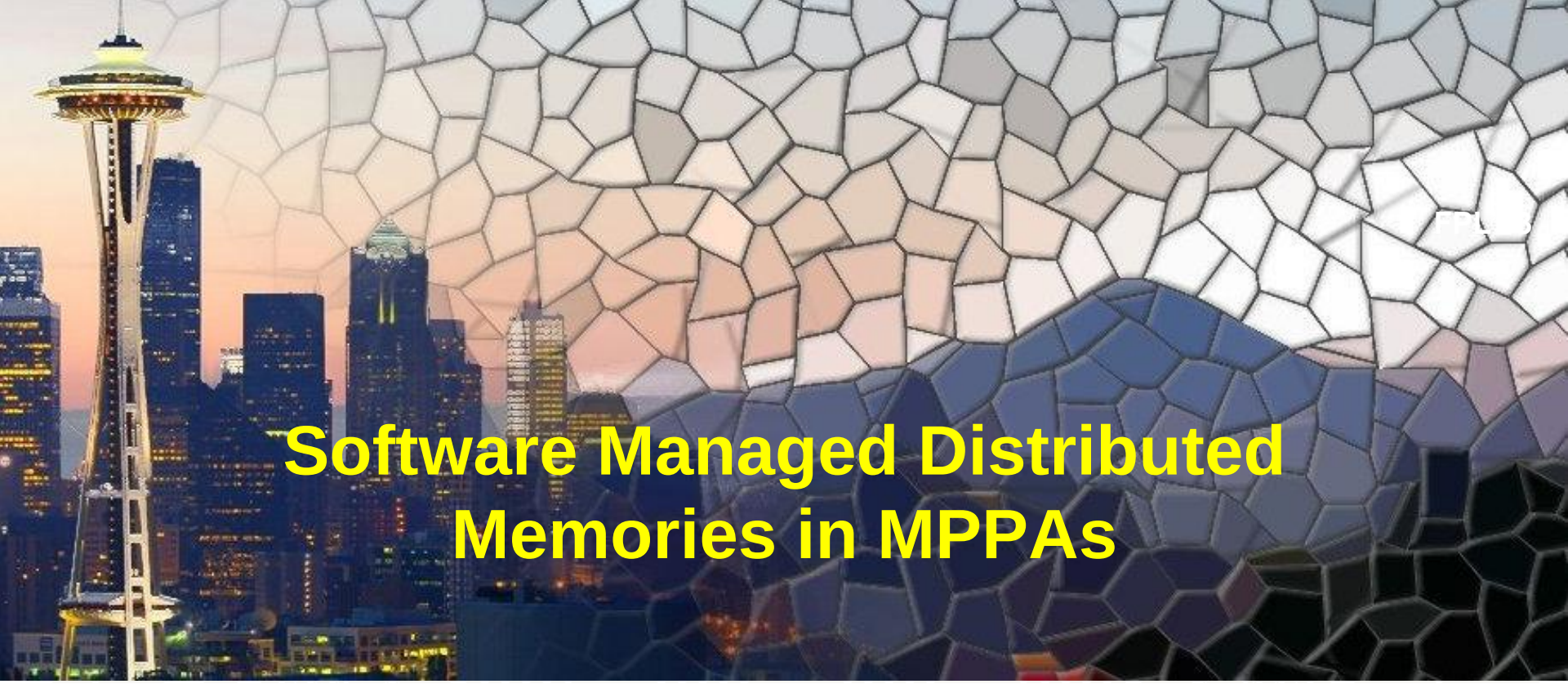


Software Managed Distributed Memories in MPPAs

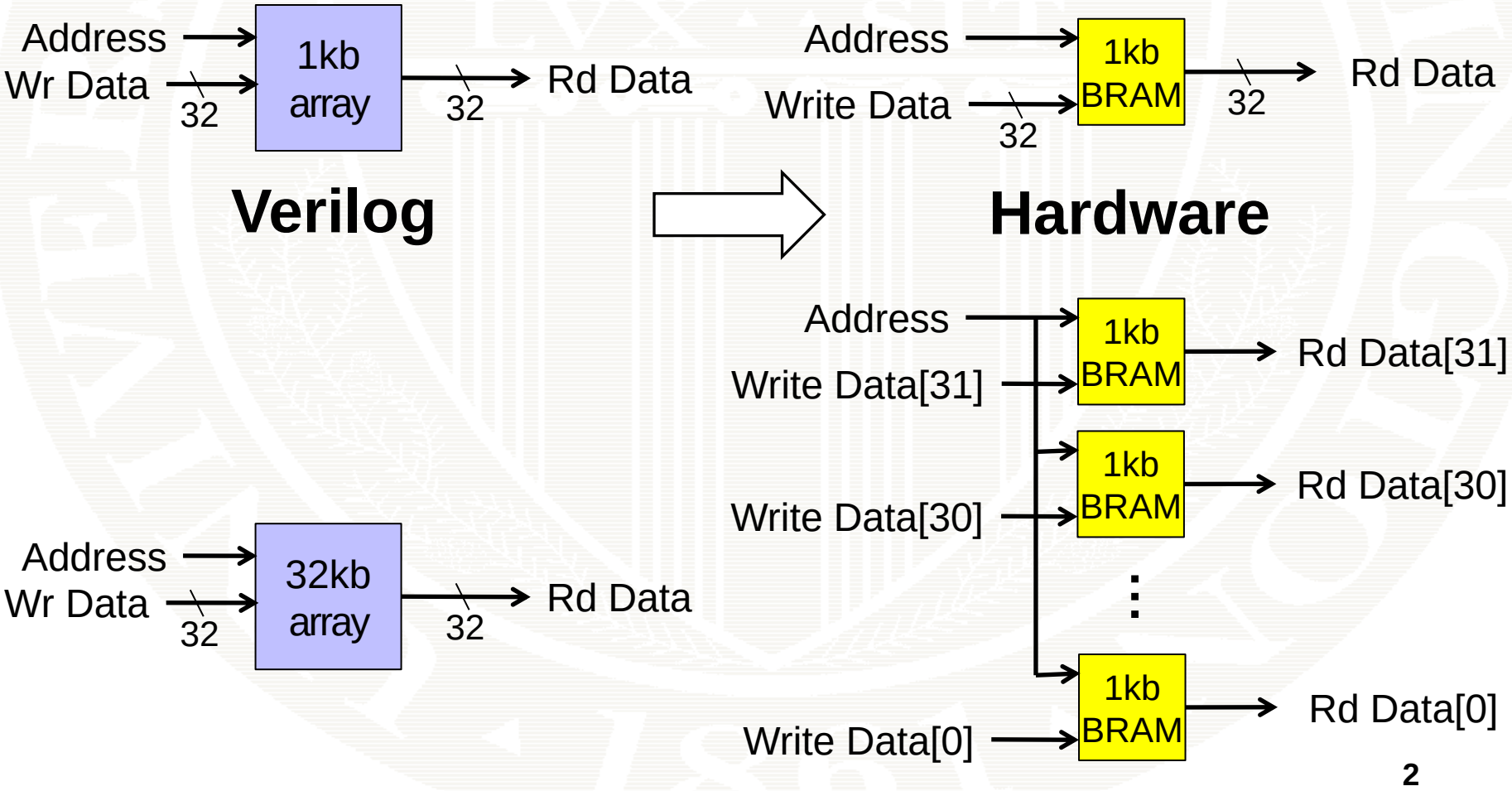
Robin Panda, Jimmy Xu, Scott Hauck

University of Washington
Department of Electrical Engineering

Why distributed memories

- Variety of applications requires a variety of memory sizes
- Provide many different memory block sizes
  - Infeasible / inflexible
- Split large memories
  - Port contention
- Build up small memories

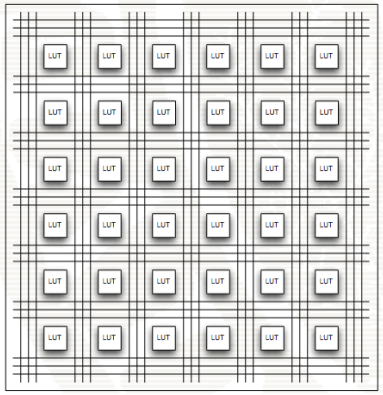
FPGAs compose large memories



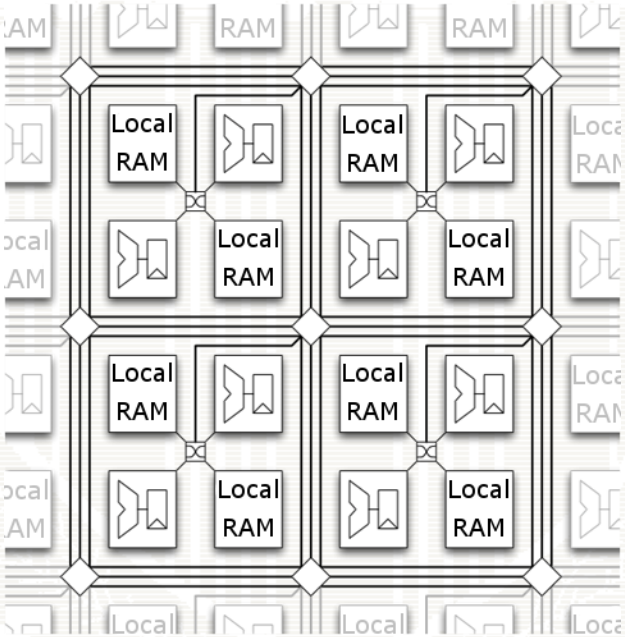
… and do it well

- Simple – built into tool flows
- FPGAs are naturally great at glue logic
  - Minimal logic unit overhead
  - Delay grows slowly as memory increases
  - Slowdown easily pipelined away
- What about word-based CGRAs and MPPAs?

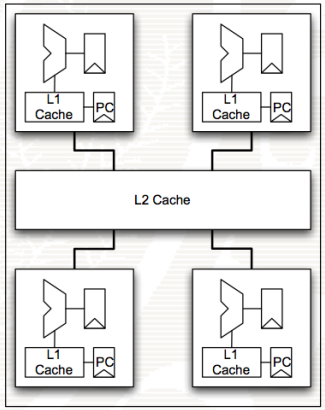
Massively Parallel Processor Arrays



FPGA



MPPA



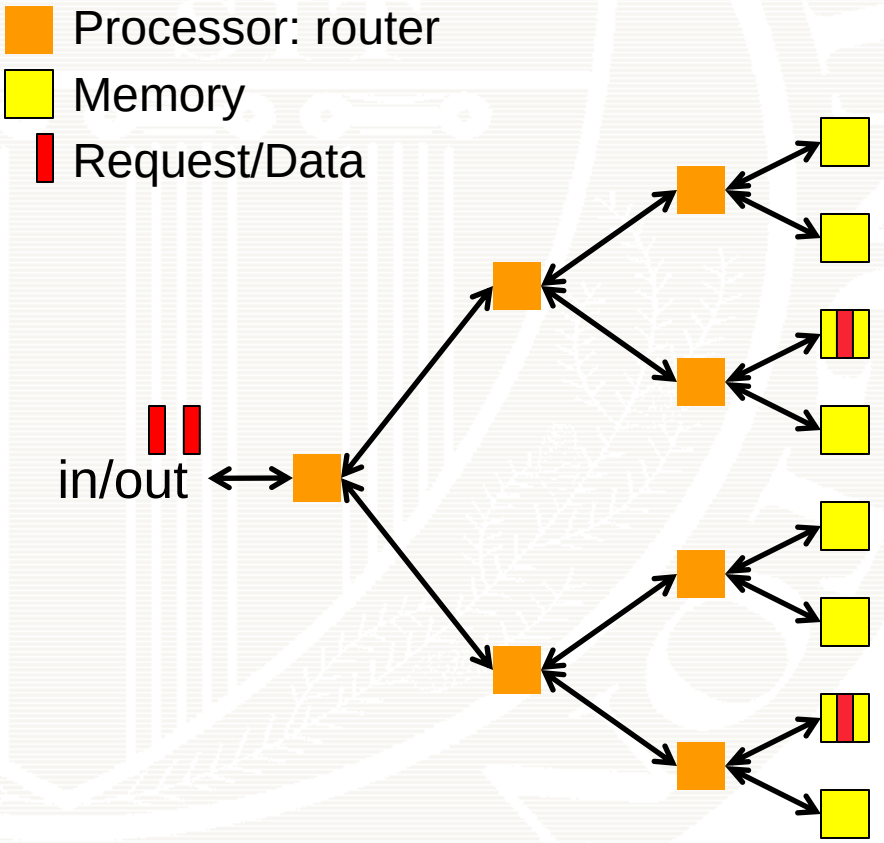
Multi-core CPU

Challenges

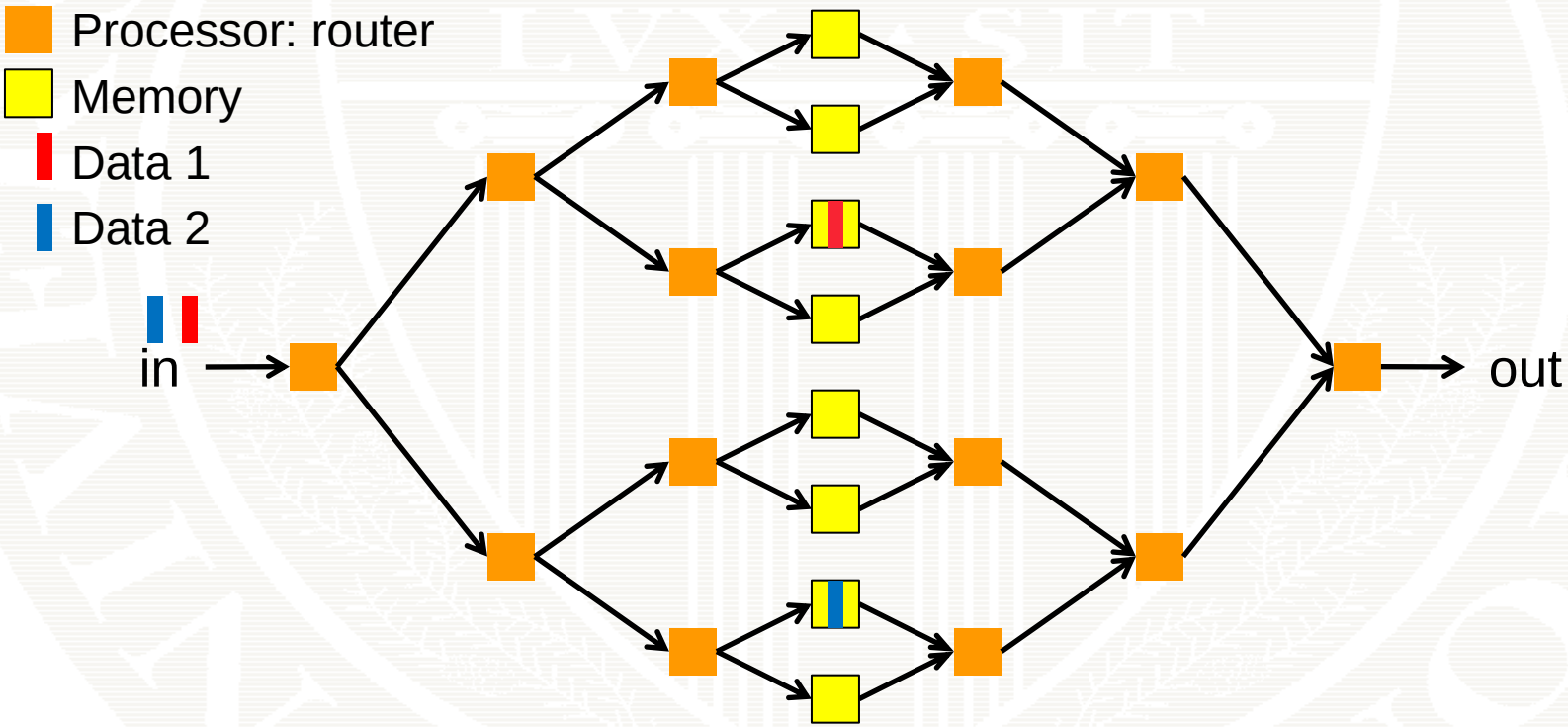
- Throughput
  - Code efficiency & network
- Synchronization
  - GALS & virtual channels
- Resource use

Simple tree network

- Latency = time to traverse network
- Throughput = transactions / sec
- No parallelism: max throughput = 1 / latency

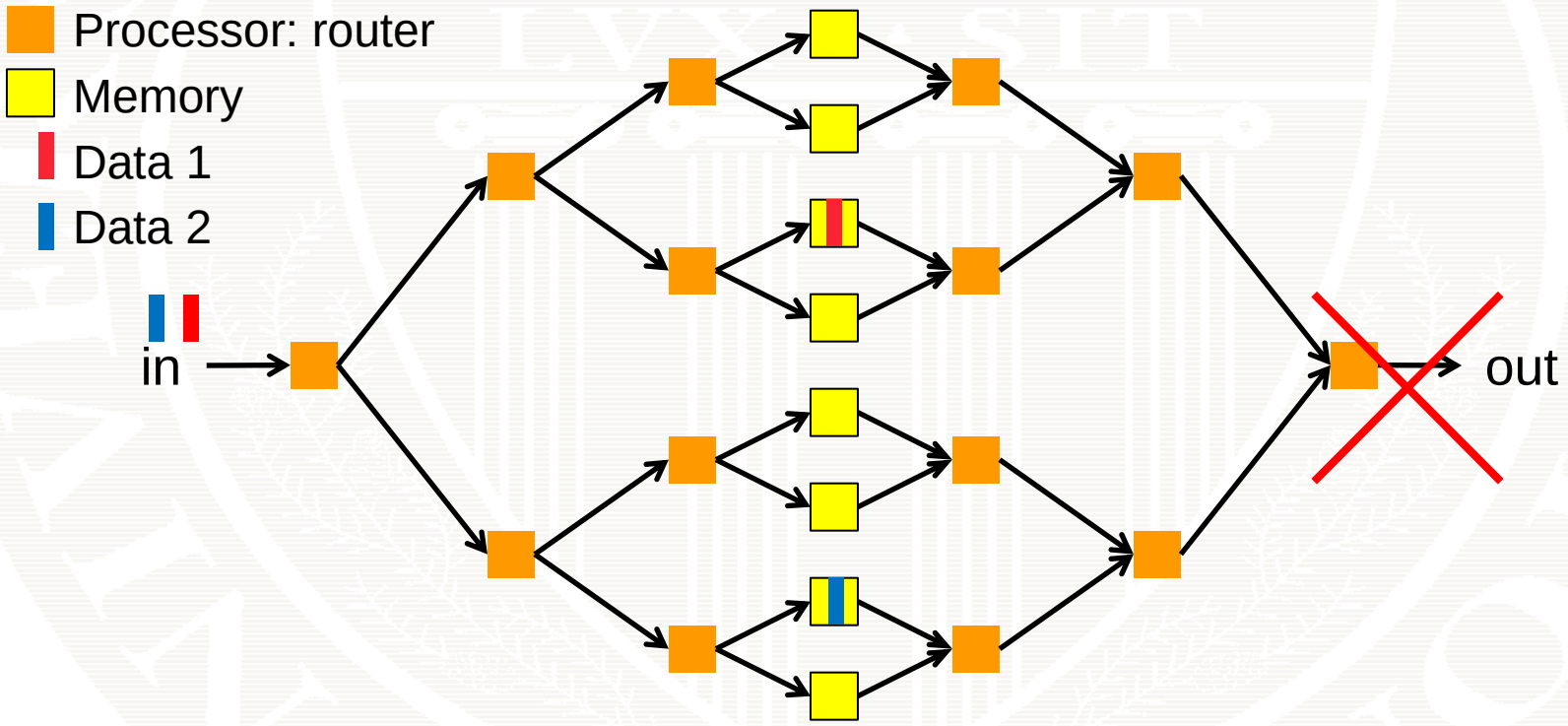


Duplicate network for pipelining



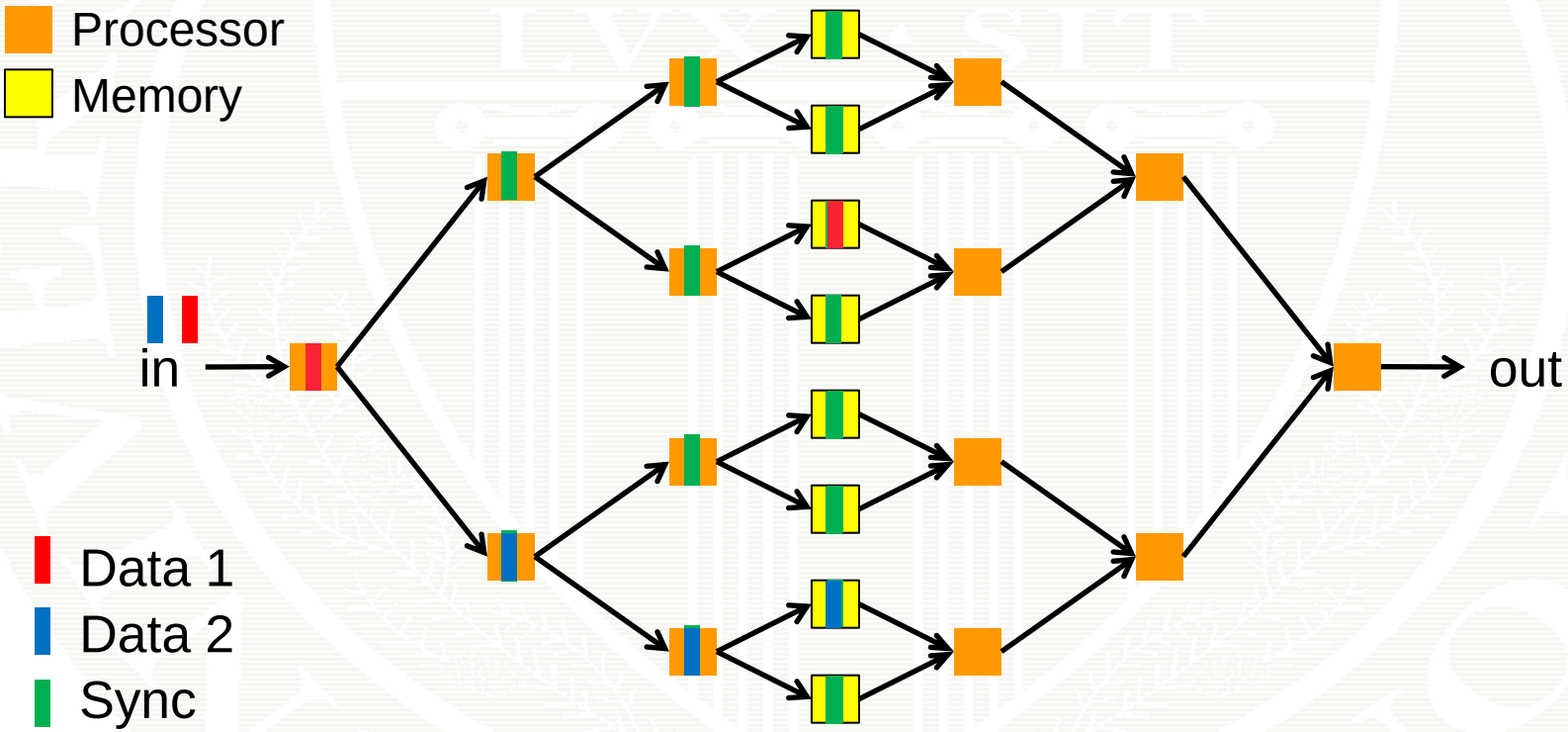
Greatly increases throughput

Asynchronicity: out of order response



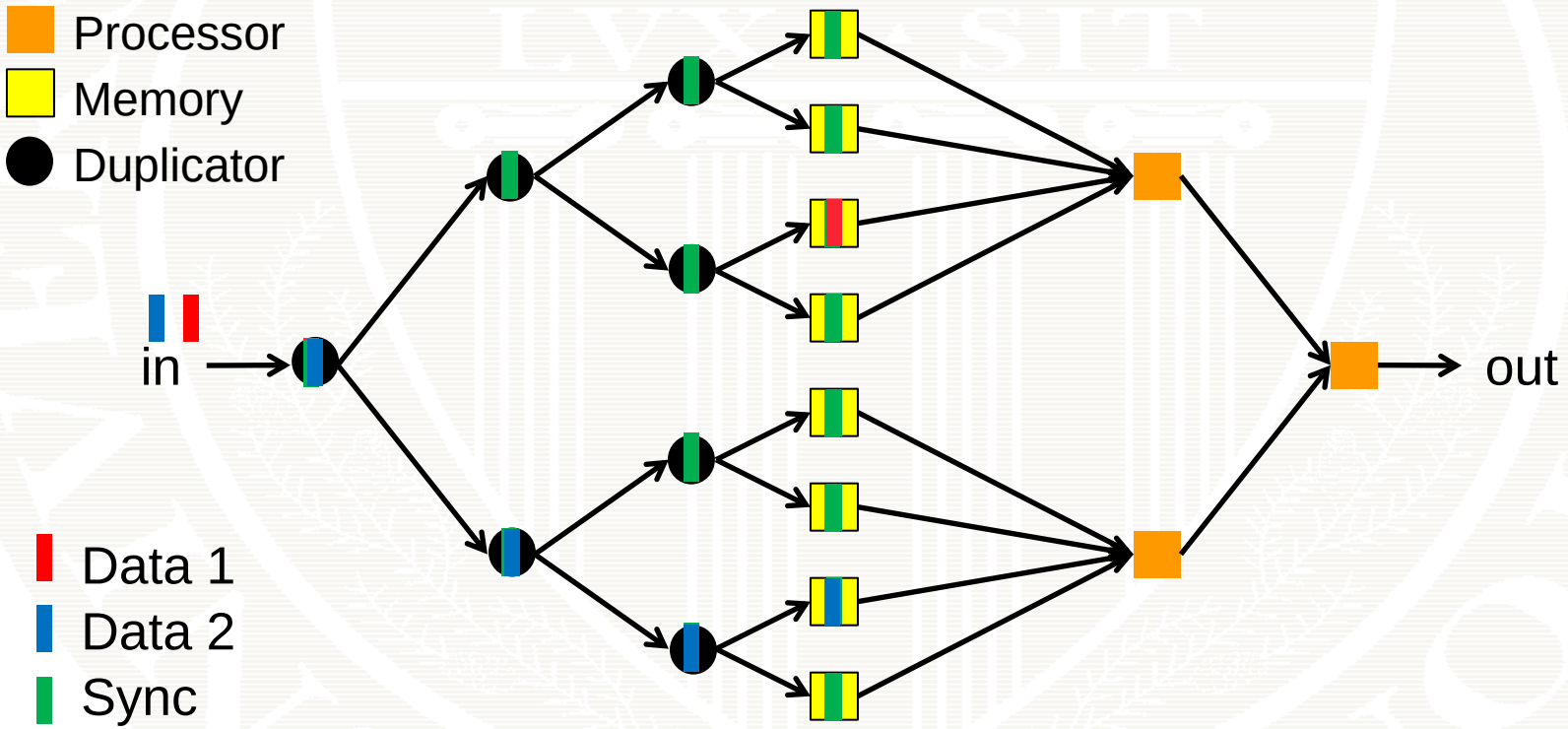
How to ensure proper functionality?

Synchronization packets



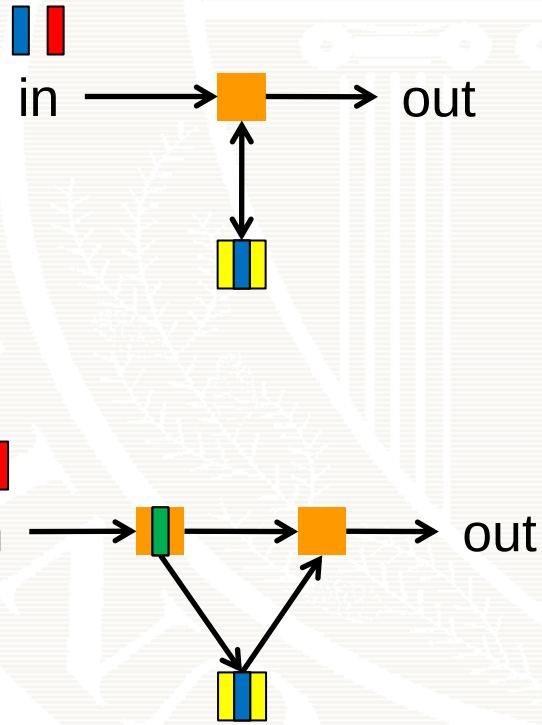
Later processors will delay fast data

Router reduction



Use full hardware capability to reduce overhead

Who controls individual memories?



Pipeline at all levels

Processor
Memory
Data 1
Data 2
Valid

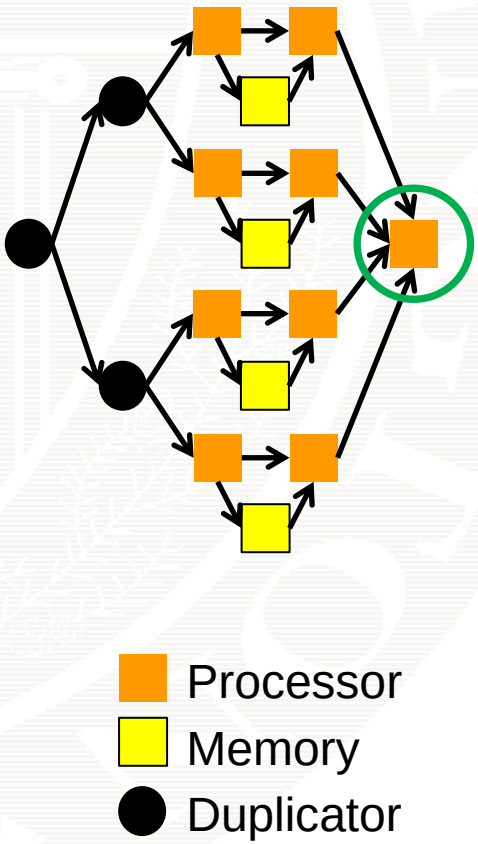
Optimizing code for Ambric

- Chose network protocol for easy decode
  - Minimize bit processing required
- Write in assembly
  - Care for delay slots
  - Use hardware looping
  - Use muxing instructions
  - Use complex instructions

Merge processors

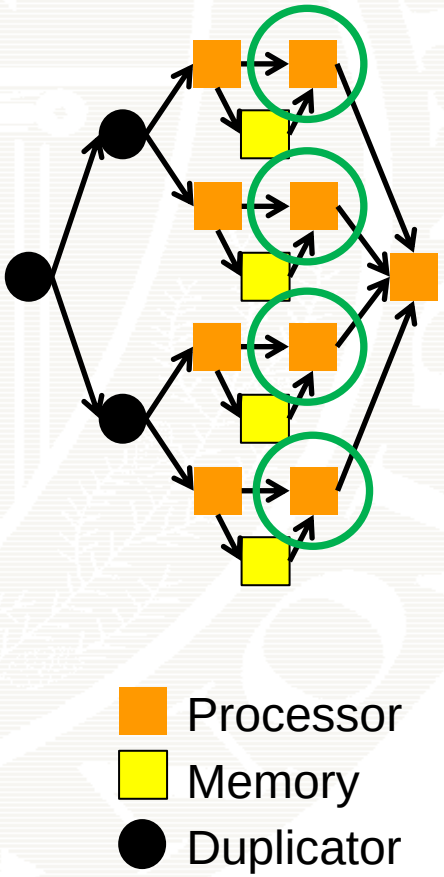
```
   loopalways END

   mov in0, r1, sink
   or in1, r1, sink
   or in2, r1, sink
END:
   or in3, r1, sink, out, 0
```



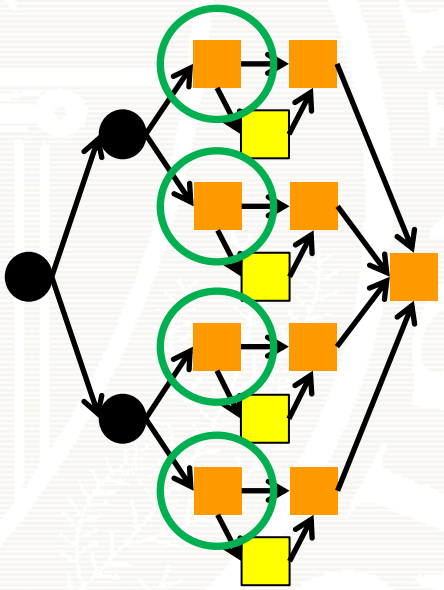
Data processors

```
        ; read from address processor
mov cross, sink
        ; mux based on prev. instruction.
        ; ones is “FFFF”
mux zero, ones, r1
        ; send data or sync packet
and memR0, r1, sink, out, de0
```



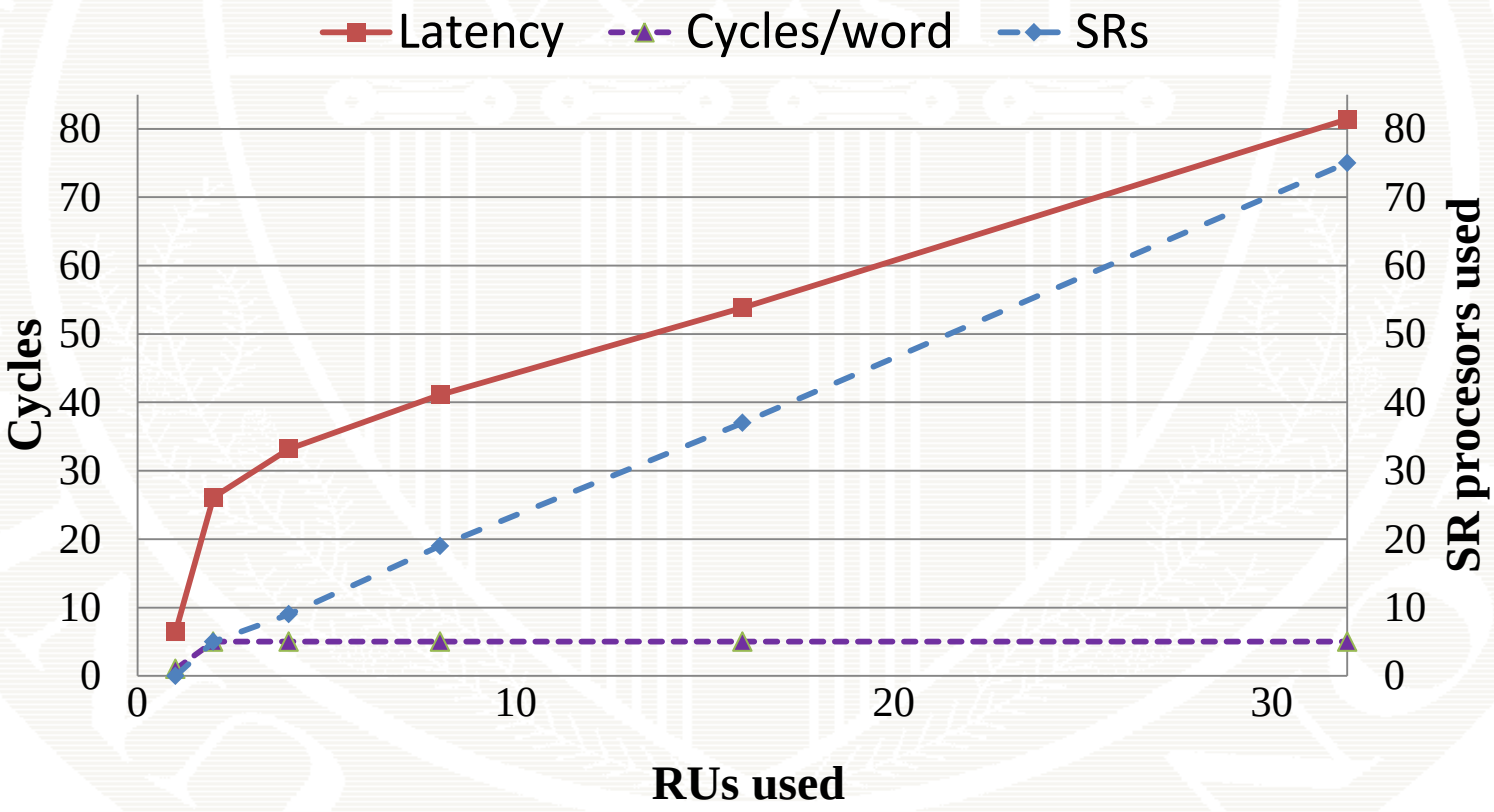
Address processors

```
        ; read in word
    mov Input, r1, sink
        ; branch if write.  brde has no delay slot.
    brde WRITE
        ; apply local address check
    xor address, r2, sink
        ; send read request to memory
    and r1, mask, sink, memW0, de0
        ; send mux select to data processor
    and r2, notmask, sink, cross, de0
WRITE:
    …
```

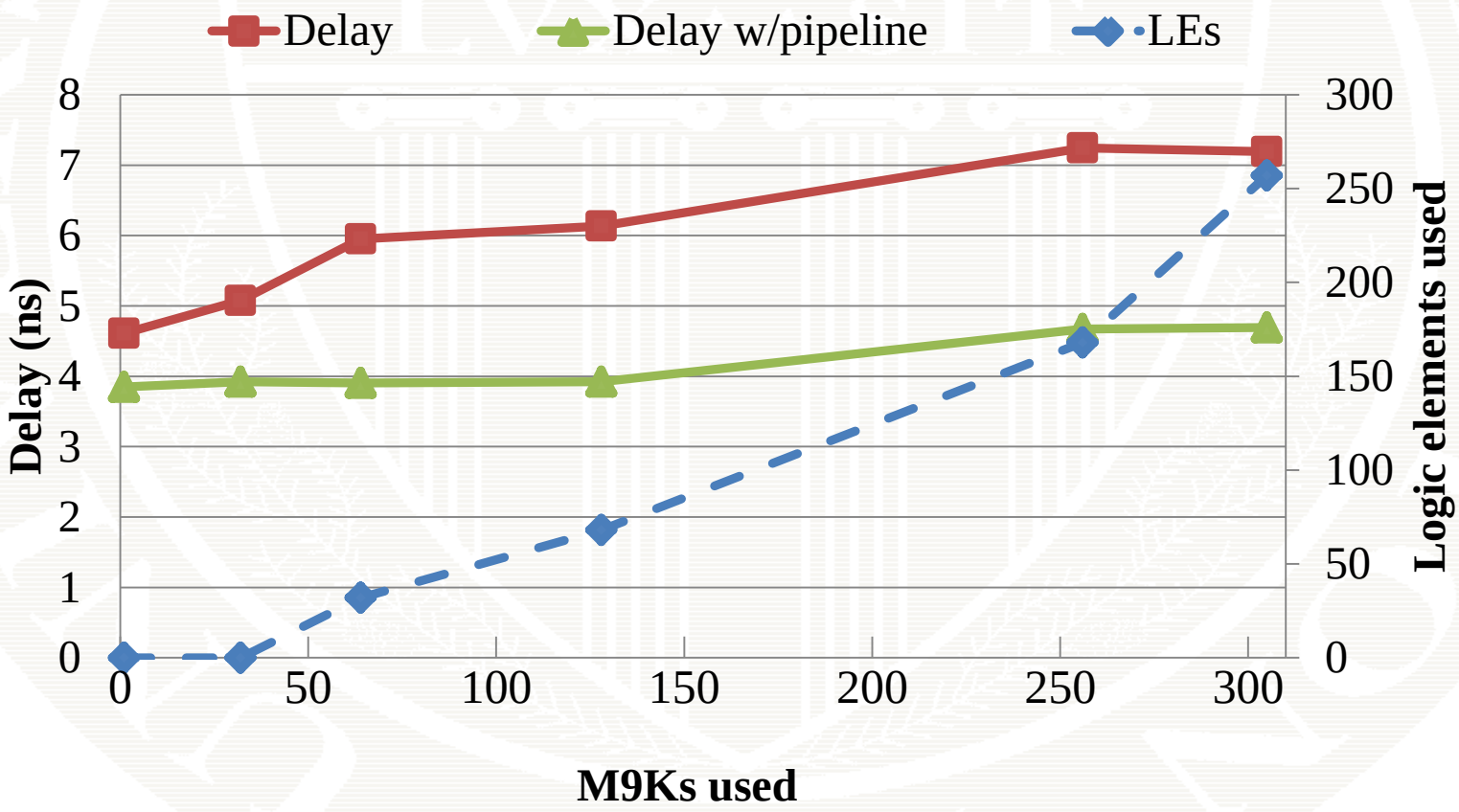


Processor
Memory
Duplicator

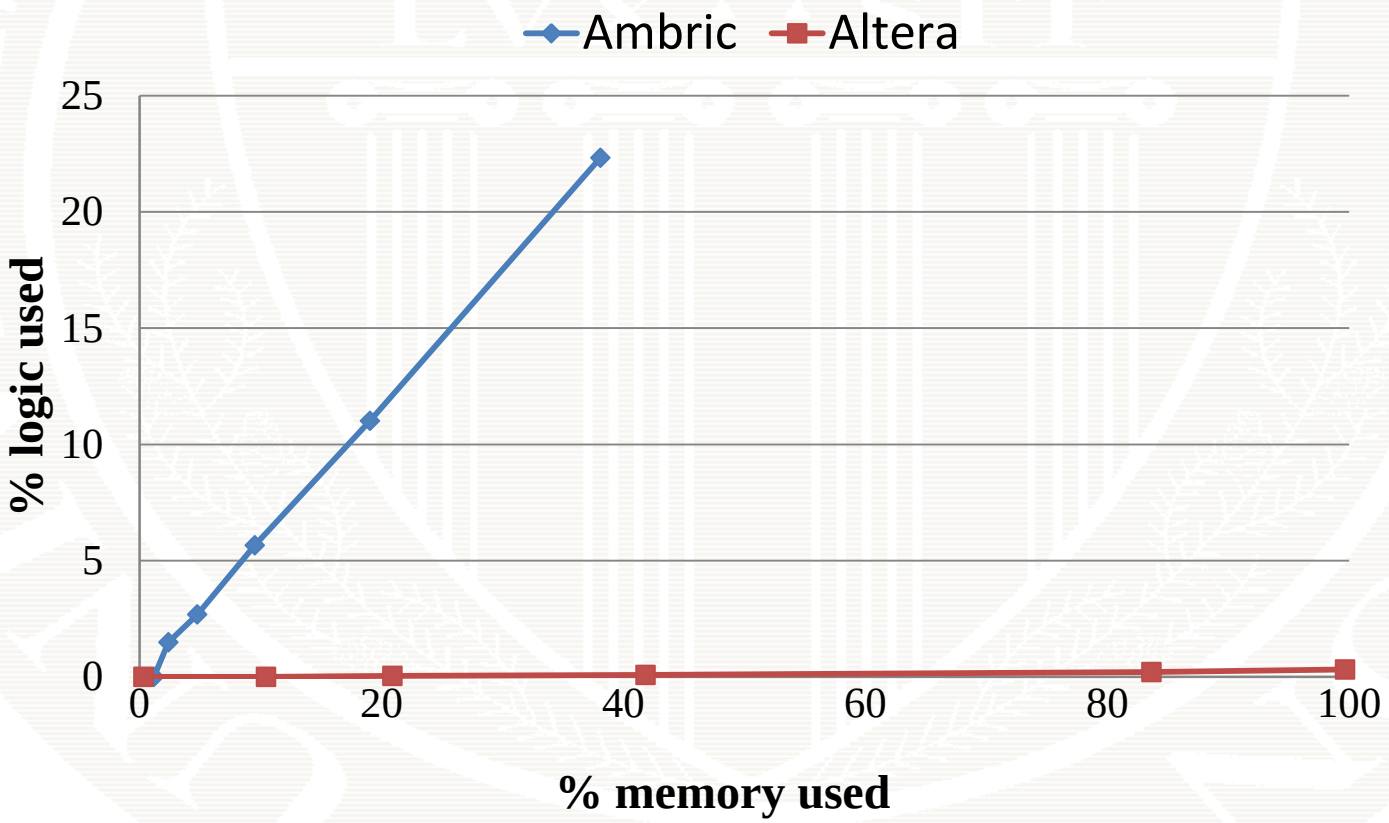
Ambric memory performance



FPGA memory performance

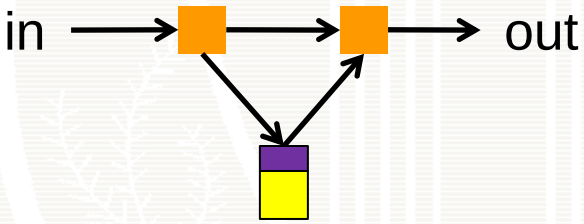


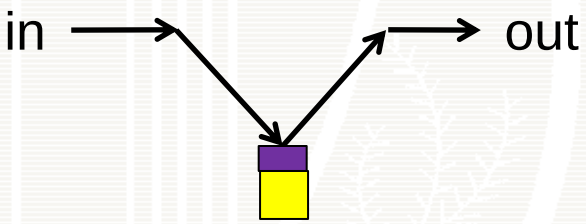
Resource overhead comparison



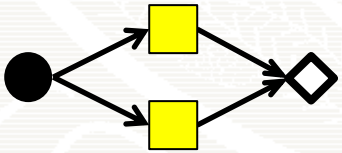
Recommendations

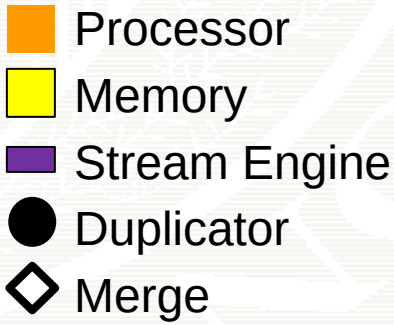
- Add address mask/match to stream engines



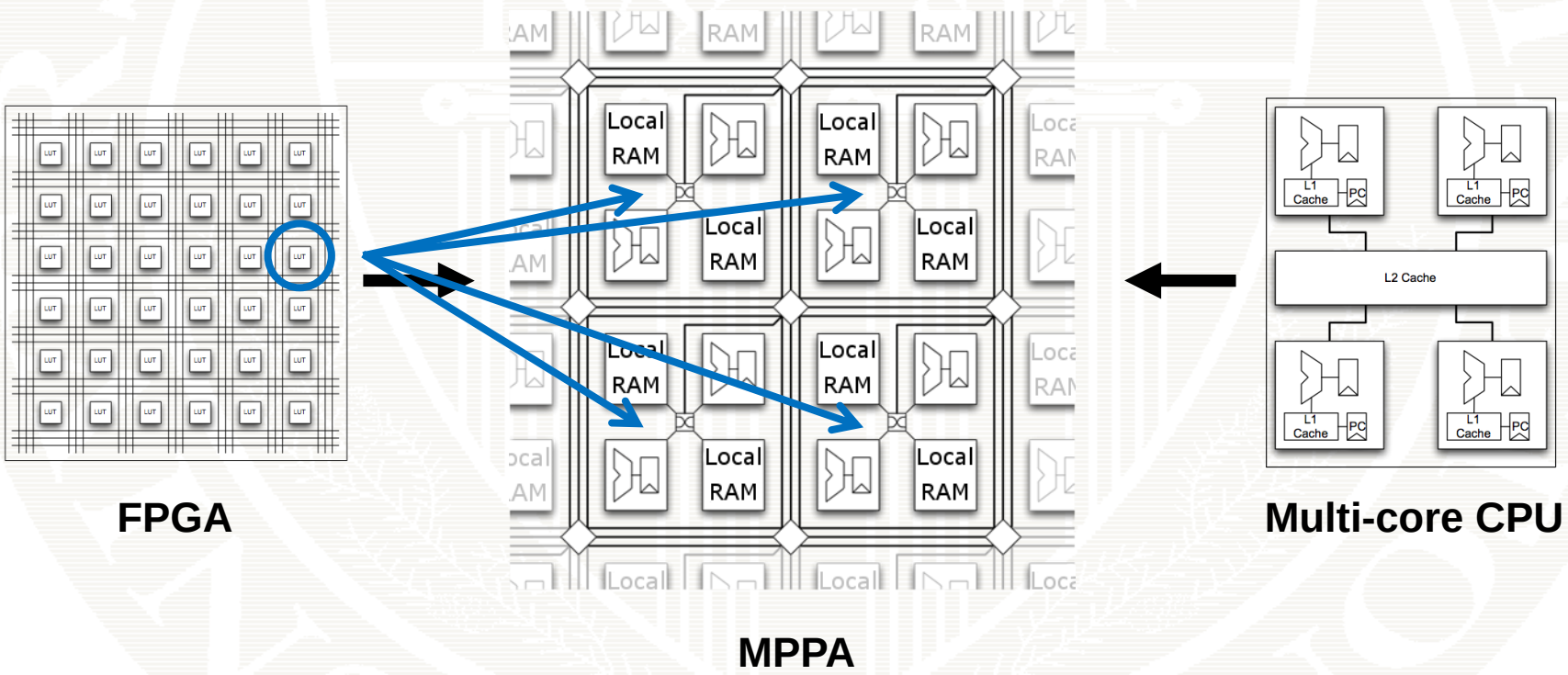


- Interconnect merges





Or add single bit capabilities



FPGA

MPPA

Multi-core CPU

Allowed for area **reduction** in Mosaic project.

Takeaway

- Can automatically support large memories with good throughput
- Memory partitioning is bit-oriented and thus, inefficient on word-oriented accelerators
- Architectural enhancement would help greatly