Optimizing Under Abstraction: Using Prefetching to Improve FPGA Performance

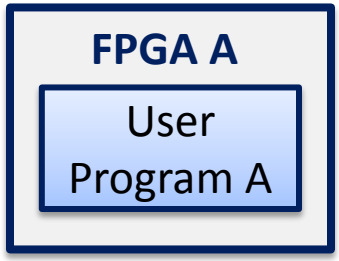
Hsin-Jung Yang[†], **Kermin E. Fleming**[‡], **Michael Adler**[‡], **and Joel Emer**[†‡]

[†] Massachusetts Institute of Technology
[‡] Intel Corporation

September 3rd, FPL 2013

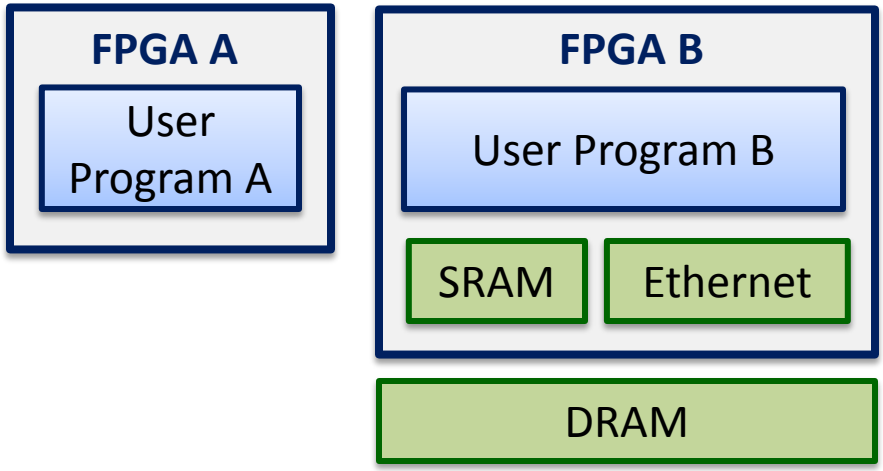
Motivation

- **Moore's Law**
  - Increasing FPGA size and capability
- **Use case for FPGA:**



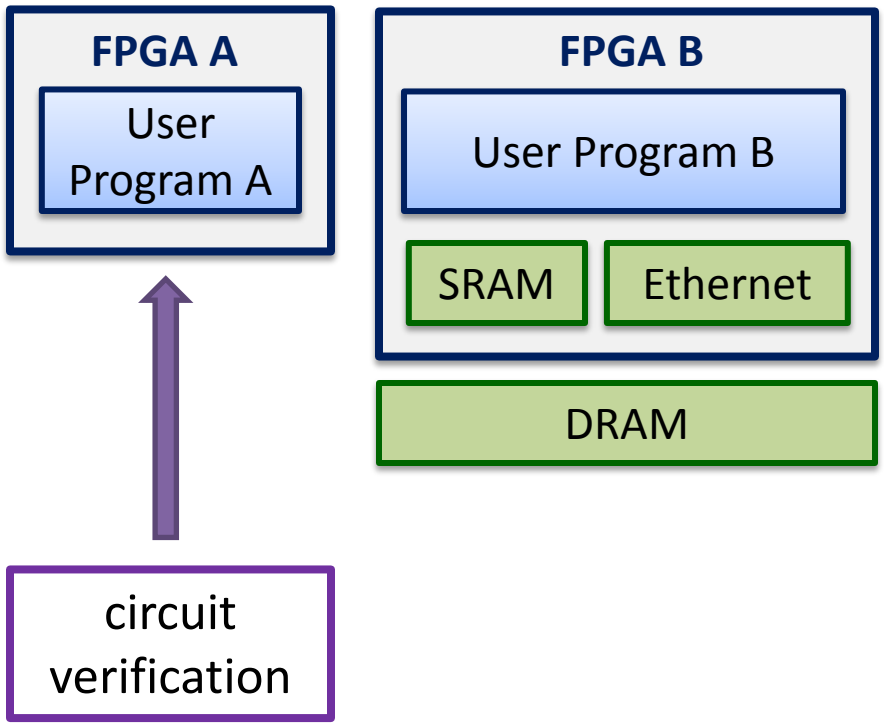
Motivation

- **Moore's Law**
  - Increasing FPGA size and capability
- **Use case for FPGA:**



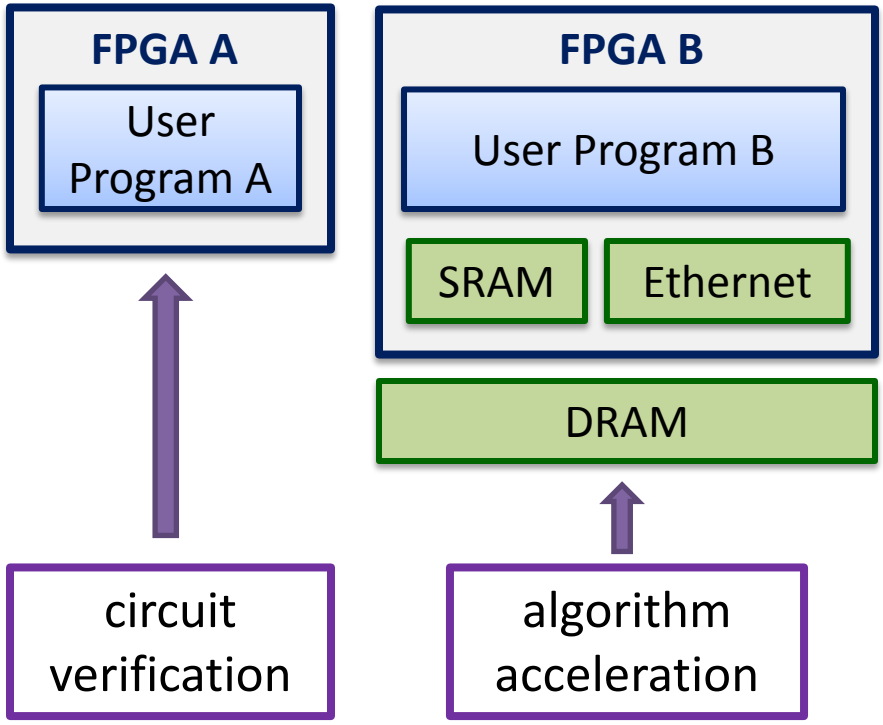
Motivation

- **Moore's Law**
  - Increasing FPGA size and capability
- **Use case for FPGA:**



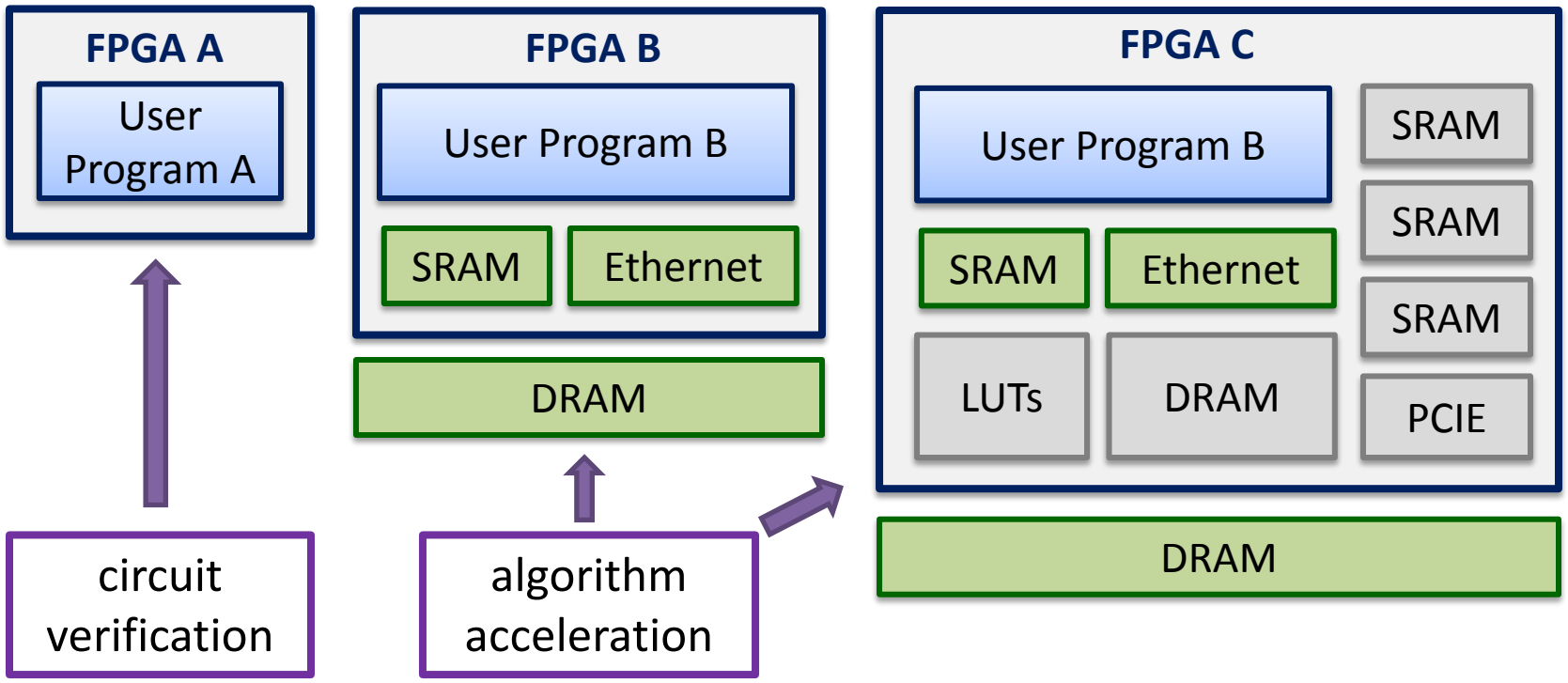
Motivation

- **Moore's Law**
  - Increasing FPGA size and capability
- **Use case for FPGA:**



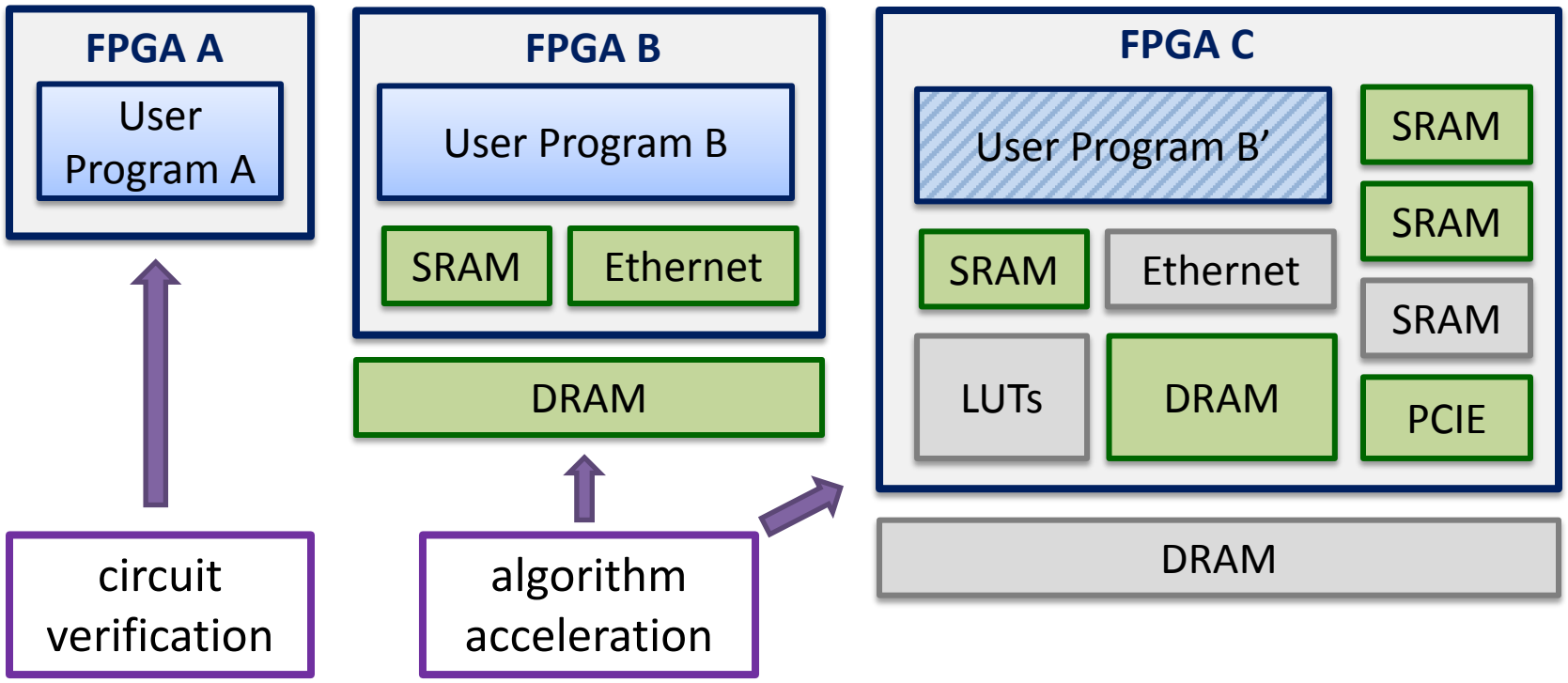
Motivation

- **Moore's Law**
  - Increasing FPGA size and capability
- **Use case for FPGA:**



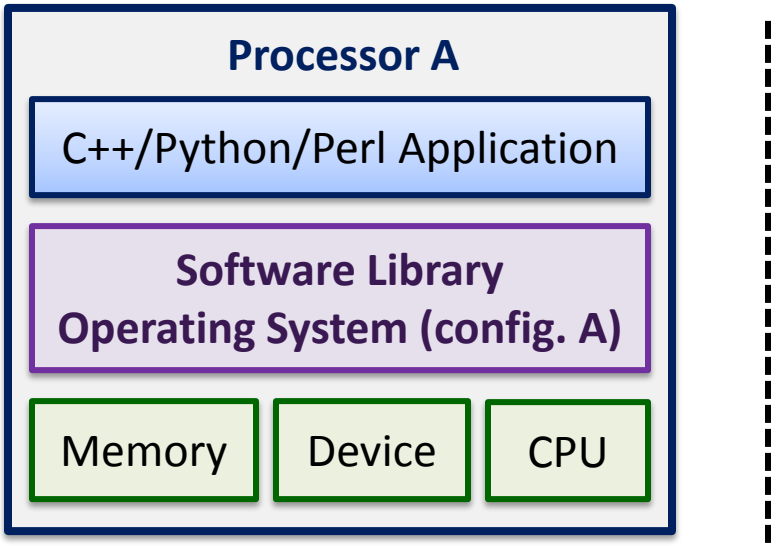
Motivation

- **Moore's Law**
  - Increasing FPGA size and capability
- **Use case for FPGA:**



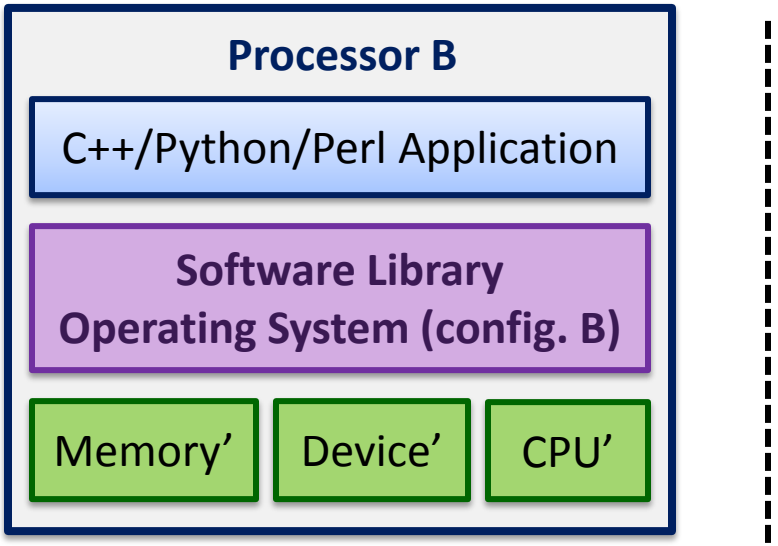
Abstraction

- **Goal: making FPGAs easier to use**



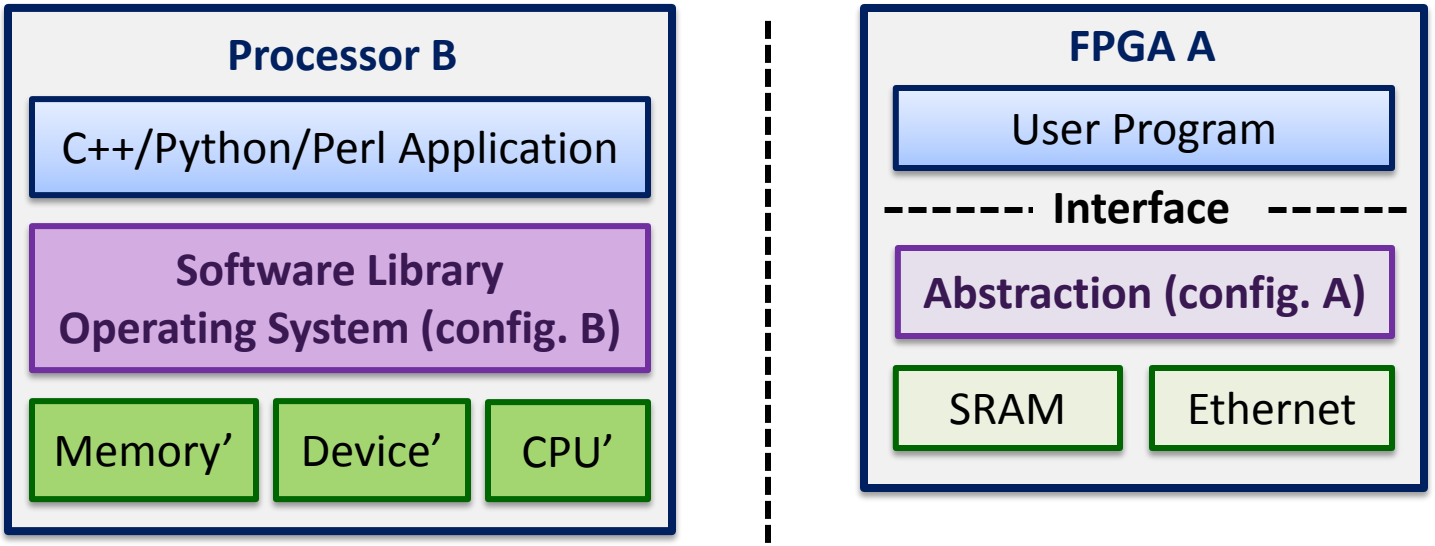
Abstraction

- **Goal: making FPGAs easier to use**



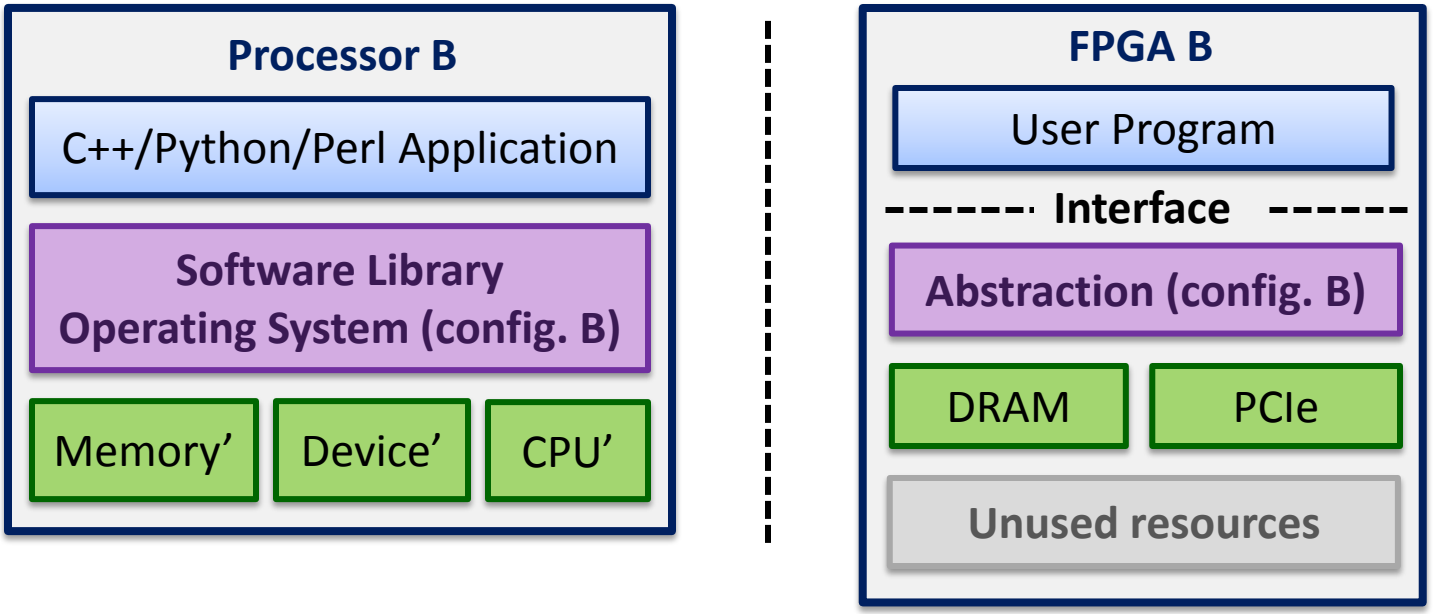
Abstraction

- **Goal: making FPGAs easier to use**



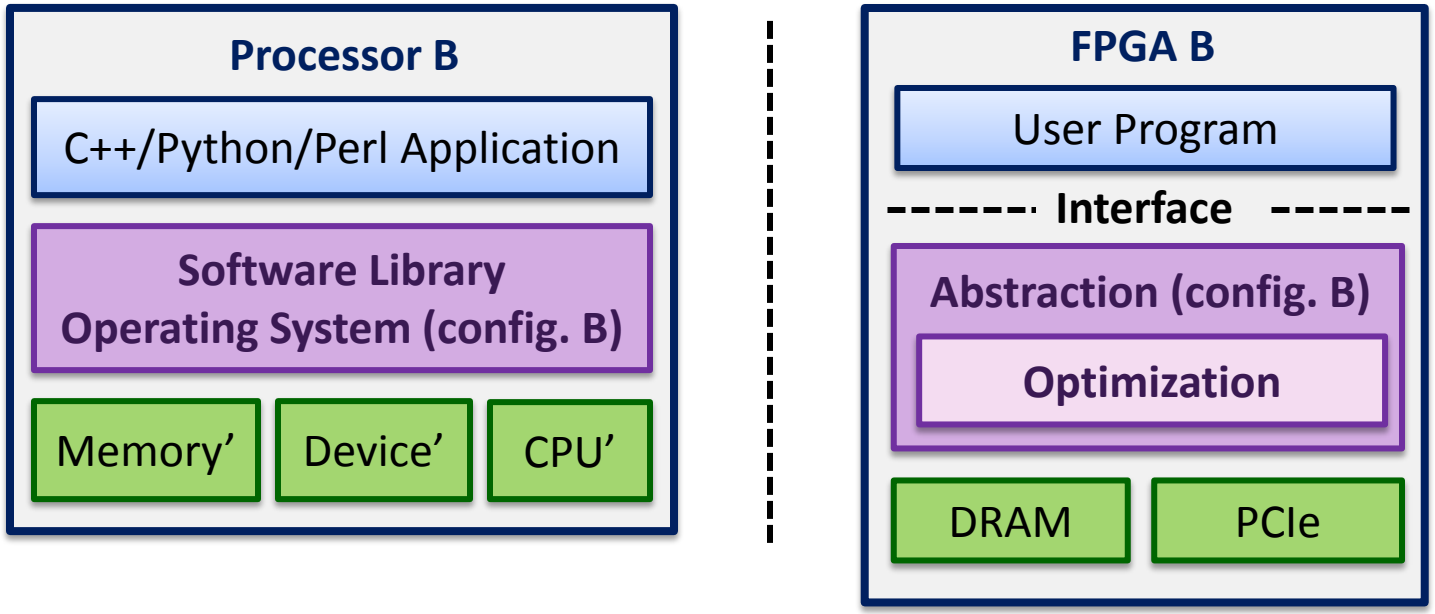
Abstraction

- **Goal: making FPGAs easier to use**



Abstraction

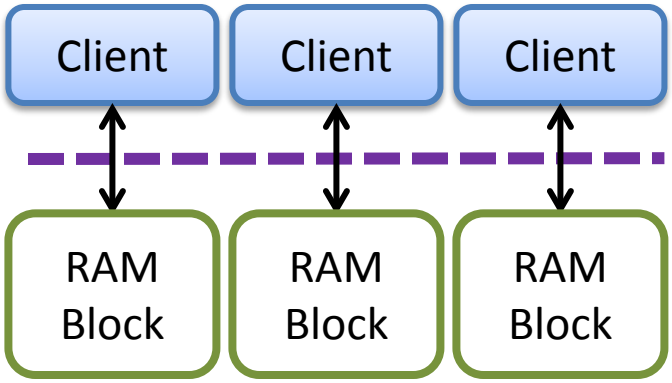
- **Goal: making FPGAs easier to use**

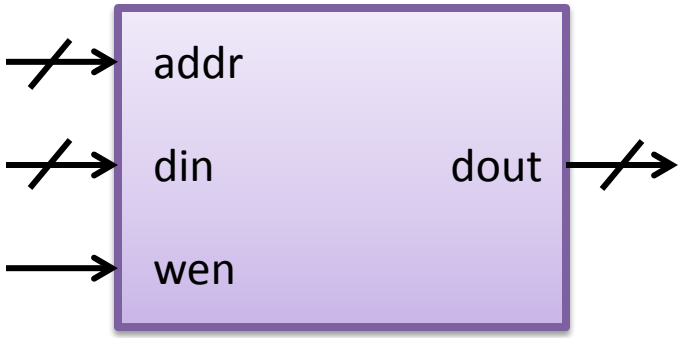


- **Optimization under abstraction**
  - Automatically accelerate FPGA applications
  - Provide **FREE** performance gain

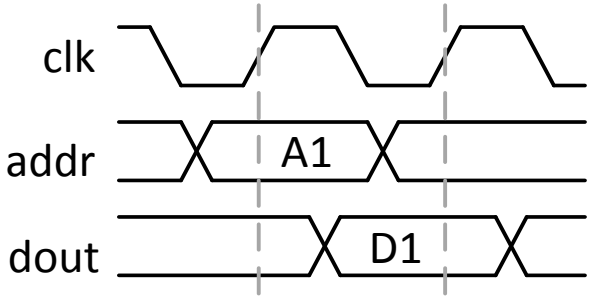
Memory Abstraction

FPGA Block RAMs



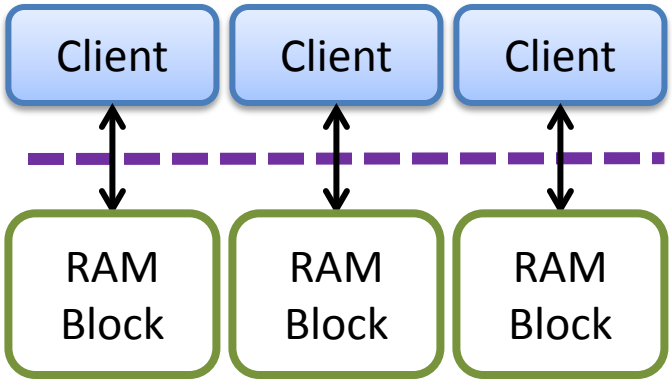


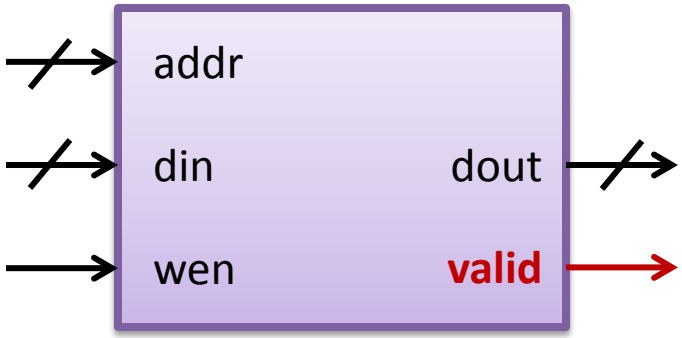

```
interface MEMORY_INTERFACE
   input:
      readReq (addr);
      write(addr, din);
   output:
      // dout is available at the next cycle of readReq
      readResp() if (readReq fired previous cycle);
endinterface
```



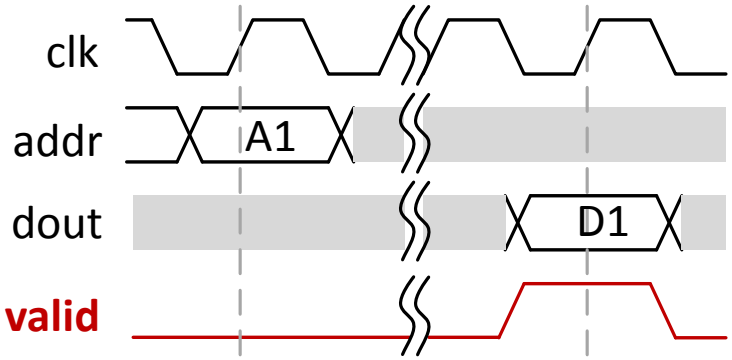
Memory Abstraction

FPGA Block RAMs

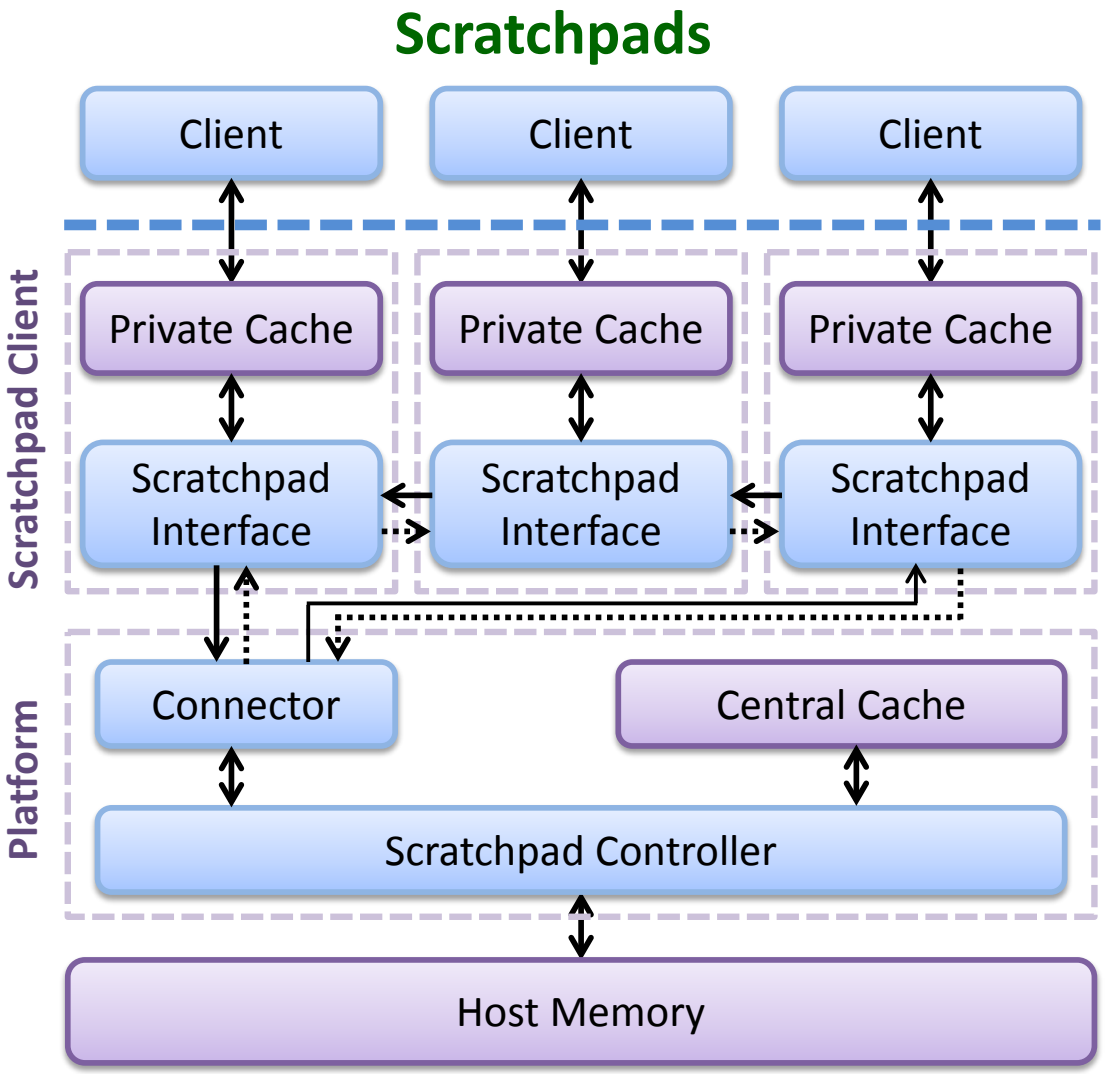





```
interface MEMORY_INTERFACE
   input:
      readReq (addr);
      write(addr, din);
   output:
      // dout is available when response is ready
      readResp() if (valid == True);
endinterface
```

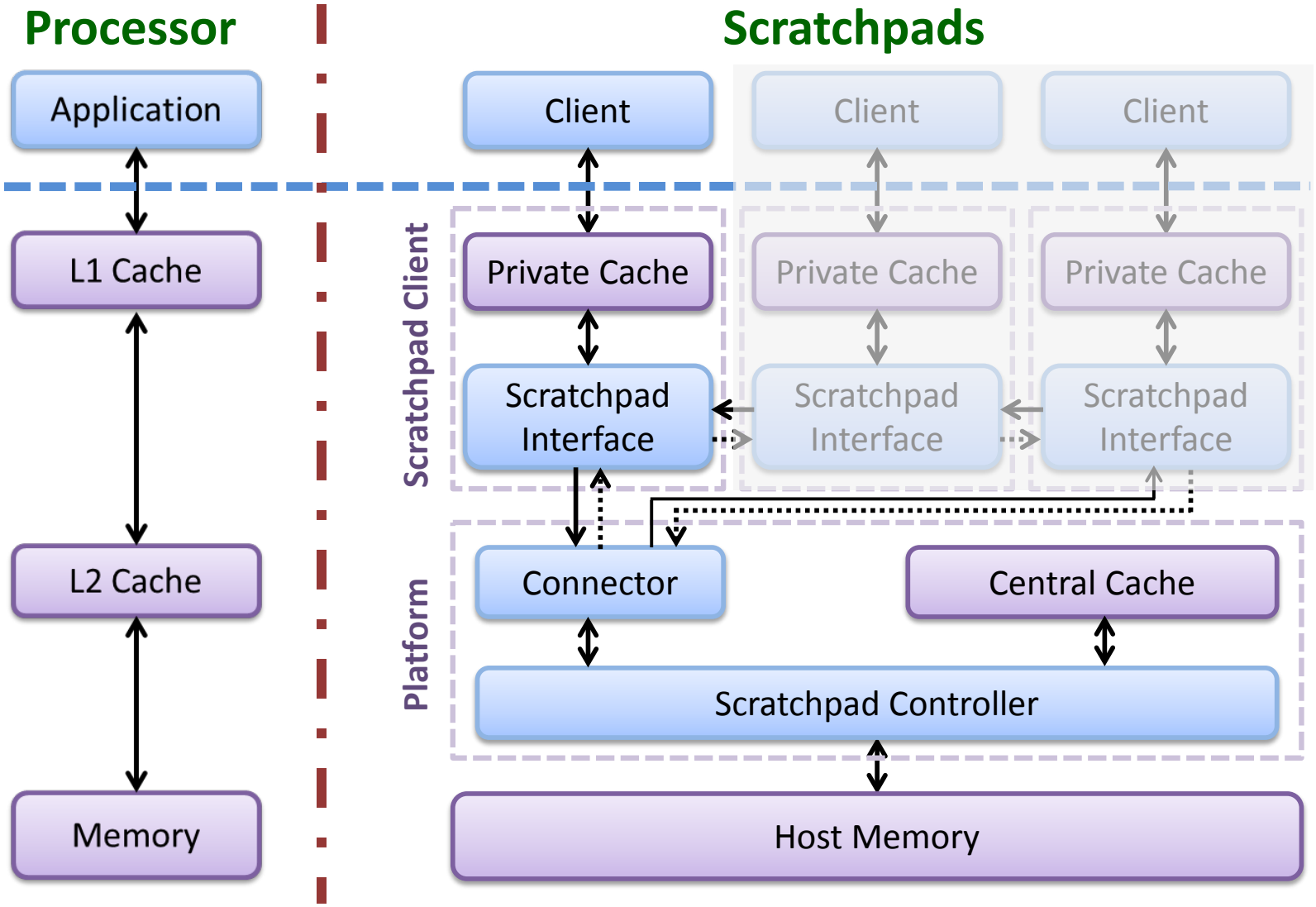


LEAP Scratchpads



M. Adler *et al.*, "LEAP Scratchpads," in FPGA, 2011.

LEAP Scratchpads



M. Adler *et al.*, "LEAP Scratchpads," in FPGA, 2011.

Scratchpad Optimization

Automatically accelerate memory-using FPGA programs

- **Reduce scratchpad latency**
- **Leverage unused resources**
- **Learn from optimization techniques in processors**
  - Larger caches, greater associativity
  - Better cache policies
  - Cache prefetching

Scratchpad Optimization

Automatically accelerate memory-using FPGA programs

- **Reduce scratchpad latency**
- **Leverage unused resources**
- **Learn from optimization techniques in processors**
  - Larger caches, greater associativity
  - Better cache policies
  - Cache prefetching

Talk Outline

- Motivation
- Introduction to LEAP Scratchpads
- **Prefetching in FPGAs vs. in processors**
- **Scratchpad Prefetcher Microarchitecture**
- **Evaluation and Prefetch Optimization**
- **Conclusion**

Prefetching Techniques

Comparison of prefetching techniques and platforms

	Static Prefetching		Dynamic Prefetching	
Platform	Processor	FPGA	Processor	FPGA
How?	User/ Compiler	User	Hardware manufacturer	Compiler
No code change			✓	✓
High prefetch accuracy	✓	✓		
No instruction overhead		✓	✓	✓
Runtime information			✓	✓

Prefetching Techniques

Comparison of prefetching techniques and platforms

	Static Prefetching		Dynamic Prefetching	
Platform	Processor	FPGA	Processor	FPGA
How?	User/ Compiler	User	Hardware manufacturer	Compiler
No code change			✓	✓
High prefetch accuracy	✓	✓		
No instruction overhead		✓	✓	✓
Runtime information			✓	✓

Dynamic Prefetching in Processor

Classic processor dynamic prefetching policies

- **When to prefetch**
  - Prefetch on cache miss
  - Prefetch on cache miss and prefetch hit
    - Also called tagged prefetch
- **What to prefetch**
  - Always prefetch next memory block
  - Learn stride-access patterns

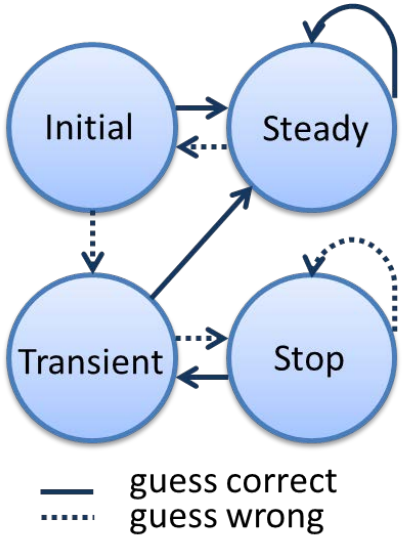
Dynamic Prefetching in Processor

Stride prefetching

- **L1 cache:** PC-based stride prefetching
- **L2 cache:** address-based stride prefetching

(fully associative cache of learners)

	Tag	Previous Address	Stride	State
learner 1	0xa001	0x1008	4	Steady
learner 2	0xa002	0x2000	0	Initial
learner 3	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮

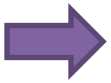


Dynamic Prefetching on FPGAs

- **Easier:**
  - Cleaner streaming memory accesses
  - No need for PC as a filter to separate streams
  - Fixed (usually plenty of) resources
- **Harder:**
  - Back-to-back memory accesses
  - Inefficient to implement CAM

Dynamic Prefetching on FPGAs

- **Easier:**
  - Cleaner streaming memory accesses
  - No need for PC as a filter to separate streams
  - Fixed (usually plenty of) resources
- **Harder:**
  - Back-to-back memory accesses
  - Inefficient to implement CAM

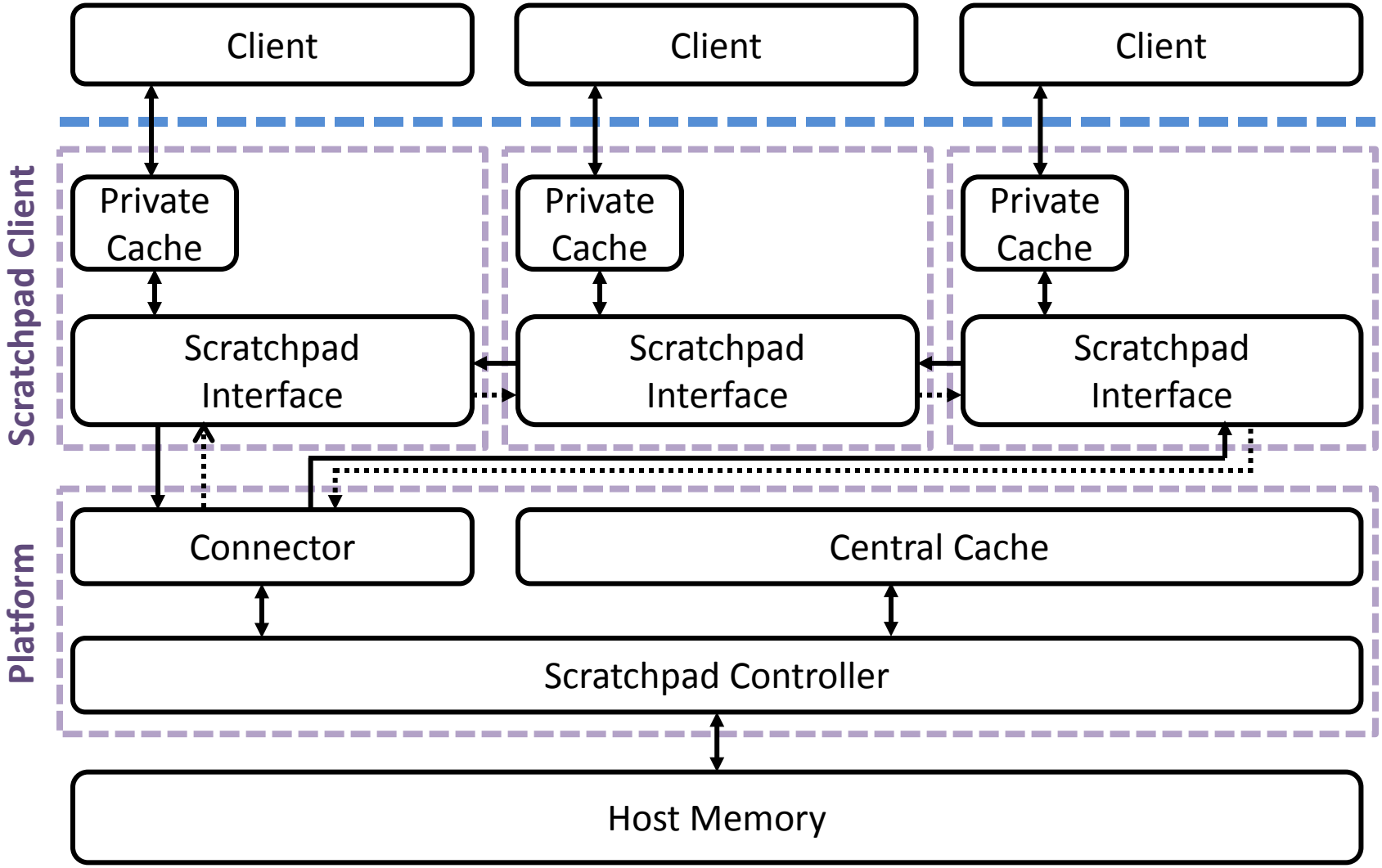


Scratchpad prefetcher uses **address-based stride prefetching** with a larger set of direct-mapped learners

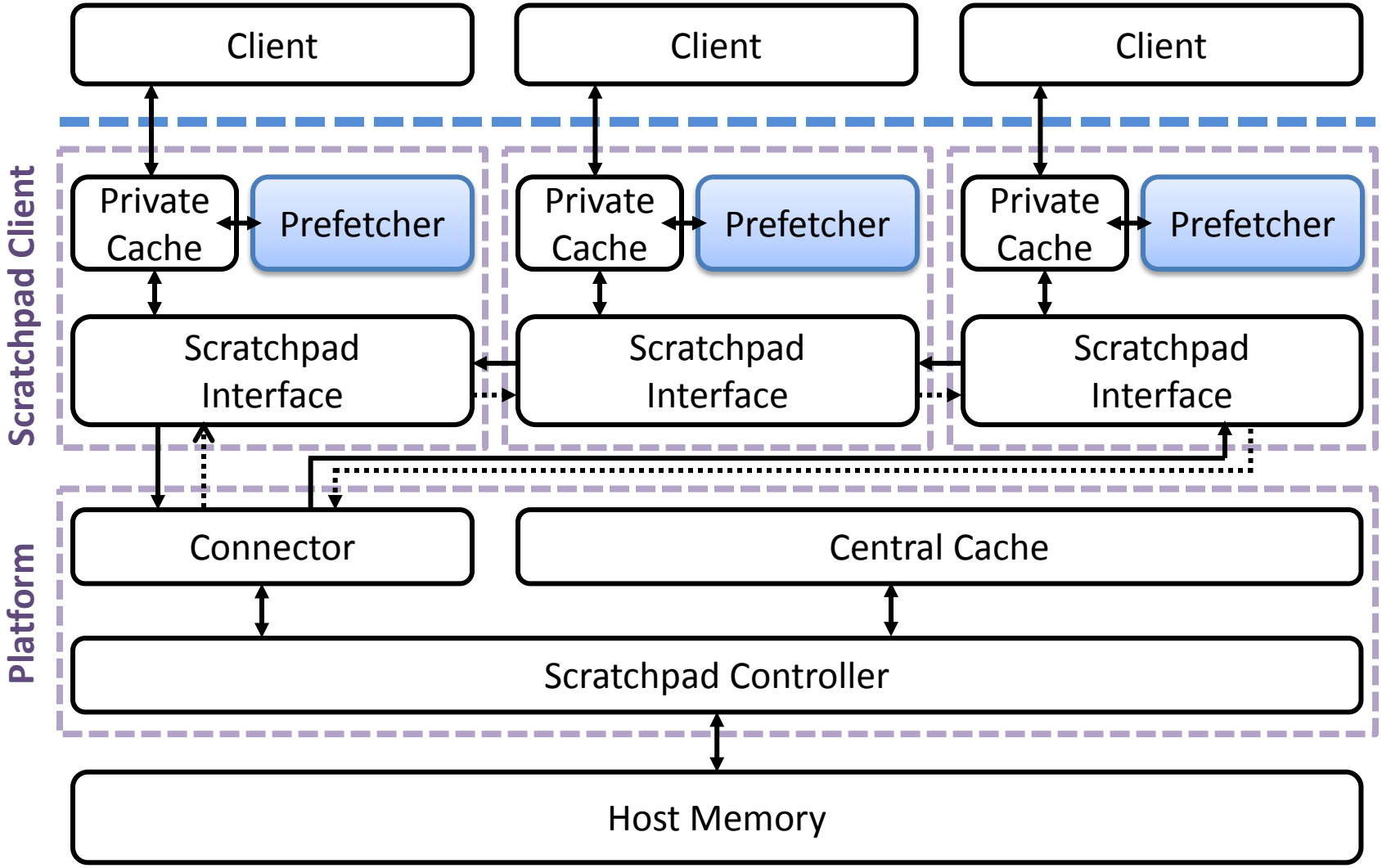
Talk Outline

- Motivation
- Introduction to LEAP Scratchpads
- Prefetching in FPGAs vs. in processors
- **Scratchpad Prefetcher Design**
- **Evaluation and Prefetch Optimization**
- **Conclusion**

Scratchpad Prefetcher



Scratchpad Prefetcher



Scratchpad Prefetching Policy

- **When to prefetch**
  - Cache line miss / prefetch hit
  - Prefetcher learns the stride pattern
- **What to prefetch**
  - Prefetch address: P = L + s * d
  - Cache line address: L
  - Learned stride: S
  - Look-ahead distance: d

Scratchpad Prefetching Policy

- **When to prefetch**
  - Cache line miss / prefetch hit
  - Prefetcher learns the stride pattern
- **What to prefetch**
  - Prefetch address: P = L + s * d
  - Cache line address: L
  - Learned stride: S
  - Look-ahead distance: d

Scratchpad Prefetching Policy

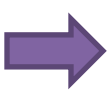
- **Look-ahead distance:**
  - Small distance? prefetch benefit ↓
  - Large distance? cache pollution
  - Suitable distance for different programs & platforms?

Scratchpad Prefetching Policy

- **Look-ahead distance:**
  - Small distance? prefetch benefit ↓
  - Large distance? cache pollution
  - Suitable distance for different programs & platforms?

Scratchpad Prefetching Policy

- **Look-ahead distance:**
  - Small distance? prefetch benefit ↓
  - Large distance? cache pollution
  - Suitable distance for different programs & platforms?

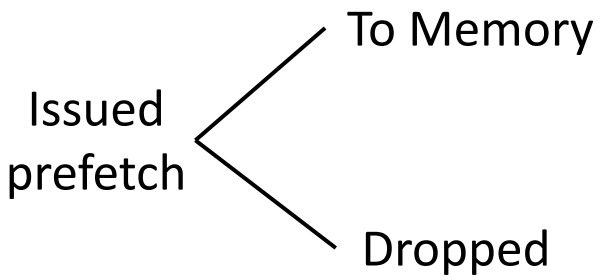


Dynamically adjust look-ahead distance

Issued prefetch

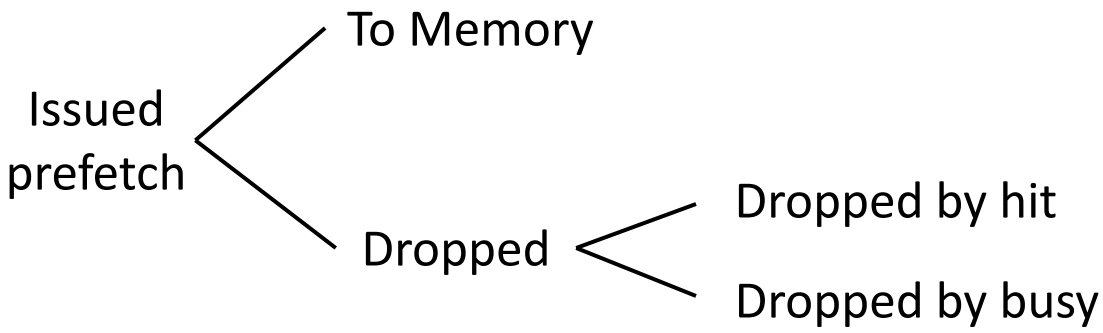
Scratchpad Prefetching Policy

- **Look-ahead distance:**
  - Small distance? prefetch benefit ↓
  - Large distance? cache pollution
  - Suitable distance for different programs & platforms?



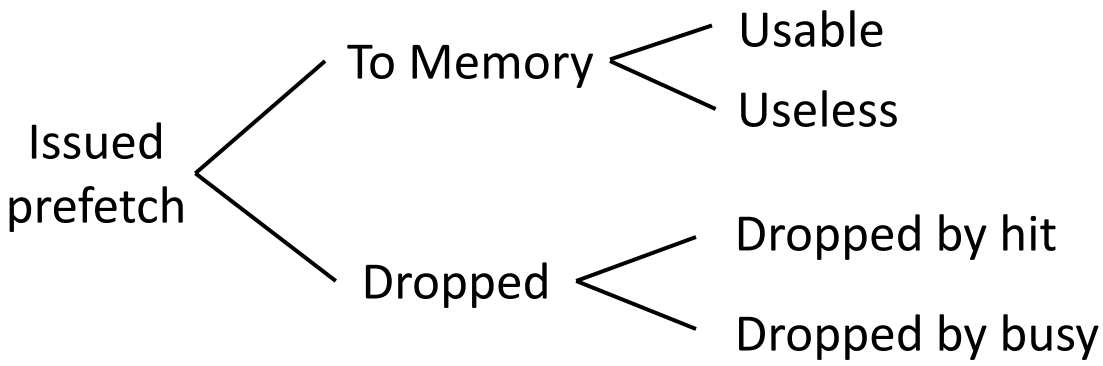
Scratchpad Prefetching Policy

- **Look-ahead distance:**
  - Small distance? prefetch benefit ↓
  - Large distance? cache pollution
  - Suitable distance for different programs & platforms?



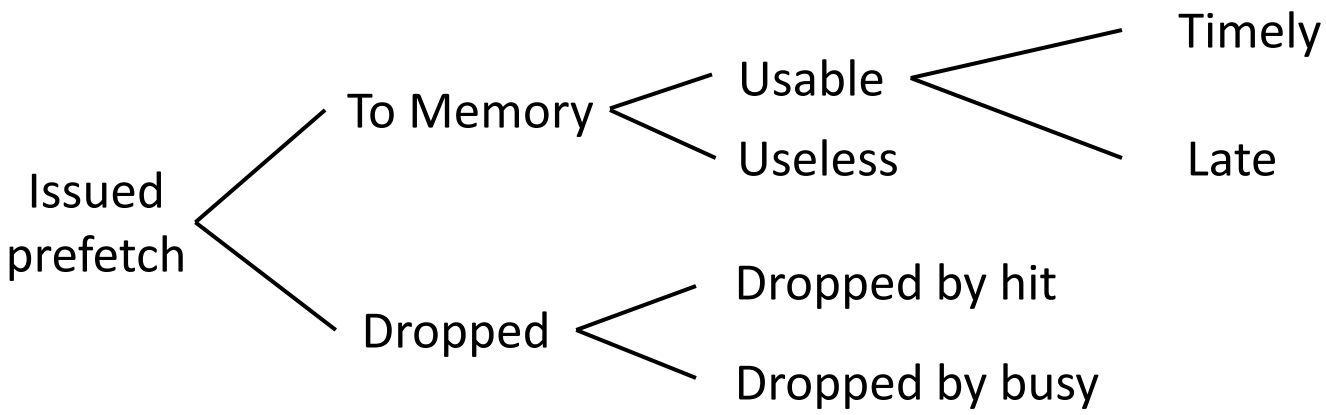
Scratchpad Prefetching Policy

- **Look-ahead distance:**
  - Small distance? prefetch benefit ↓
  - Large distance? cache pollution
  - Suitable distance for different programs & platforms?



Scratchpad Prefetching Policy

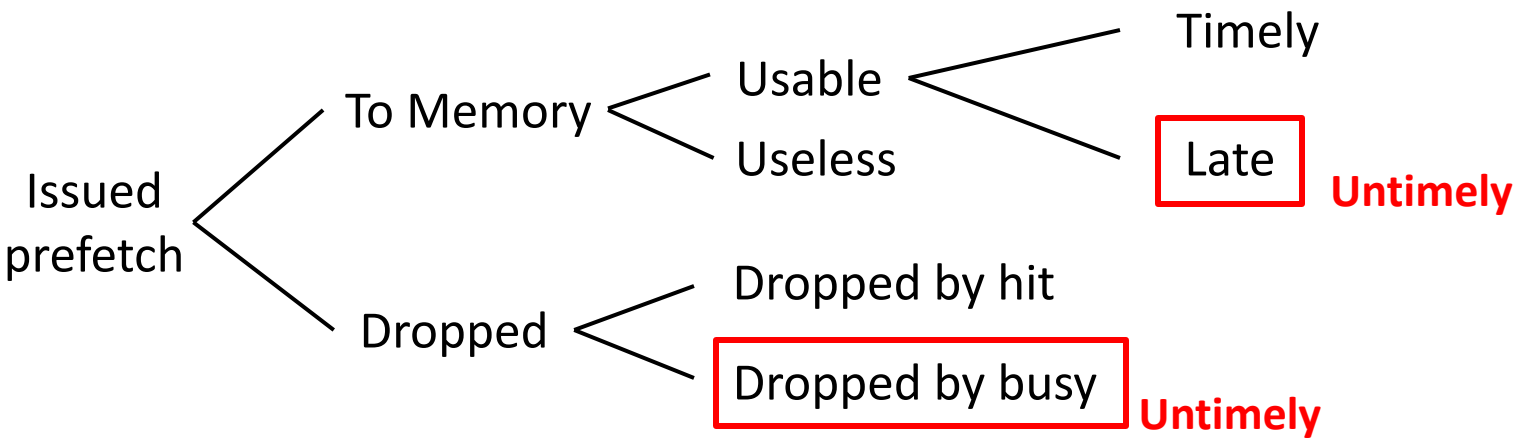
- **Look-ahead distance:**
  - Small distance? prefetch benefit ↓
  - Large distance? cache pollution
  - Suitable distance for different programs & platforms?



Scratchpad Prefetching Policy

- **Look-ahead distance:**
  - Small distance? prefetch benefit ↓
  - Large distance? cache pollution
  - Suitable distance for different programs & platforms?

➡ Dynamically adjust look-ahead distance

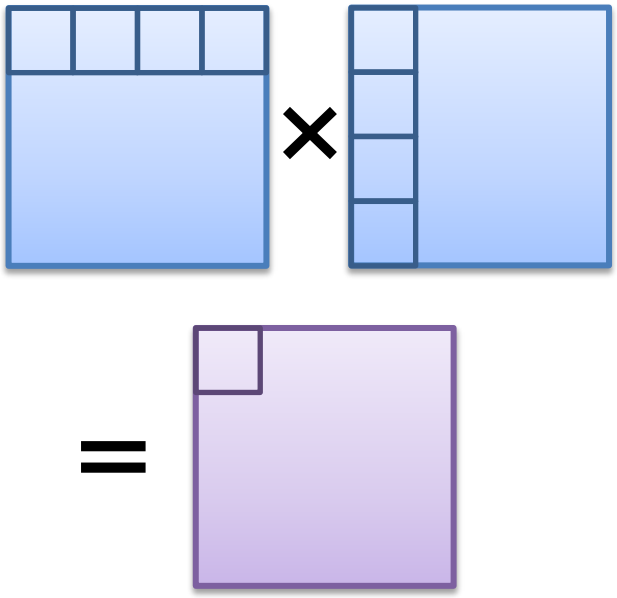


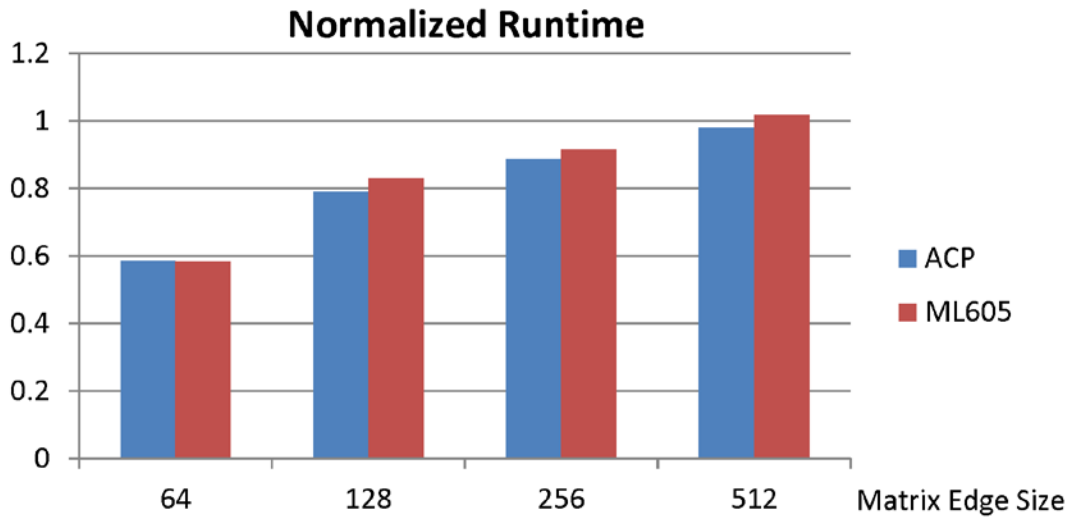
Talk Outline

- Motivation
- Introduction to LEAP Scratchpads
- Prefetching in FPGAs vs. in processors
- Scratchpad Prefetcher Microarchitecture
- **Evaluation and Prefetch Bandwidth Control**
- **Conclusion**

Evaluation

- **Blocked matrix-matrix multiplication (MMM)**

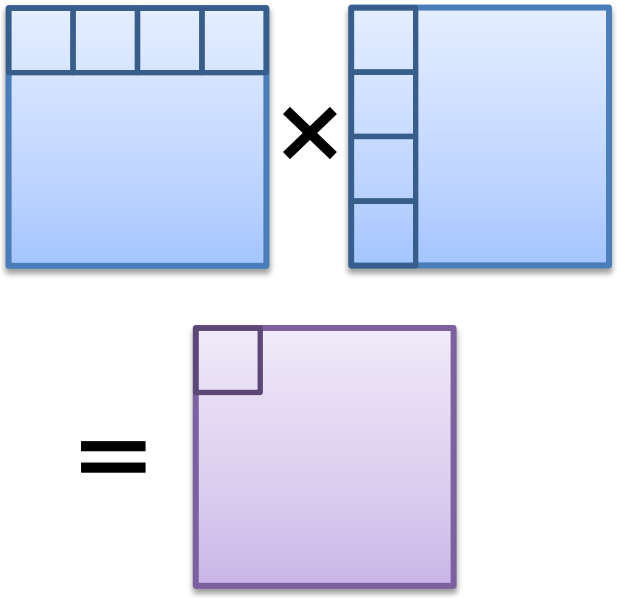


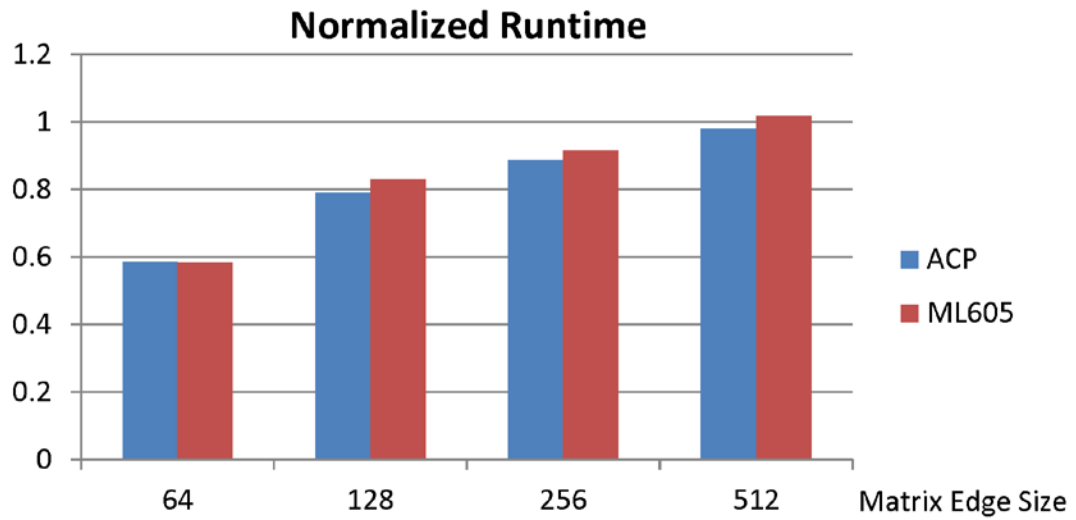


N. Dave *et al.*, "Hardware acceleration of matrix multiplication on a xilinx fpga," in MEMOCODE, 2007.

Evaluation

- **Blocked matrix-matrix multiplication (MMM)**



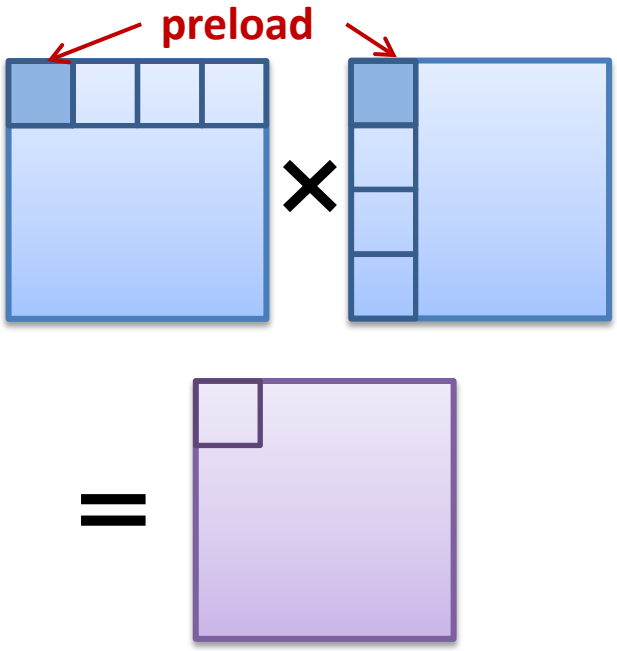


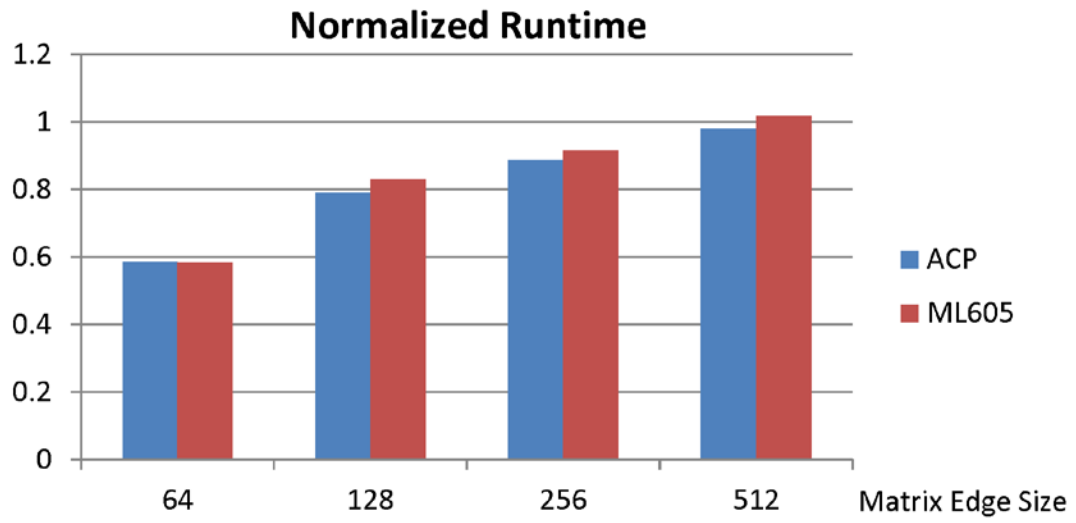
- Prefetching has larger gains in smaller matrices.
- Prefetching helps in edge conditions.

N. Dave *et al.*, "Hardware acceleration of matrix multiplication on a xilinx fpga," in MEMOCODE, 2007.

Evaluation

- **Blocked matrix-matrix multiplication (MMM)**



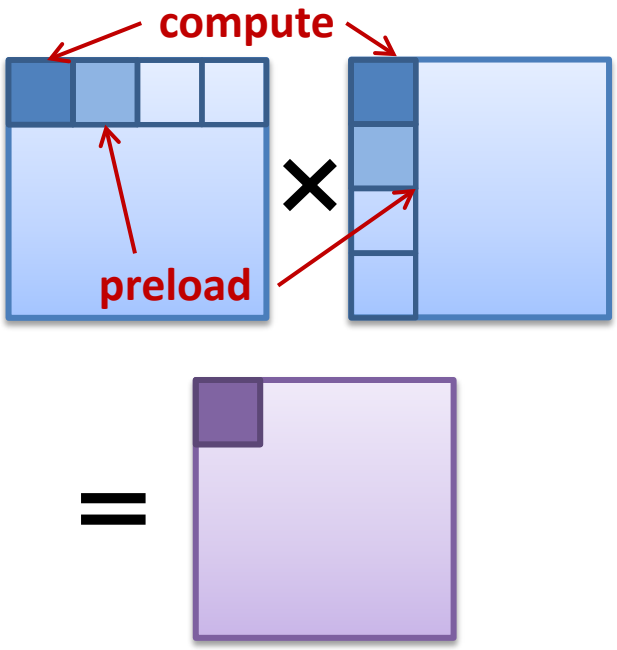


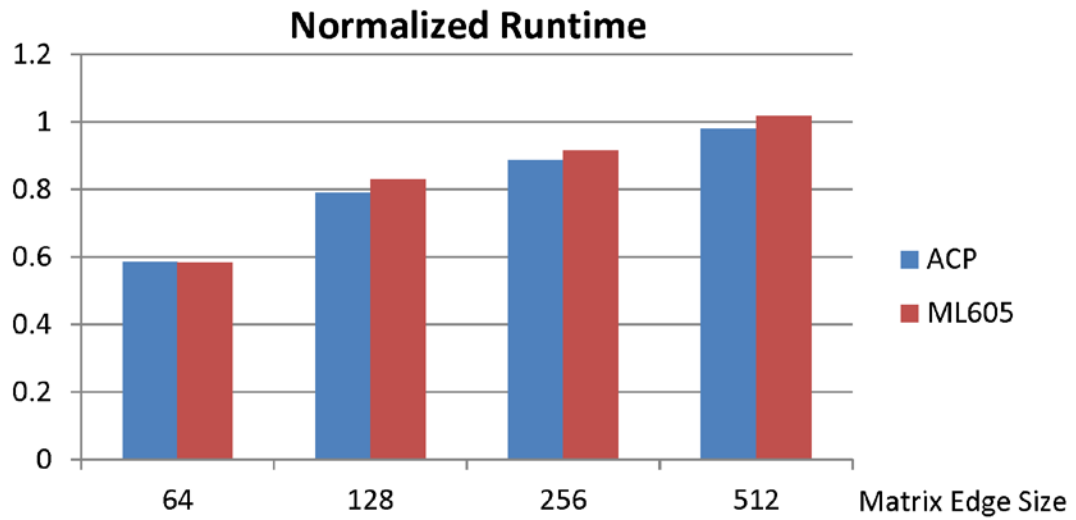
- Prefetching has larger gains in smaller matrices.
- Prefetching helps in edge conditions.

N. Dave *et al.*, "Hardware acceleration of matrix multiplication on a xilinx fpga," in MEMOCODE, 2007.

Evaluation

- **Blocked matrix-matrix multiplication (MMM)**



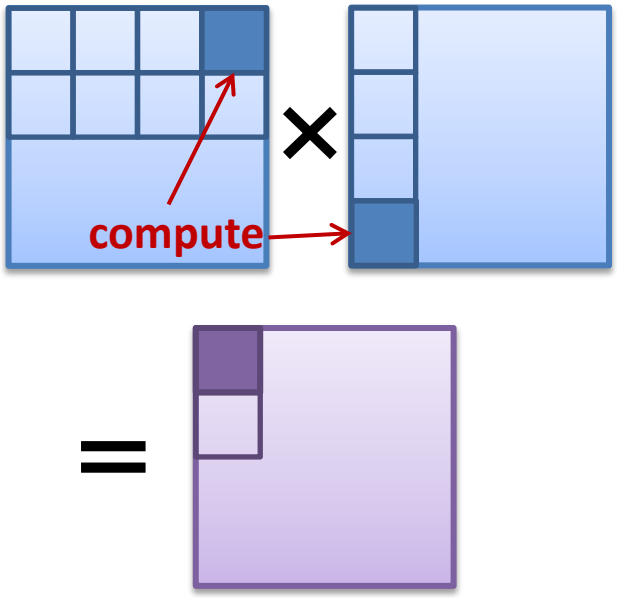


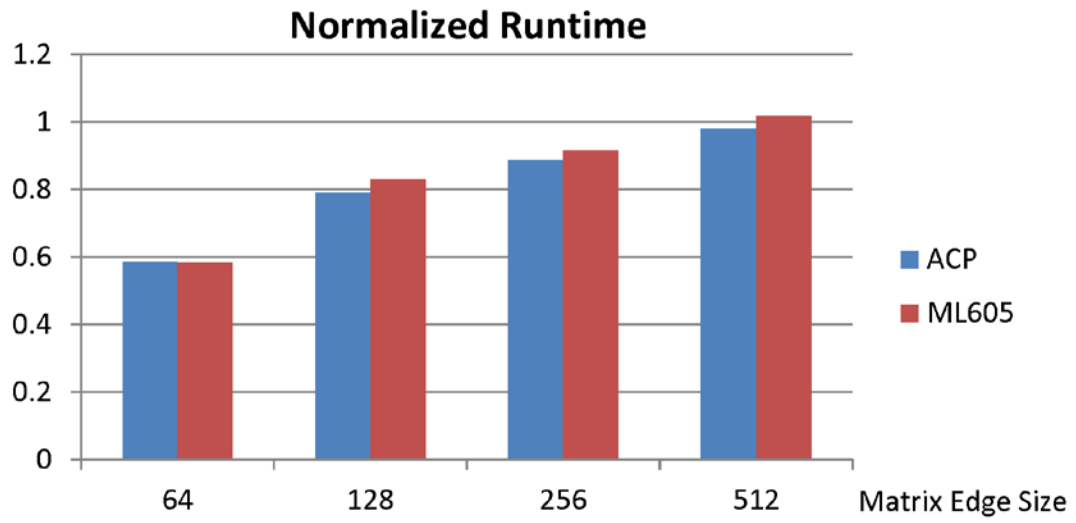
- Prefetching has larger gains in smaller matrices.
- Prefetching helps in edge conditions.

N. Dave *et al.*, "Hardware acceleration of matrix multiplication on a xilinx fpga," in MEMOCODE, 2007.

Evaluation

- **Blocked matrix-matrix multiplication (MMM)**



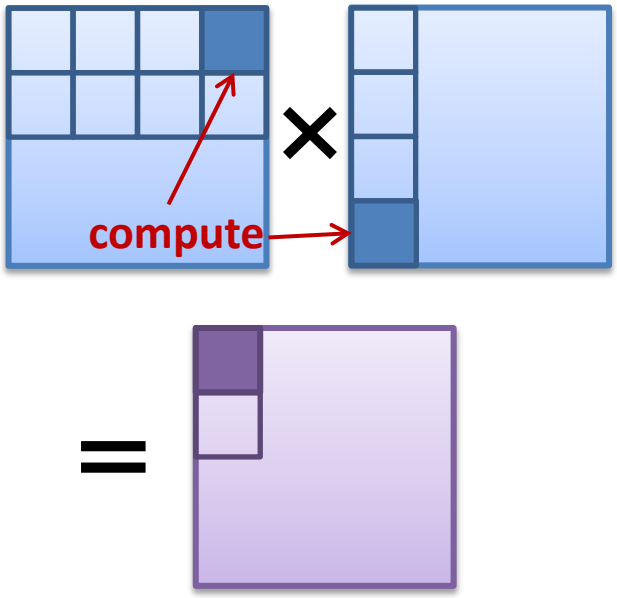


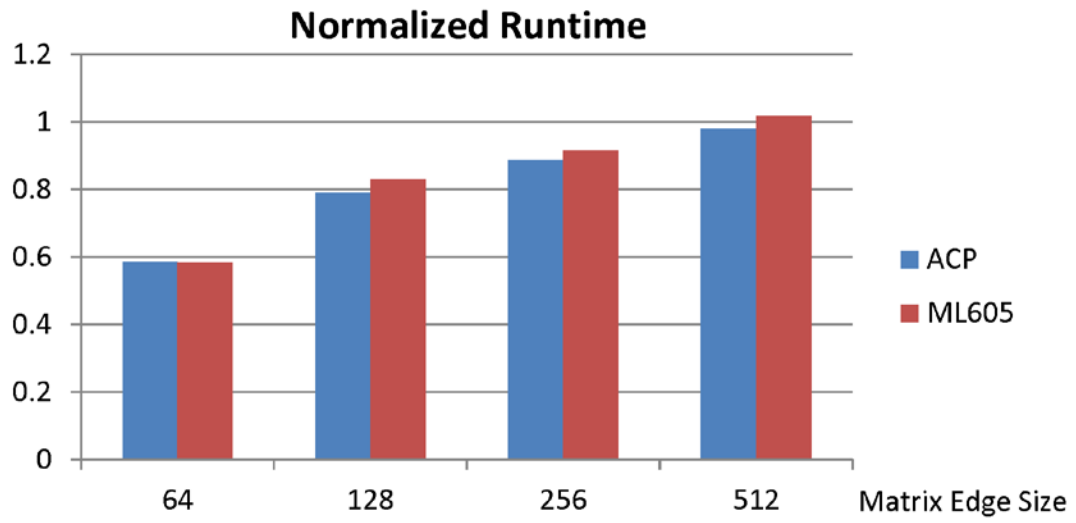
- Prefetching has larger gains in smaller matrices.
- Prefetching helps in edge conditions.

N. Dave *et al.*, "Hardware acceleration of matrix multiplication on a xilinx fpga," in MEMOCODE, 2007.

Evaluation

- **Blocked matrix-matrix multiplication (MMM)**





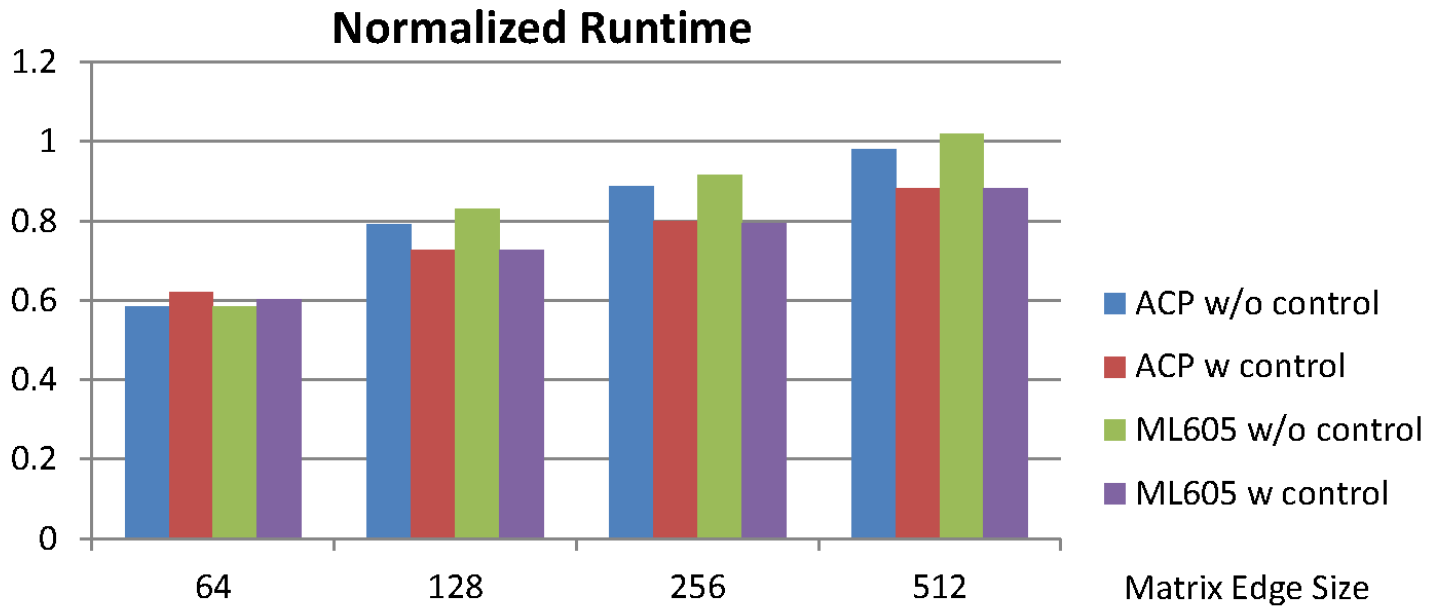
- Prefetching has larger gains in smaller matrices.
- Prefetching helps in edge conditions.

N. Dave *et al.*, "Hardware acceleration of matrix multiplication on a xilinx fpga," in MEMOCODE, 2007.

Memory Bandwidth Control

- **MMM with memory bandwidth control**

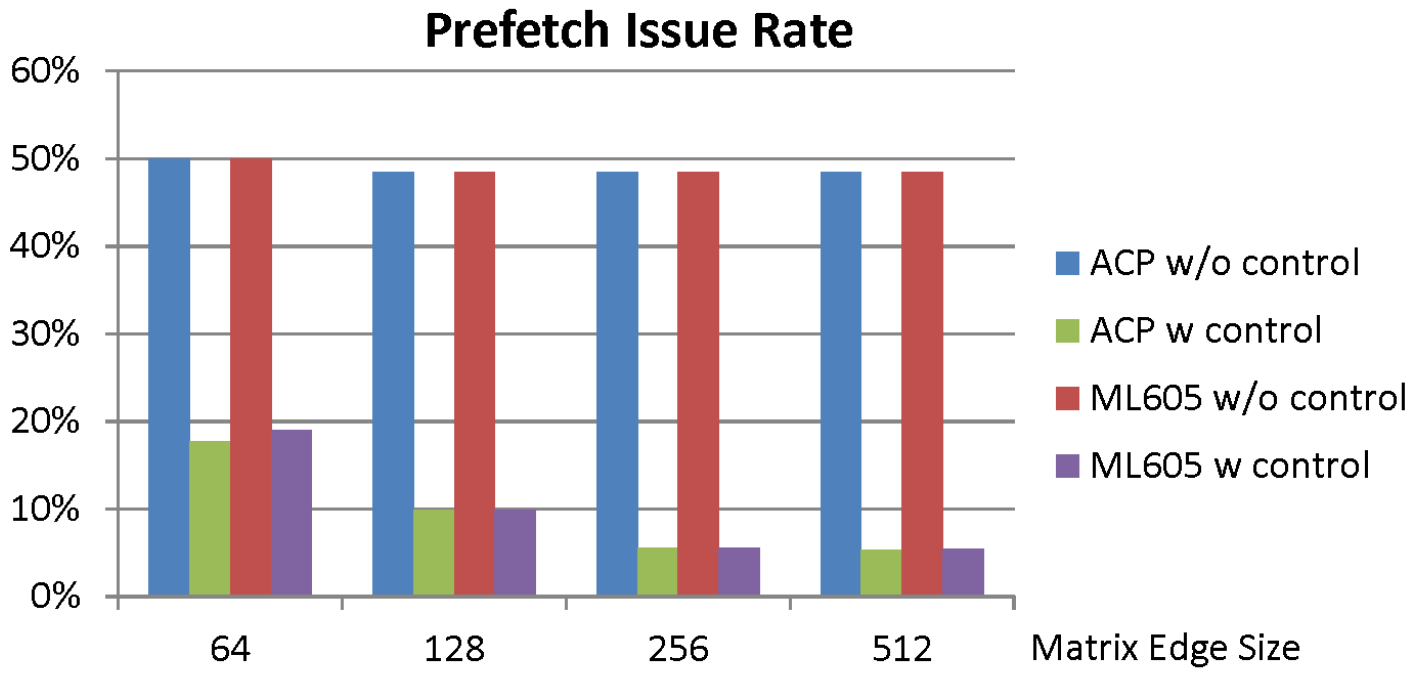
  Prefetcher automatically stops issuing requests when there are too many requests inflight



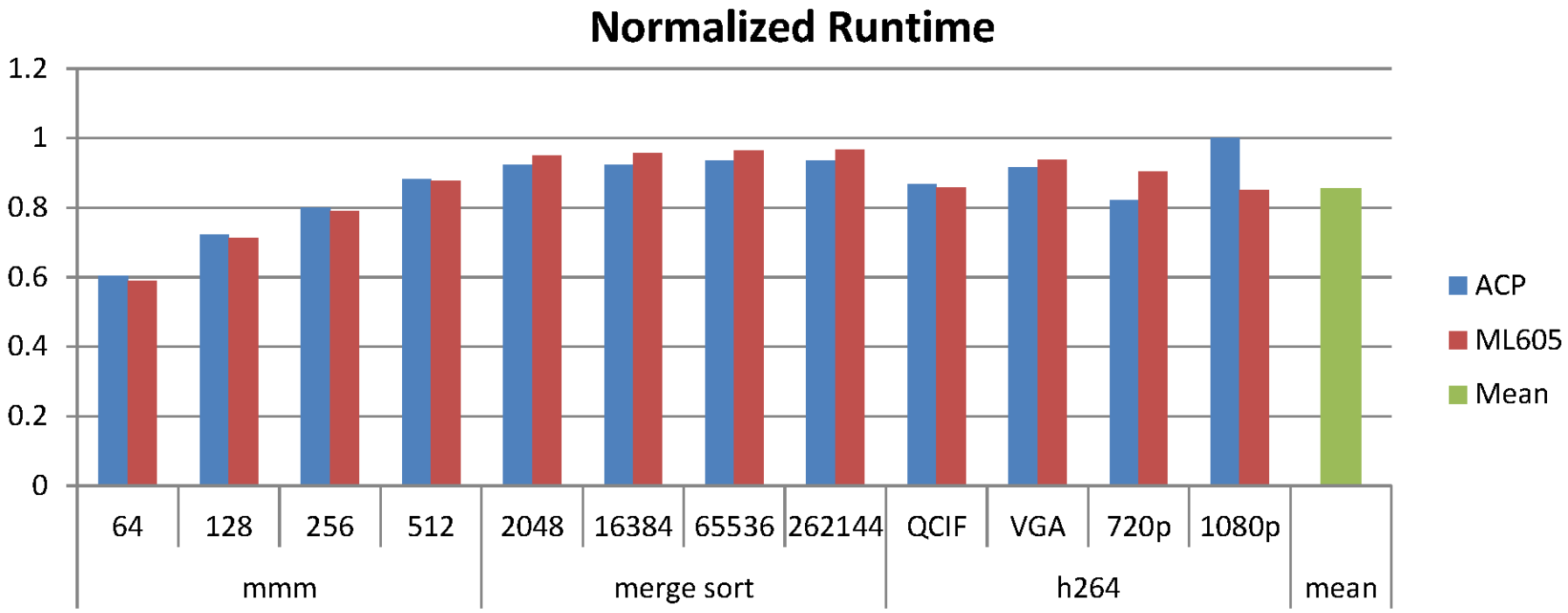
Memory Bandwidth Control

- **MMM with memory bandwidth control**

  Prefetcher automatically stops issuing requests when there are too many requests inflight



Prefetch Performance Summary



K. Fleming *et al.*, "H.264 Decoder: A Case Study in Multiple Design Points," in MEMOCODE, 2008

Prefetcher Resource Utilization

Area of different prefetching logic implementations

	Slice Registers	Slice LUTs	BRAM	f_{max}
32 learners, LUTRAM	333	1045	0	127 MHz
32 learners, BRAM	419	1275	2	131 MHz
H.264, Baseline Profile	60770	86364	99	80 MHz

- Area requirements of FPGA prefetching are small.
  - — less than 0.5% of total chip area on the ML605 board

Conclusion

- FPGA programs are hard to write
- FPGA programs usually do not fully utilize resources
- Optimizations under abstraction leverage unused resources and provide FREE performance gain
- Adding prefetching to LEAP Scratchpads speeds up existing streaming applications
  - No program code changes in target design
  - 15% average runtime improvement
- There are many other possible optimizations to the FPGA memory system

Thank You