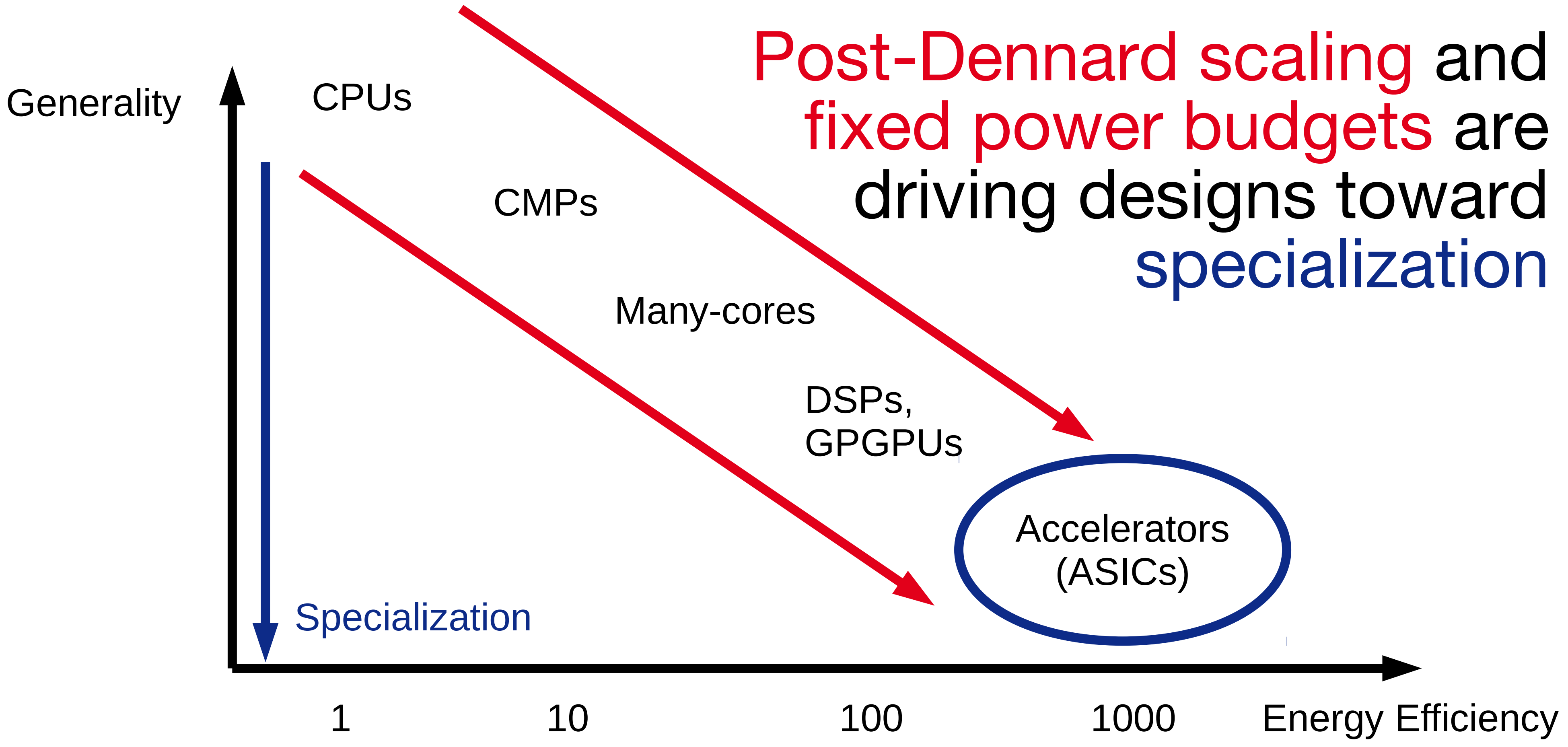
An Analysis of Accelerator Coupling in Heterogeneous Architectures

DAC'15, San Francisco, CA, USA

Emilio G. Cota
Paolo Mantovani
Giuseppe Di Guglielmo
Luca P. Carloni

Columbia University

Generality vs. Efficiency



Post-Dennard scaling and fixed power budgets are driving designs toward specialization

Accelerators have become become essential for high-efficiency systems, e.g. SoCs

Our Goal

Analysis of Accelerator Couplings

A major **trade-off** in accelerator design, since it determines how **memory** is accessed

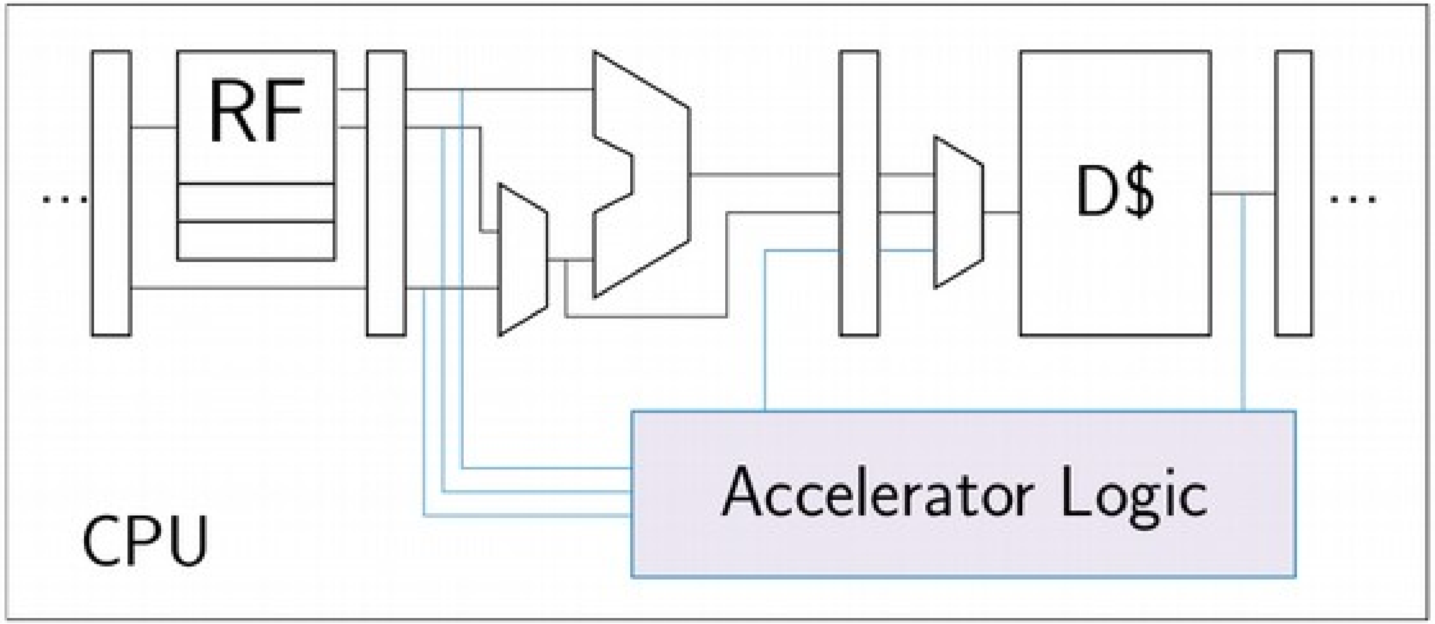
Our goal: to draw observations about **performance, efficiency and programmability** of accelerators with different couplings

Two main options w.r.t. CPUs:

- Tightly-Coupled (TCAs)
- Loosely-Coupled (LCAs)

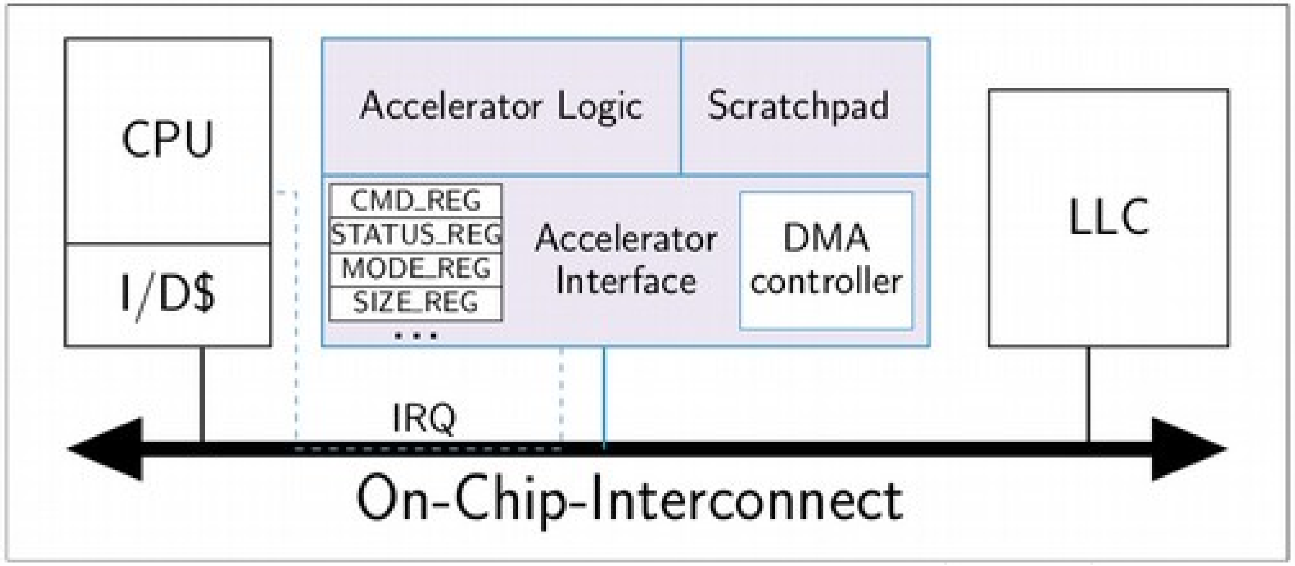
Tightly-Coupled (TCAs)

a.k.a. “coprocessor model”



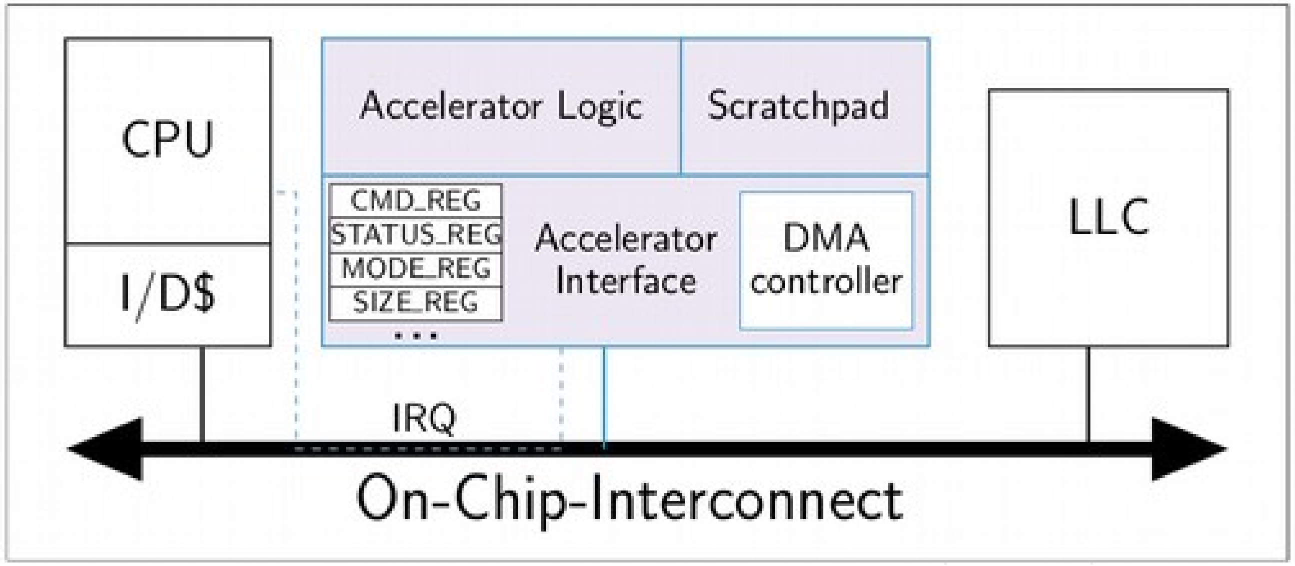
- ✓ Nil invocation overhead (via ISA extensions)
- ✓ No internal storage: direct access to L1 cache
- ✗ Limited portability: design heavily tied to CPU

Loosely-Coupled (LCAs)
a.k.a. “SoC-like model”



- ✔ Good design reuse: no CPU-specific knowledge
- ✗ Fixed set-up costs due to driver invocation and DMA
- ✔ Freedom to tailor private memories (scratchpads), e.g. providing different banks, ports, and bit widths
  - ✗ Scratchpads require large area expenses

Loosely-Coupled (LCAs)
a.k.a. “SoC-like model”



Two flavors:

- LLC-DMA
- DRAM-DMA

✔ Good design reuse: no CPU-specific knowledge

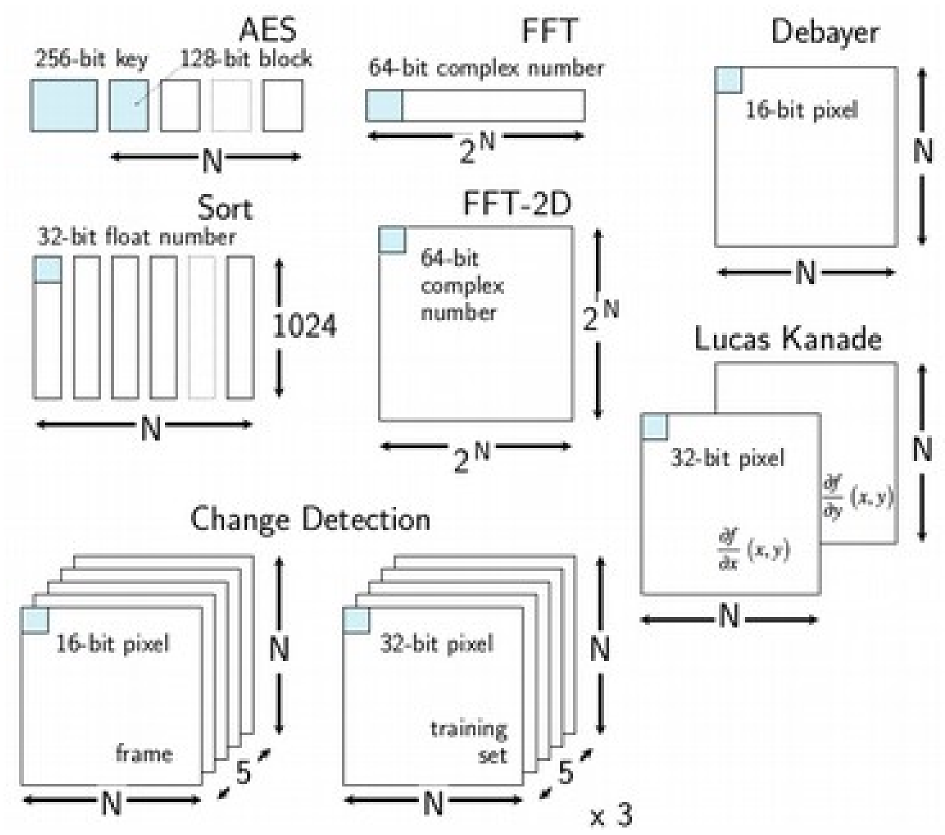
✗ Fixed set-up costs due to driver invocation and DMA

✔ Freedom to tailor private memories (scratchpads), e.g. providing different banks, ports, and bit widths

  ✗ Scratchpads require large area expenses

Target Applications

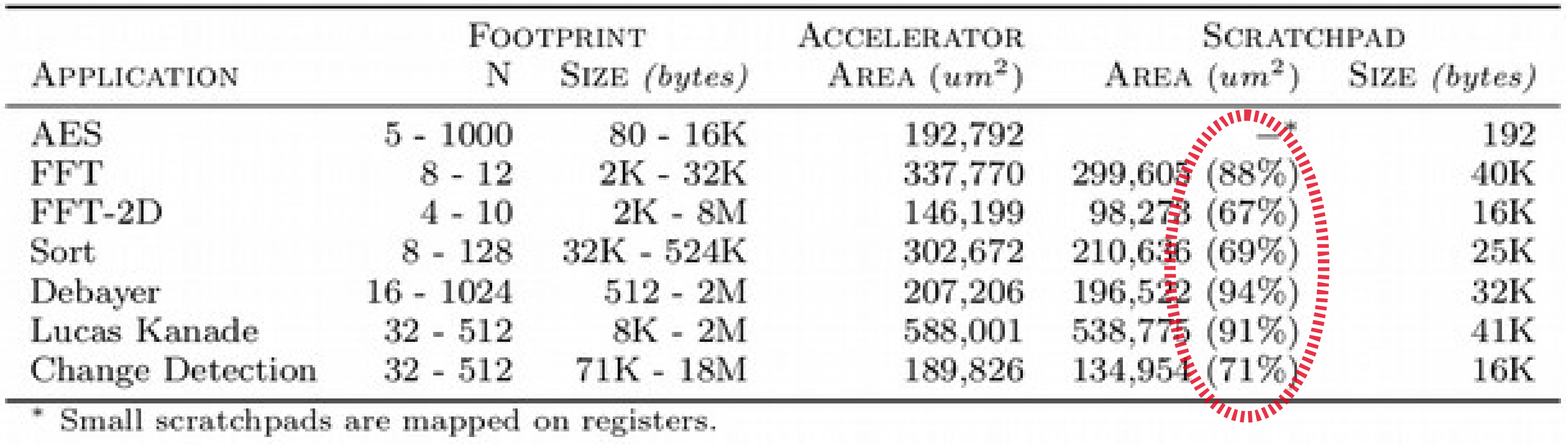
- Seven high-throughput applications from the PERFECT Benchmark Suite[*]



Application	Footprint N	Size (bytes)
AES	5 - 1000	80 - 16K
FFT	8 - 12	2K - 32K
FFT-2D	4 - 10	2K - 8M
Sort	8 - 128	32K - 524K
Debayer	16 - 1024	512 - 2M
Lucas Kanade	32 - 512	8K - 2M
Change Detection	32 - 512	71K - 18M

[*] http://hpc.pnl.gov/PERFECT

Accelerator Design



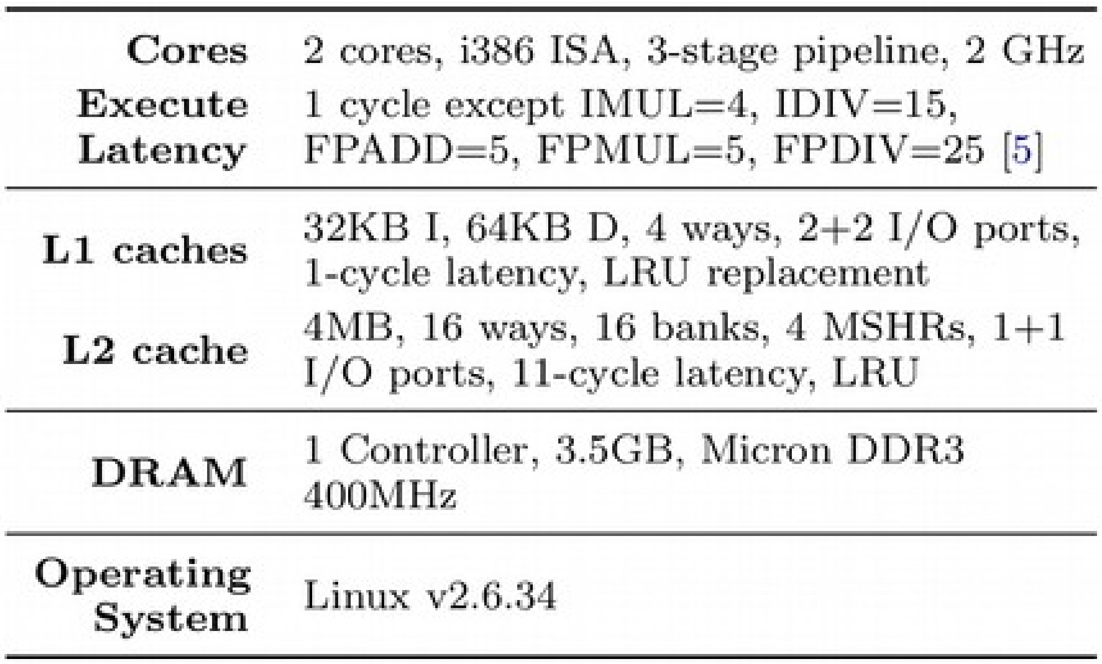
Application	Footprint N	Footprint Size *(bytes)*	Accelerator Area *(um²)*	Scratchpad Area *(um²)*	Scratchpad Size *(bytes)*
AES	5 - 1000	80 - 16K	192,792	—*	192
FFT	8 - 12	2K - 32K	337,770	299,605 (88%)	40K
FFT-2D	4 - 10	2K - 8M	146,199	98,273 (67%)	16K
Sort	8 - 128	32K - 524K	302,672	210,636 (69%)	25K
Debayer	16 - 1024	512 - 2M	207,206	196,522 (94%)	32K
Lucas Kanade	32 - 512	8K - 2M	588,001	538,775 (91%)	41K
Change Detection	32 - 512	71K - 18M	189,826	134,954 (71%)	16K

* Small scratchpads are mapped on registers.

- Used High-Level Synthesis for productivity
- Most effort is on the memory subsystem to exploit parallelism, i.e. a large number of operations per clock cycle
  - Most accelerator area is therefore memory

Experimental Methodology

- Full-system simulation running Linux
- In-order embedded-like i386 cores

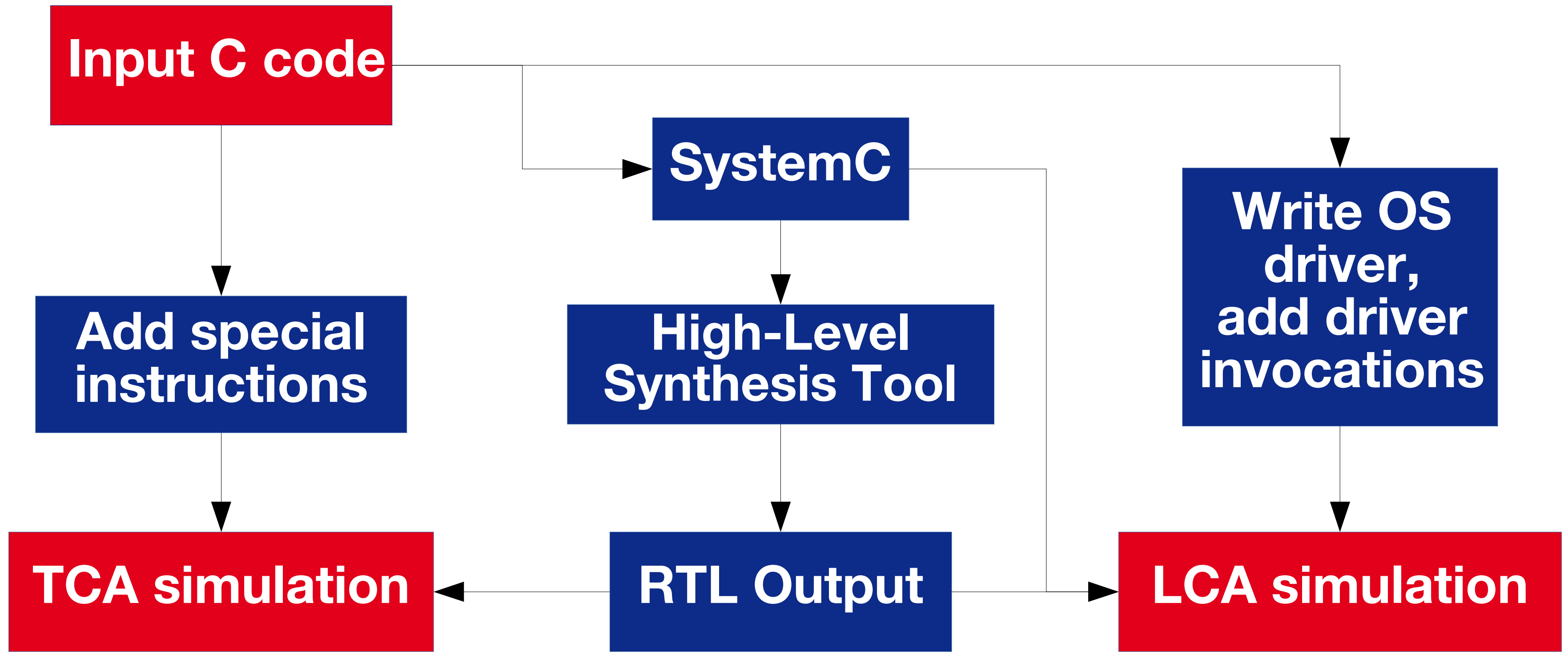


Cores	2 cores, i386 ISA, 3-stage pipeline, 2 GHz
Execute Latency	1 cycle except IMUL=4, IDIV=15, FPADD=5, FPMUL=5, FPDIV=25 [5]
L1 caches	32KB I, 64KB D, 4 ways, 2+2 I/O ports, 1-cycle latency, LRU replacement
L2 cache	4MB, 16 ways, 16 banks, 4 MSHRs, 1+1 I/O ports, 11-cycle latency, LRU
DRAM	1 Controller, 3.5GB, Micron DDR3 400MHz
Operating System	Linux v2.6.34

- Detailed Level-1 and Level-2 cache models
- Accurate DRAM simulation with DRAMSim2

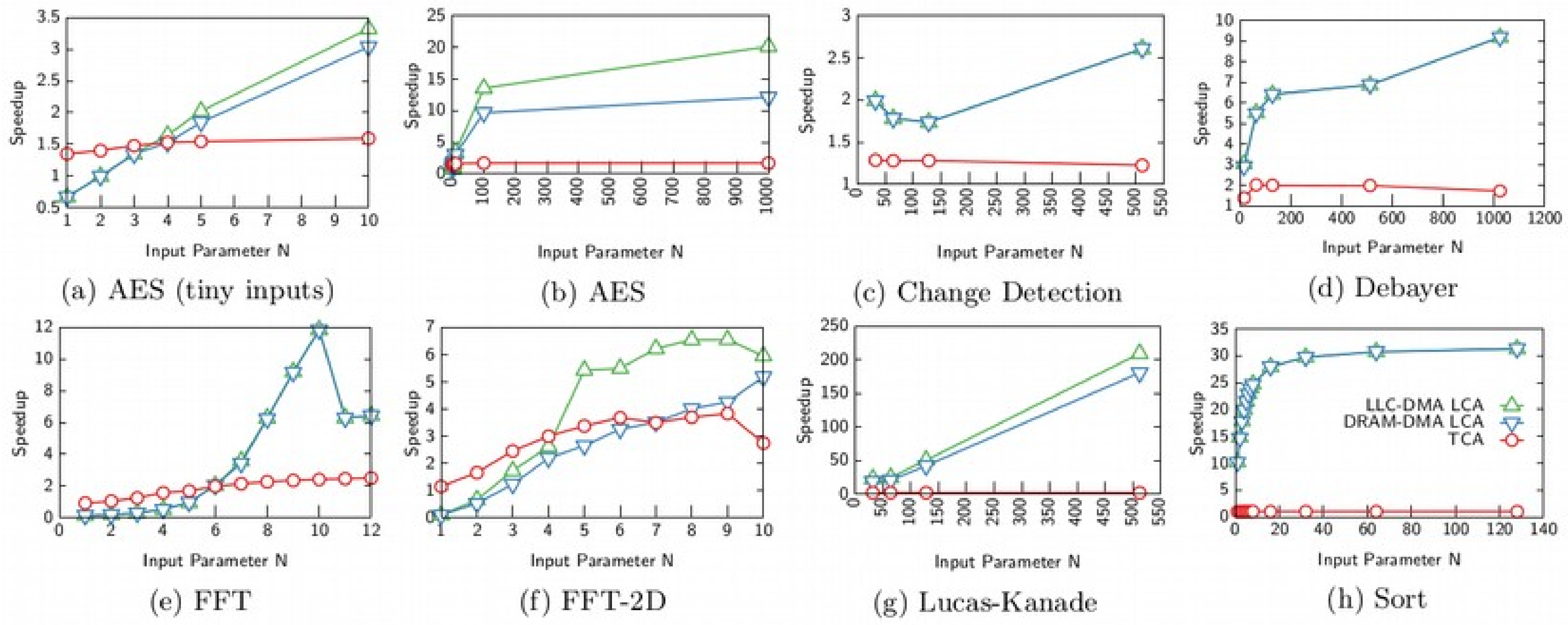
Experimental Methodology

Heterogeneous System Simulation



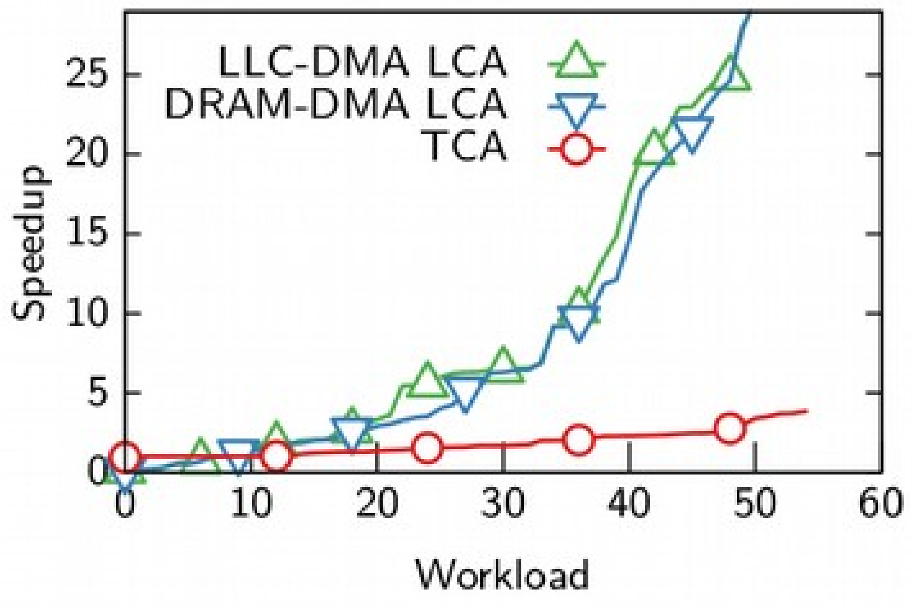
- Latencies from RTL are back-annotated into the simulator (for TCAs) and SystemC (LCAs)
- LCAs: SystemC accelerator simulation run in parallel with the simulator, synchronizing every 100 cycles

Speedup over Software

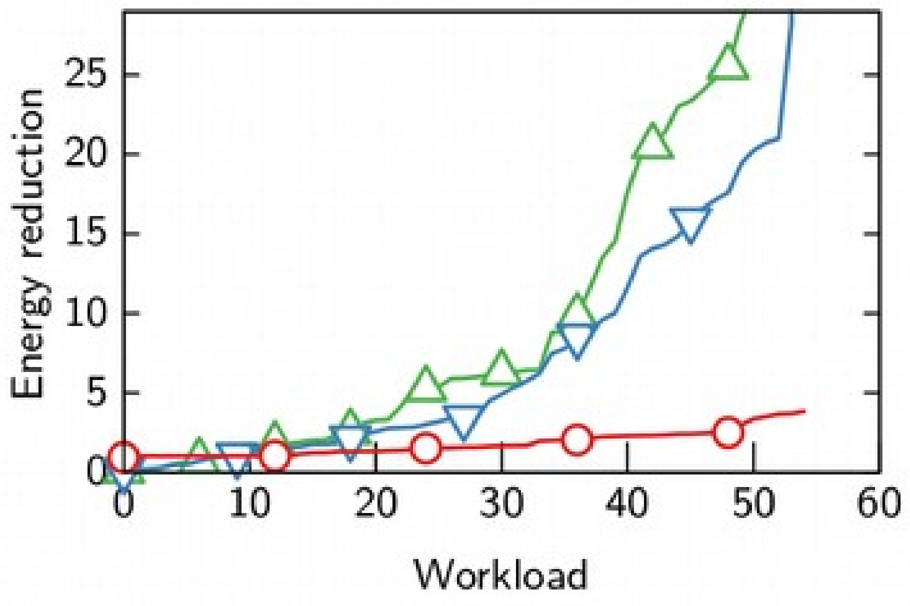


- LLC-DMA LCA > DRAM-DMA LCA > TCA
- Ratio of scratchpad vs. input size matters, e.g. FFT
- DRAM bandwidth bottleneck on accelerators with communication >> computation, e.g. sort

Performance & Energy



(a) Perf sorted



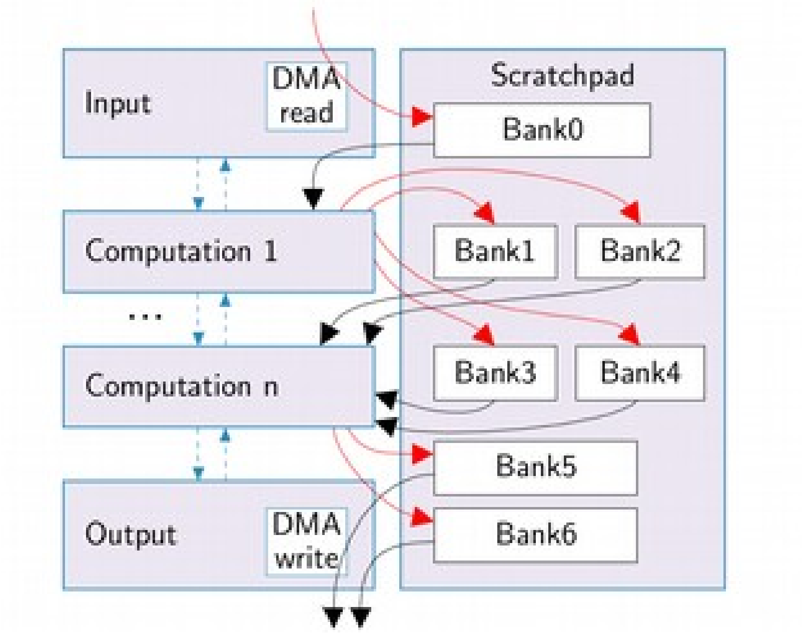
(b) Energy sorted

- LLC-DMA LCA > DRAM-DMA LCA > TCA
- Efficiency gap between LCAs due to difference in off-chip accesses
- LLC pollution study results in paper/poster

Concluding Observations

- LCAs best positioned to deliver high throughput given non-trivial inputs amenable to computation in bursts
  - DRAM bandwidth can limit this potential
- Programming LCAs is not conceptually complex
  - Operating Systems have simple, well-defined interfaces for this

- Why **LCAs > TCAs**: **Tailored, many-ported scratchpads are key to performance**
  - L1s cannot provide this parallelism (at most 2 ports!)


Input
DMA read
Scratchpad
Bank0
Computation 1
Bank1
Bank2
...
Computation n
Bank3
Bank4
Bank5
Output
DMA write
Bank6