experiments. Sniper provided data processing tools used were: `gen_topology.py`, `cpi-stack.py`, and `mcpat.py`.

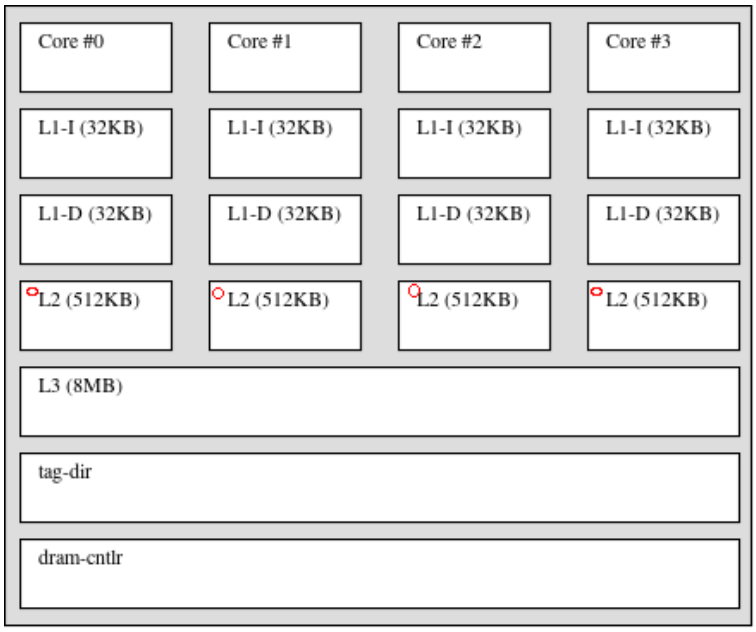
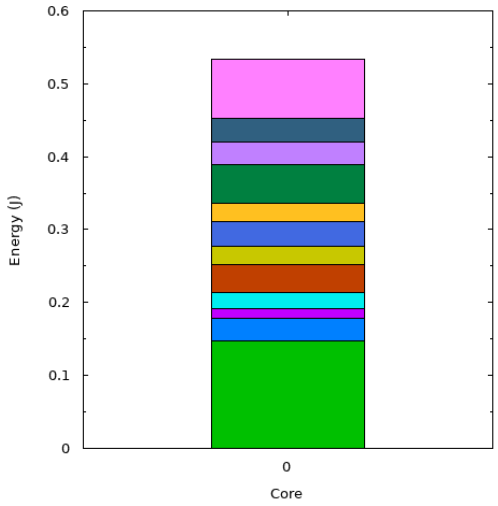


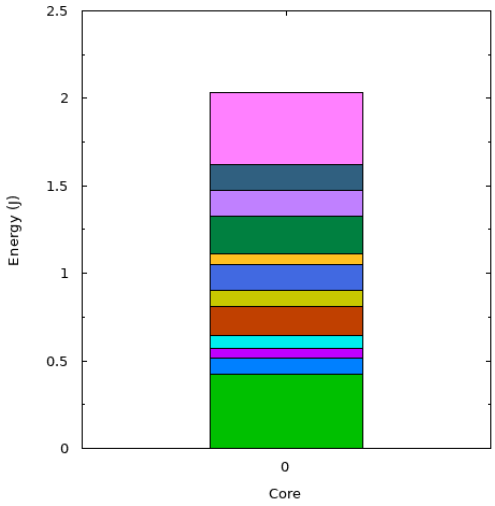
Figure 1: Topology for `fft_O2`, `ocean.cont`, and `radix` benchmark tests with 512 KB L2 cache size (topologies for the 1 MB and 4 MB L2 (marked red dot) tests are the same). All benchmarks were run in `Sniper-7.3` with the `gainestown` configuration using the `--viz` and `--roi` options.

3 Energy Results

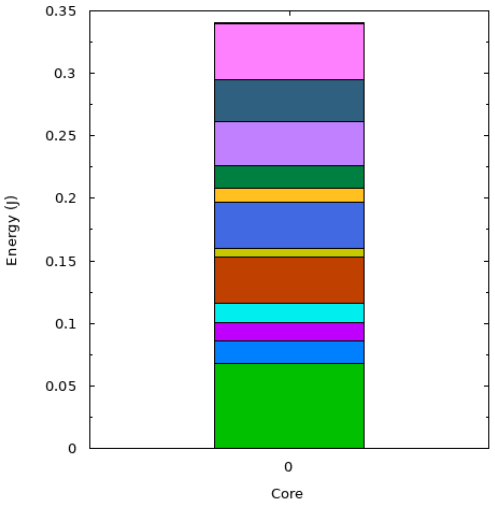
The following discusses the energy and cpi stacks for common L2 cache sized benchmarks. For post processed data, see Fig. 7, Fig. 11, and Fig. 16 for the energy stacks of core 0 of the benchmarks `fft_02`, `ocean.cont`, and `radix`, respectively, and Fig. 9, Fig. 13, and Fig. 18 for the corresponding CPI stacks. Specific values are also in the Appendix for each stack. We analyze the pattern in the following stacks (color legend at the end):



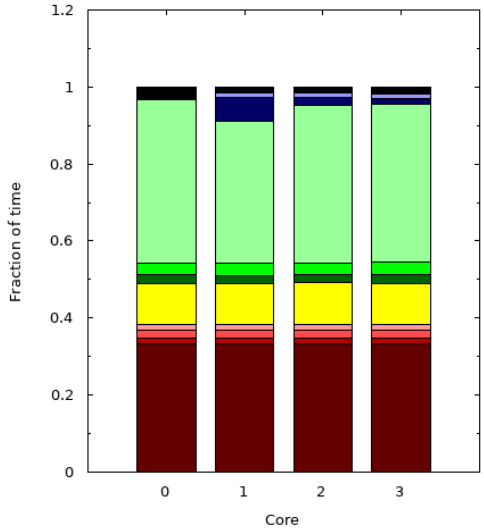
(a) `fft_02`: L2 enery 0.03 J; Total energy 0.53 J



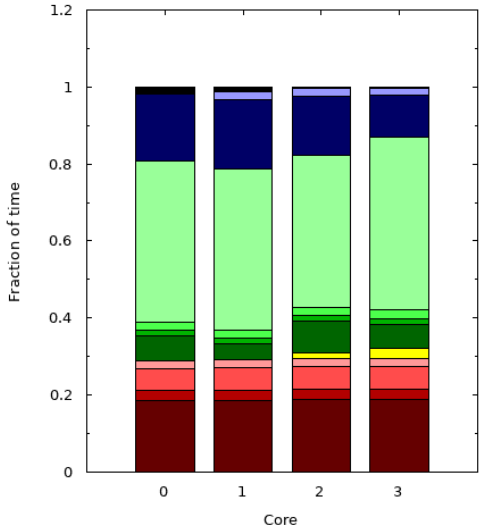
(b) `ocean.cont`: L2 energy 0.15 J; Total energy 2.03 J



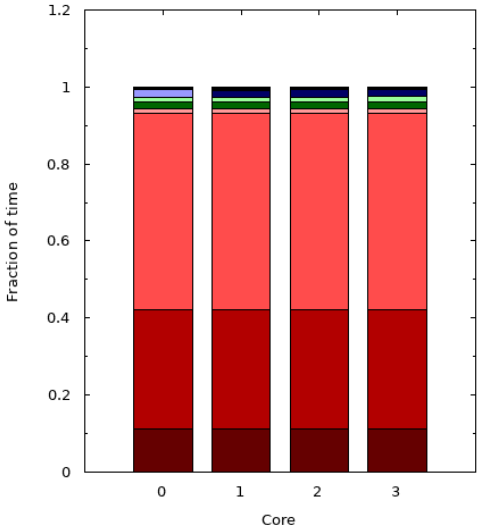
(c) `radix`: L2 energy 0.04 J; Total energy 0.34 J



(d) `fft_02`

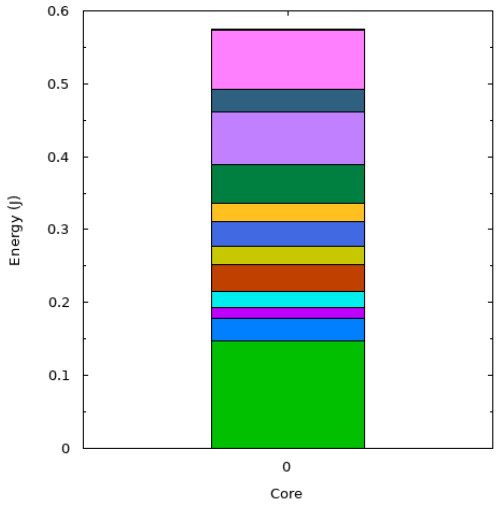


(e) `ocean.cont`

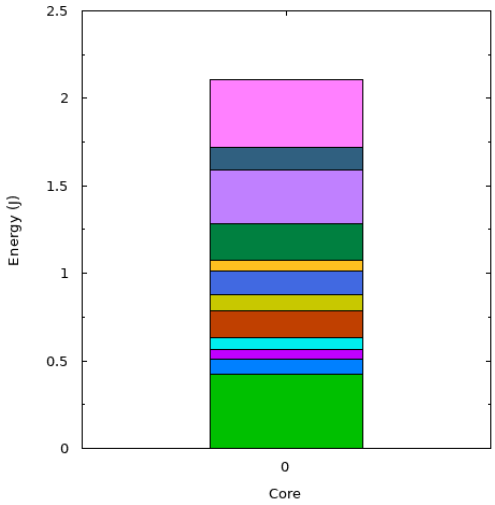


(f) `radix`

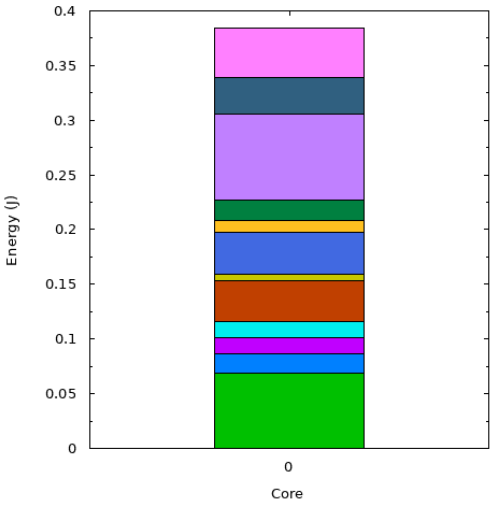
Figure 2: **512 KB L2 Cache Size:** Energy stacks for (a,d) `fft_02`, (b,e) `ocean.cont`, and (c.f) `radix` benchmarks using **512 KB** L2 cache size on core 0, and their corresponding CPI stacks.



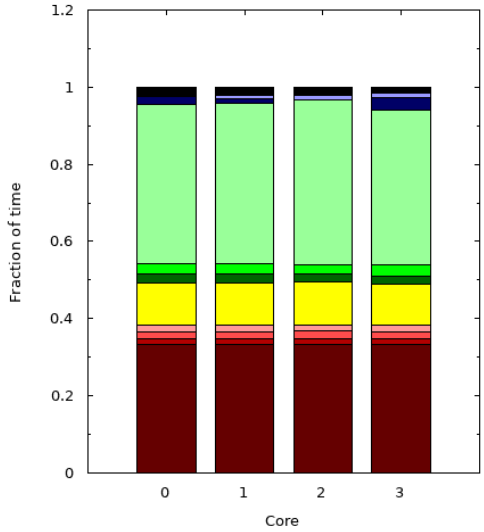
(a) `fft_02`: L2 energy 0.07 J; Total energy 0.57 J



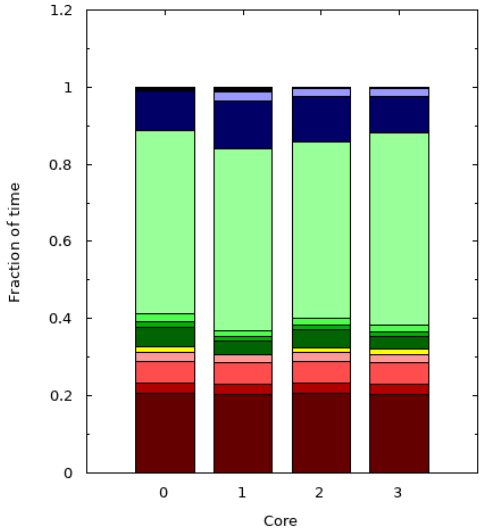
(b) `ocean.cont`: L2 energy 0.30 J; Total energy 2.11 J



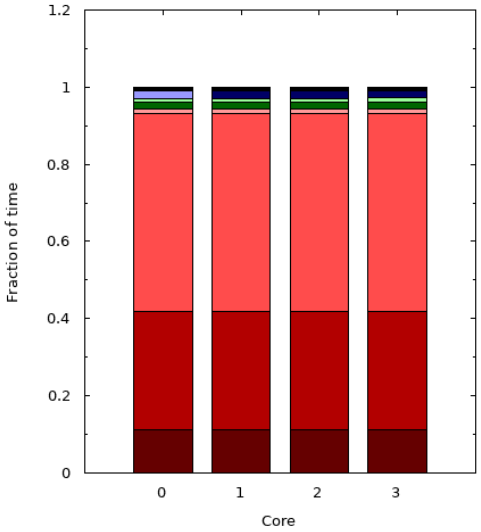
(c) `radix`: L2 energy 0.08 J; Total energy 0.38 J



(d) `fft_02`

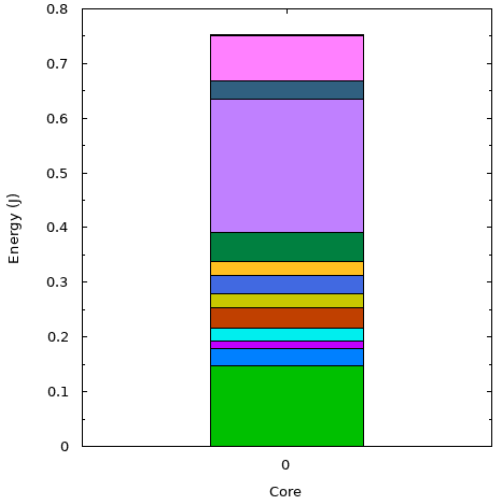


(e) `ocean.cont`

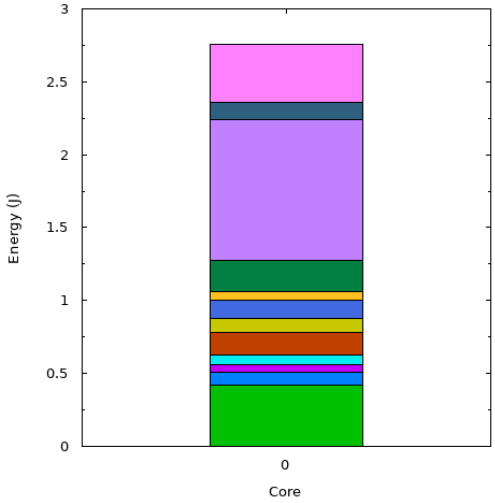


(f) `radix`

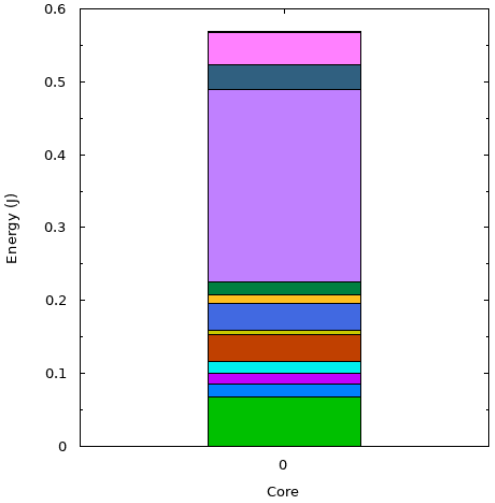
Figure 3: **1 MB L2 Cache Size:** Energy stacks for (a,d) `fft_02`, (b,e) `ocean.cont`, and (c.f) `radix` benchmarks using **1 MB** L2 cache size on core 0, and their corresponding CPI stacks.



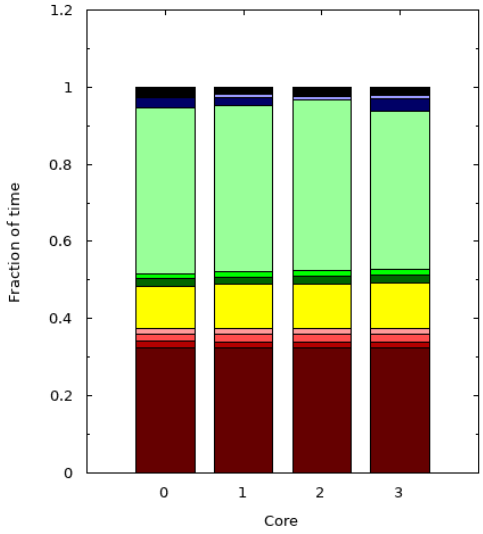
(a) `fft_02`: L2 enery 0.24 J; Total energy 0.75 J



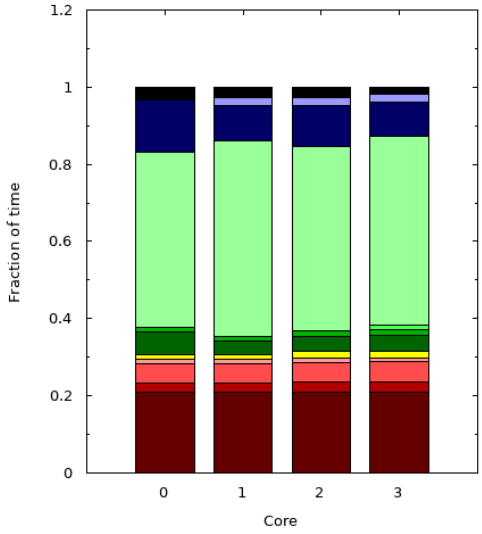
(b) `ocean.cont`: L2 energy 0.96 J; Total energy 2.76 J



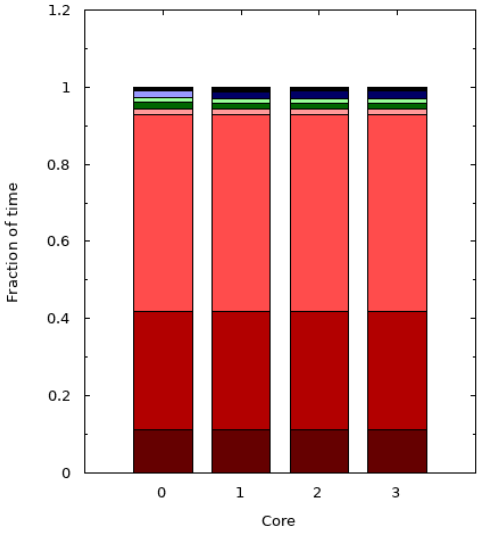
(c) `radix`: L2 energy 0.26 J; Total energy 0.57 J



(d) `fft_02`

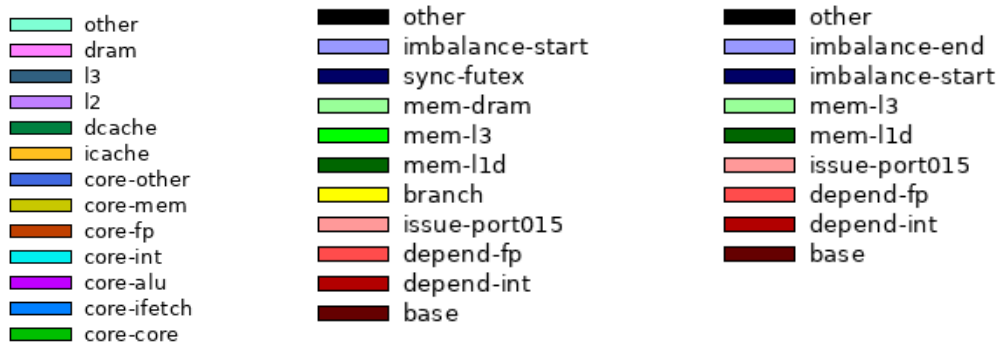


(e) `ocean.cont`

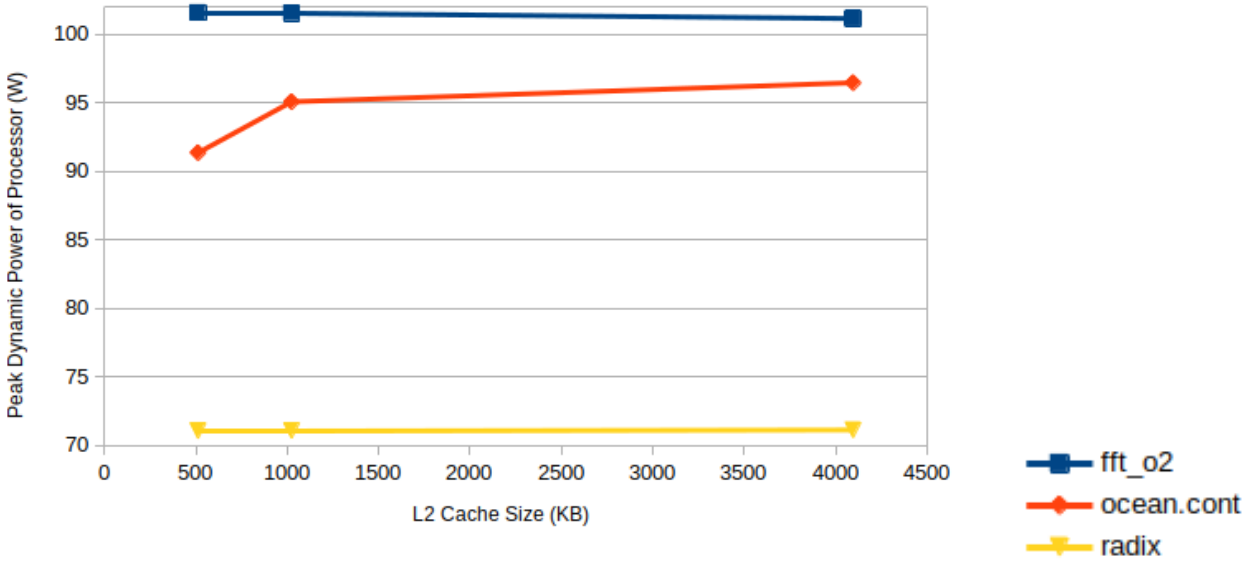


(f) `radix`

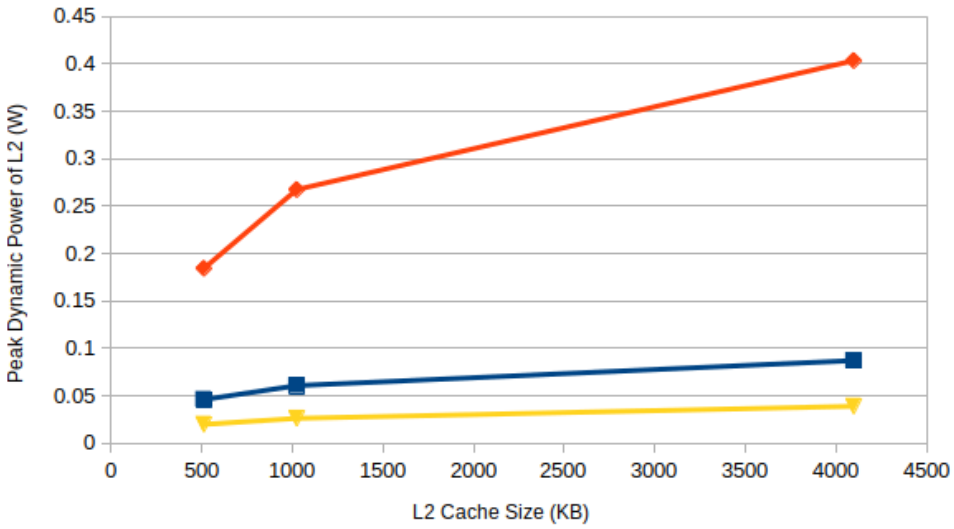
Figure 4: **4 MB L2 Cache Size:** Energy stacks for (a,d) `fft_02`, (b,e) `ocean.cont`, and (c.f) `radix` benchmarks using 4 MB L2 cache size on core 0, and their corresponding CPI stacks.



From Fig. 2 - 4 we can see that the increased diversity and complexity of the operations and instructions in the `ocean.cont` benchmark resulted in higher overall energy consumption; the y-axis energy scale for Fig. 2b, 3b, 4b is bigger than those for the other two benchmarks. The `ocean` suite of test studies large-scale ocean movements based on currents, and uses 4D array grids and a red-black Gauss-Seidel multigrid equation solver[4]. The `radix` suite uses an iterative radix sort algorithm that generates histograms and has each processor permute array index keys, a process that depends on processors communicating in order to determine keys thorough writes. And `FFT` is described as a complex, optimized 1D version of radix that minimizes interprocess communication. From their CPI stacks, we can see that the `ocean.cont` benchmark spent a greater fraction of time on `imbalance-start`, `sync-futex`, and `other` kinds of instructions, so these are likley what contributed to its higher engery consumption over `ftt_O2` and `radix`. Notice, too, for other kinds of instructions, `ocean.cont` spent fractions of time between the other two benchmarks; as in, `oceant.cont` had less `branch` and `base` instructions than `ftt_O2`, but more than `radix`; likewise, `oceant.cont` had less `depend-fp` and `depend-int` instructions than `radix`, but more than `ftt_O2`; and `oceant.cont` had almost similar time on `mem` related instructions than `fft_O2` but more than `radix`. The energy differences are likely to come from `imbalance-start`, `sync-futex`, and `other` kinds of instructions, which must be consume more energy if they have to execute. And finally, the `ftt_O2` benchmark consumed more overall energy than `radix`, evident by `ftt_O2` being a complex version of and algorithm that `radix` derives. For example, `ftt_O2` has clearly more `mem-dram` operations, while `radix` is dominated by integer and floating point related operations that likely didn't require memory access, seen in its bigger deep purple strip on the energy stack that represents `core-alu` and the smaller pink strip that represents `dram` (c). This pattern is observed across all the L2 cache size sweep.



(a)



(b)

Figure 5: L2 cache sizes plotted against peak dynamic power of (a) the processor, and (b) the L2 cache, for each of the three benchmarks given in the legend.

For each of the individual L2 cache sizes accross the three benchmarks, peak dynamic power (DP) of the processor (Fig. 5a) for `fft_O2` always peaked higher than `ocean.cont` which always peaked higher than `radix`. Sweeping across the L2 cache size, `fft_O2` DP always peaked higher than `radix` for the reasons it consumed more enery than `radix` discussed on the previous page. `fft_O2` DP peaking higher than `ocean.cont` each time, though, even though `ocean.cont` consumed more total energy, may be due to `fft_O2` having more branch instructions than the `ocean.cont` benchmark (see the yellow strip in the cpi-stacks in Fig. 4). Things like speculative execution and missed branch predictions are costly in power, and

those potentials may increase as branch insturction increase.

For each of the individual L2 cache sizes accross the three benchmarks, peak DP of the L2 (Fig. 5b) was as expected, where no matter the L2 cache size, the `ocean.cont` always peaked higher thant `fft_O2` which peaked higher than `radix`. Looking back at Fig. 4a - 4c, this matches what we see, where in `ocean.cont`, the L2 consumed more energy than the L2 in `fft_O2`, which consumed more energy than the L2 in `radix`. In Fig. 4d - 4f, we can also see that tests with more `mem-dram` instructions consumer more energy; in particular, `ocean.cont` spends a larger fraction of time on these instructions than any other, which we see `fft_O2` spends almost equal time between `mem-dram` and `base` operations, which likely contributes to why it's L2 doesn't peak higher than that of `ocean.cont`.

4 Performance Analysis

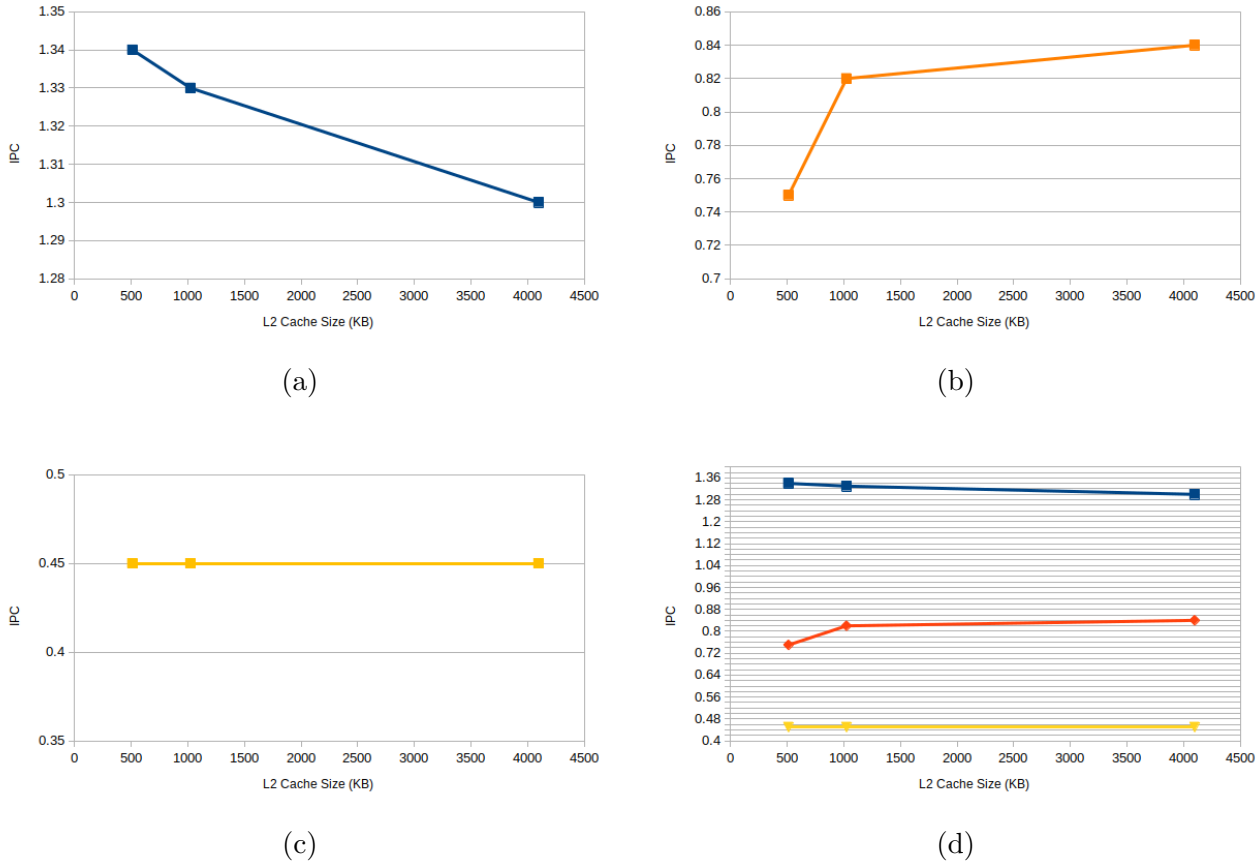


Figure 6: L2 Cache Size vs. IPC plots for the (a) `fft_O2` (blue), (b) `ocean.cont` (orange), and (c) `radix` (yellow) benchmarks; All plots are displayed together in Fig. 6d in order to see the IPC across the three benchmarks for one L2 cache size.

Fig. 6 shows that L2 cache size affects IPC differently for each benchmark. Firstly, Fig. 6d shows all the benchmarks together; for a given L2 size, `fft_O2` has greater IPC than `ocean.cont`, which is greater than `radix`. `FTT` was already noted to be more optimized than `Radix`[4]. It is surprising that given the lower energy stack and particular CPI stack for `radix` in Fig. 4, that is has lower IPC than the other benchmarks; this may be due to inefficient sorting algorithms rather than the type of instructions it uses. Since it does not have much in ways of memory accessing instructions (Fig. (2-4)f), it makes sense that in Fig. 6c, the IPC doesn't change with the L2 cache size for `radix`.

In Fig. 6a, IPC decreases with L2 cache size for `fft_O2`: Typically, we'd see that increasing total cache size improves miss rate[5] (Appendix B), and therefore overall performance, yet IPC decreases with the L2 sweep. Since associativity and block sized remained the same across the sweep, this dip in performance with larger L2 could be the result of incurring a greater miss and hit access times for memory because it is larger. This is supported by the fact that in `fft_O2`, the L2 consumes the most energy relative to the other components when L2 is 4 MB (Fig. 4a), and that a substantial fraction of its time is spend on memory

accessing instructions (Fig. 4d). Penalties in memory access could be negatively affecting other performance.

Conversly, in Fig. 6b, IPC increases with L2 cache size for `ocean.cont`: This benchmark probably does benefit from the improved miss rate of a larger total L2 cache size without incurring higher miss/hit penalty for accessing memory. `ocean.cont` operations lean heavily more on memory access instructions than the other benchmarks (Fig. (2-4)e), and at the larger 4 MB L2 cache size, more energy is consumed by L2 than any other component (Fig. 4b). Differently than `fft_O2`, DP peaks for L2 higher for `ocean.cont` than `fft_O2` (Fig. 5b); so, by Amdahl's Law, improvemnts in L2 size would benefit performance in `ocean.cont` proportionally more than for `fft_O2`.

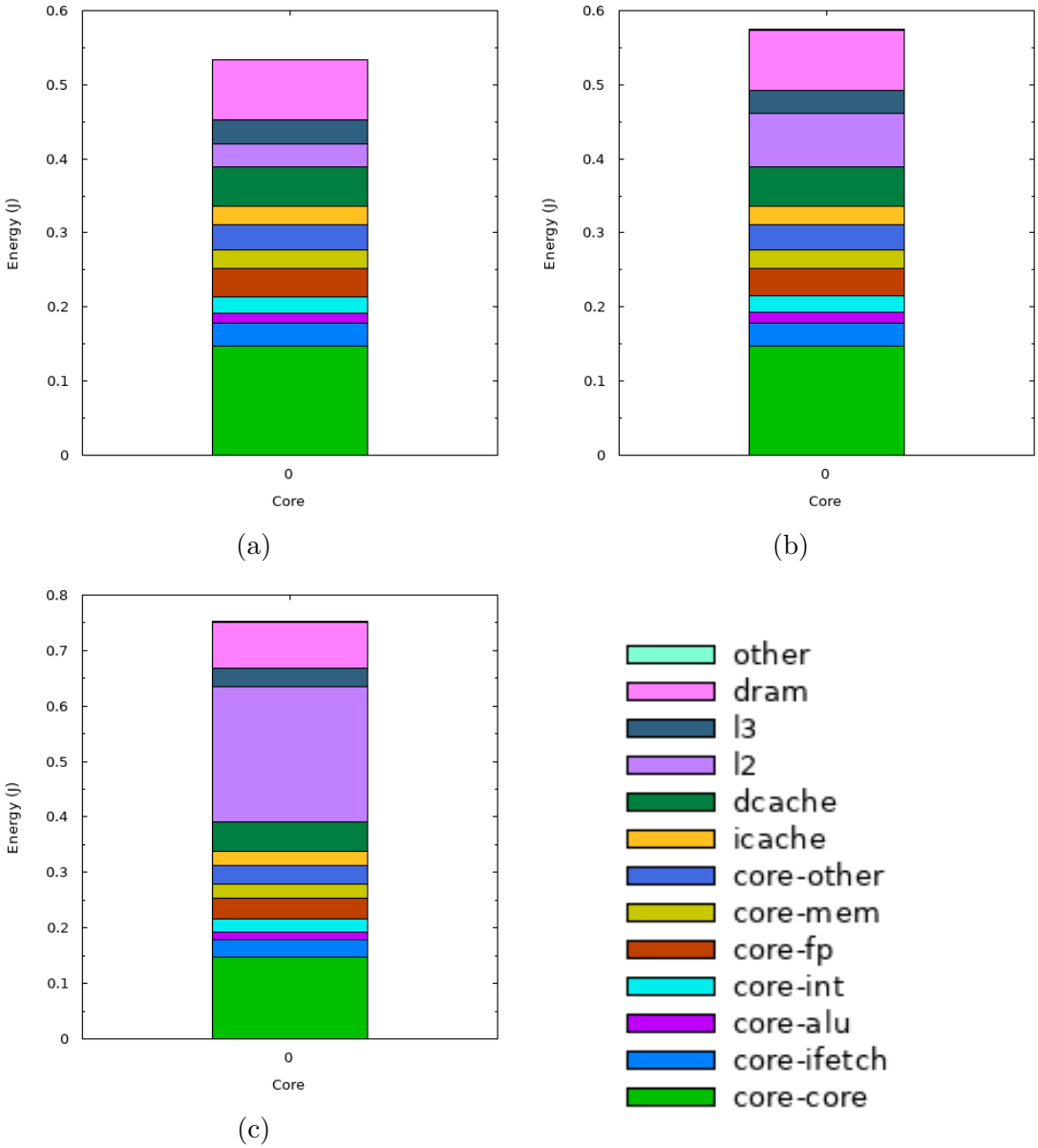
5 Conclusion

Using the `SniperSim` tool, we were able to observe affects of L2 cache sizes across three benchmarks that each had observable differences between their CPI stacks. Unsurprisingly, as L2 size increased, so did it's energy consumption; across different benchmarks' performance, only those that had larger fractions on time with memory access operations were sensitive to the L2 sweep. A surprising result was that increasing the L2 cache size negatively affected IPC of the `fft_O2` benchmark. If time permitted for more follow-up simulations, I suggest running these benchmarks with additional sweeps on cache associativity and block size. The current associativity of 8 may be high enough that is negatively affects performance in `fft_O2` at the larger L2 cache size because of the cost of searching 8 ways in parallel, so I would expect it to improve for `fft_O2`. If it gets low enough, it is likely to negatively impact performance for `fft_O2` and `ocean.cont`, because they have significant fraction of time spent doing memory access instructions, while all this won't affect IPC of `radix` just as it wasn't affect in the L2 cache size sweep. Likewise, it may be interesting to see a sweep of cache block size, because a too low or too high block size (range of about 16 to 256 byte) can both negatively impact performance for a particular total cache size[5] (Appendix B). I would expect to see the IPC data to have a bell shape with a cache block size sweep.

6 Appendix: Post Processed Data (Figures and Output Values)

6.1 fft_O2

6.1.1 Power Results



(a) (b)

(c)

Figure 7: Power for core 0 in `fft_O2` benchmark with L2 cache size of (a) 512 KB, (b) 1 MB, and (c) 4 MB.

	Power	Energy	Energy %
core-core	16.59 W	0.15 J	27.62%
core-ifetch	3.48 W	0.03 J	5.80%
core-alu	1.58 W	0.01 J	2.63%
core-int	2.50 W	0.02 J	4.16%
core-fp	4.28 W	0.04 J	7.13%
core-mem	2.81 W	0.02 J	4.68%
core-other	3.78 W	0.03 J	6.29%
icache	2.84 W	0.03 J	4.73%
dcache	5.93 W	0.05 J	9.88%
l2	3.57 W	0.03 J	5.95%
l3	3.55 W	0.03 J	5.90%
dram	9.12 W	0.08 J	15.19%
other	0.03 W	0.31 mJ	0.06%
core	35.01 W	0.31 J	58.29%
cache	15.89 W	0.14 J	26.46%
total	60.05 W	0.53 J	100.00%

(a)

	Power	Energy	Energy %
core-core	16.52 W	0.15 J	25.66%
core-ifetch	3.47 W	0.03 J	5.39%
core-alu	1.58 W	0.01 J	2.45%
core-int	2.49 W	0.02 J	3.87%
core-fp	4.27 W	0.04 J	6.64%
core-mem	2.80 W	0.02 J	4.35%
core-other	3.79 W	0.03 J	5.89%
icache	2.83 W	0.03 J	4.40%
dcache	5.91 W	0.05 J	9.18%
l2	8.02 W	0.07 J	12.45%
l3	3.53 W	0.03 J	5.48%
dram	9.14 W	0.08 J	14.20%
other	0.04 W	0.34 mJ	0.06%
core	34.93 W	0.31 J	54.23%
cache	20.29 W	0.18 J	31.51%
total	64.40 W	0.57 J	100.00%

(b)

	Power	Energy	Energy %
core-core	16.17 W	0.15 J	19.61%
core-ifetch	3.41 W	0.03 J	4.14%
core-alu	1.57 W	0.01 J	1.90%
core-int	2.46 W	0.02 J	2.98%
core-fp	4.24 W	0.04 J	5.15%
core-mem	2.74 W	0.03 J	3.33%
core-other	3.73 W	0.03 J	4.52%
icache	2.79 W	0.03 J	3.38%
dcache	5.80 W	0.05 J	7.04%
l2	26.71 W	0.24 J	32.40%
l3	3.51 W	0.03 J	4.25%
dram	9.25 W	0.08 J	11.22%
other	0.06 W	0.53 mJ	0.07%
core	34.32 W	0.31 J	41.63%
cache	38.81 W	0.35 J	47.08%
total	82.44 W	0.75 J	100.00%

(c)

Figure 8: Specific values for each components' power consumption (See. Fig. 7), for `fft_02` benchmark with L2 cache size of (a) 512 KB, (b) 1 MB, and (c) 4 MB.

6.1.2 CPI Stacks

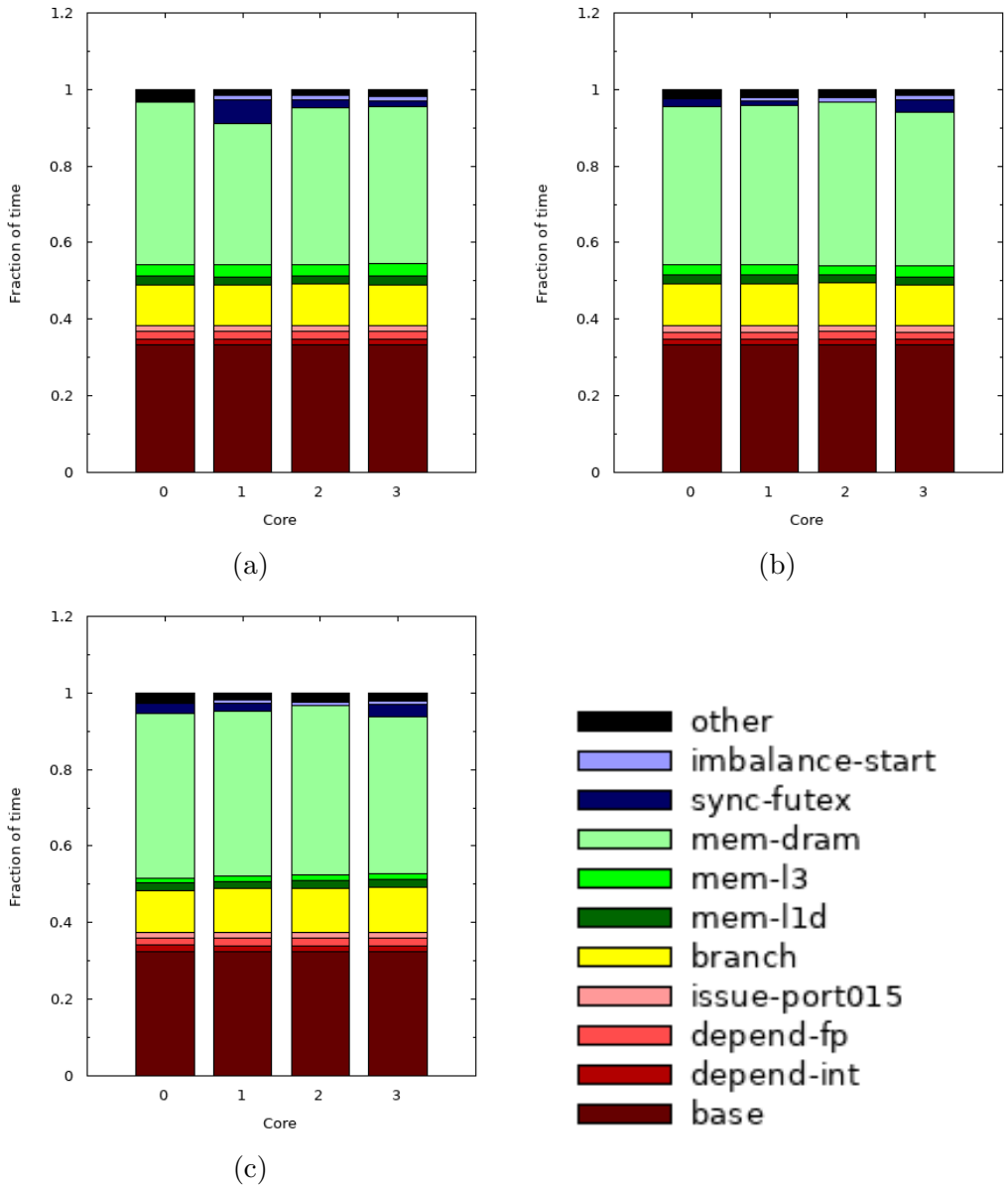


Figure 9: CPI stack for `fft_O2` benchmark with L2 cache size of (a) 512 KB, (b) 1 MB, and (c) 4 MB. Specific values for each components' fraction of time given in Fig. 10

CPI	Core 0	Core 1	Core 2	Core 3
base	0.25	0.25	0.25	0.25
depend-int	0.01	0.01	0.01	0.01
depend-fp	0.01	0.01	0.01	0.01
issue-port015	0.01	0.01	0.01	0.01
branch	0.08	0.08	0.08	0.08
mem-l1d	0.02	0.02	0.02	0.02
mem-l3	0.02	0.02	0.02	0.02
mem-dram	0.32	0.28	0.31	0.31
sync-futex	0.00	0.05	0.02	0.01
imbalance-start	0.00	0.01	0.01	0.01
other	0.02	0.01	0.01	0.01
total	0.75	0.75	0.75	0.75

(a)

CPI	Core 0	Core 1	Core 2	Core 3
base	0.25	0.25	0.25	0.25
depend-int	0.01	0.01	0.01	0.01
depend-fp	0.01	0.01	0.02	0.01
issue-port015	0.01	0.01	0.01	0.01
branch	0.08	0.08	0.08	0.08
mem-l1d	0.02	0.02	0.02	0.02
mem-l3	0.02	0.02	0.02	0.02
mem-dram	0.31	0.31	0.32	0.30
sync-futex	0.02	0.01	0.00	0.02
imbalance-start	0.00	0.01	0.01	0.01
other	0.02	0.02	0.02	0.01
total	0.75	0.75	0.75	0.75

(b)

CPI	Core 0	Core 1	Core 2	Core 3
base	0.25	0.25	0.25	0.25
depend-int	0.01	0.01	0.01	0.01
depend-fp	0.02	0.01	0.02	0.02
issue-port015	0.01	0.01	0.01	0.01
branch	0.08	0.09	0.09	0.09
mem-l1d	0.01	0.02	0.02	0.02
mem-l3	0.01	0.01	0.01	0.01
mem-dram	0.33	0.33	0.34	0.31
sync-futex	0.02	0.01	0.00	0.02
imbalance-start	0.00	0.01	0.01	0.01
other	0.02	0.01	0.02	0.02
total	0.77	0.77	0.77	0.77

(c)

Figure 10: Specific values for each components' CPI stack fraction of time (See. Fig. 9), for `fft_O2` benchmark with L2 cache size of (a) 512 KB, (b) 1 MB, and (c) 4 MB.

6.2 ocean.cont

6.2.1 Power Results

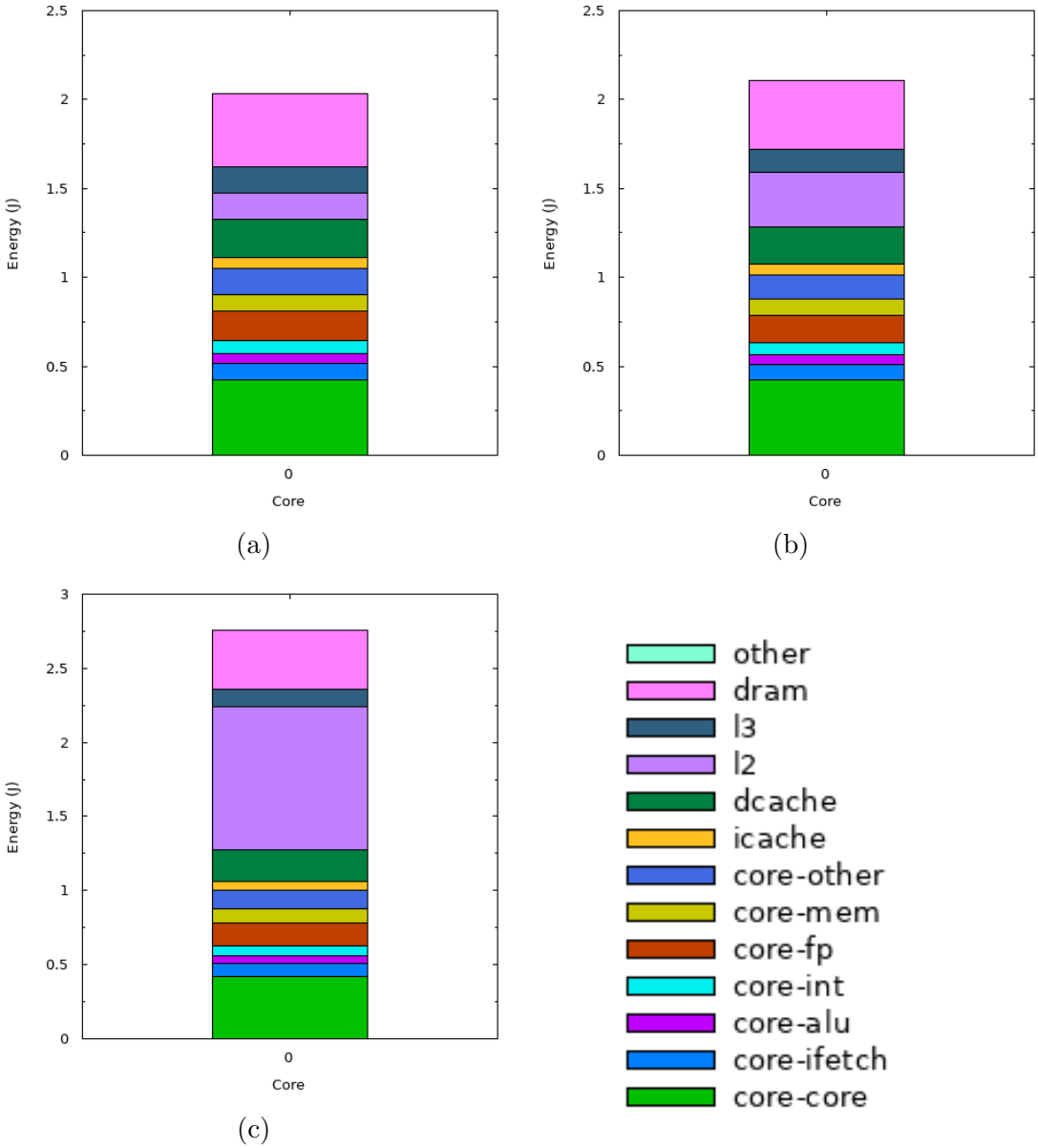


Figure 11: Power for core 0 in `ocean.cont` benchmark with L2 cache size of (a) 512 KB, (b) 1 MB, and (c) 4 MB.

	Power	Energy	Energy %
core-core	10.96 W	0.43 J	20.94%
core-ifetch	2.36 W	0.09 J	4.51%
core-alu	1.41 W	0.05 J	2.70%
core-int	1.89 W	0.07 J	3.62%
core-fp	4.22 W	0.16 J	8.07%
core-mem	2.40 W	0.09 J	4.58%
core-other	3.78 W	0.15 J	7.21%
icache	1.58 W	0.06 J	3.02%
dcache	5.59 W	0.22 J	10.68%
l2	3.84 W	0.15 J	7.34%
l3	3.68 W	0.14 J	7.03%
dram	10.59 W	0.41 J	20.22%
other	0.03 W	1.35 mJ	0.07%
core	27.03 W	1.05 J	51.64%
cache	14.70 W	0.57 J	28.07%
total	52.35 W	2.03 J	100.00%

(a)

	Power	Energy	Energy %
core-core	11.91 W	0.42 J	20.08%
core-ifetch	2.49 W	0.09 J	4.20%
core-alu	1.44 W	0.05 J	2.42%
core-int	1.96 W	0.07 J	3.31%
core-fp	4.34 W	0.15 J	7.31%
core-mem	2.59 W	0.09 J	4.37%
core-other	3.79 W	0.13 J	6.40%
icache	1.65 W	0.06 J	2.78%
dcache	6.00 W	0.21 J	10.12%
l2	8.52 W	0.30 J	14.37%
l3	3.62 W	0.13 J	6.11%
dram	10.95 W	0.39 J	18.47%
other	0.04 W	1.36 mJ	0.06%
core	28.52 W	1.01 J	48.09%
cache	19.79 W	0.70 J	33.37%
total	59.30 W	2.11 J	100.00%

(b)

	Power	Energy	Energy %
core-core	12.15 W	0.42 J	15.34%
core-ifetch	2.52 W	0.09 J	3.18%
core-alu	1.44 W	0.05 J	1.82%
core-int	1.98 W	0.07 J	2.50%
core-fp	4.37 W	0.15 J	5.51%
core-mem	2.64 W	0.09 J	3.33%
core-other	3.73 W	0.13 J	4.71%
icache	1.67 W	0.06 J	2.10%
dcache	6.11 W	0.21 J	7.71%
l2	27.66 W	0.96 J	34.92%
l3	3.58 W	0.12 J	4.52%
dram	11.31 W	0.39 J	14.28%
other	0.06 W	2.01 mJ	0.07%
core	28.84 W	1.00 J	36.40%
cache	39.02 W	1.36 J	49.25%
total	79.22 W	2.76 J	100.00%

(c)

Figure 12: Specific values for each components' power consumption (See. Fig. 11), for `ocean.cont` benchmark with L2 cache size of (a) 512 KB, (b) 1 MB, and (c) 4 MB.

6.2.2 CPI Stacks

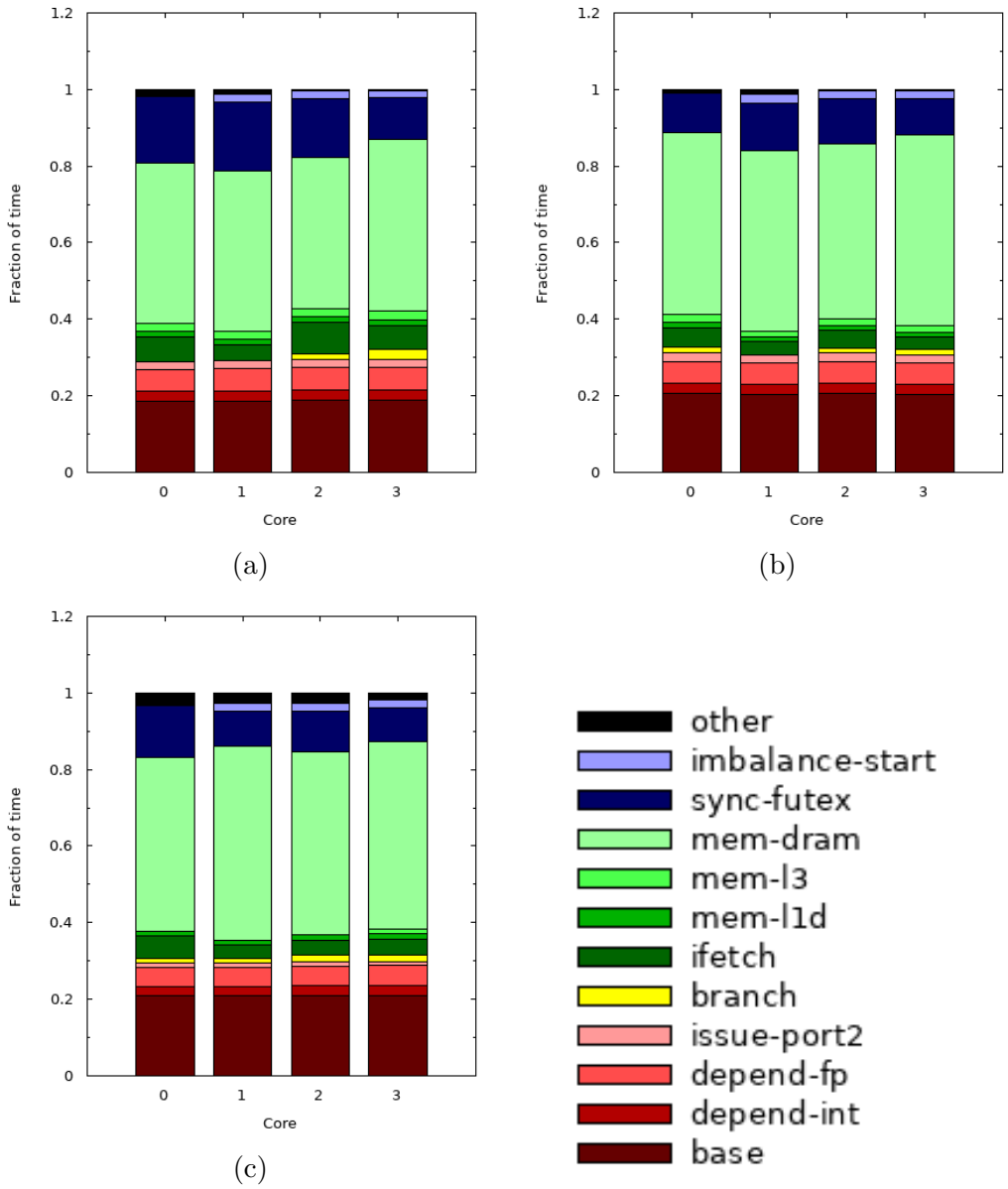


Figure 13: CPI stack for `ocean.cont` benchmark with L2 cache size of (a) 512 KB, (b) 1 MB, and (c) 4 MB. Specific values for each components' fraction of time given in Fig. 15

CPI	Core 0	Core 1	Core 2	Core 3
base	0.25	0.25	0.25	0.25
depend-int	0.03	0.03	0.03	0.03
depend-fp	0.08	0.08	0.08	0.08
issue	0.03	0.03	0.03	0.03
branch	0.00	0.00	0.02	0.04
ifetch	0.09	0.06	0.11	0.08
mem-l1d	0.02	0.02	0.02	0.02
mem-l3	0.03	0.03	0.03	0.03
mem-dram	0.56	0.56	0.52	0.59
sync-futex	0.23	0.24	0.21	0.14
imbalance-start	0.00	0.03	0.03	0.03
other	0.02	0.02	0.00	0.00
total	1.34	1.34	1.33	1.33

(a)

CPI	Core 0	Core 1	Core 2	Core 3
base	0.25	0.25	0.25	0.25
depend-int	0.03	0.03	0.03	0.03
depend-fp	0.07	0.07	0.07	0.07
issue	0.03	0.03	0.03	0.03
branch	0.02	0.00	0.02	0.02
ifetch	0.06	0.04	0.06	0.04
mem-l1d	0.02	0.02	0.02	0.02
mem-l3	0.02	0.02	0.02	0.02
mem-dram	0.58	0.58	0.56	0.61
sync-futex	0.12	0.15	0.14	0.12
imbalance-start	0.00	0.03	0.03	0.03
other	0.01	0.02	0.00	0.00
total	1.21	1.23	1.22	1.22

(b)

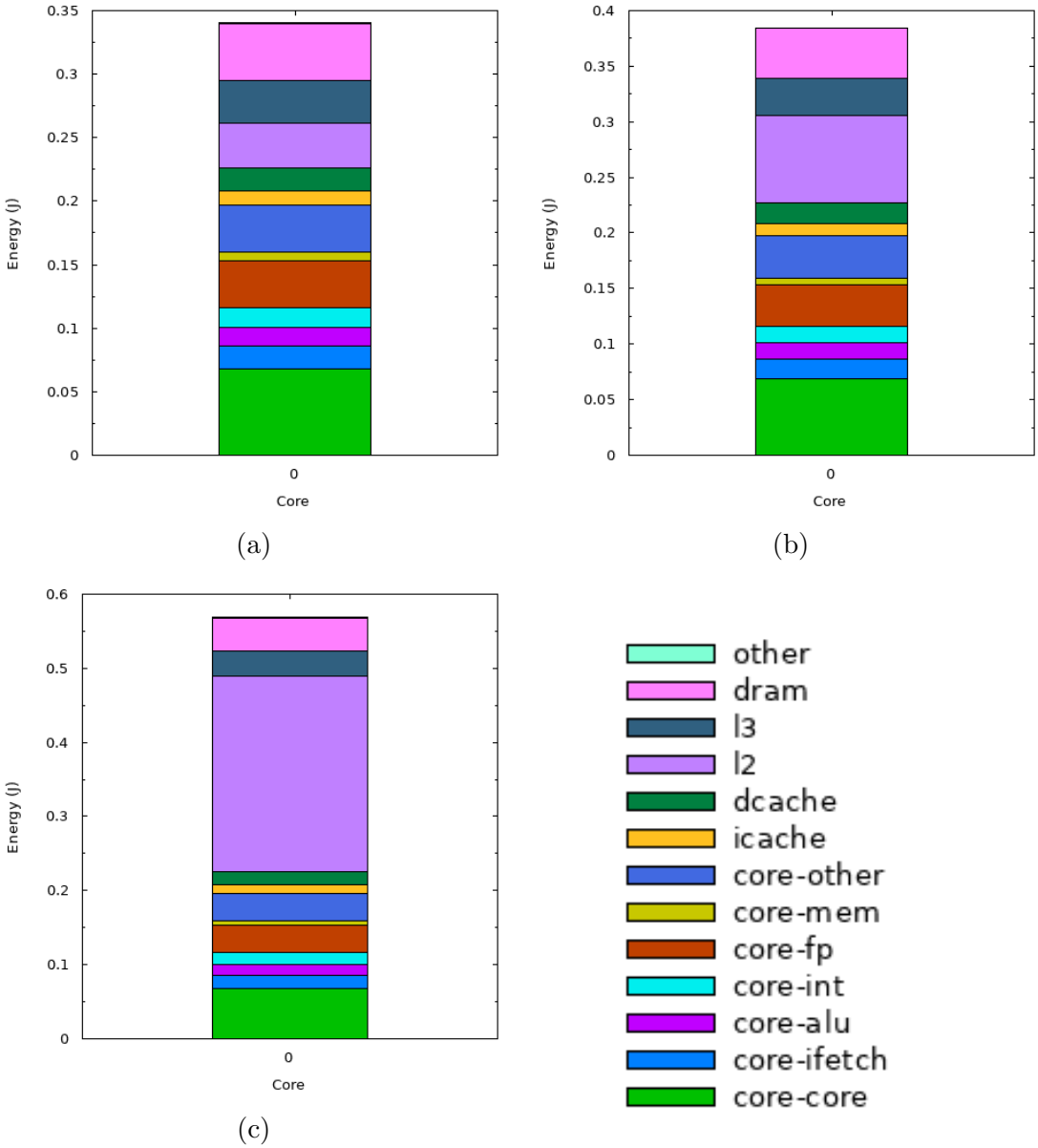
CPI	Core 0	Core 1	Core 2	Core 3
base	0.25	0.25	0.25	0.25
depend-int	0.03	0.03	0.03	0.03
depend-fp	0.06	0.06	0.06	0.06
issue-port2	0.01	0.01	0.01	0.01
branch	0.02	0.01	0.02	0.02
ifetch	0.07	0.04	0.05	0.05
mem-l1d	0.02	0.02	0.02	0.02
mem-l3	0.00	0.00	0.00	0.01
mem-dram	0.54	0.61	0.57	0.58
sync-futex	0.16	0.11	0.13	0.11
imbalance-start	0.00	0.03	0.03	0.03
other	0.04	0.03	0.03	0.02
total	1.20	1.20	1.19	1.19

(a)

Figure 15: Specific values for each components' CPI stack fraction of time (See. Fig. 13), for `ocean.cont` benchmark with L2 cache size of (a) 512 KB, (b) 1 MB, and (c) 4 MB.

6.3 radix

6.3.1 Power Results



(a) (b) (c)

Figure 16: Power for core 0 in `radix` benchmark with L2 cache size of (a) 512 KB, (b) 1 MB, and (c) 4 MB.

	Power	Energy	Energy %
core-core	6.91 W	0.07 J	20.08%
core-ifetch	1.79 W	0.02 J	5.22%
core-alu	1.51 W	0.01 J	4.38%
core-int	1.54 W	0.02 J	4.47%
core-fp	3.74 W	0.04 J	10.86%
core-mem	0.66 W	6.55 mJ	1.93%
core-other	3.78 W	0.04 J	10.98%
icache	1.09 W	0.01 J	3.16%
dcache	1.87 W	0.02 J	5.45%
l2	3.54 W	0.04 J	10.30%
l3	3.40 W	0.03 J	9.89%
dram	4.54 W	0.04 J	13.20%
other	0.03 W	0.34 mJ	0.10%
core	19.92 W	0.20 J	57.90%
cache	9.91 W	0.10 J	28.80%
total	34.40 W	0.34 J	100.00%

(a)

	Power	Energy	Energy %
core-core	6.91 W	0.07 J	17.77%
core-ifetch	1.79 W	0.02 J	4.62%
core-alu	1.51 W	0.01 J	3.87%
core-int	1.54 W	0.02 J	3.96%
core-fp	3.74 W	0.04 J	9.62%
core-mem	0.66 W	6.55 mJ	1.70%
core-other	3.79 W	0.04 J	9.77%
icache	1.09 W	0.01 J	2.80%
dcache	1.87 W	0.02 J	4.82%
l2	7.98 W	0.08 J	20.54%
l3	3.40 W	0.03 J	8.75%
dram	4.54 W	0.04 J	11.69%
other	0.04 W	0.38 mJ	0.10%
core	19.94 W	0.20 J	51.31%
cache	14.34 W	0.14 J	36.91%
total	38.85 W	0.38 J	100.00%

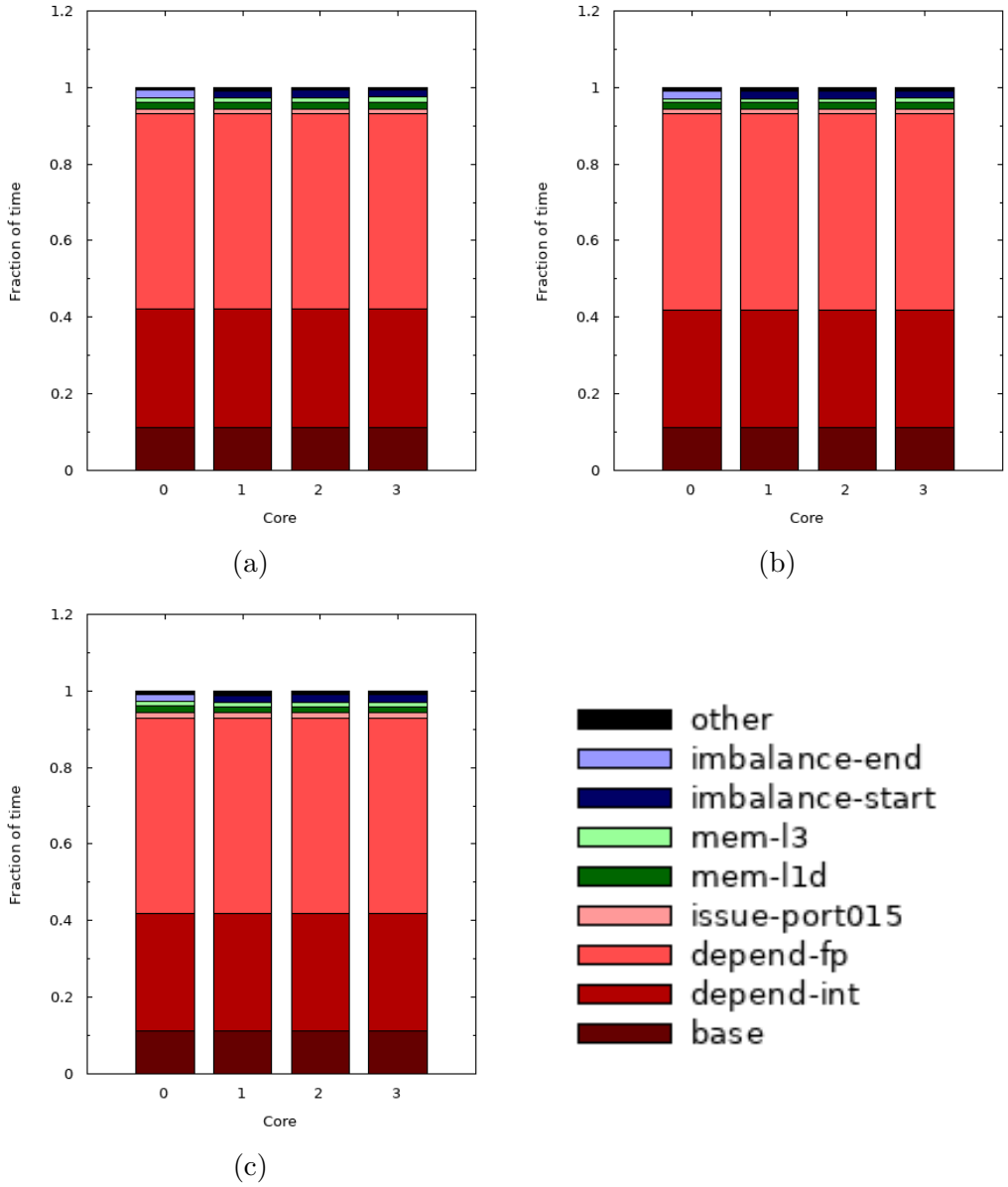
(b)

	Power	Energy	Energy %
core-core	6.90 W	0.07 J	12.01%
core-ifetch	1.79 W	0.02 J	3.12%
core-alu	1.51 W	0.01 J	2.62%
core-int	1.54 W	0.02 J	2.67%
core-fp	3.74 W	0.04 J	6.50%
core-mem	0.66 W	6.56 mJ	1.15%
core-other	3.73 W	0.04 J	6.49%
icache	1.09 W	0.01 J	1.89%
dcache	1.87 W	0.02 J	3.26%
l2	26.64 W	0.26 J	46.36%
l3	3.40 W	0.03 J	5.91%
dram	4.54 W	0.04 J	7.90%
other	0.06 W	0.57 mJ	0.10%
core	19.86 W	0.20 J	34.57%
cache	33.00 W	0.33 J	57.43%
total	57.46 W	0.57 J	100.00%

(c)

Figure 17: Specific values for each components' power consumption (See. Fig. 16), for `radix` benchmark with L2 cache size of (a) 512 KB, (b) 1 MB, and (c) 4 MB.

6.3.2 CPI Stacks



(a) (b)

(c)

Figure 18: CPI stack for `radix` benchmark with L2 cache size of (a) 512 KB, (b) 1 MB, and (c) 4 MB. Specific values for each components' fraction of time given in Fig. 19

CPI	Core 0	Core 1	Core 2	Core 3
base	0.25	0.25	0.25	0.25
depend-int	0.69	0.69	0.69	0.69
depend-fp	1.14	1.15	1.14	1.14
issue-port015	0.03	0.03	0.03	0.03
mem-l1d	0.04	0.04	0.04	0.04
mem-l3	0.03	0.02	0.03	0.03
imbalance-start	0.00	0.04	0.04	0.04
imbalance-end	0.04	0.00	0.00	0.00
other	0.02	0.02	0.02	0.01
total	2.24	2.24	2.24	2.24

(a)

CPI	Core 0	Core 1	Core 2	Core 3
base	0.25	0.25	0.25	0.25
depend-int	0.69	0.69	0.69	0.69
depend-fp	1.14	1.15	1.14	1.14
issue-port015	0.03	0.03	0.03	0.03
mem-l1d	0.04	0.04	0.04	0.04
mem-l3	0.03	0.02	0.02	0.03
imbalance-start	0.00	0.04	0.04	0.04
imbalance-end	0.04	0.00	0.00	0.00
other	0.02	0.02	0.02	0.02
total	2.24	2.24	2.24	2.24

(b)

CPI	Core 0	Core 1	Core 2	Core 3
base	0.25	0.25	0.25	0.25
depend-int	0.69	0.69	0.69	0.69
depend-fp	1.14	1.15	1.14	1.14
issue-port015	0.03	0.03	0.03	0.03
mem-l1d	0.04	0.04	0.04	0.04
mem-l3	0.03	0.02	0.02	0.03
imbalance-start	0.00	0.04	0.04	0.04
imbalance-end	0.04	0.00	0.00	0.00
other	0.02	0.02	0.02	0.02
total	2.24	2.24	2.24	2.24

(c)

Figure 19: Specific values for each components' CPI stack fraction of time (See. Fig. 18), for `radix` benchmark with L2 cache size of (a) 512 KB, (b) 1 MB, and (c) 4 MB.