A Run-time Configurable Cache/Scratchpad Memory with Virtualized User-Level RDMA Capability

G. Nikiforos, G. Kalokerinos, V. Papaefstathiou,
S. Kavadias, D. Pnevmatikatos and **M. Katevenis**

FORTH-ICS - SARC project

FORTH

6th HiPEAC Industrial Workshop
November 26, 2008
Paris

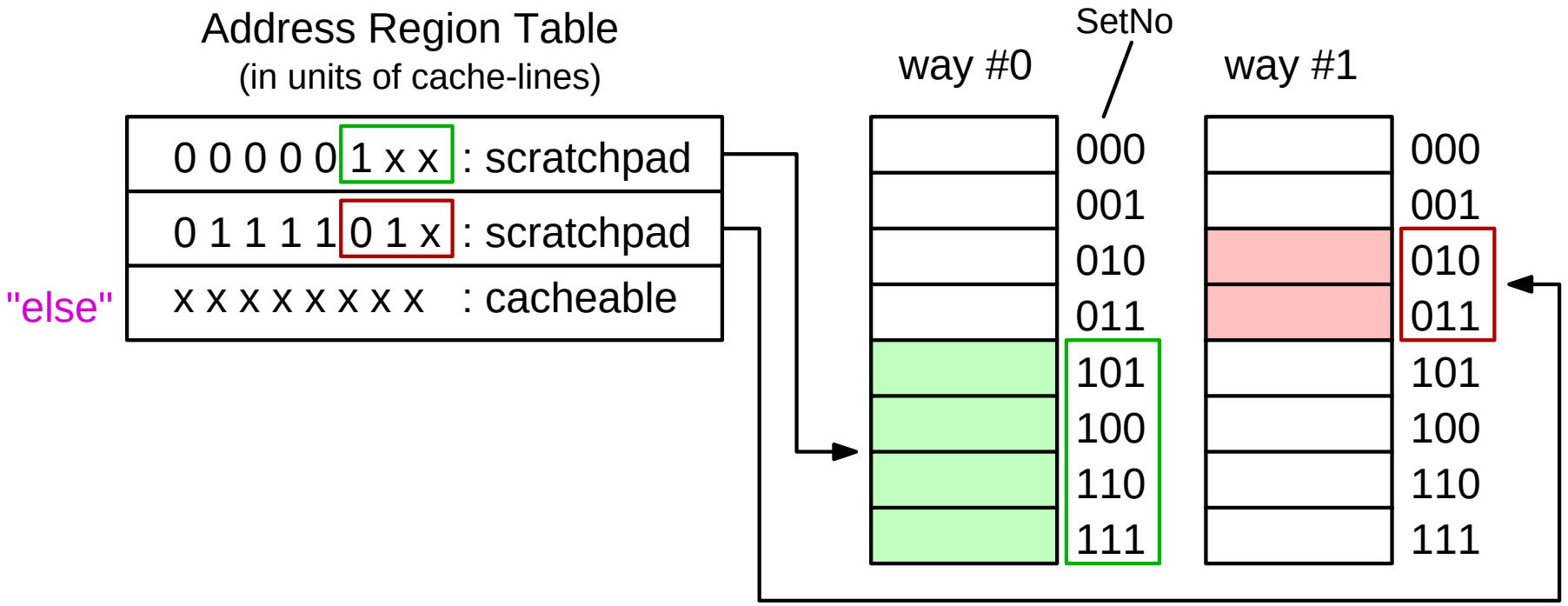
Memory Hierarchies and Locality Management

- Multilevel Coherent Caches:
  - + easy programming: “let the data adaptively find their way”
  - – poor scalability to large number of cores
  - – difficult to optimize or achieve deterministic performance, in the cases where application has knowledge about locality
- Scratchpads with DMA capabilities:
  - + allow optimizations and predictable performance ⇨ scalable
  - – hard to program
- **Merged:** get the best of both worlds …

Talk Outline

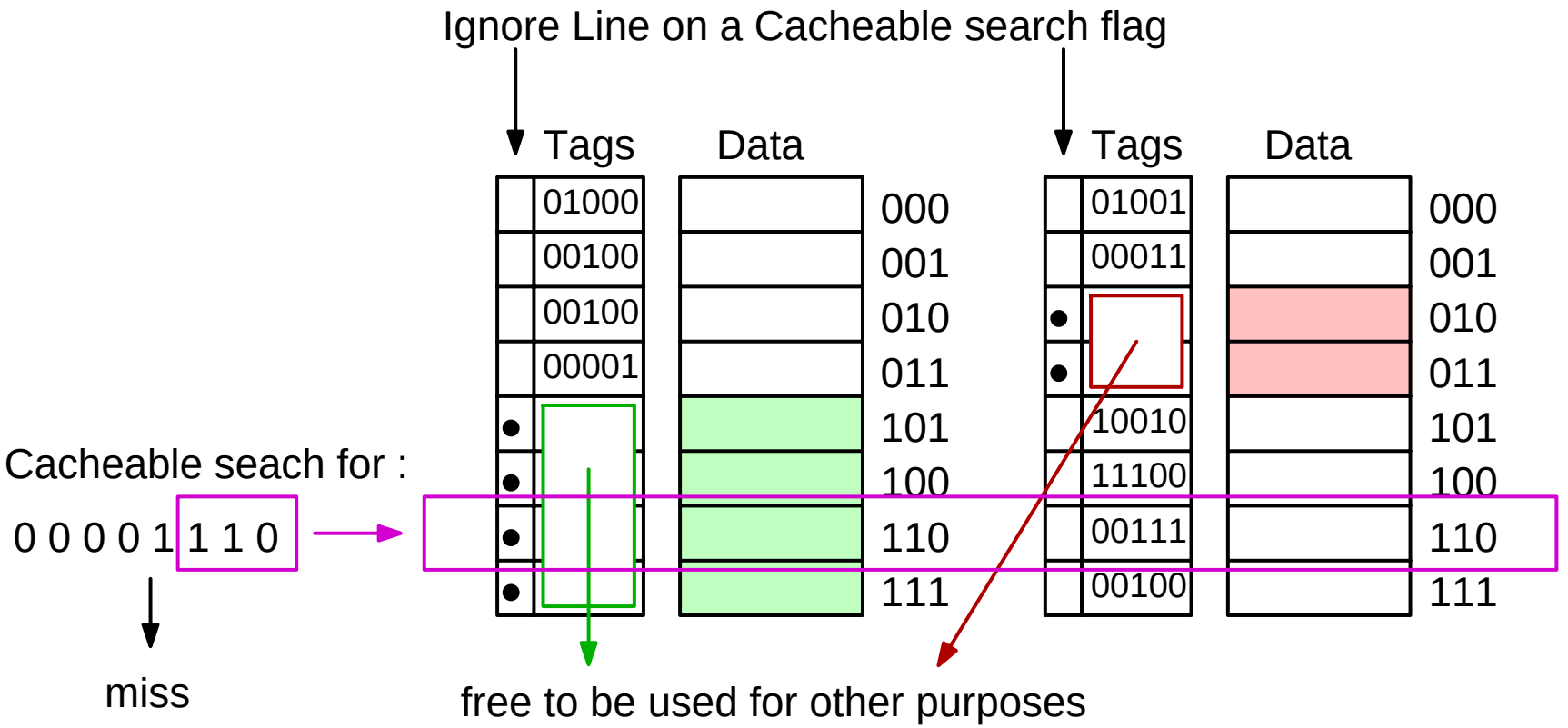
1. **Run-time Configurable Scratchpads**
2. Virtualized User Level DMA
3. FPGA Implementation - Results

Configurable Scratchpad Regions inside a Cache



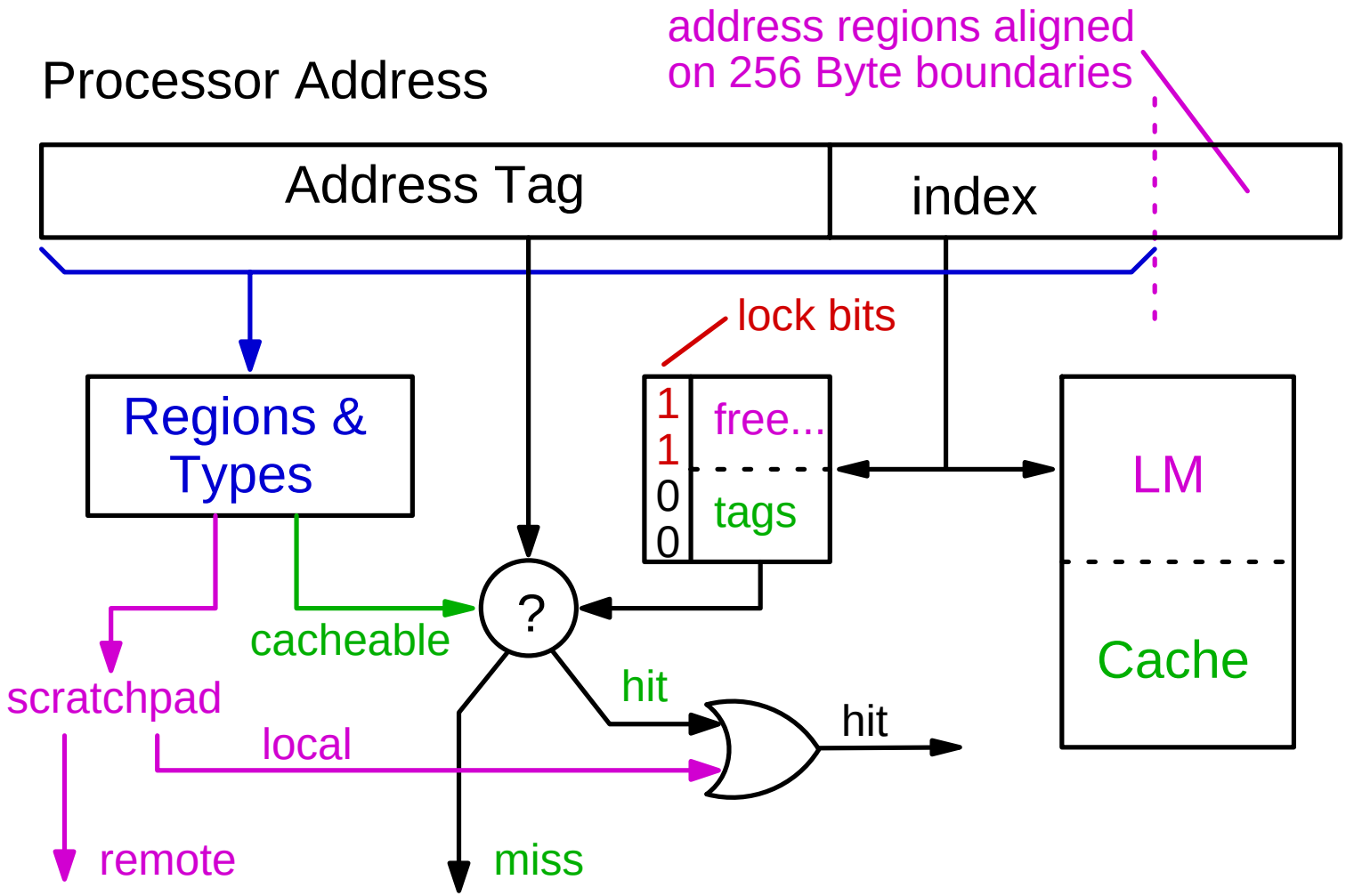
- physical allocation alignment in cache must be compatible with scratchpad address region alignment.
  - No translation of address index bits when accessing the cache
- a single Table entry suffices for a large contiguous region

Tags of Locked Lines are freed up



- set lock bits on all scratchpad cache-lines
  - comparators and replacement ignore these lines
- tag bits are freed – used for NI meta-data e.g. linked-list pointers

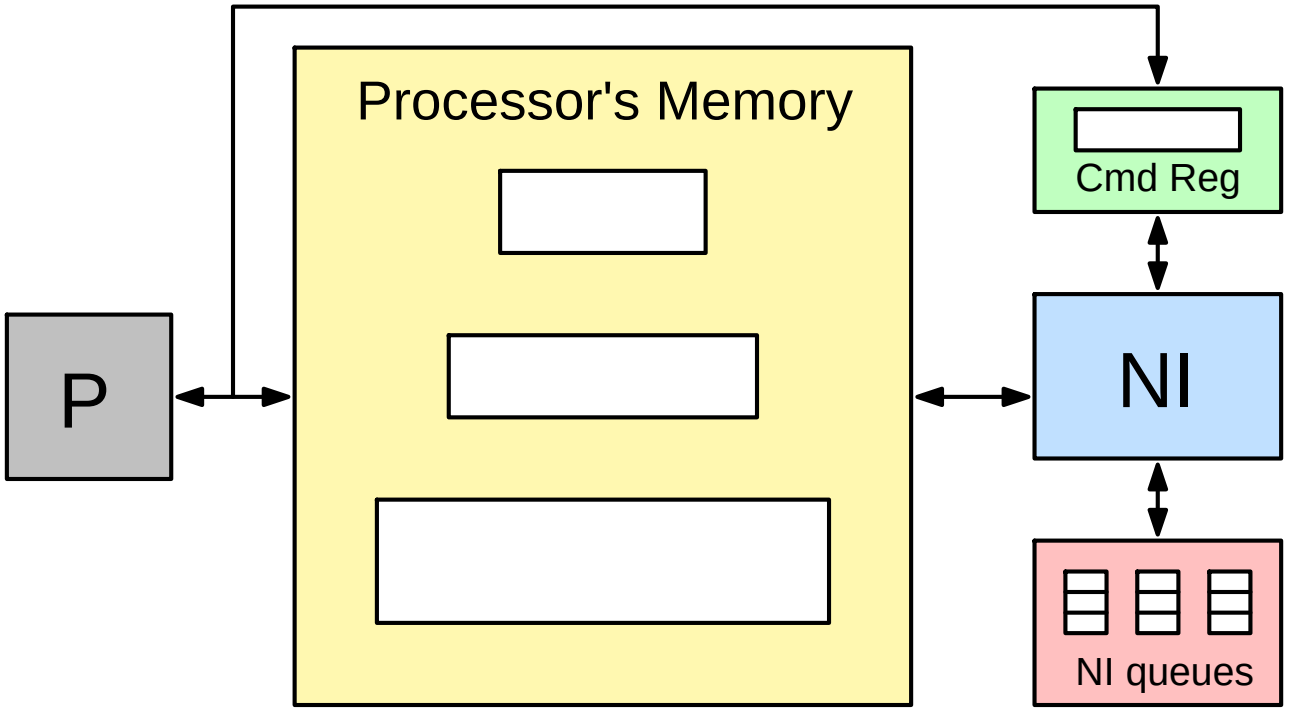
Memory Access Flow



Talk Outline

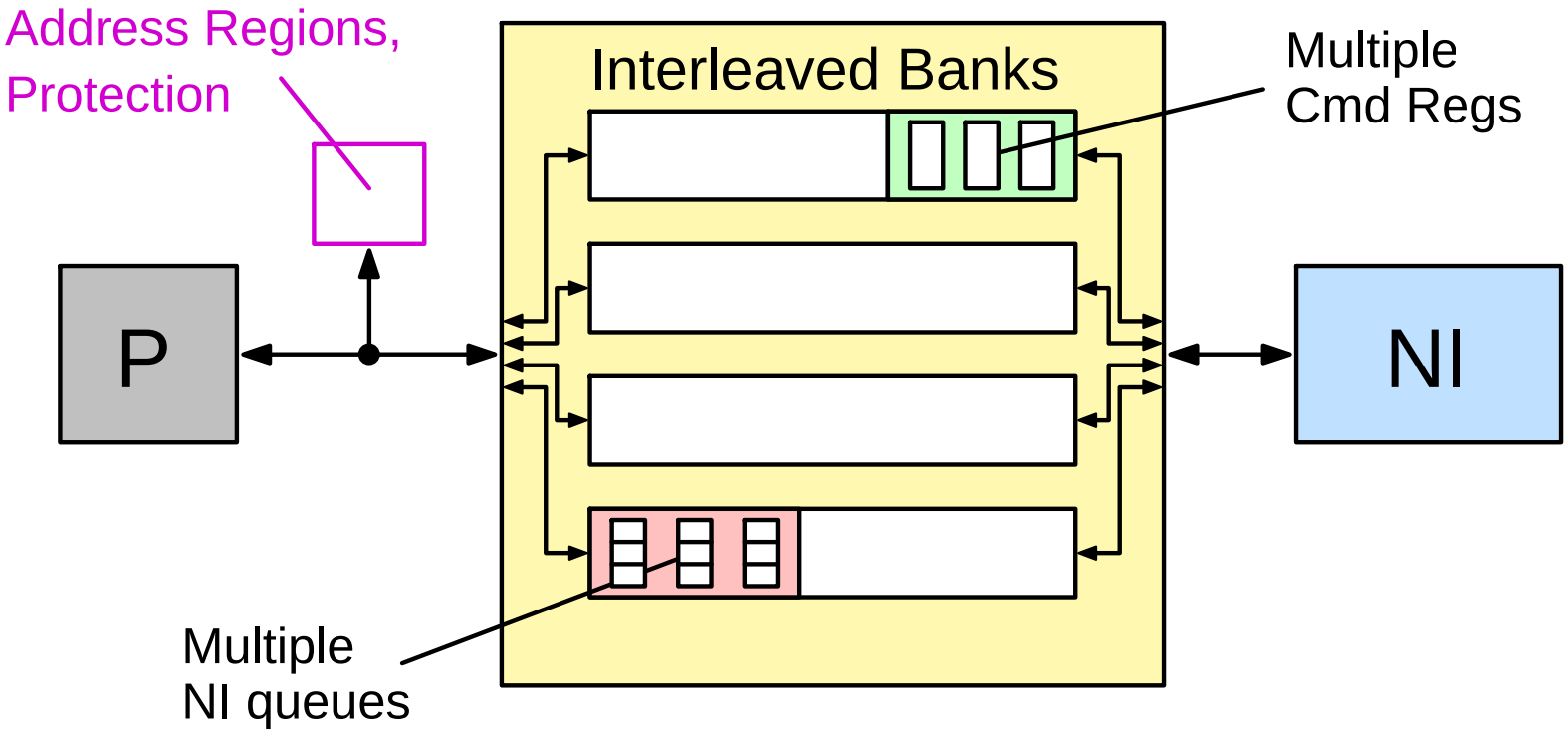
1. Run-time Configurable Scratchpads
2. **Virtualized User Level DMA**
3. FPGA Implementation - Results

Old Style: NI as a Peripheral (I/O) Device



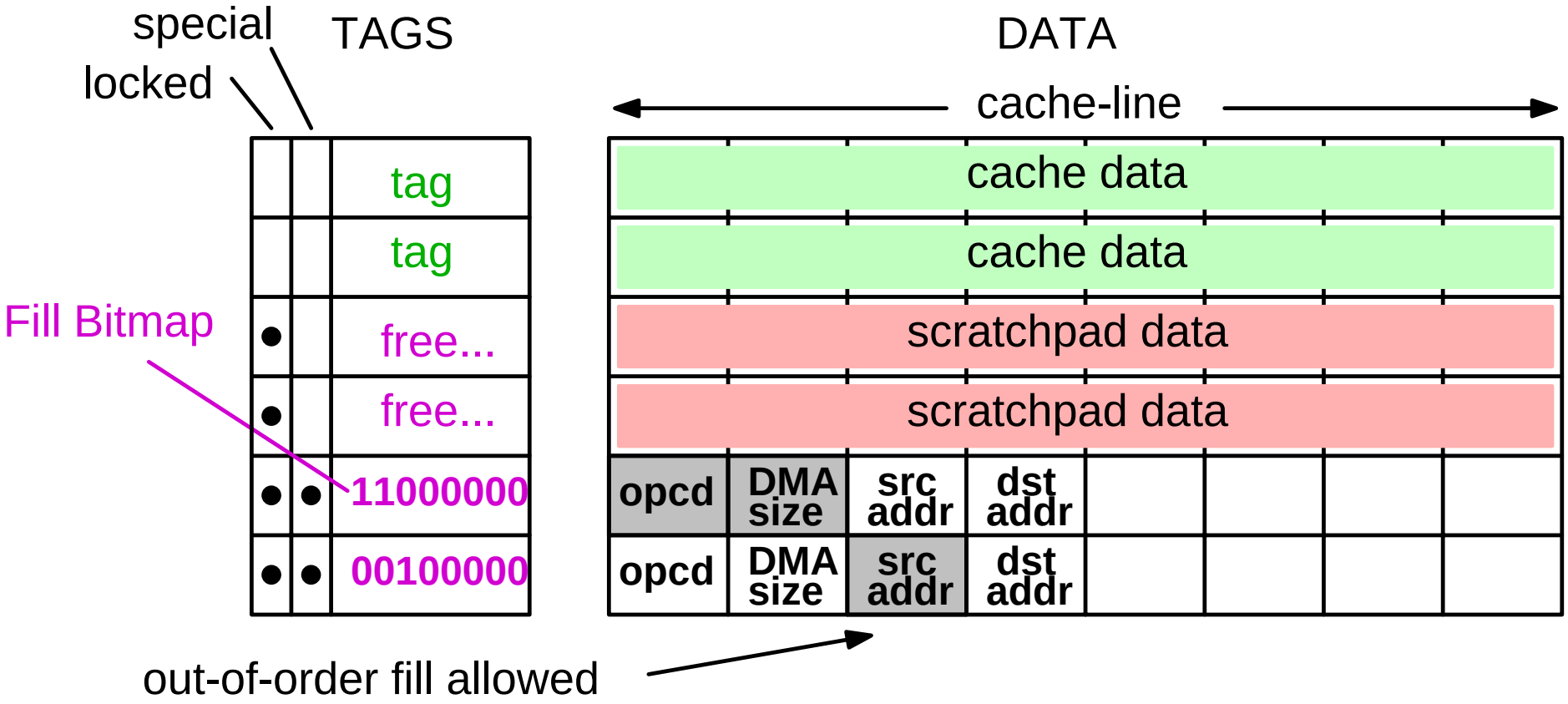
- fixed command registers
  - OS control (system call) needed, to share among processes
- dedicated memory for NI tables and meta-data
  - dedicating memories underutilizes all but one of them

New Style: Virtualized NI, tightly coupled to Cache



- SW allocates as many command buffers as desired
  - per domain (thread/process) allocation
  - protection table defines who has access to what
  - allows for asynchronous, user-level accesses to NI commands
- interleaved banks: sufficient bandwidth for P and NI accesses

NI Command Buffers



- allocated inside scratchpad areas; share the same ART entry
- DMAs issued as a series of stores ⇨ low overhead, latency
  - store-once, even OoO; monitor mod's to trigger completion

Talk Outline

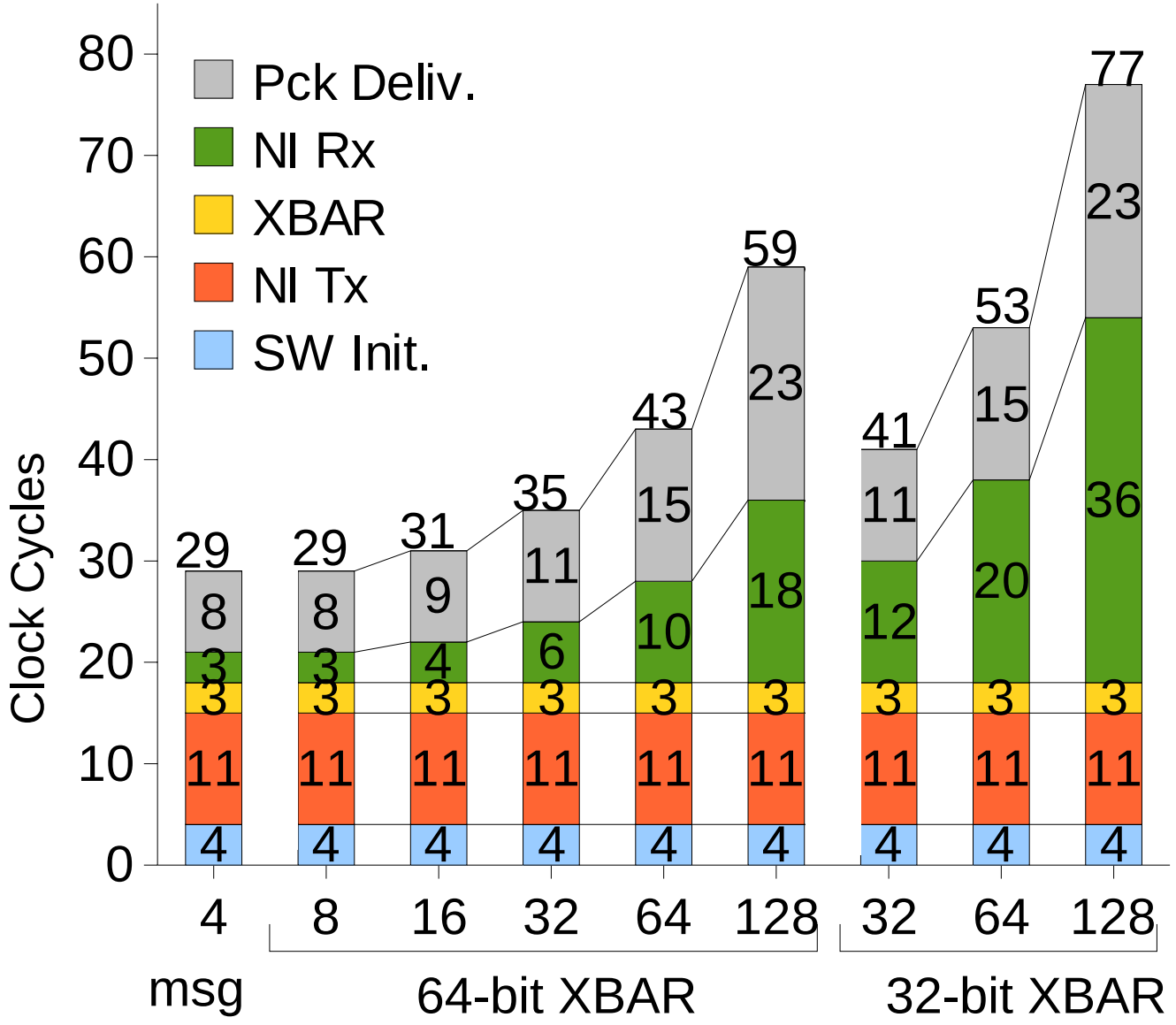
1. Run-time Configurable Scratchpads
2. Virtualized User Level DMA
3. **FPGA Implementation - Results**

FPGA Prototype

- Implementation currently in a Xilinx Virtex II-Pro FPGA
- First prototype includes:
  - two 32-bit MicroBlaze in-order soft-processors (4 P's in Virtex-5)
  - 3-port, 64-bit crossbar as NoC
  - controller for off-chip DRAM (256MB DDR SDRAM)
  - per processor:
    - private L1 cache: 16KB, write-through, 64-bit fill path
    - private L2 cache: 64KB, write-back, L1 inclusive, featuring integrated cache controller and our custom network interface
  - no cache-coherence yet …
  - currently operates at 50MHz
- Read Hit: 1 clock (L1), or 4 clocks (L2)
- Write Hits: Pipelined, at 1 word/clock for both L1 and L2

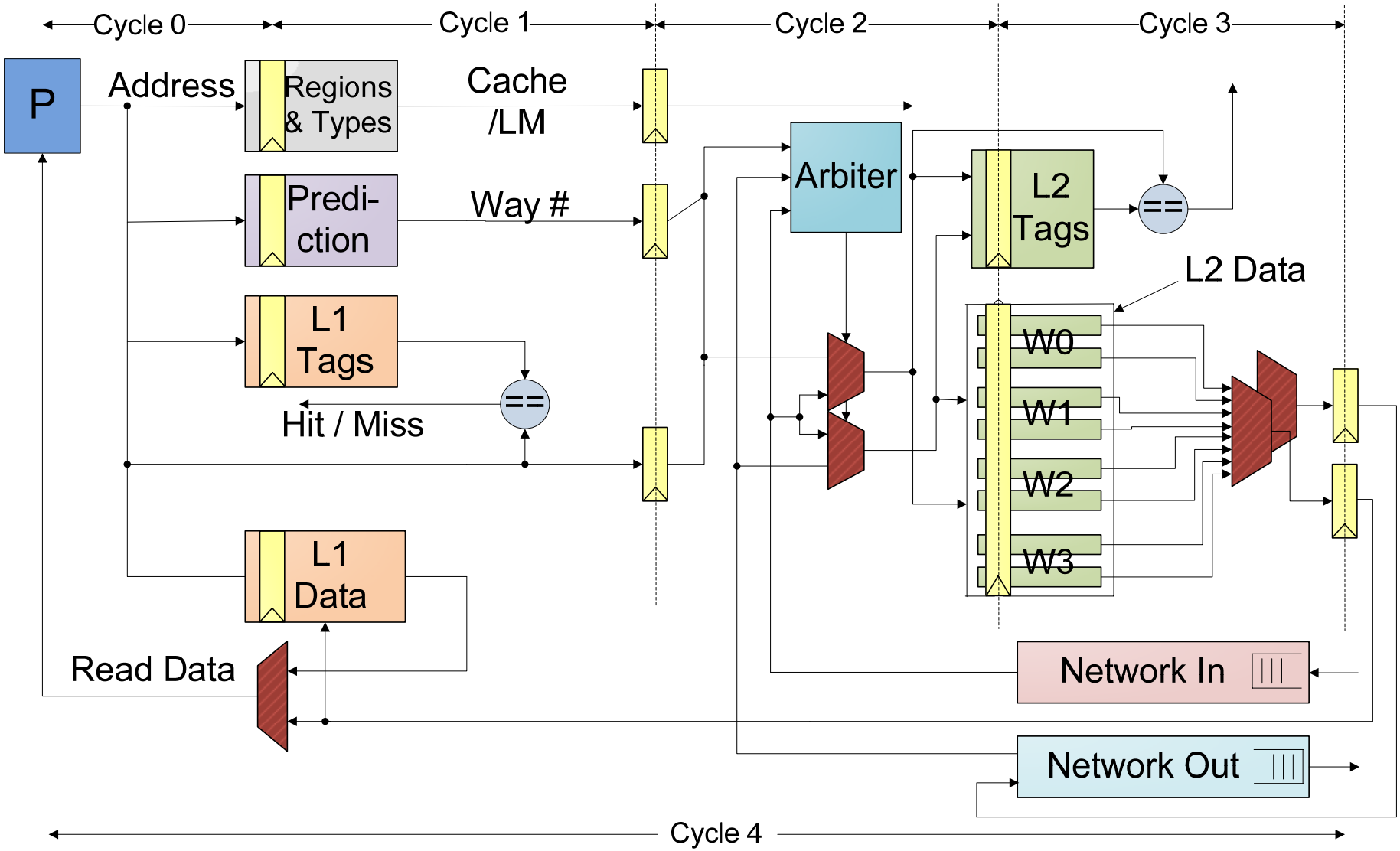
NI Latency

- End-to-end minimum sized DMA costs 29 cycles
- Cut-through Tx (red & yellow)
- Store&Fwd Rx
  - CRC check (green, grey)

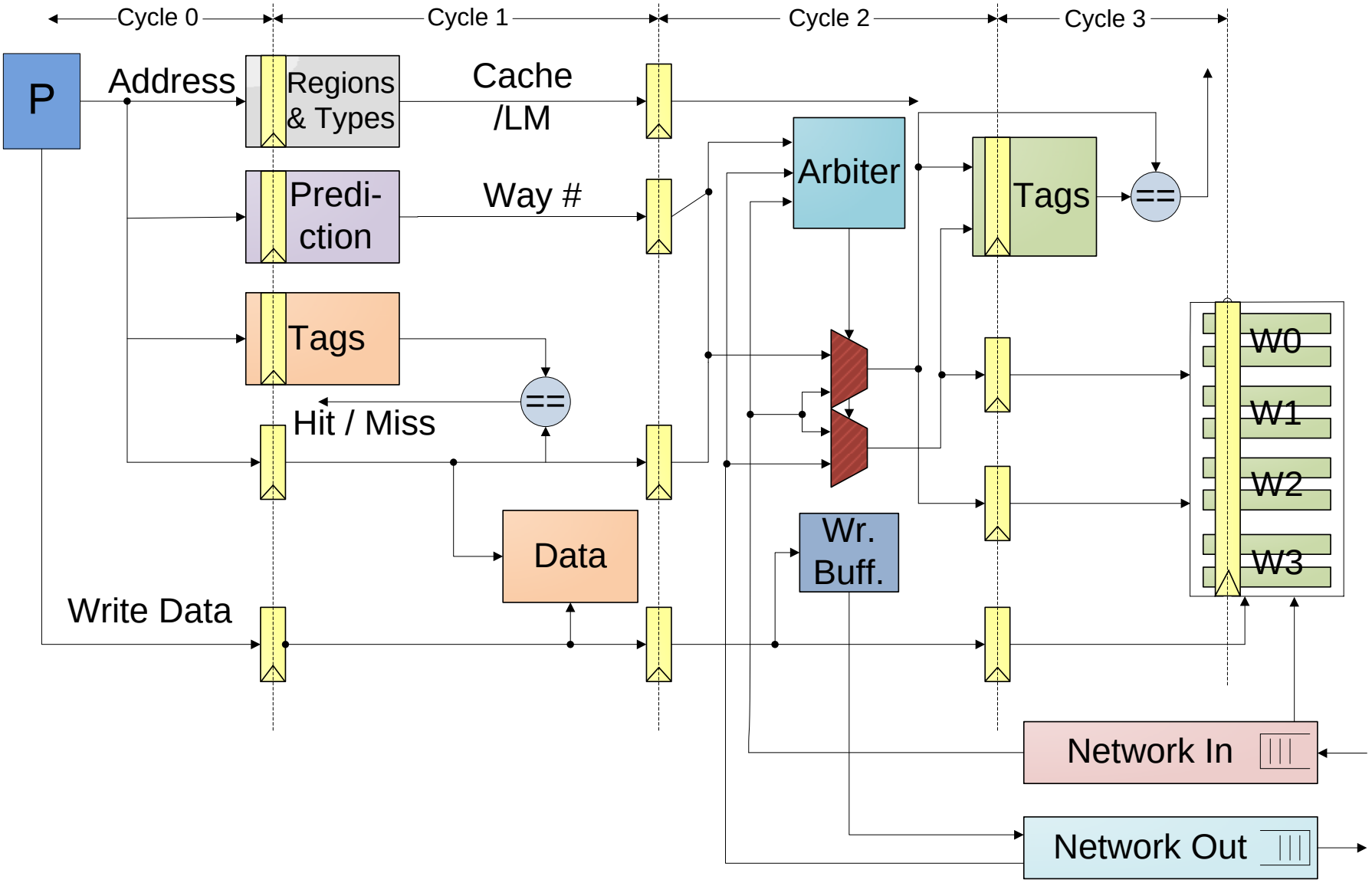


Latency vs. Transfer Size (bytes)

Design: Read Path



Design: Write Path



Circuit Cost

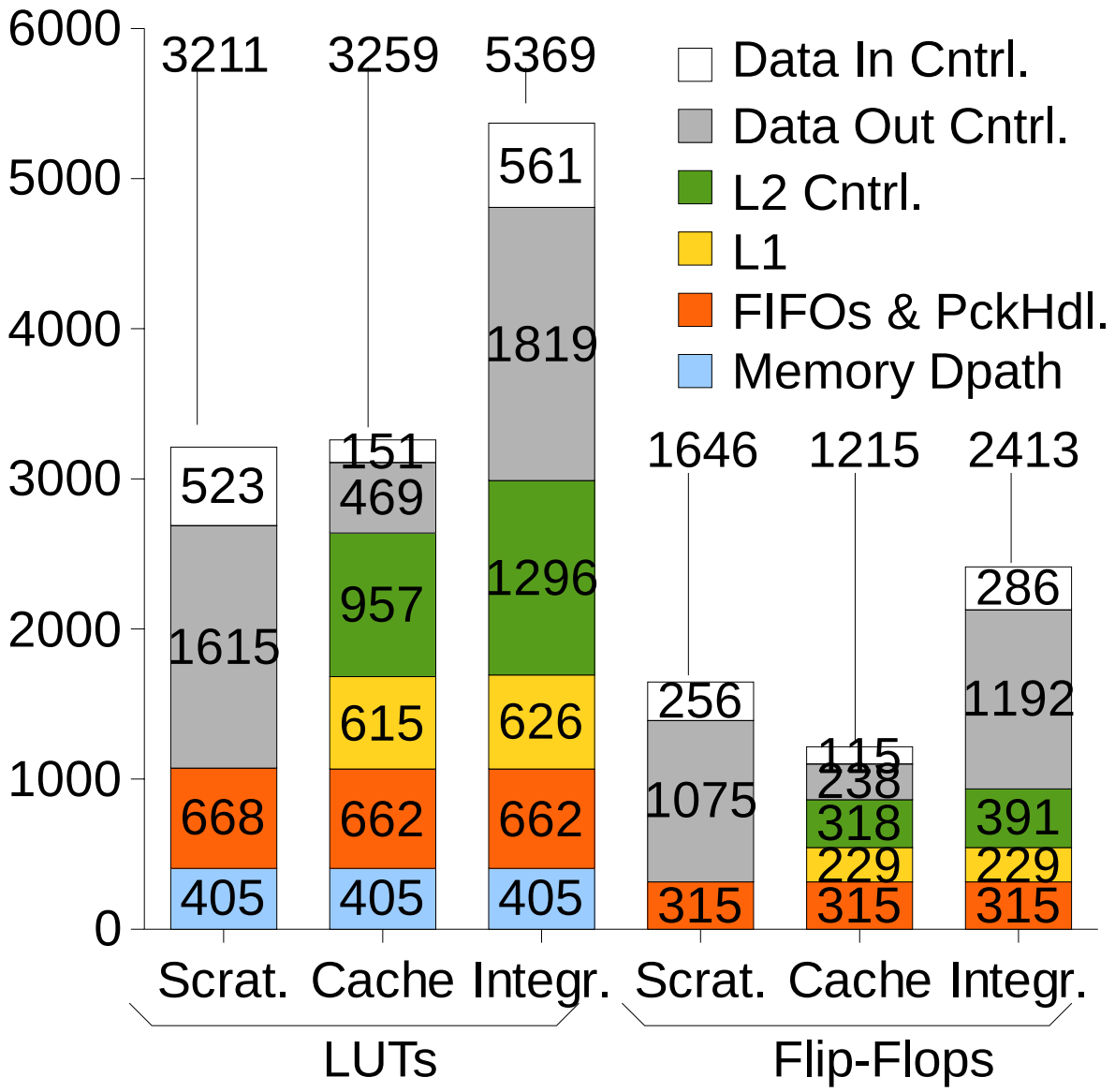
Comparing:

1.Scratch.&DMA

2.Cache only

3.Integrated

(does not include SRAM arrays)



L2 Cache Controller & NI: Cost & Sharing

- Common datapath used by Cache and NI DMA controllers
  - 2 architectural ports on 8 interleaved memory banks
  - 3 competing agents (Processor/Cache, Outgoing NI, Incoming NI) ⇨ 3 x 2 switch
- Data movement controllers of Cache and Scratchpad have not been unified yet …
  - initial design shows a 7.5% overlap
- Adding advanced features to the cache (coherence, multiple outstanding misses) and careful merging with NI is likely to increase the sharing of resources

Conclusions

- Local SRAM Configurable as Cache/Scratchpad
  - lock cache lines against eviction:
    easy to implement, helps towards Deterministic Computing
  - make cache lines addressable by user programs: allows data delivery directly into local memory ⇨ locality optimization
- Unified Cache Controller / DMA Controller / Network Interface
  - shared datapath for accesses by (i) Processor, (ii) Network
  - accesses to lines can have side-effects, depending on tag/state:
    - cache miss handling
    - virtualized NI command registers (next: counters, queues)
- FPGA Prototyping
  - reasonable cost (few K LUTs, K FFs)
  - low latency communication (few tens of clock cycles)