High-Speed Soft-Processor Architecture For FPGA Overlays

Charles Eric LaForest

SGS Final Oral Examination
December 5th, 2014

Motivation

- Designing on FPGAs remains difficult
  - Larger systems
  - Longer CAD processing times
  - Increases time-to-market and engineering costs

FPGA Design Processes

- Hardware Description Languages (Verilog, VHDL)
  - Precise implementation
  - Low-level and tedious
  - Long CAD processing time

FPGA Design Processes

- High-Level Synthesis (LegUp, Bluespec)
  - Easier, faster design and exploration
  - Mostly same performance as HDL
  - “Black-Box” implementations
  - Long CAD processing time

FPGA Design Processes

- Overlays (soft-processors)
  - Easiest and fastest: design as software
  - Co-design hardware only if necessary
  - Fast overall design cycle
  - Lower performance
  - Higher area

FPGA Design Processes

- Soft-processor vs. underlying FPGA (Stratix IV)
  - Logic Fabric: 800 MHz
  - Block RAM: 550 MHz
  - DSP Block: 480 MHz
  - Nios II/f: 240 MHz

FPGA Design Processes

- ***How do we improve overlay performance?***

Overlay Design Goals

- Abundant Parallelism
  - SIMD and MIMD

Overlay Design Goals

- Abundant Parallelism
  - SIMD and MIMD
- High Clock Frequency (Fmax)

Overlay Design Goals

- Abundant Parallelism
  - SIMD and MIMD
- High Clock Frequency (Fmax)
- Low Architectural Overhead
  - Low CPI, instruction count

Overlay Design Goals

- Abundant Parallelism
  - SIMD and MIMD
- High Clock Frequency (Fmax)
- Low Architectural Overhead
  - Low CPI, instruction count
- Few Stalls
  - Data and Control dependencies, Memory latency

Overlay Design Goals

- Abundant Parallelism
  - SIMD and MIMD
- High Clock Frequency (Fmax)
- Low Architectural Overhead
  - Low CPI, instruction count
- Few Stalls
  - Data and Control dependencies, Memory latency
- Simple and Minimal

Overlay Design Goals

- Abundant Parallelism
  - SIMD and MIMD
- High Clock Frequency (Fmax)
- Low Architectural Overhead
  - Low CPI, instruction count
- Few Stalls
  - Data and Control dependencies, Memory latency
- Simple and Minimal
- Congruent to underlying FPGA
  - Word widths, pipeline depths, primitives

Multi-Threaded Overlay Architecture

- Must pipeline to absorb FPGA delays
  - Problem: dependencies between pipeline stages

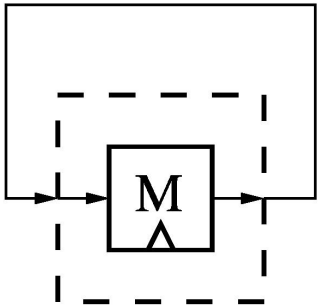
Multi-Threaded Overlay Architecture

- Must pipeline to absorb FPGA delays
  - Problem: dependencies between pipeline stages
- Proposed solution: fully-pipelined multi-threading
  - Full pipelining to maximize Fmax
  - “Single-cycle” thread instructions over entire pipeline
  - Multiple threads for SIMD and MIMD parallelism

Multi-Threaded Overlay Architecture

- Must pipeline to absorb FPGA delays
  - Problem: dependencies between pipeline stages
- Proposed solution: fully-pipelined multi-threading
  - Full pipelining to maximize Fmax
  - “Single-cycle” thread instructions over entire pipeline
  - Multiple threads for SIMD and MIMD parallelism
- **Only allow fixed round-robin scheduling**
  - Unlike HEP, Tera, UTMT II, CUSTARD, NetThreads
  - No pipeline dependencies (almost...)
  - Determinism enables thread composition

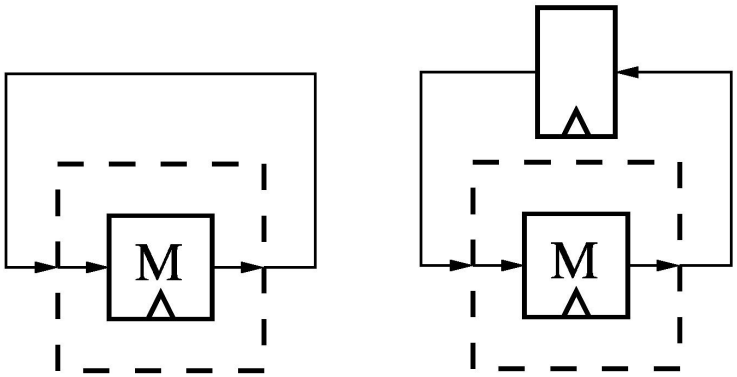
Self-Loop Characterization (BRAM)



398 MHz

- Accounts for interconnect and clock-to-out delay

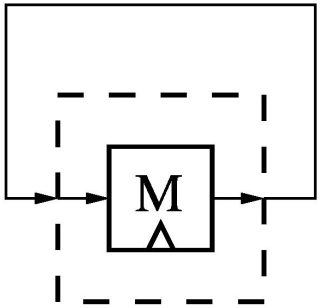
Self-Loop Characterization (BRAM)



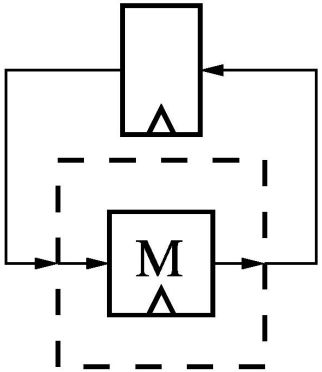
398 MHz 656 MHz

- Accounts for interconnect and clock-to-out delay

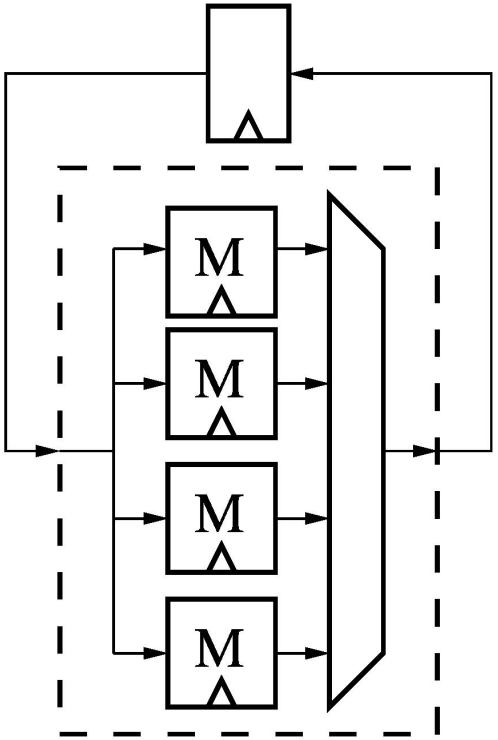
Self-Loop Characterization (BRAM)



398 MHz



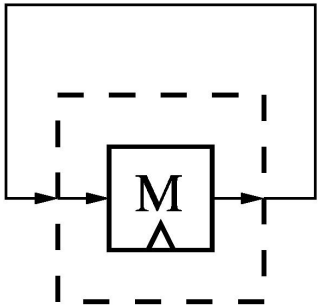
656 MHz



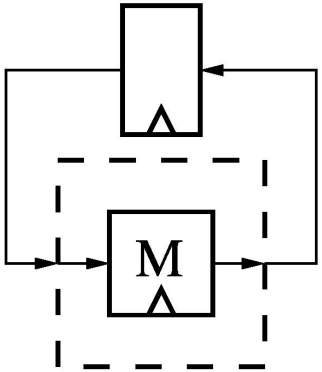
531 MHz

- Accounts for interconnect and clock-to-out delay

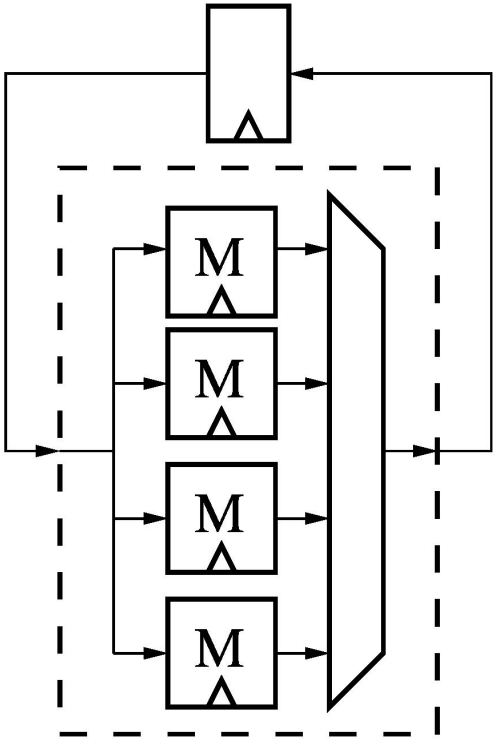
Self-Loop Characterization (BRAM)



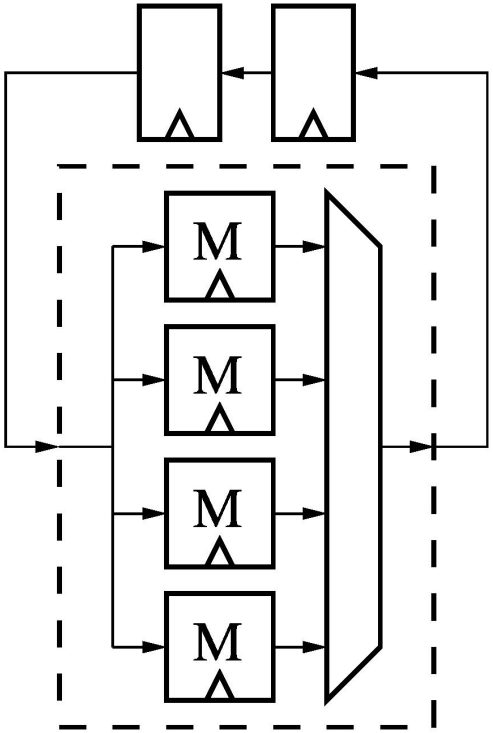
398 MHz



656 MHz



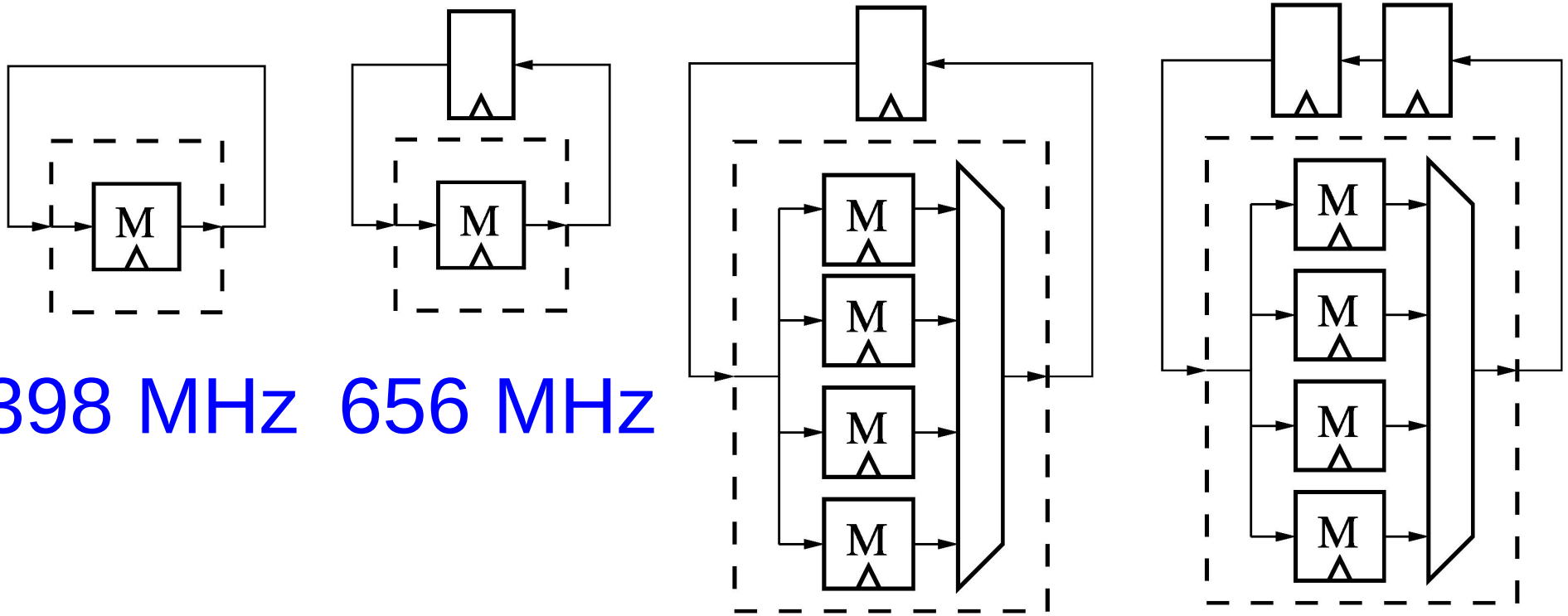
531 MHz



710 MHz

- Accounts for interconnect and clock-to-out delay

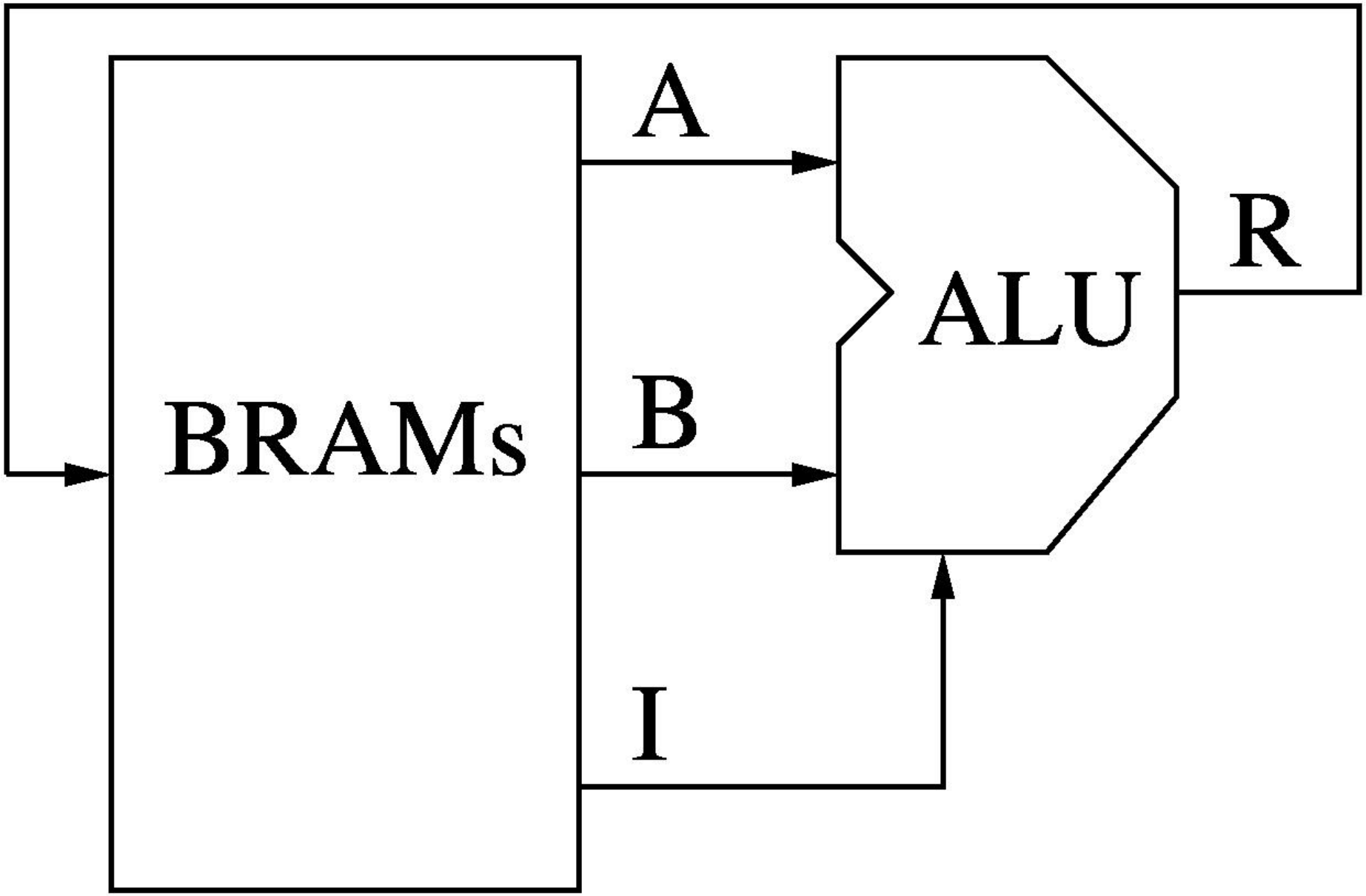
Self-Loop Characterization (BRAM)



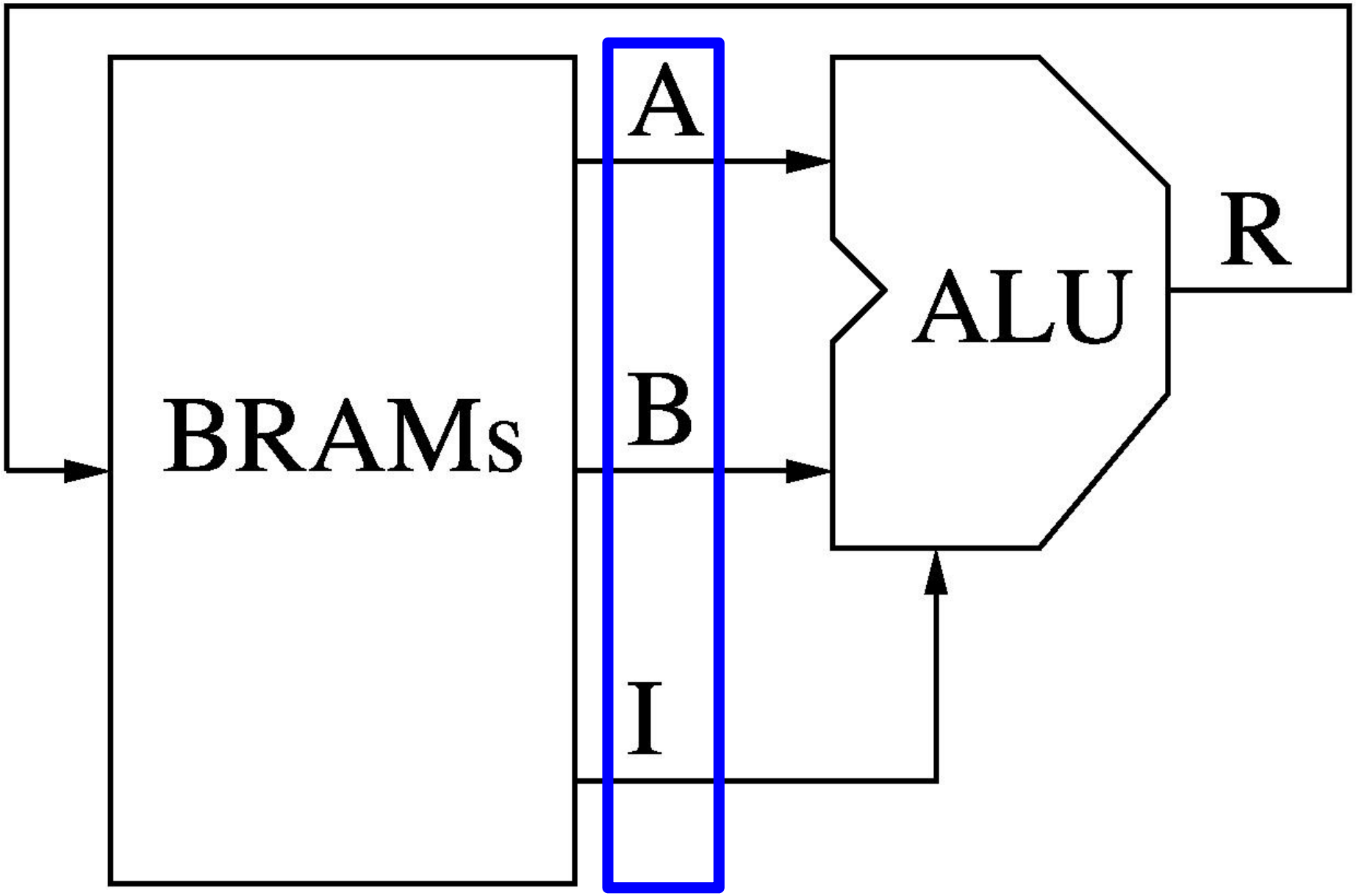
398 MHz 656 MHz 531 MHz 710 MHz

- Accounts for interconnect and clock-to-out delay
- Minimum clock pulse width of 500 to 550 MHz
- Absolute upper frequency limit on Stratix IV

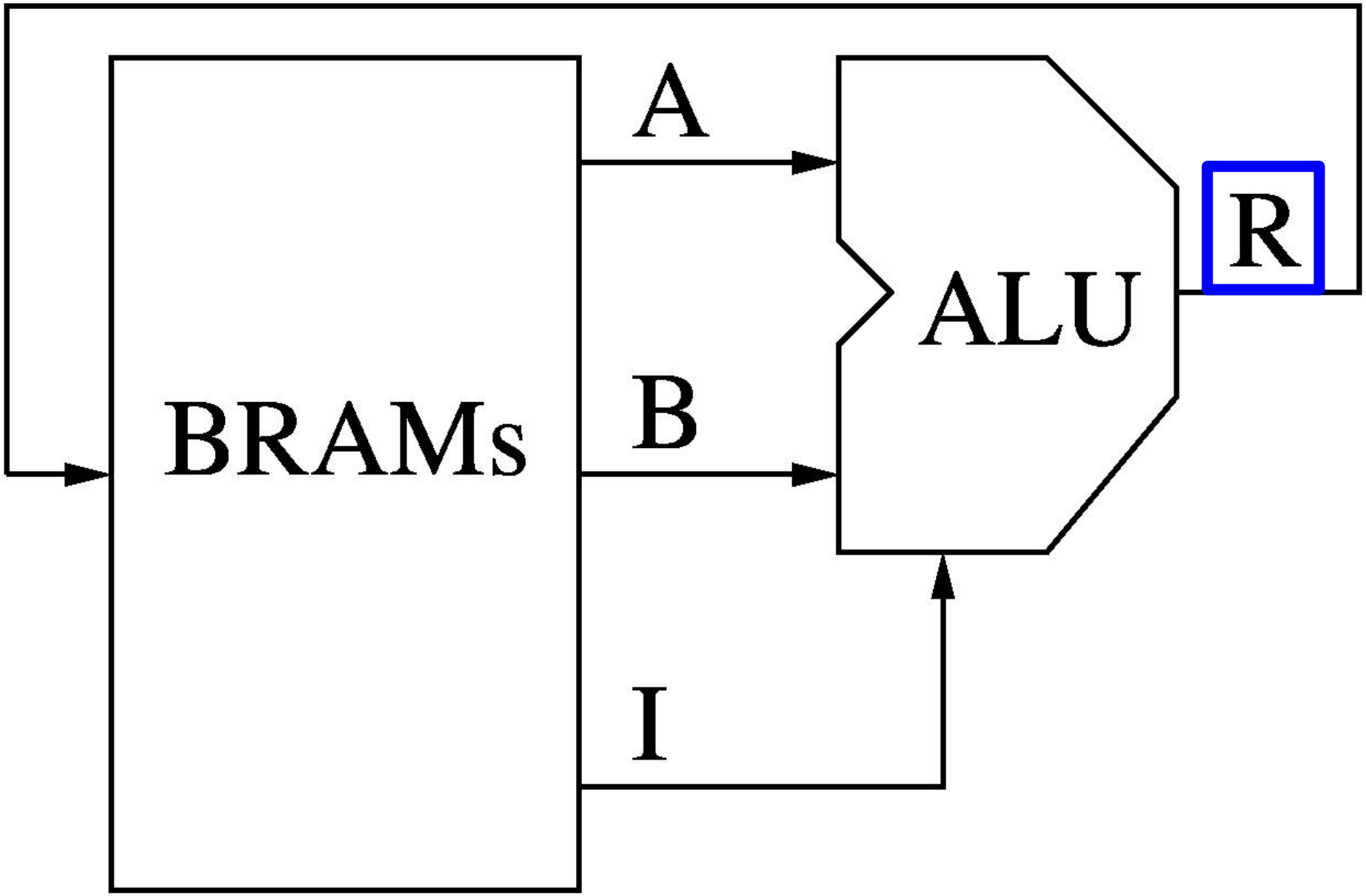
Overlay High-Level Architecture



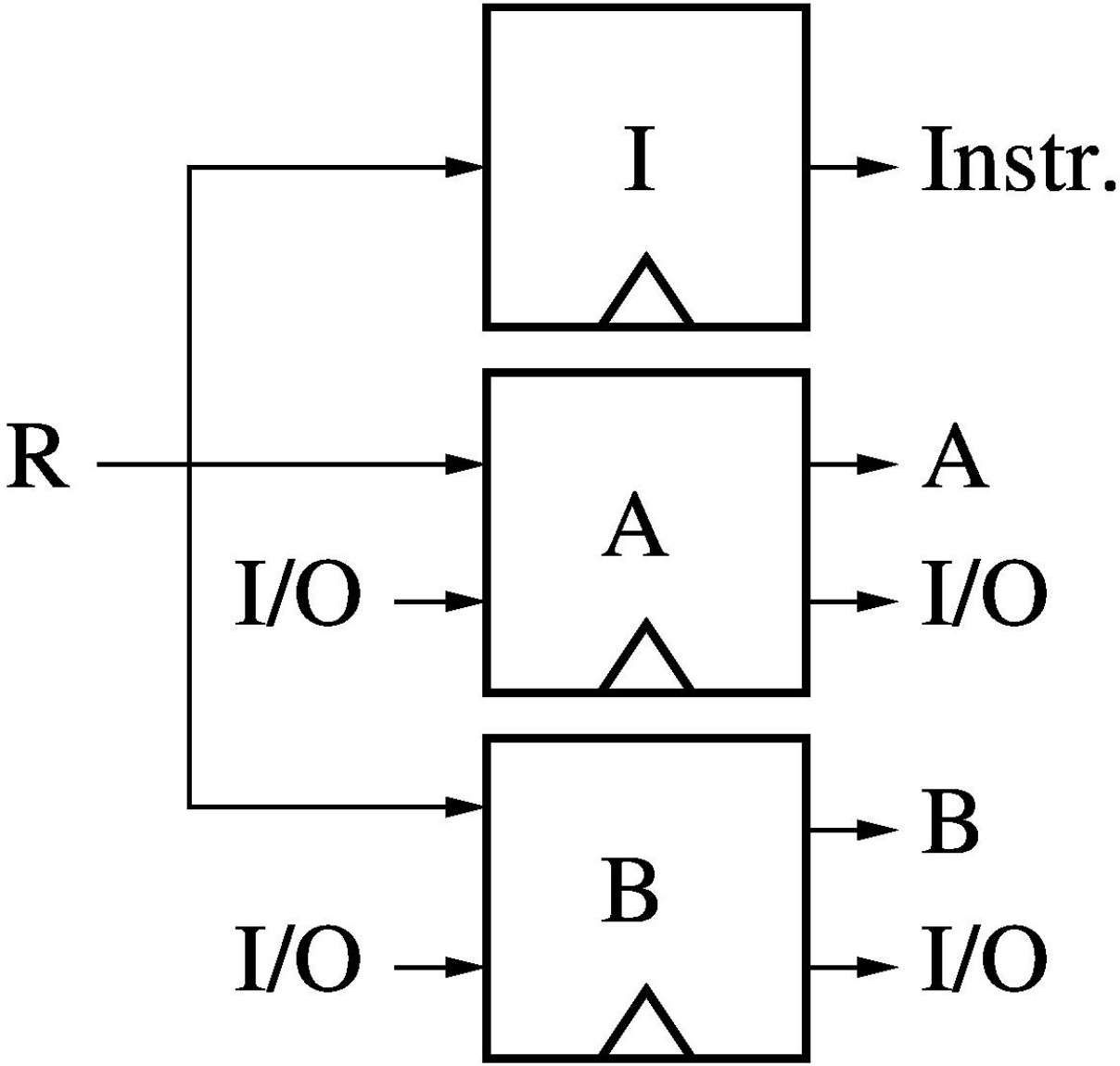
Overlay High-Level Architecture



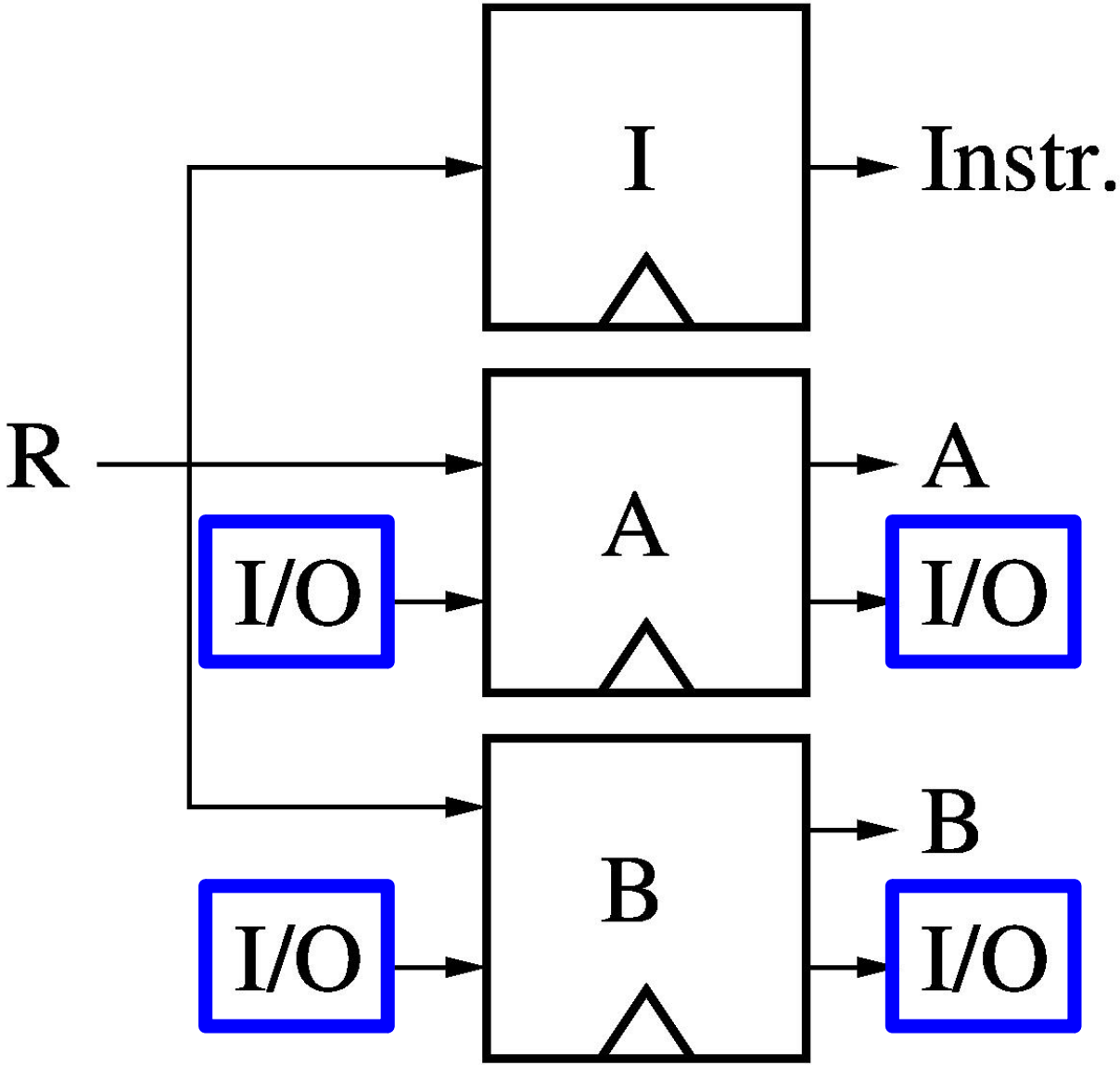
Overlay High-Level Architecture



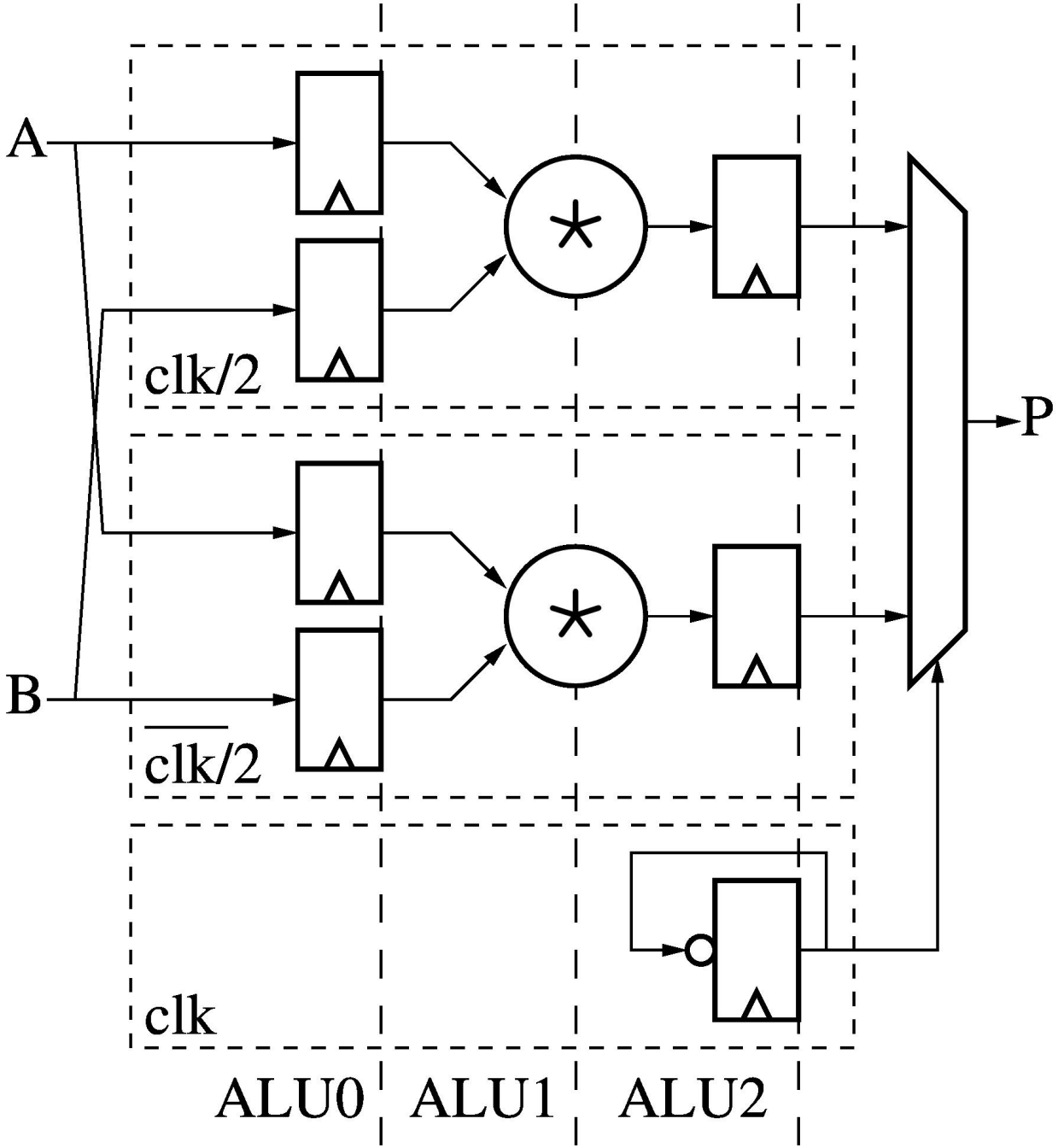
Memory High-Level Architecture



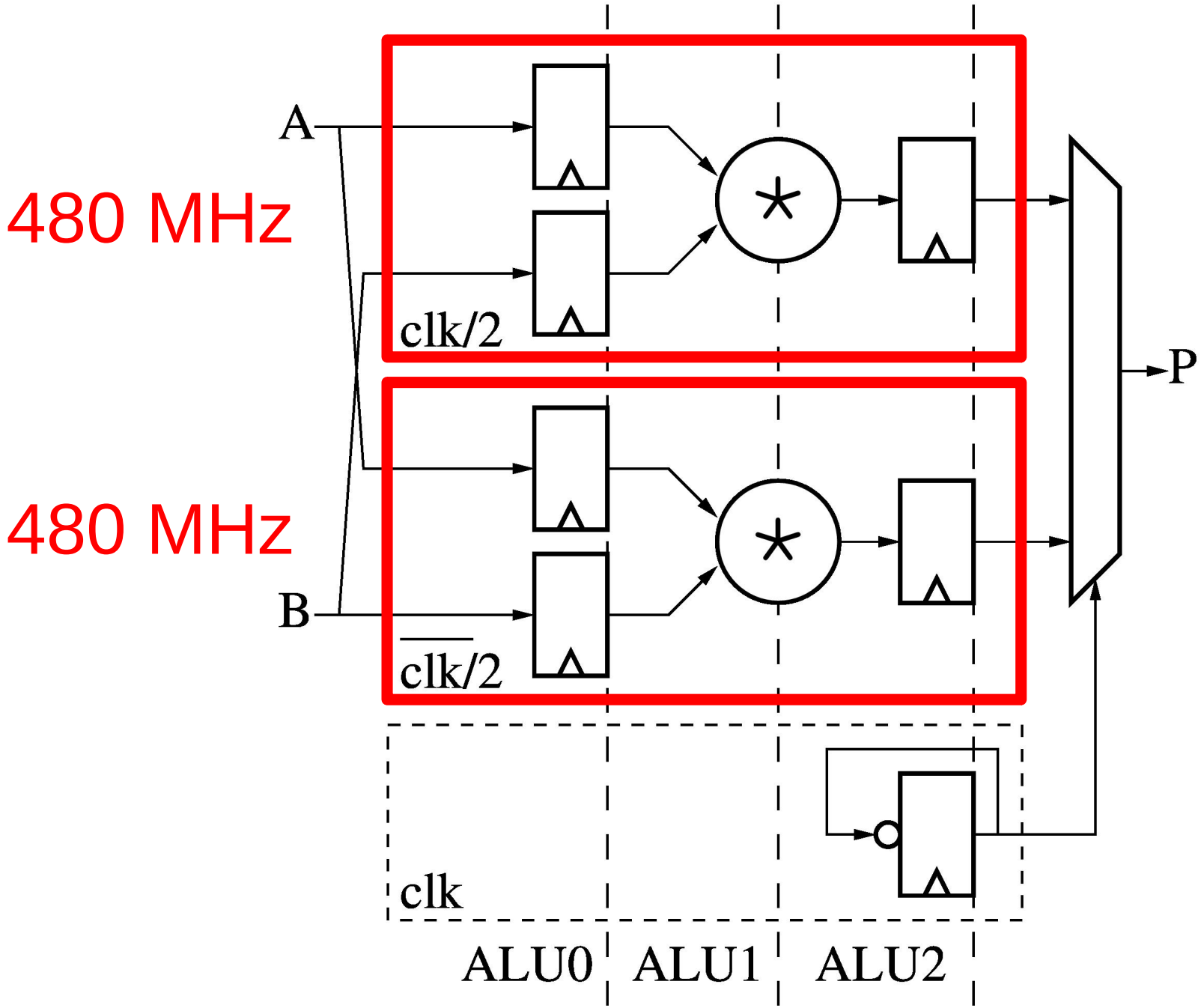
Memory High-Level Architecture



Dual-Pipeline Multiplier

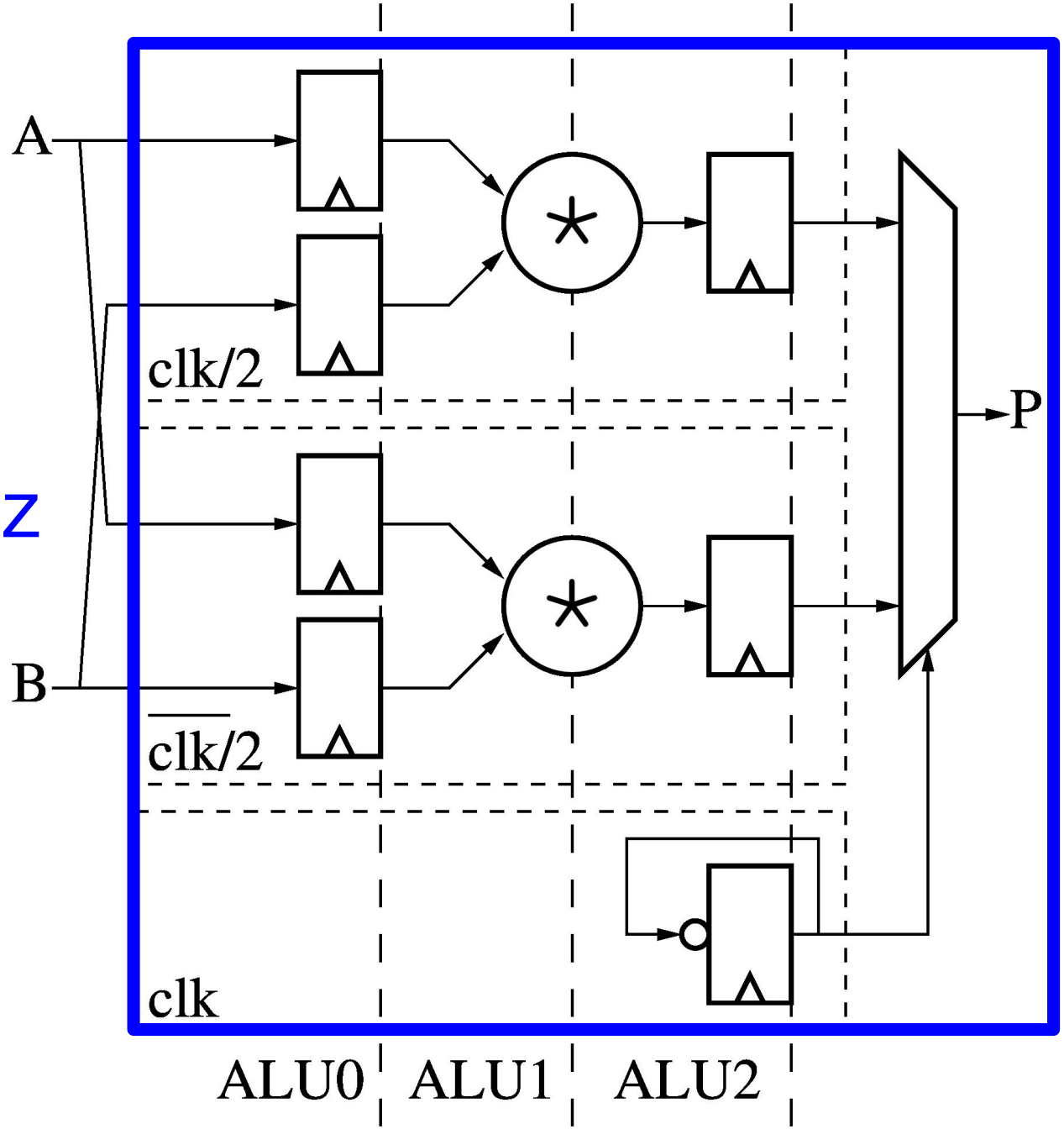


Dual-Pipeline Multiplier

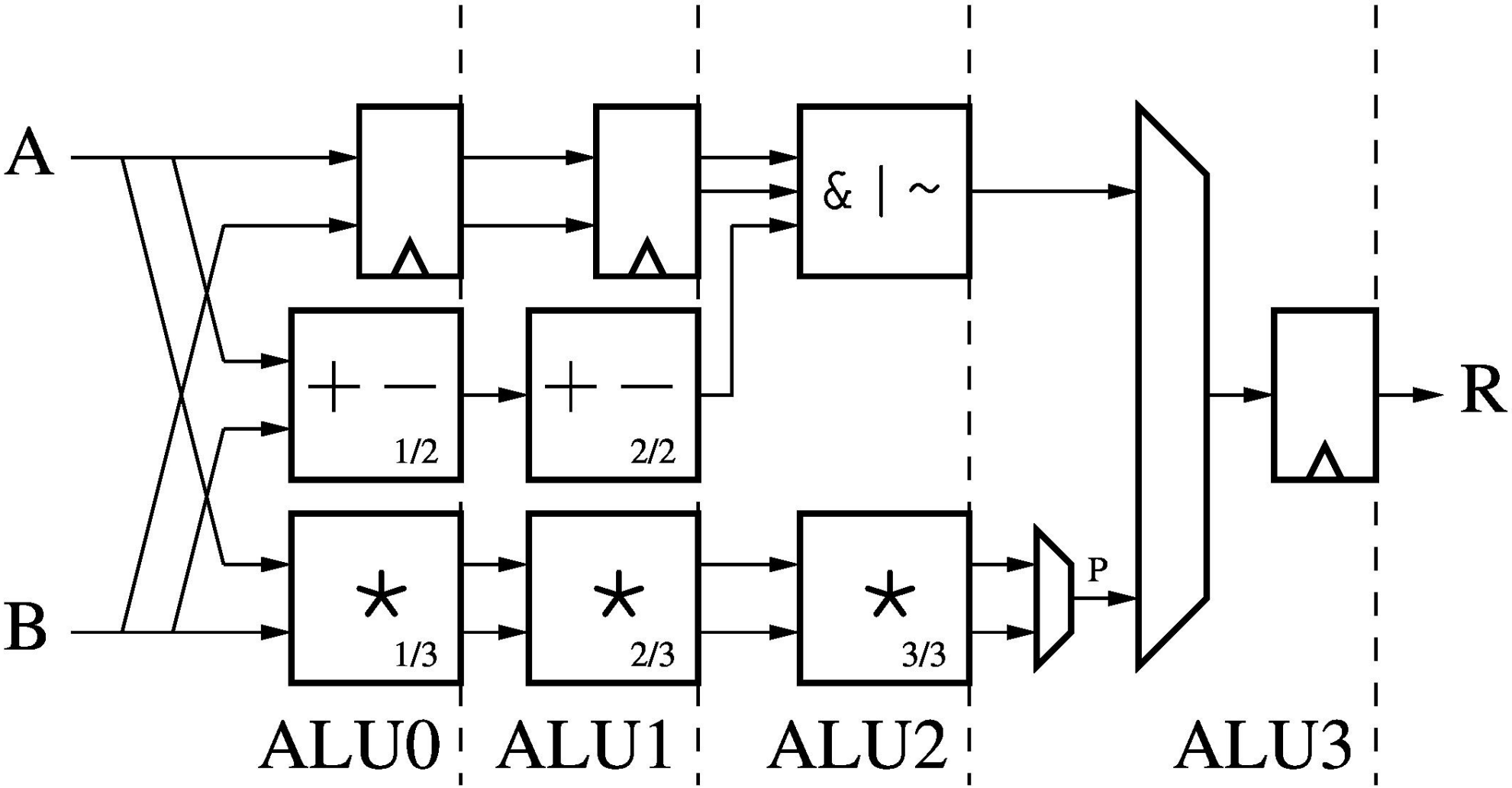


Dual-Pipeline Multiplier

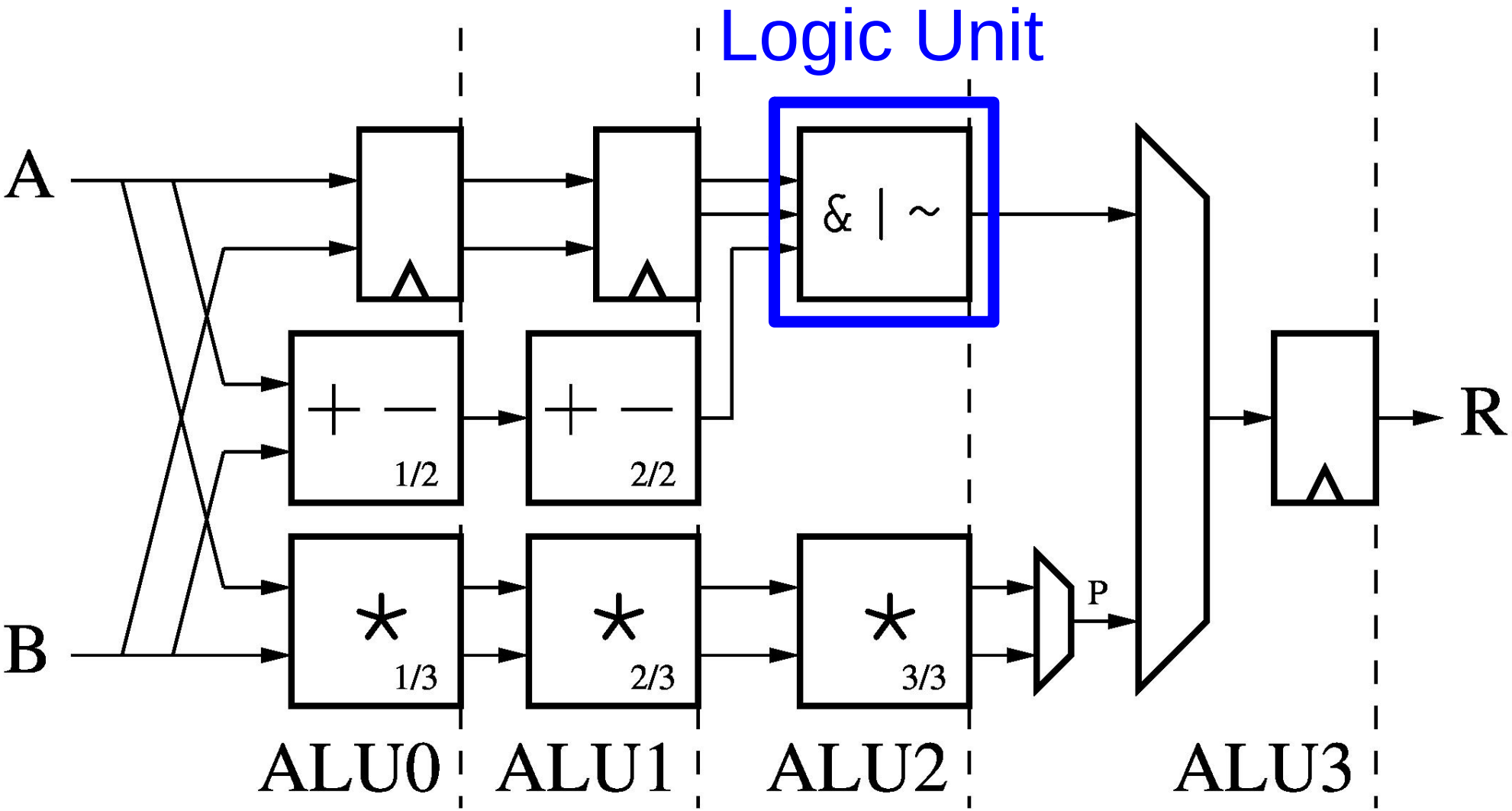
600 MHz



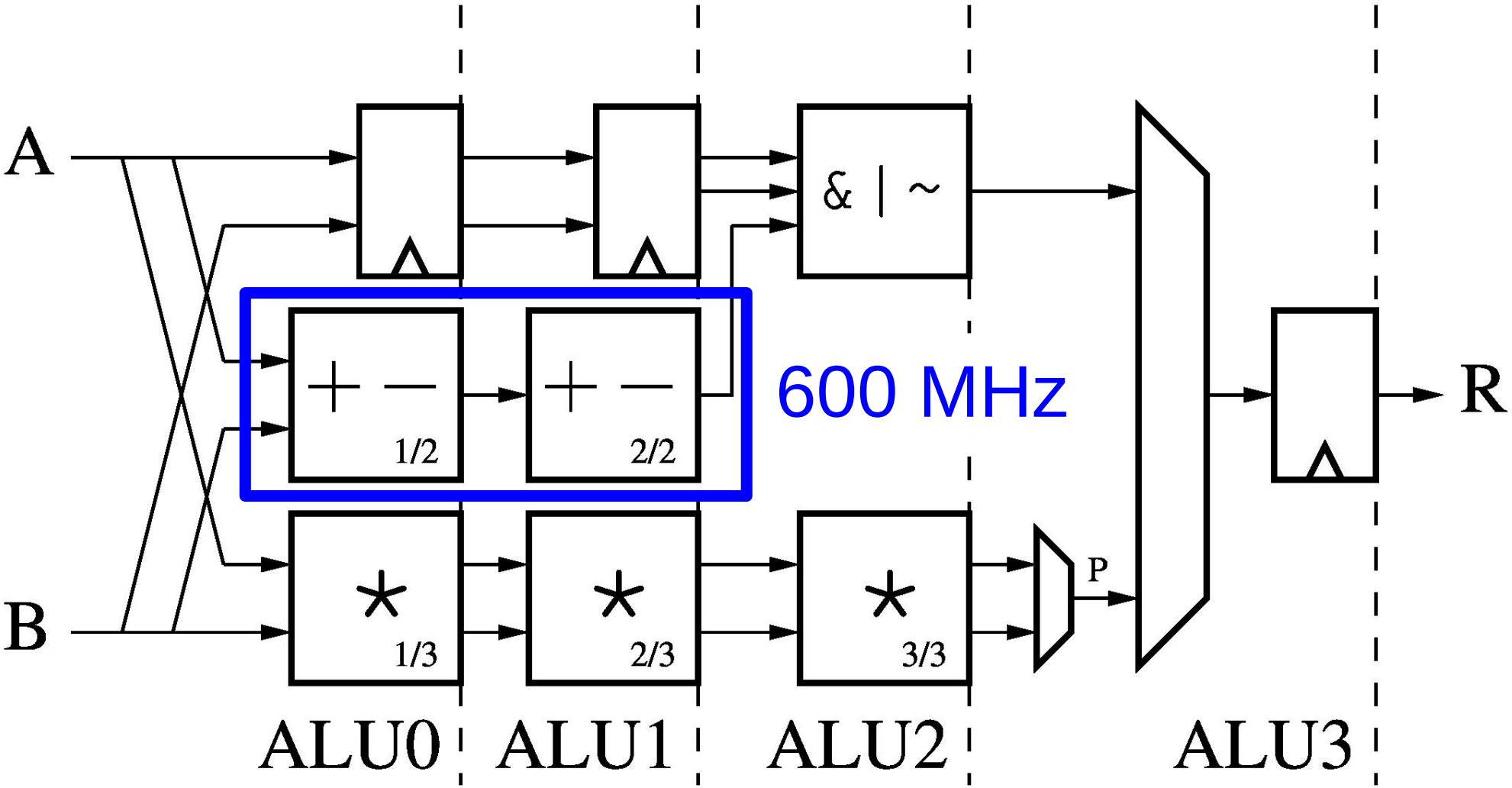
Fully-Pipelined ALU



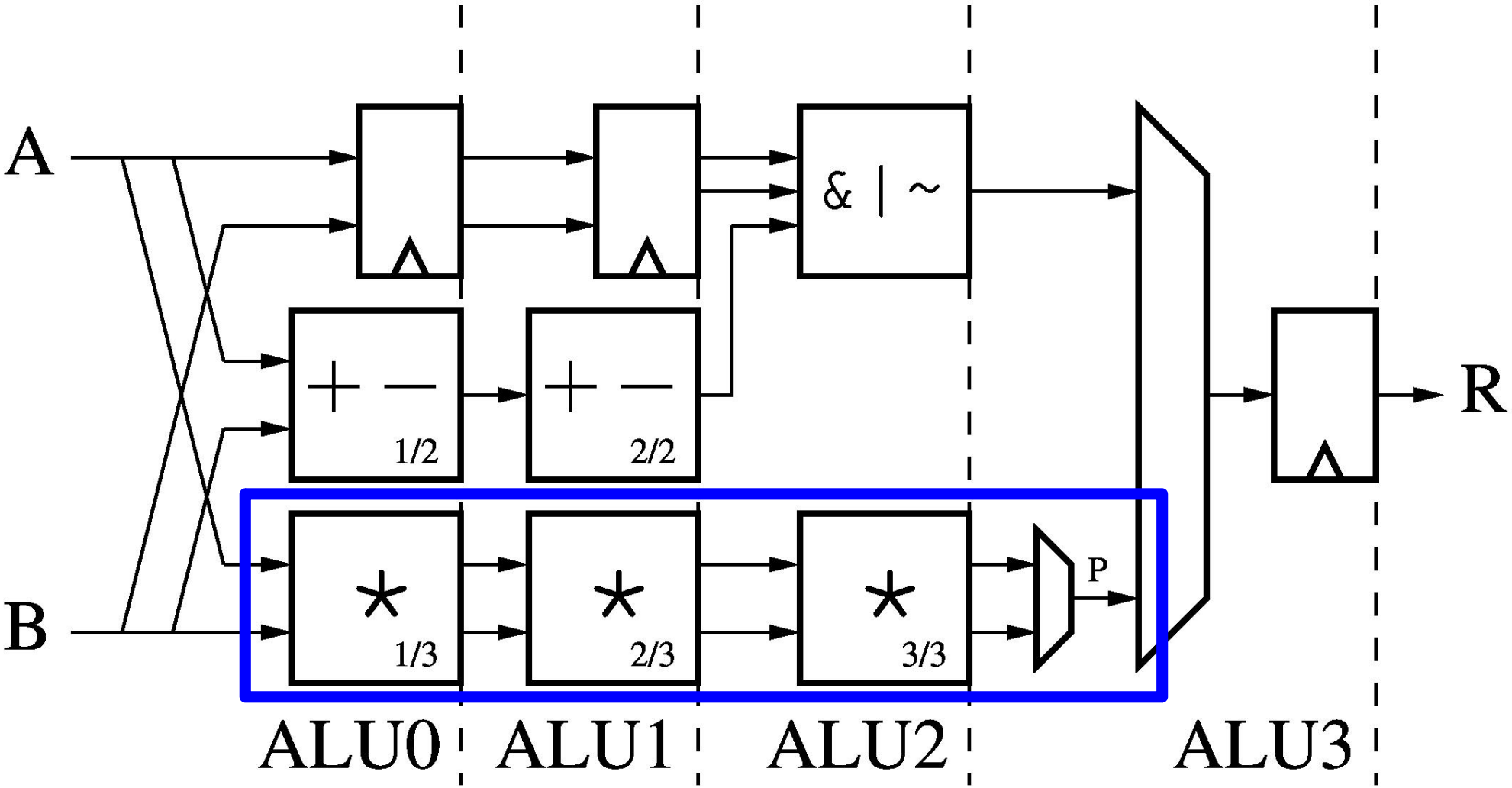
Fully-Pipelined ALU



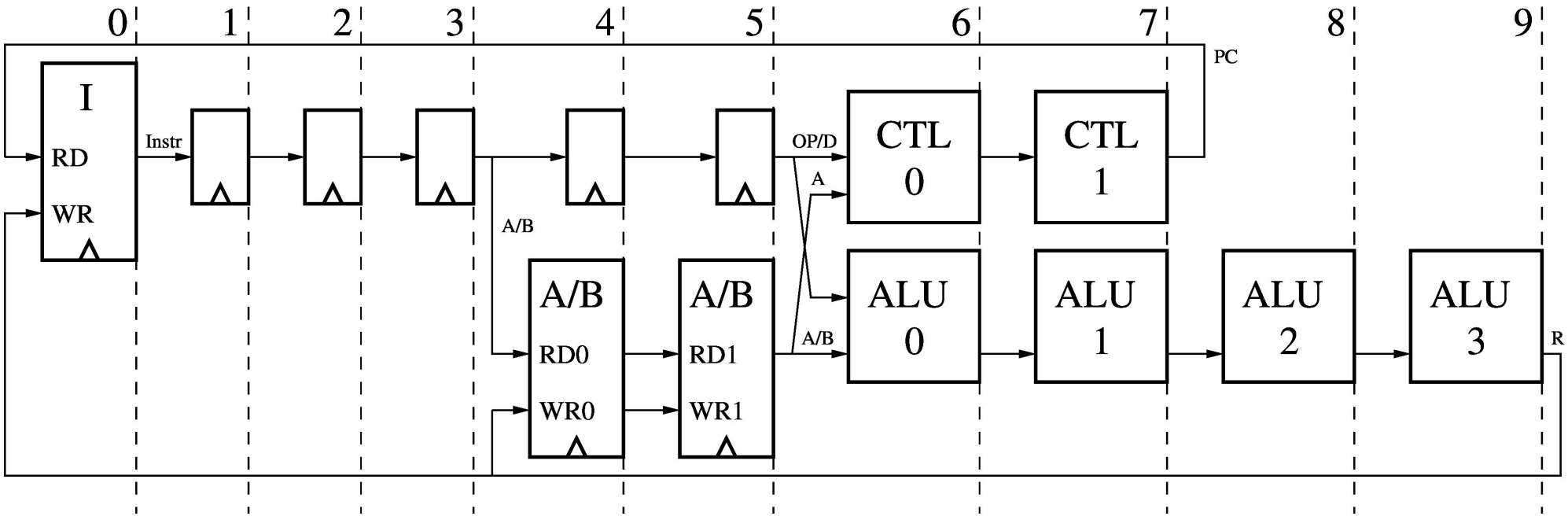
Fully-Pipelined ALU



Fully-Pipelined ALU

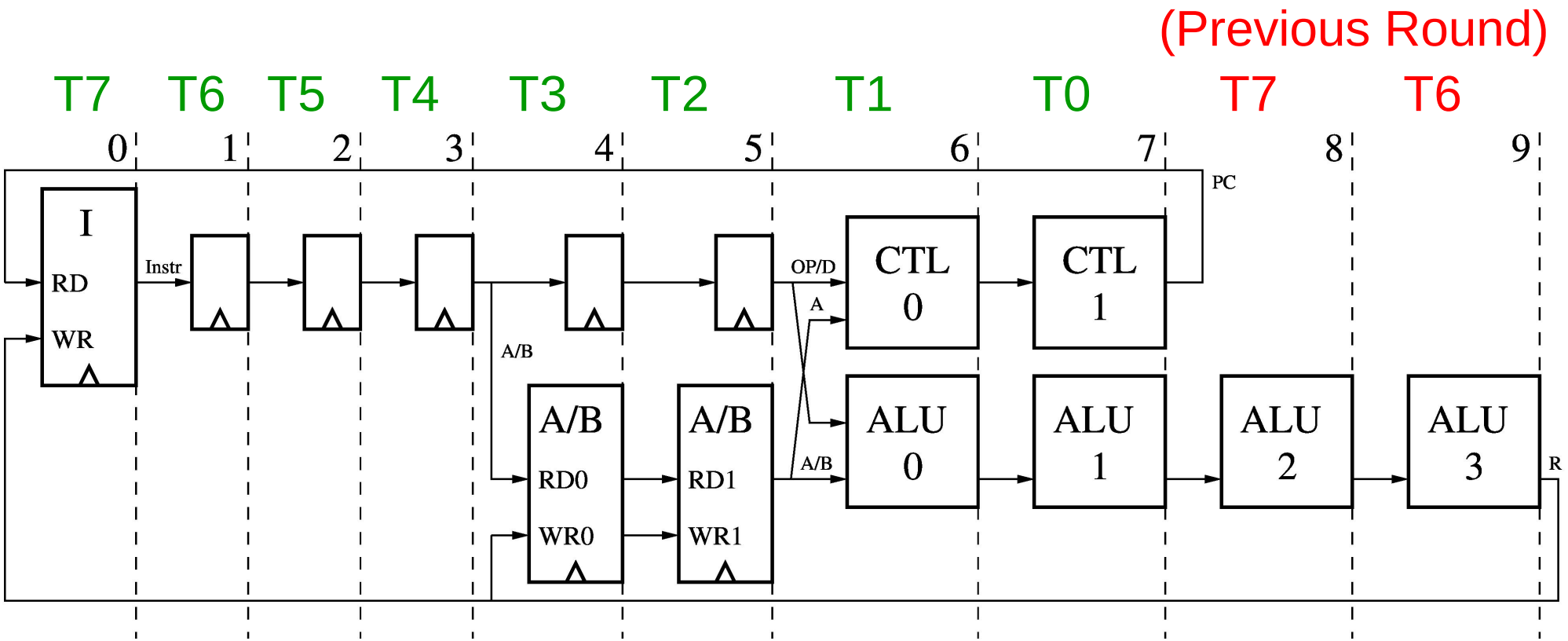


Octavo Soft-Processor



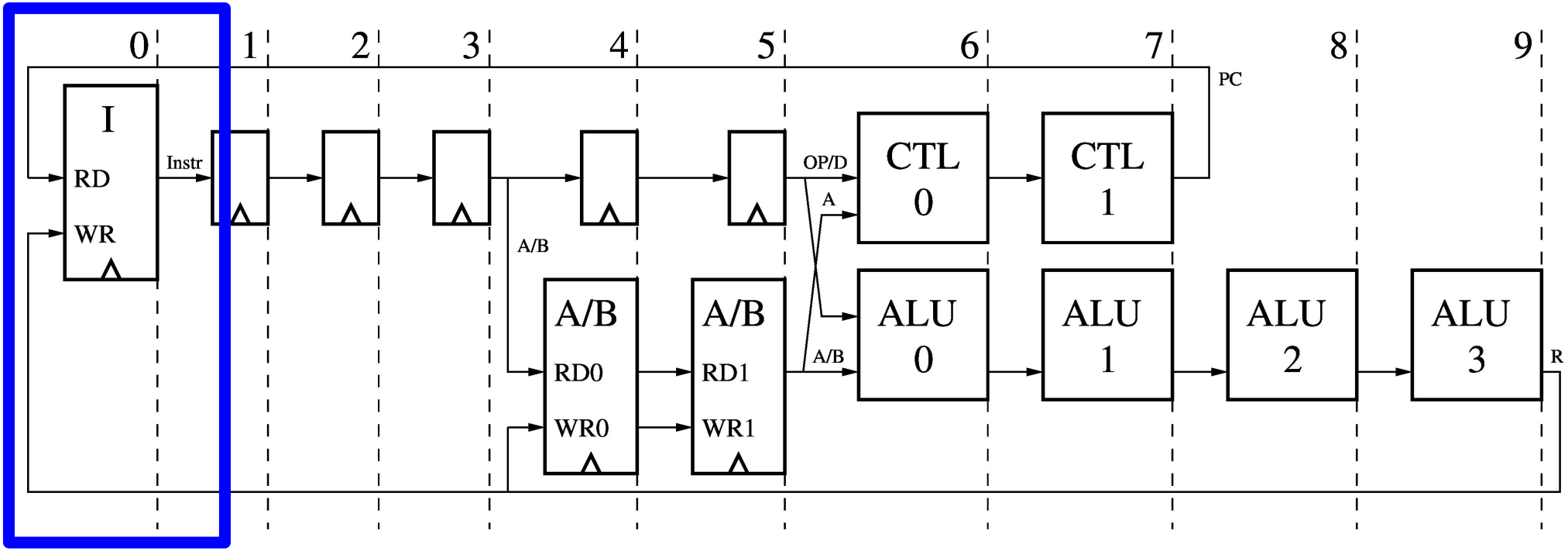
- Reaches 550 MHz on Stratix IV FPGA
- 8 threads (fixed round-robin)
- 1024 36-bit integer words for each I/A/B memory

Octavo Soft-Processor

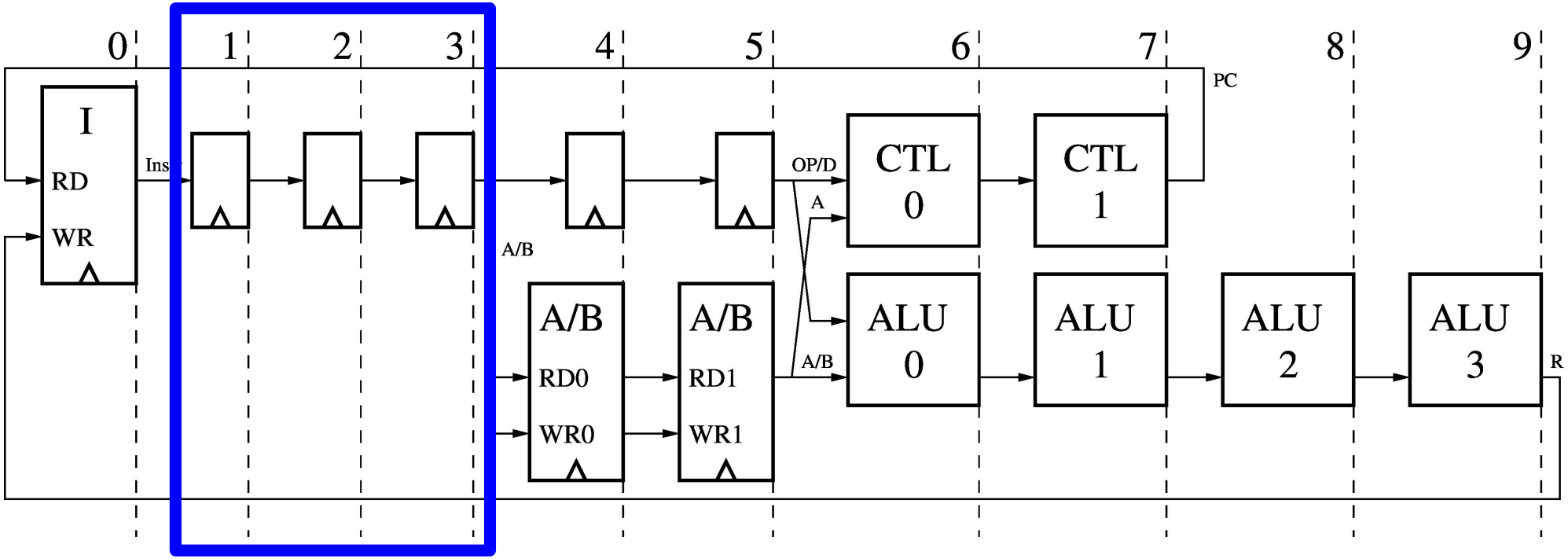


- Reaches 550 MHz on Stratix IV FPGA
- 8 threads (fixed round-robin)
- 1024 36-bit integer words for each I/A/B memory

Instruction Memory

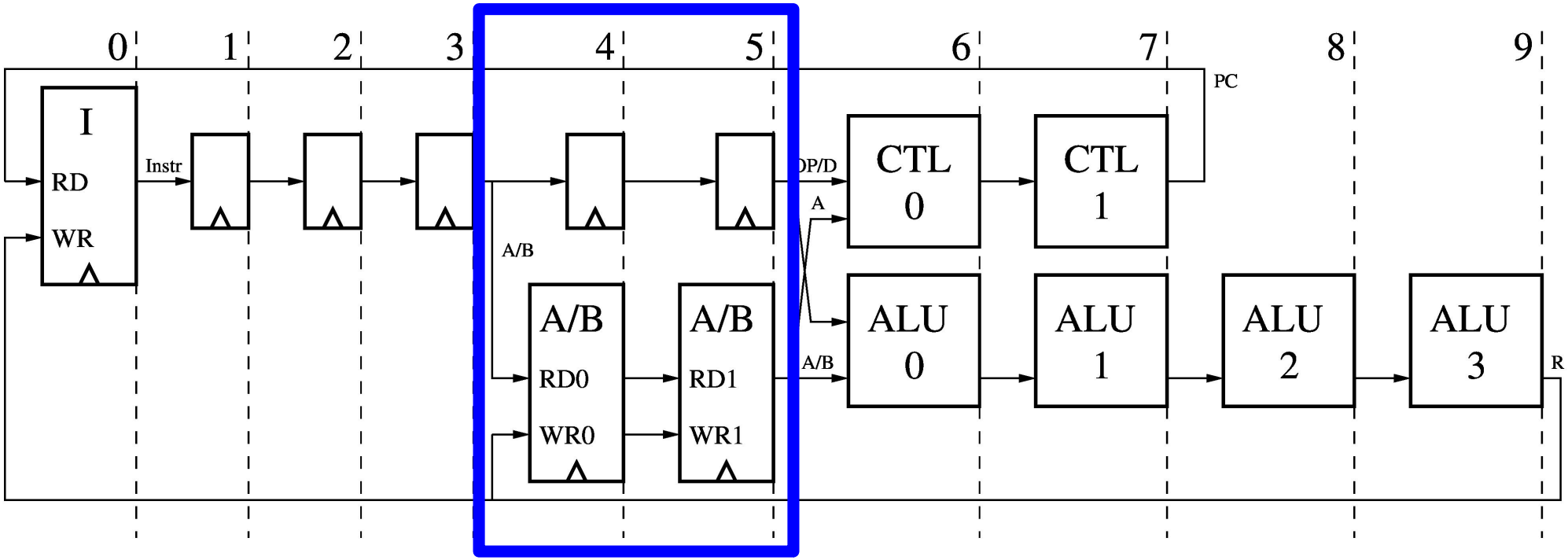


Empty Pipeline Stages



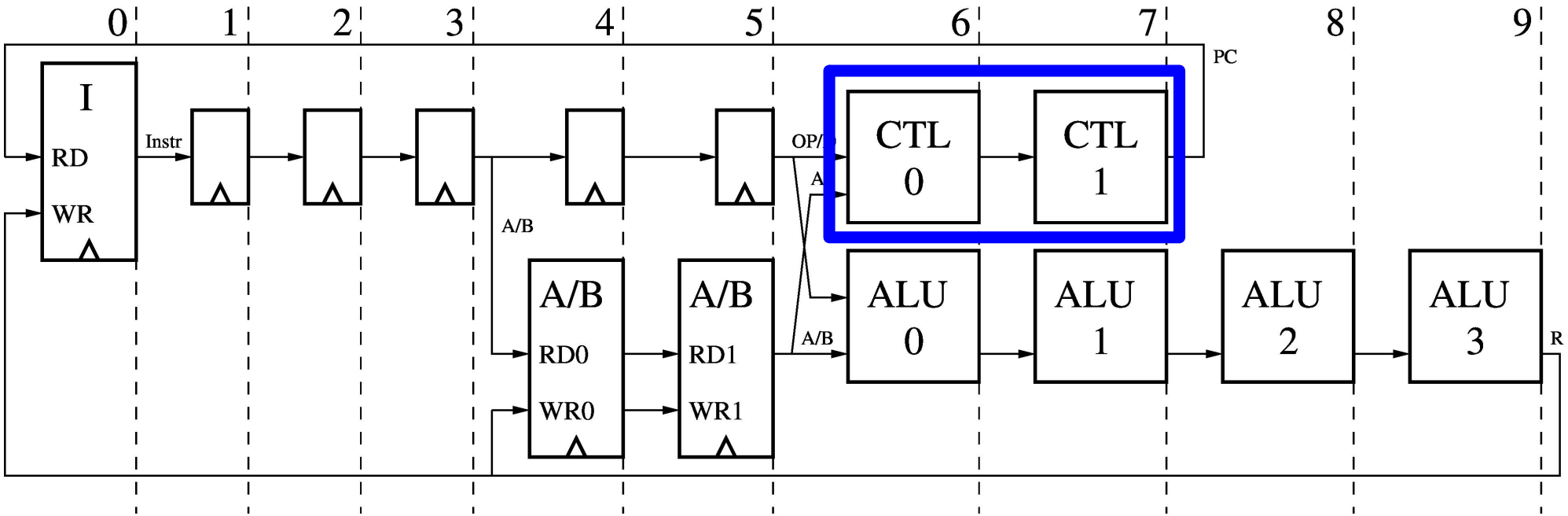
- Derived from BRAM self-loop characterization
- Used for special functions later...

A and B Data Memories



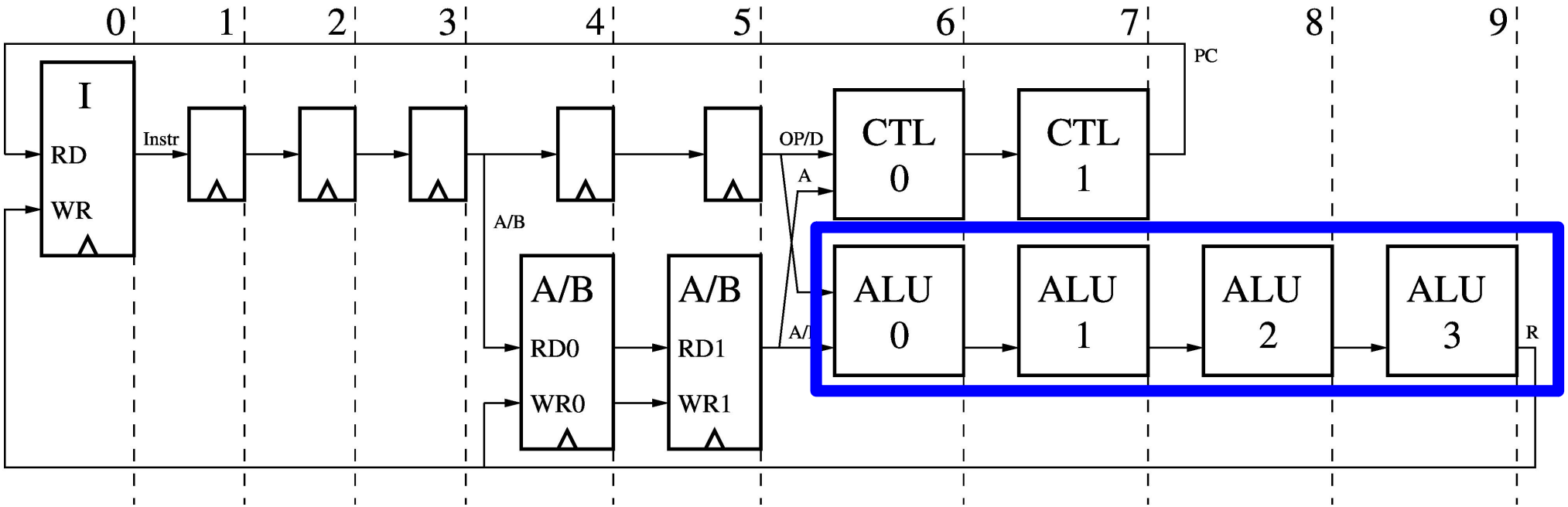
- Memory-mapped I/O ports for Accelerators

Controller



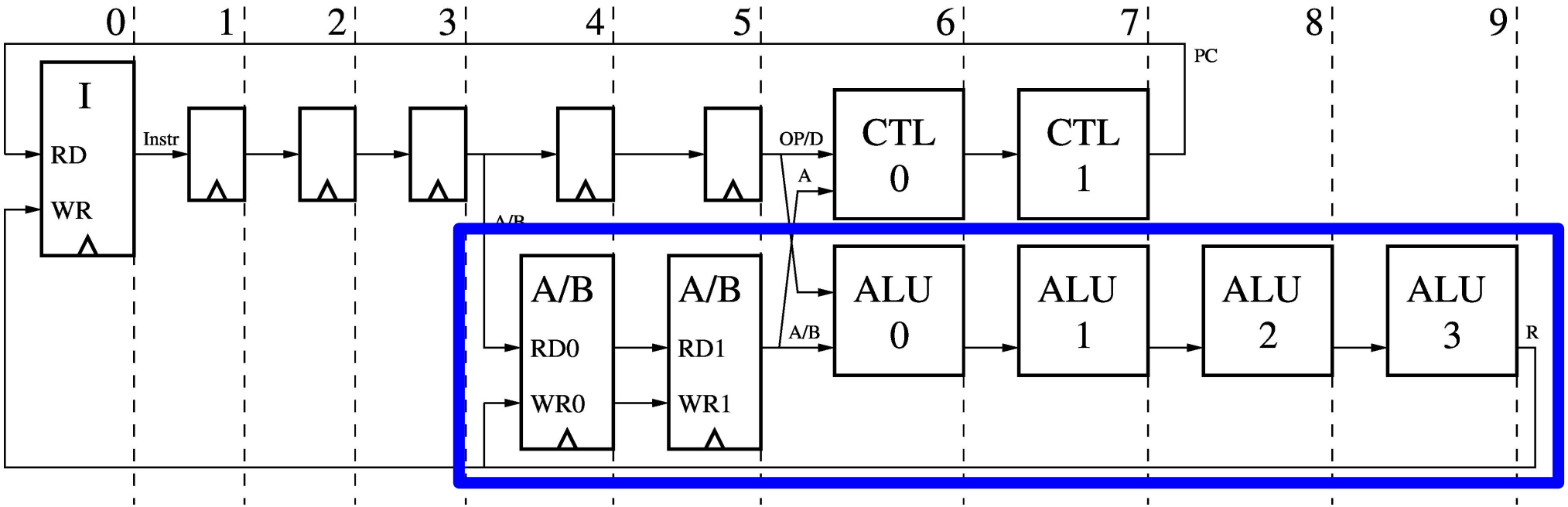
- Computes next PC for each thread (8 PCs)

ALU



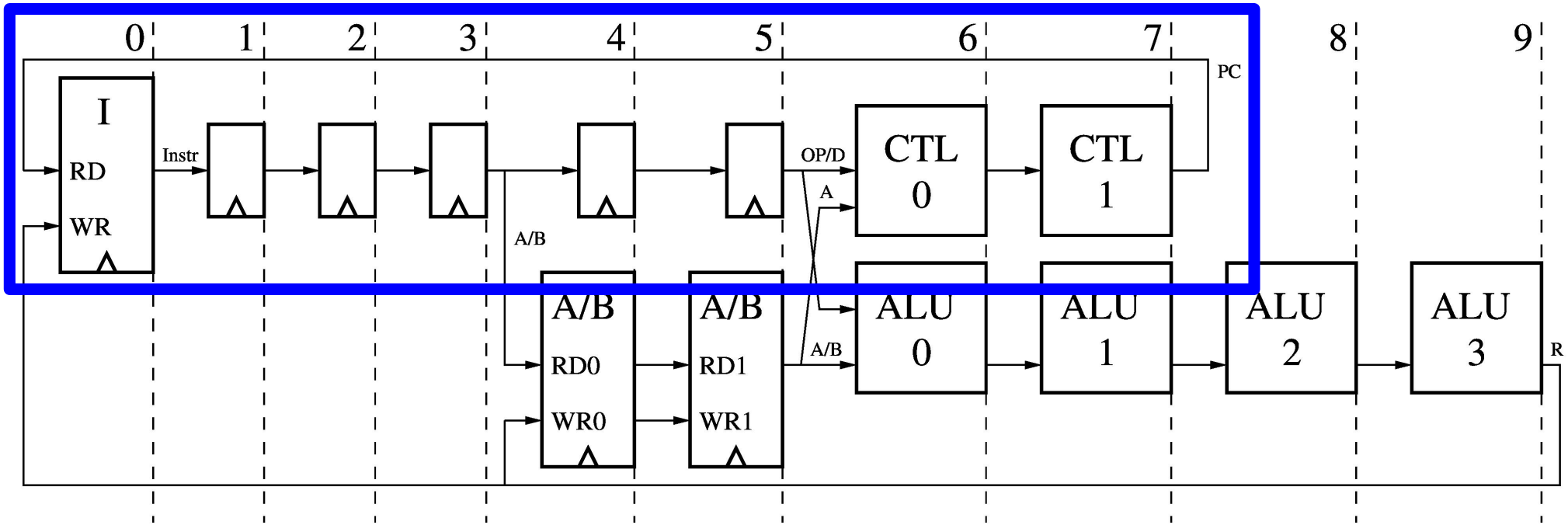
- Output (R) written to all memories

Data Path



- 8 stages (2 read, 4 compute, 2 write)

Control Path



- 8 stages to match Data Path
- Offset due to empty stages (1,2,3)
- 1-cycle RAW hazard from ALU to Instr. Mem.

Table 3.1: Octavo's Instruction Word Format.

Size:	4 bits	a bits	a bits	a bits
Field:	Opcode (OP)	Destination (D)	Source (A)	Source (B)

Table 3.2: Octavo's Instruction Set and Opcode Encoding.

Mnemonic	Opcode	Action
Logic Unit		
XOR	0000	$D \leftarrow A \oplus B$
AND	0001	$D \leftarrow A \wedge B$
OR	0010	$D \leftarrow A \vee B$
SUB	0011	$D \leftarrow A - B$
ADD	0100	$D \leftarrow A + B$
—	0101	*(Unused, for expansion)*
—	0110	*(Unused, for expansion)*
—	0111	*(Unused, for expansion)*
Multiplier		
MHS	1000	$D \leftarrow A \cdot B$ (High Word Signed)
MLS	1001	$D \leftarrow A \cdot B$ (Low Word Signed)
MHU	1010	$D \leftarrow A \cdot B$ (High Word Unsigned)
Controller		
JMP	1011	$PC \leftarrow D$
JZE	1100	if $(A = 0)$ $PC \leftarrow D$
JNZ	1101	if $(A \neq 0)$ $PC \leftarrow D$
JPO	1110	if $(A \geq 0)$ $PC \leftarrow D$
JNE	1111	if $(A < 0)$ $PC \leftarrow D$

Table 3.1: Octavo's Instruction Word Format.

Size:	4 bits	a bits	a bits	a bits
Field:	Opcode (OP)	Destination (D)	Source (A)	Source (B)

Table 3.2: Octavo's Instruction Set and Opcode Encoding.

Mnemonic	Opcode	Action
Logic Unit		
XOR	0000	$D \leftarrow A \oplus B$
AND	0001	$D \leftarrow A \wedge B$
OR	0010	$D \leftarrow A \vee B$
SUB	0011	$D \leftarrow A - B$
ADD	0100	$D \leftarrow A + B$
—	0101	*(Unused, for expansion)*
—	0110	*(Unused, for expansion)*
—	0111	*(Unused, for expansion)*
Multiplier		
MHS	1000	$D \leftarrow A \cdot B$ (High Word Signed)
MLS	1001	$D \leftarrow A \cdot B$ (Low Word Signed)
MHU	1010	$D \leftarrow A \cdot B$ (High Word Unsigned)
Controller		
JMP	1011	$PC \leftarrow D$
JZE	1100	if $(A = 0)$ $PC \leftarrow D$
JNZ	1101	if $(A \neq 0)$ $PC \leftarrow D$
JPO	1110	if $(A \geq 0)$ $PC \leftarrow D$
JNE	1111	if $(A < 0)$ $PC \leftarrow D$

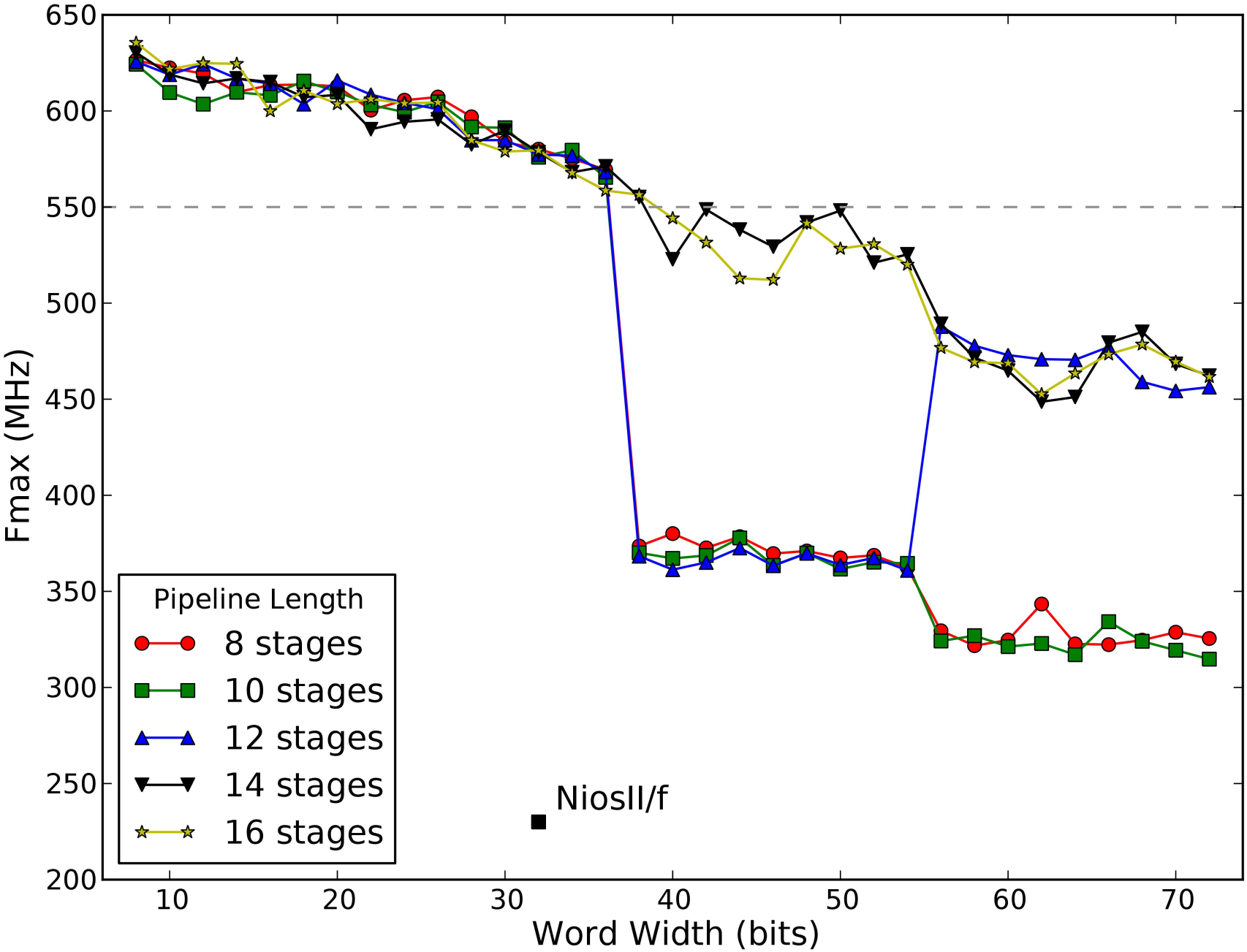
Table 3.1: Octavo's Instruction Word Format.

Size:	4 bits	a bits	a bits	a bits
Field:	Opcode (OP)	Destination (D)	Source (A)	Source (B)

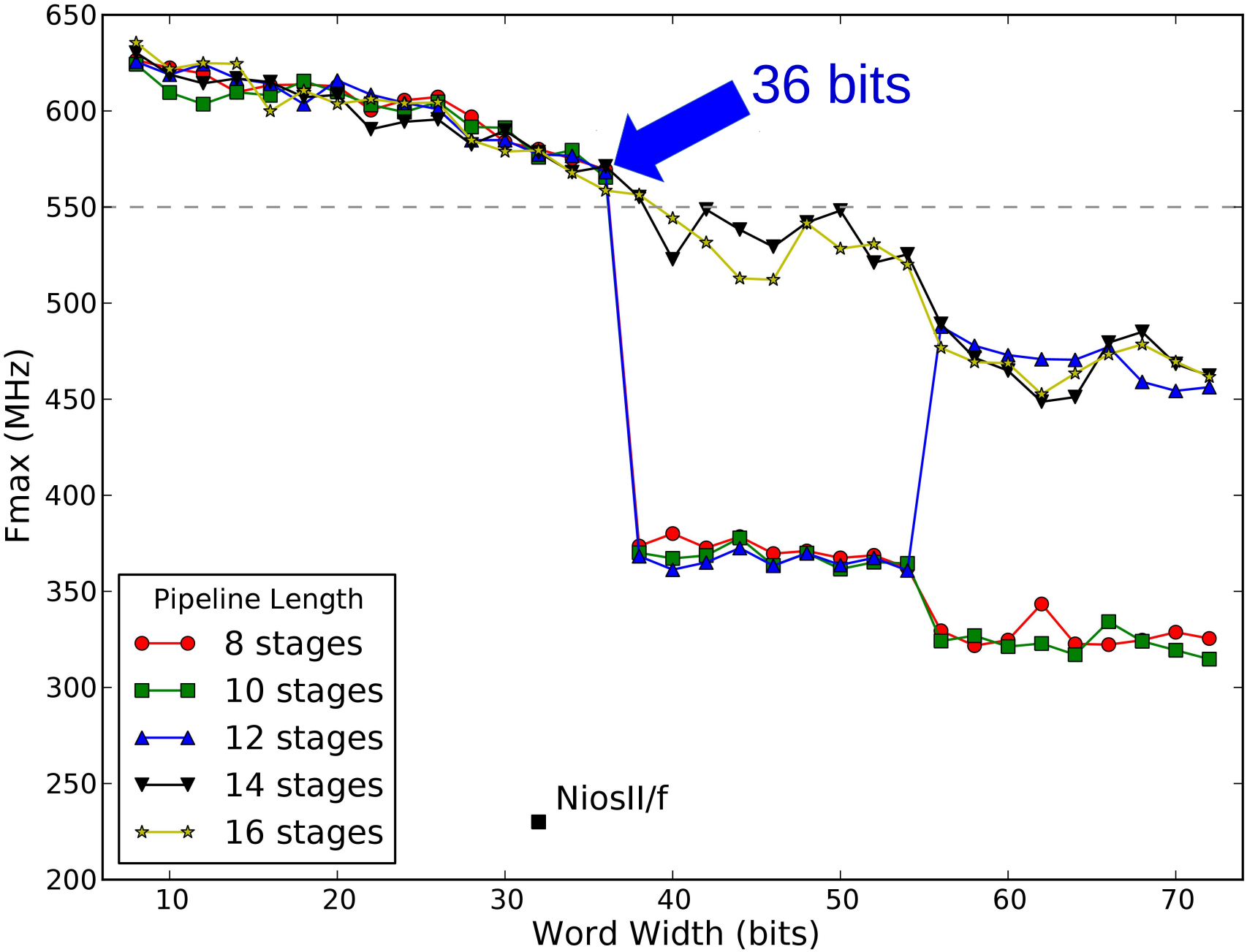
Table 3.2: Octavo's Instruction Set and Opcode Encoding.

Mnemonic	Opcode	Action
	Logic Unit	
XOR	0000	$D \leftarrow A \oplus B$
AND	0001	$D \leftarrow A \wedge B$
OR	0010	$D \leftarrow A \vee B$
SUB	0011	$D \leftarrow A - B$
ADD	0100	$D \leftarrow A + B$
—	0101	*(Unused, for expansion)*
—	0110	*(Unused, for expansion)*
—	0111	*(Unused, for expansion)*
	Multiplier	
MHS	1000	$D \leftarrow A \cdot B$ (High Word Signed)
MLS	1001	$D \leftarrow A \cdot B$ (Low Word Signed)
MHU	1010	$D \leftarrow A \cdot B$ (High Word Unsigned)
	Controller	
JMP	1011	$PC \leftarrow D$
JZE	1100	if $(A = 0)$ $PC \leftarrow D$
JNZ	1101	if $(A \neq 0)$ $PC \leftarrow D$
JPO	1110	if $(A \geq 0)$ $PC \leftarrow D$
JNE	1111	if $(A < 0)$ $PC \leftarrow D$

High Fmax over Design Space



High Fmax over Design Space



Tiling Overlay Architectures

- Tiling: duplicating in 2-D for parallelism
  - Datapaths: SIMD
  - Processors: MIMD

Tiling Overlay Architectures

- Tiling: duplicating in 2-D for parallelism
  - Datapaths: SIMD
  - Processors: MIMD
- CAD optimizations now worsen performance!

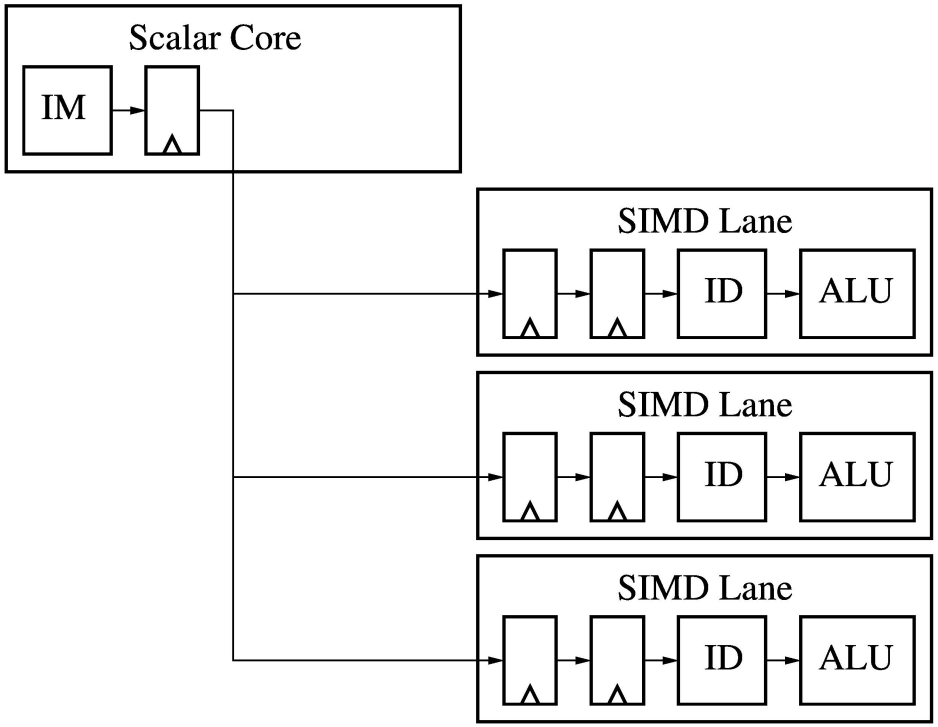
Tiling Overlay Architectures

- Tiling: duplicating in 2-D for parallelism
  - Datapaths: SIMD
  - Processors: MIMD
- CAD optimizations now worsen performance!
- Simple way to steer CAD tool
  - ...without source annotations or per-node CAD
    - R. Scoville, "*Register Duplication for Timing Closure*", Altera Wiki, 2011
  - ...without increasing CAD processing time

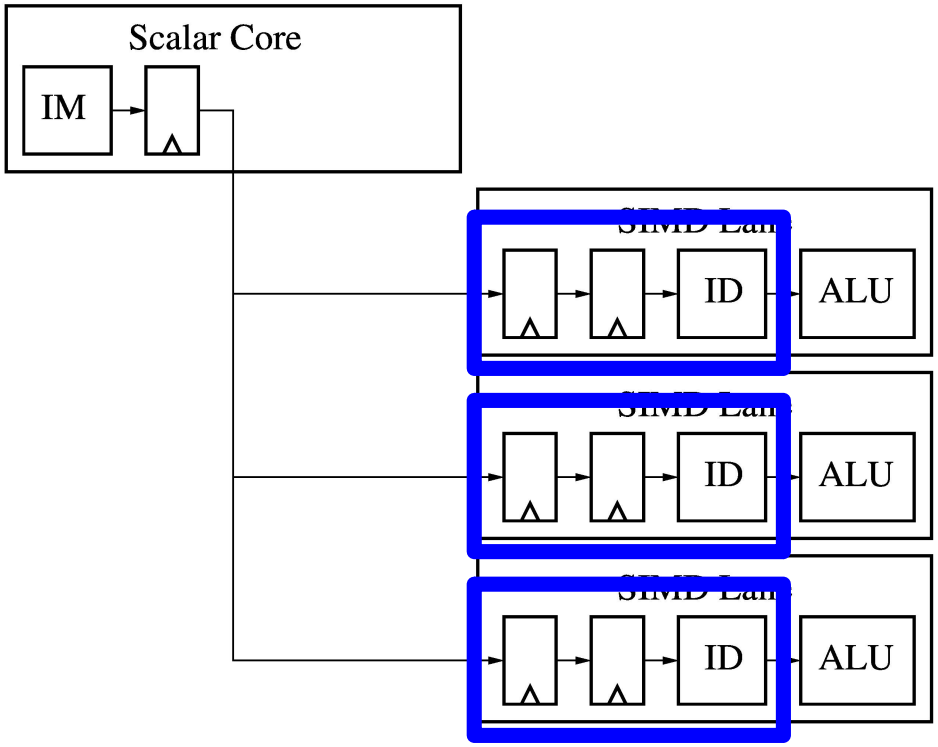
Tiling Overlay Architectures

- Tiling: duplicating in 2-D for parallelism
  - Datapaths: SIMD
  - Processors: MIMD
- CAD optimizations now worsen performance!
- Simple way to steer CAD tool
  - ...without source annotations or per-node CAD
    - R. Scoville, "*Register Duplication for Timing Closure*", Altera Wiki, 2011
  - ...without increasing CAD processing time
- Meshes of Scalar and SIMD Octavo Cores

Tiling Datapaths for SIMD

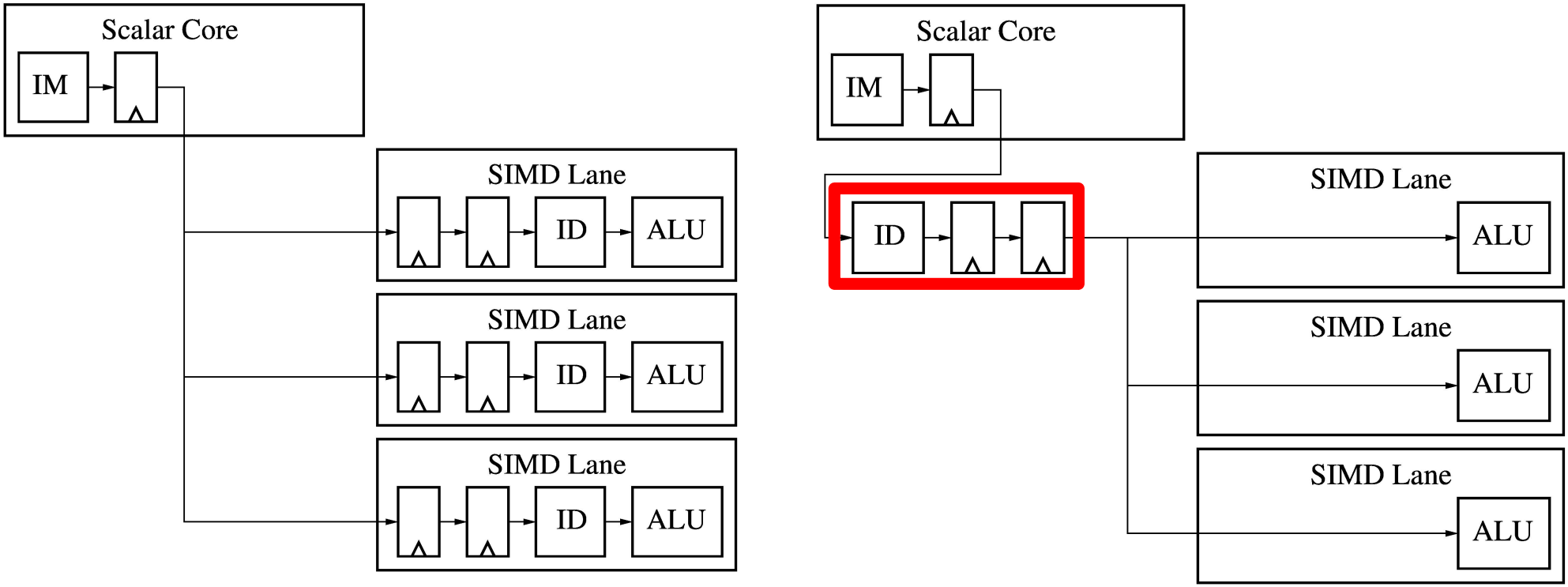


Multi-Local Logic



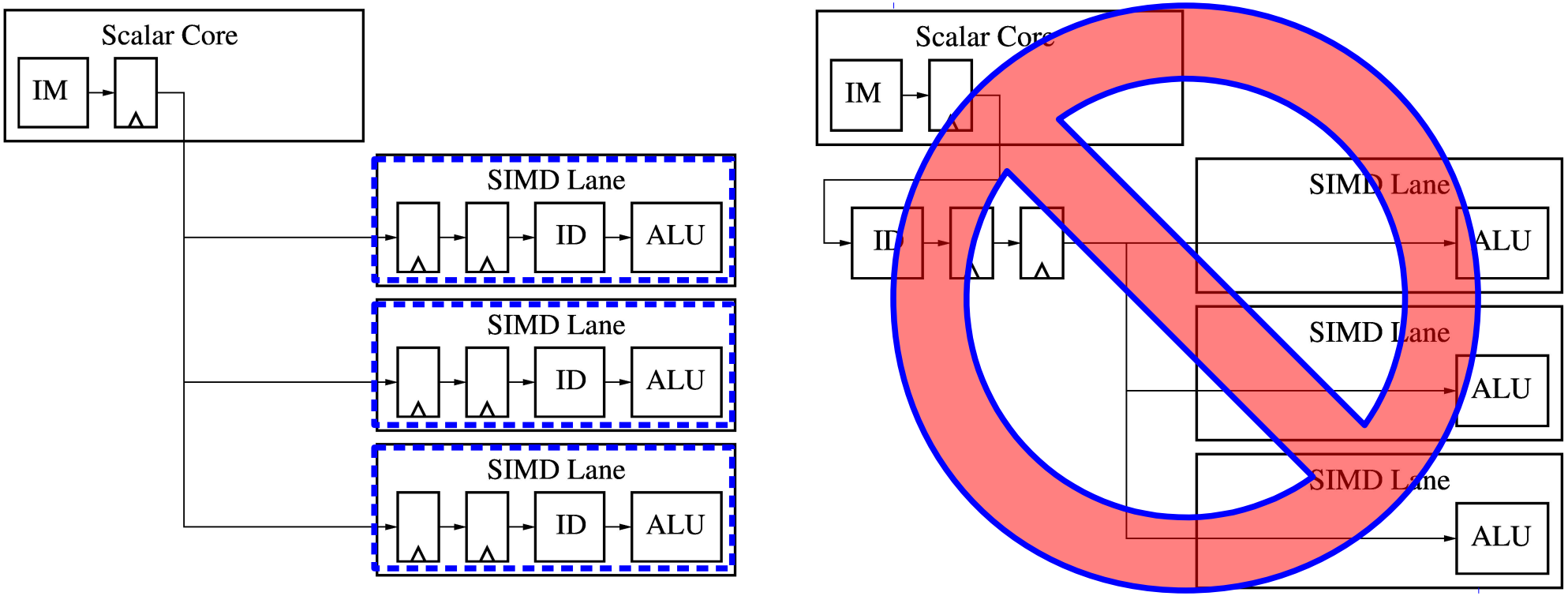
- Same simultaneous inputs and/or states
- CAD tool “de-duplicates” to save area

Harmful Optimization



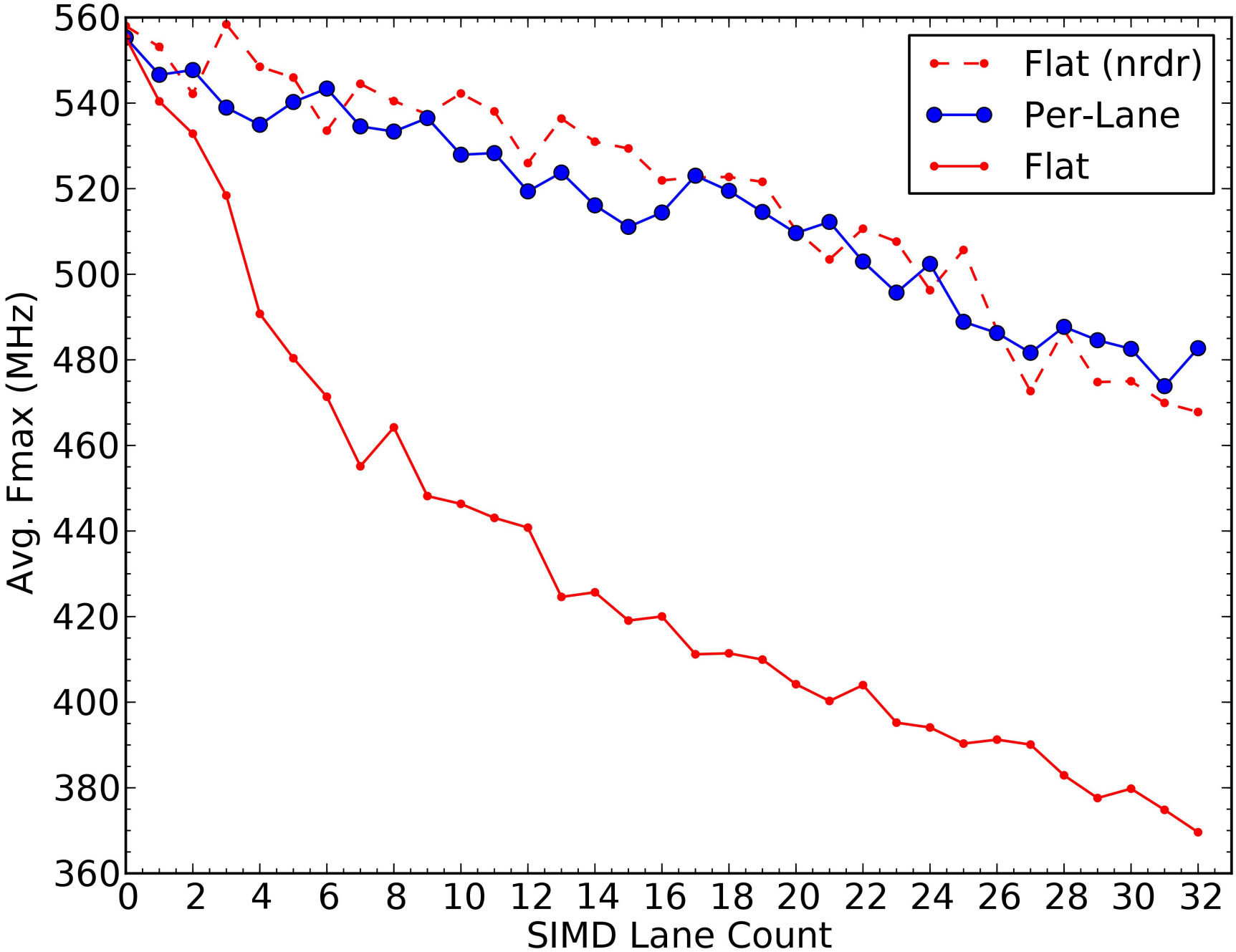
- Same simultaneous inputs and/or states
- CAD tool “de-duplicates” to save area
- But creates artificial critical paths!

Logical Partitioning

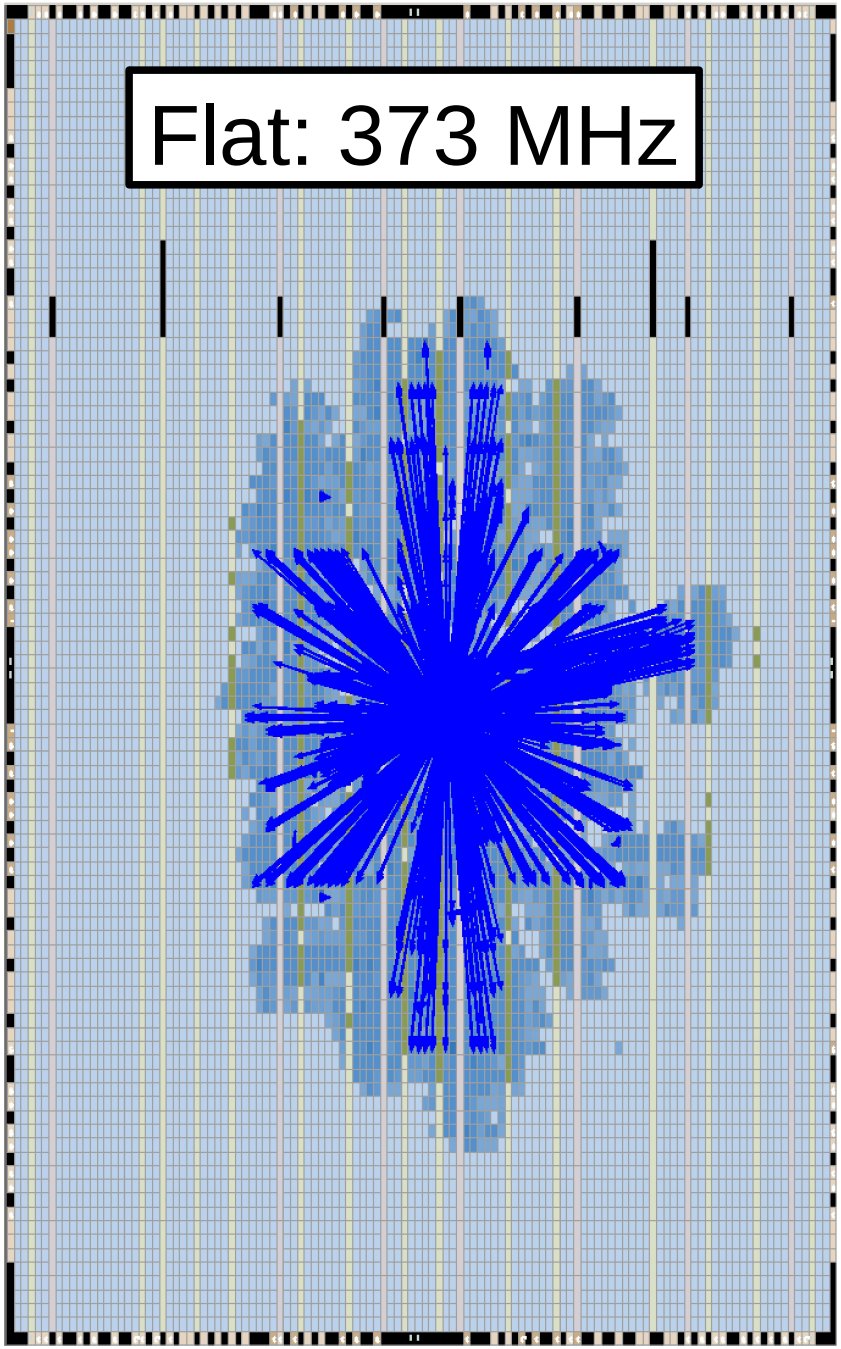


- Partition each Lane as a separate netlist
- Prevents optimizations across partitions
- Easily avoids harmful optimizations...
- ...without preventing useful ones!

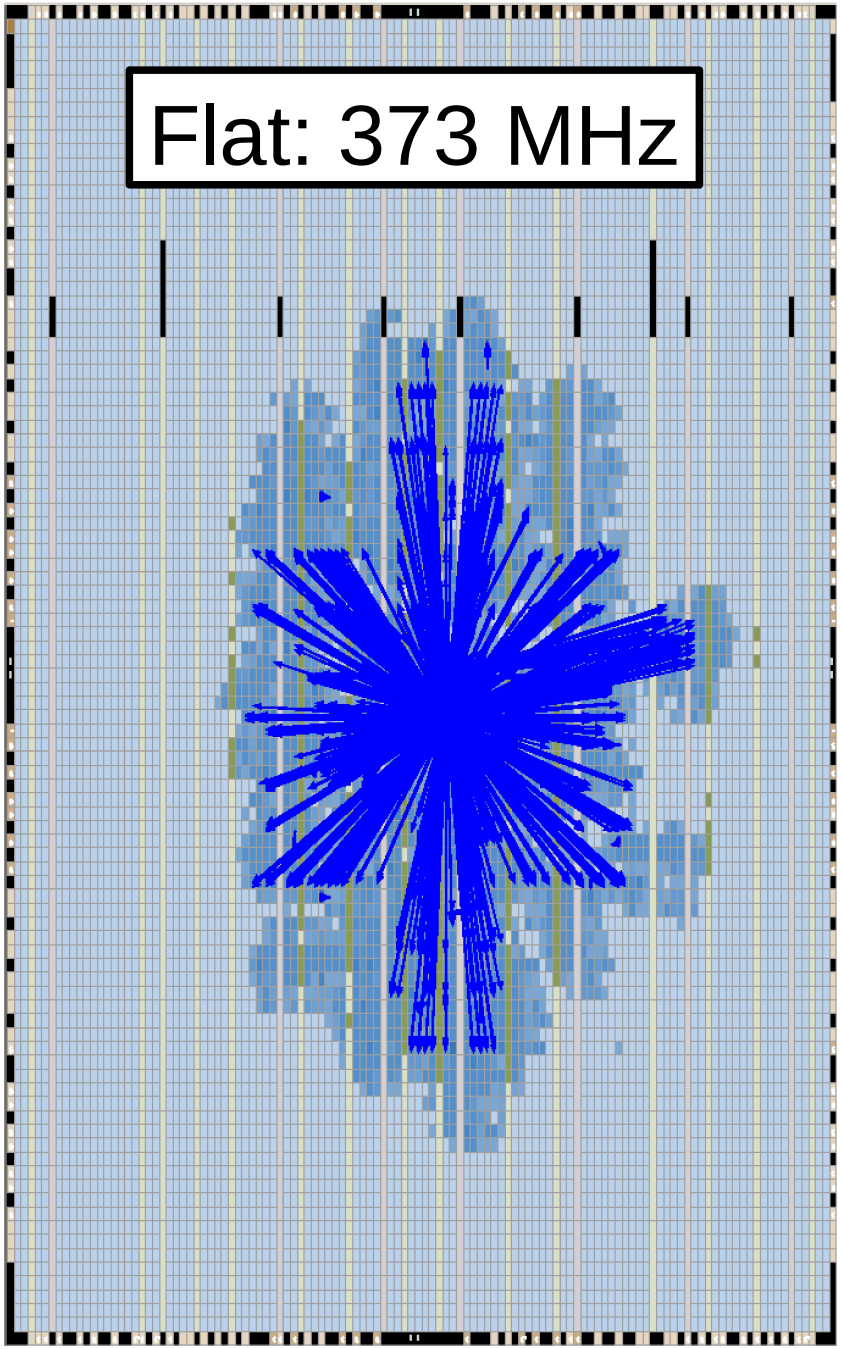
Impact on Speed

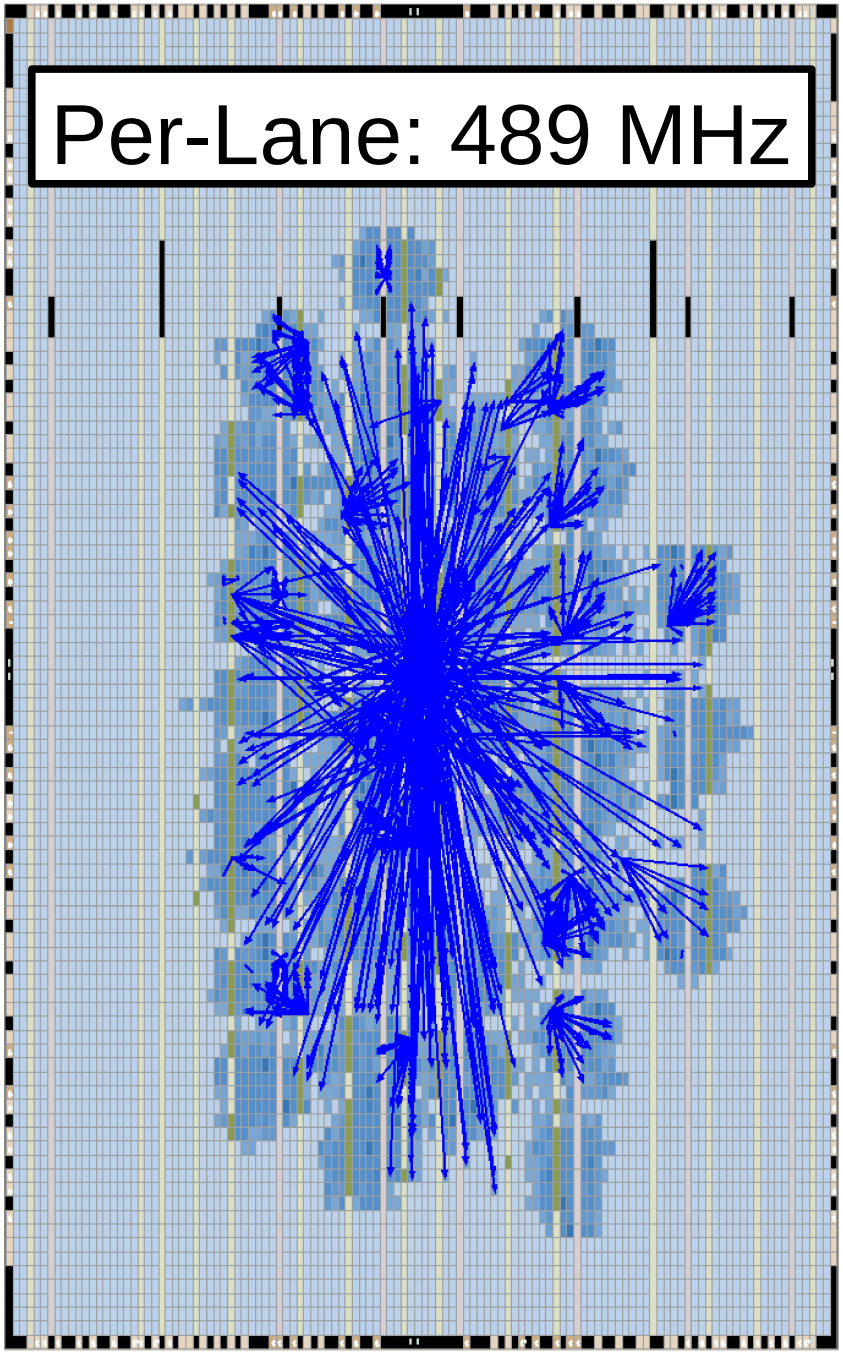


Partitioning a 32-Way SIMD Octavo

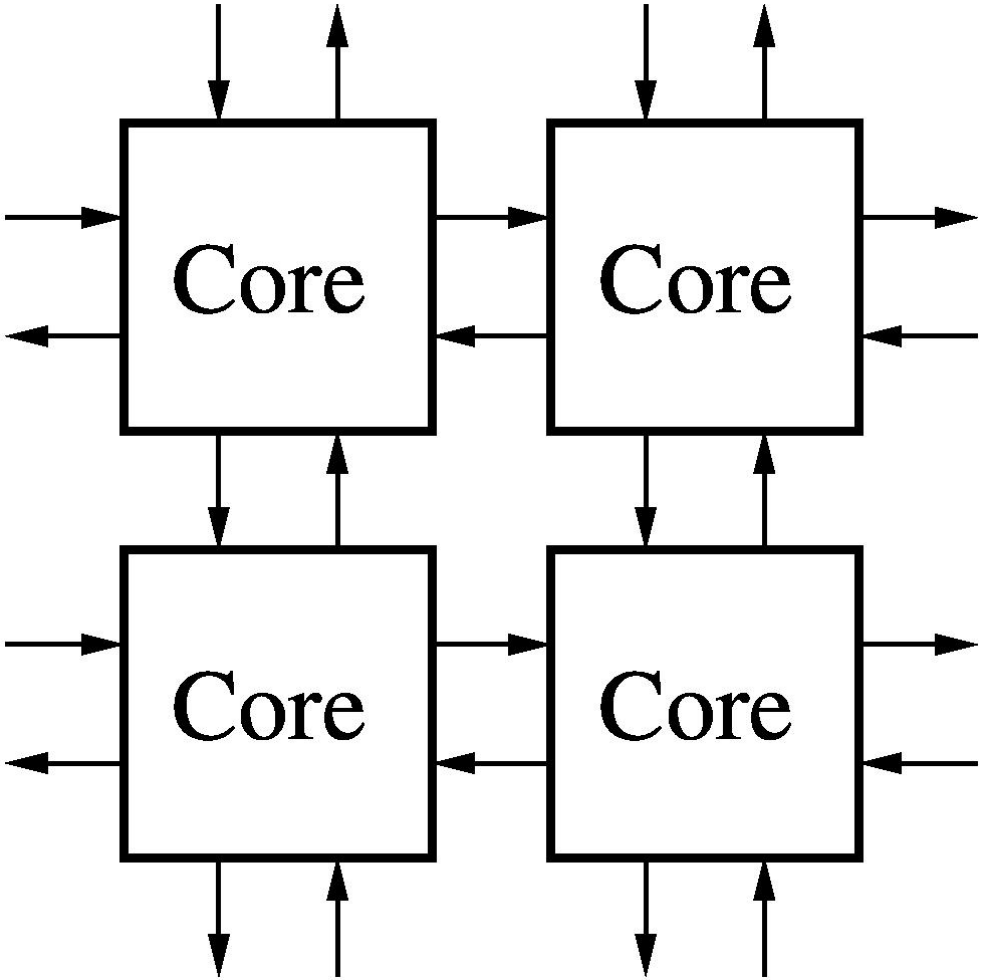


Partitioning a 32-Way SIMD Octavo



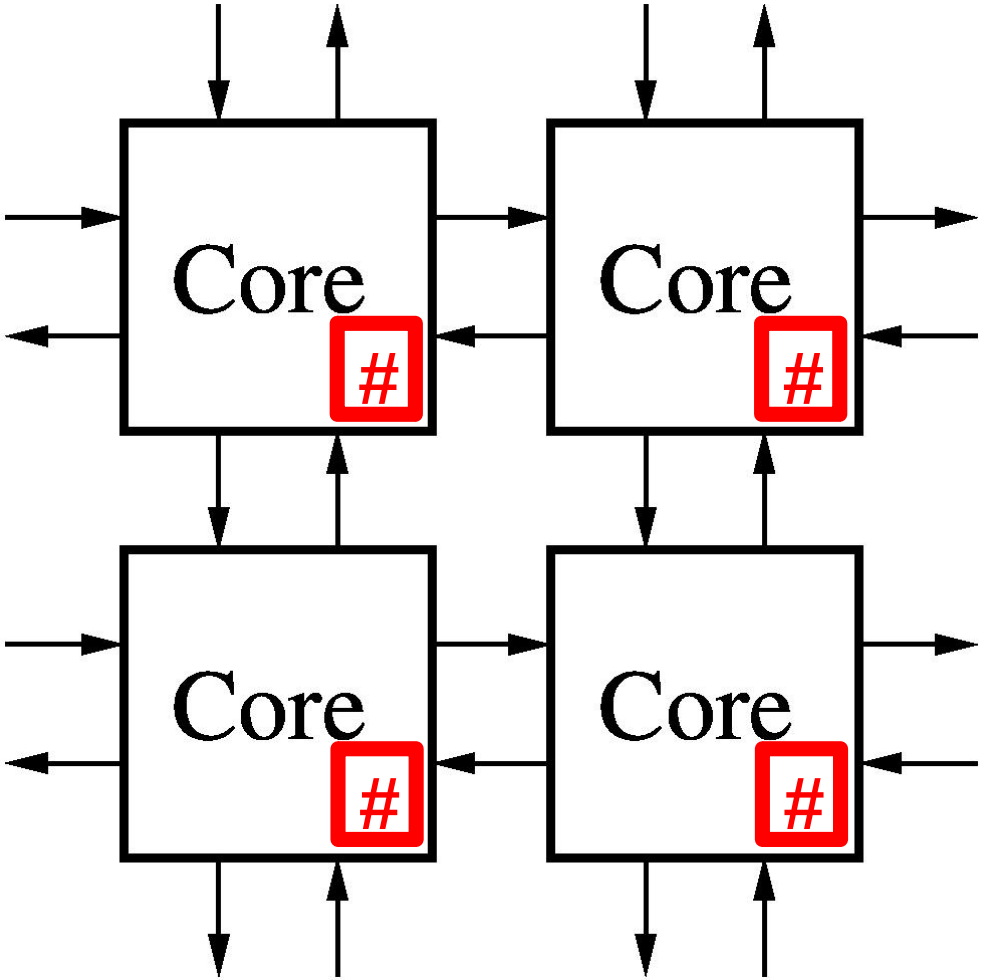


Meshes of Octavo Cores



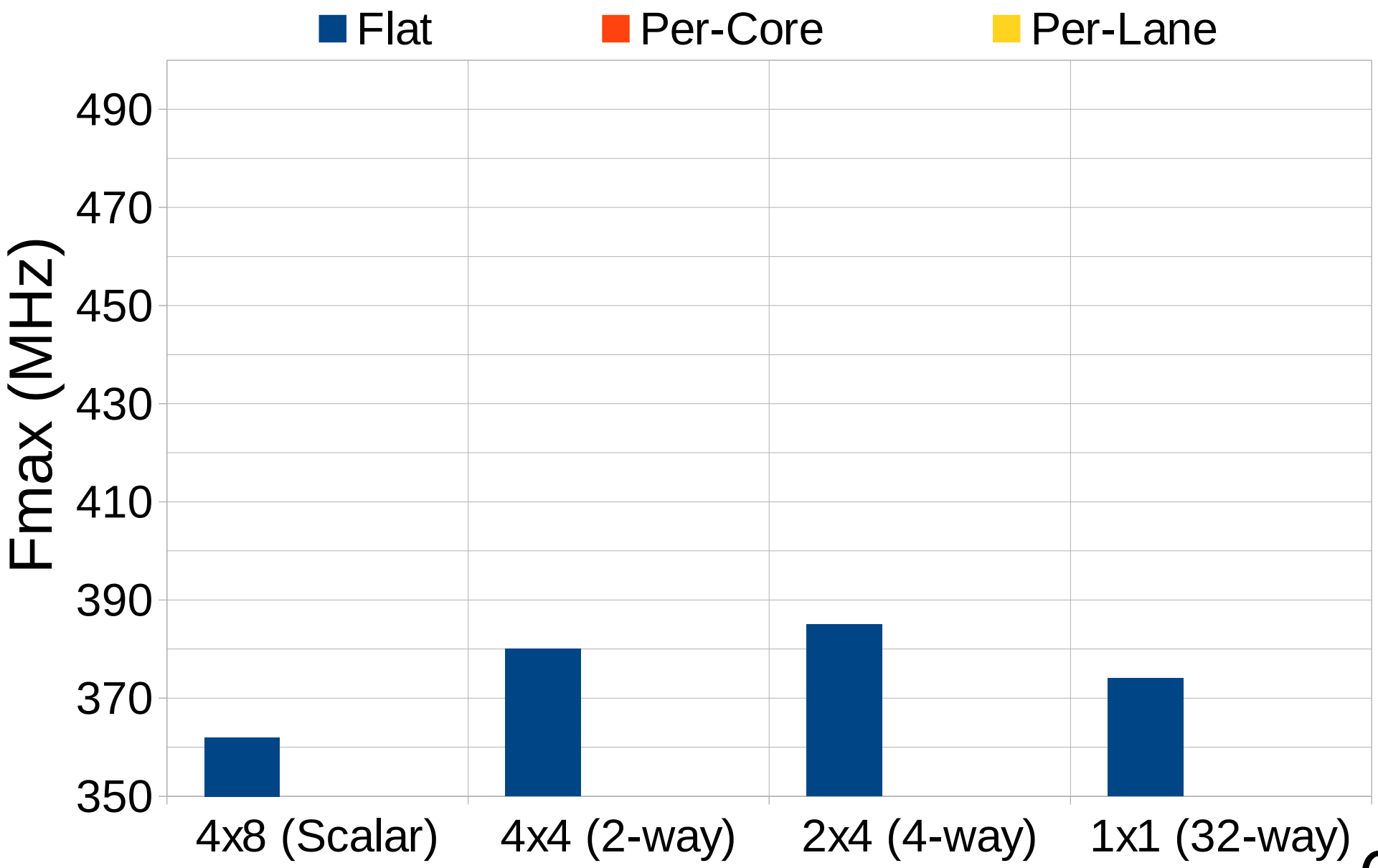
- Scalar or SIMD Cores
- New Multi-Localities

Meshes of Octavo Cores

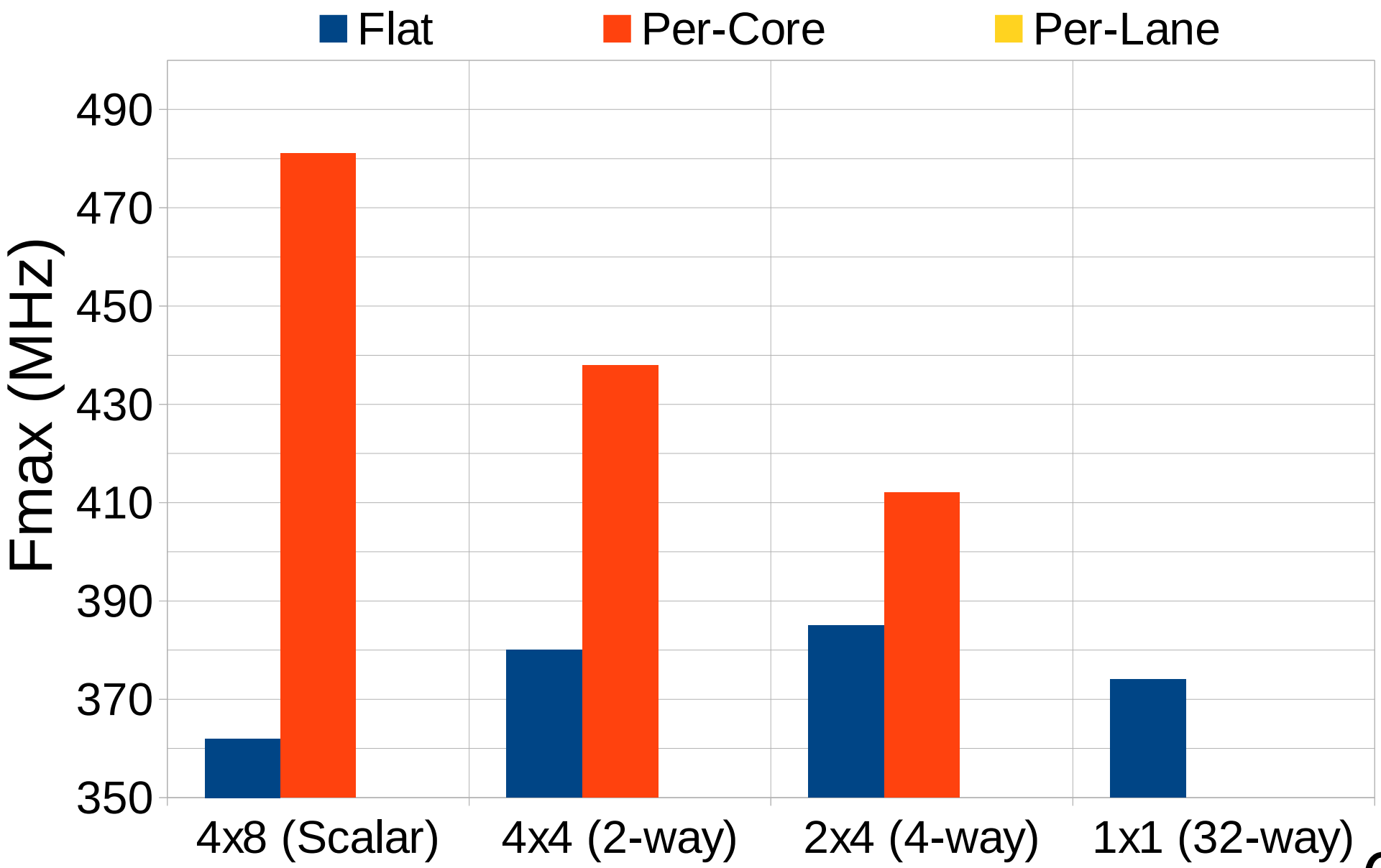


- Scalar or SIMD Cores
- New Multi-Localities
  - 3-bit thread counter
  - 1 per Core

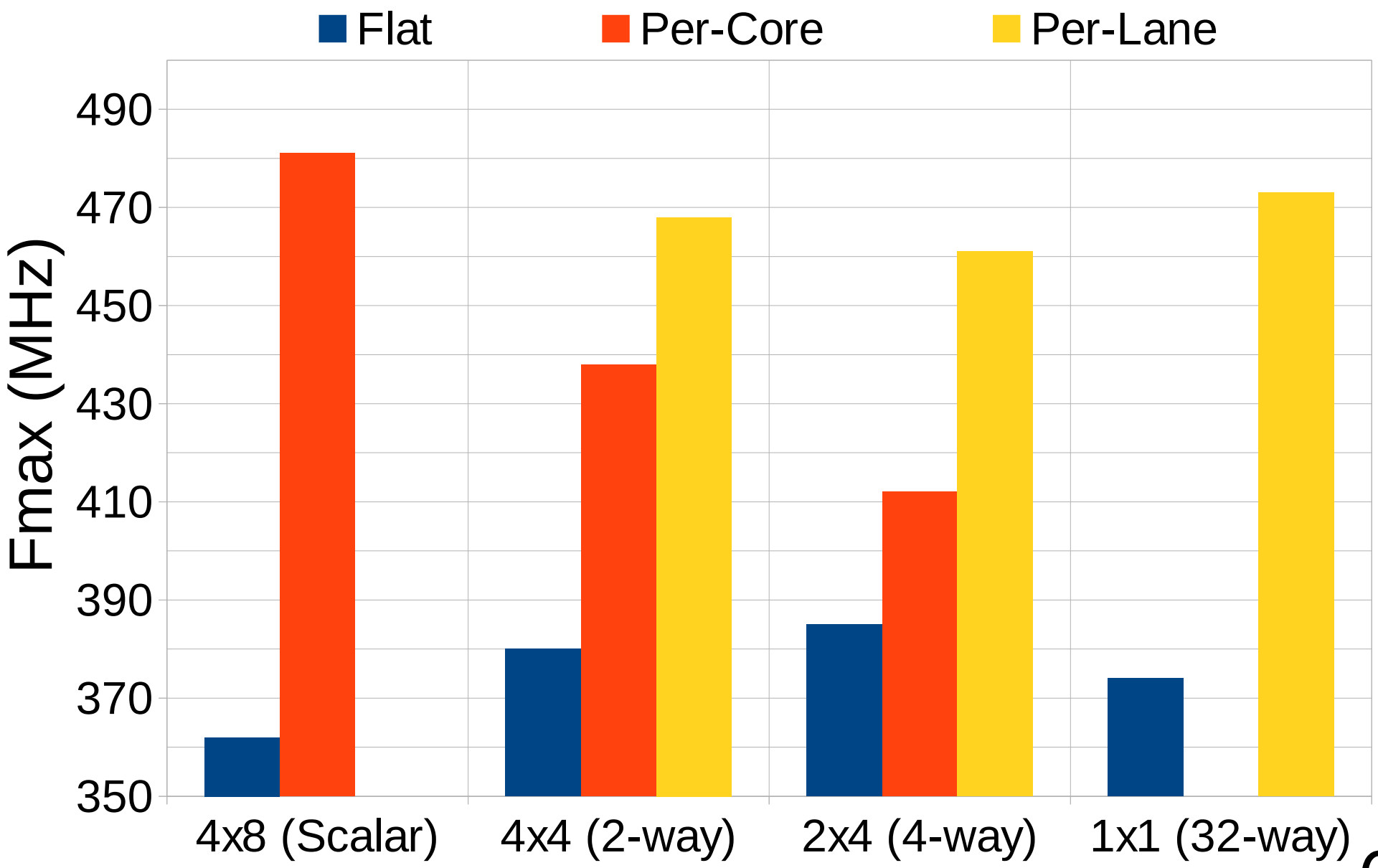
Meshes with 32 Datapaths Total



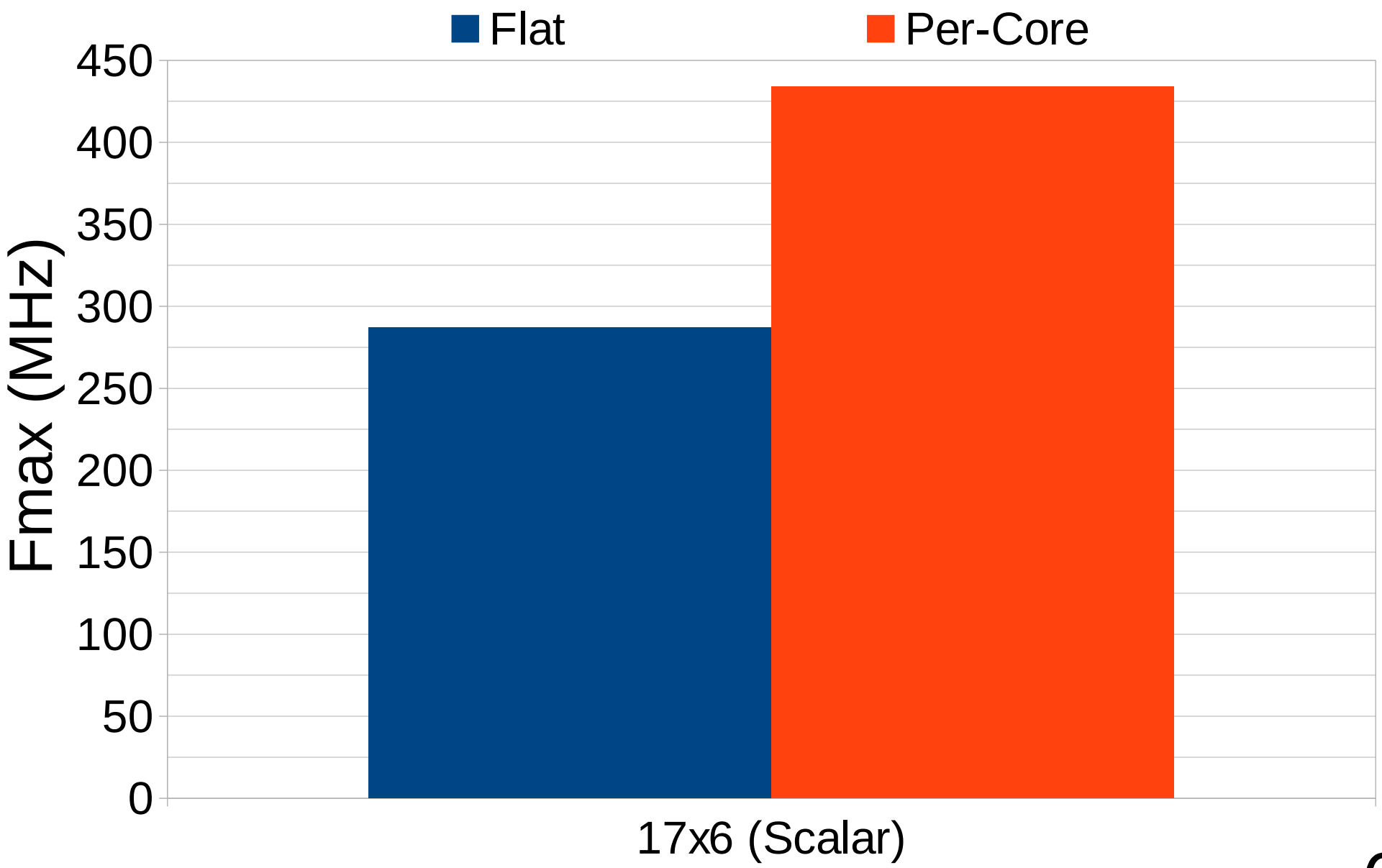
Meshes with 32 Datapaths Total



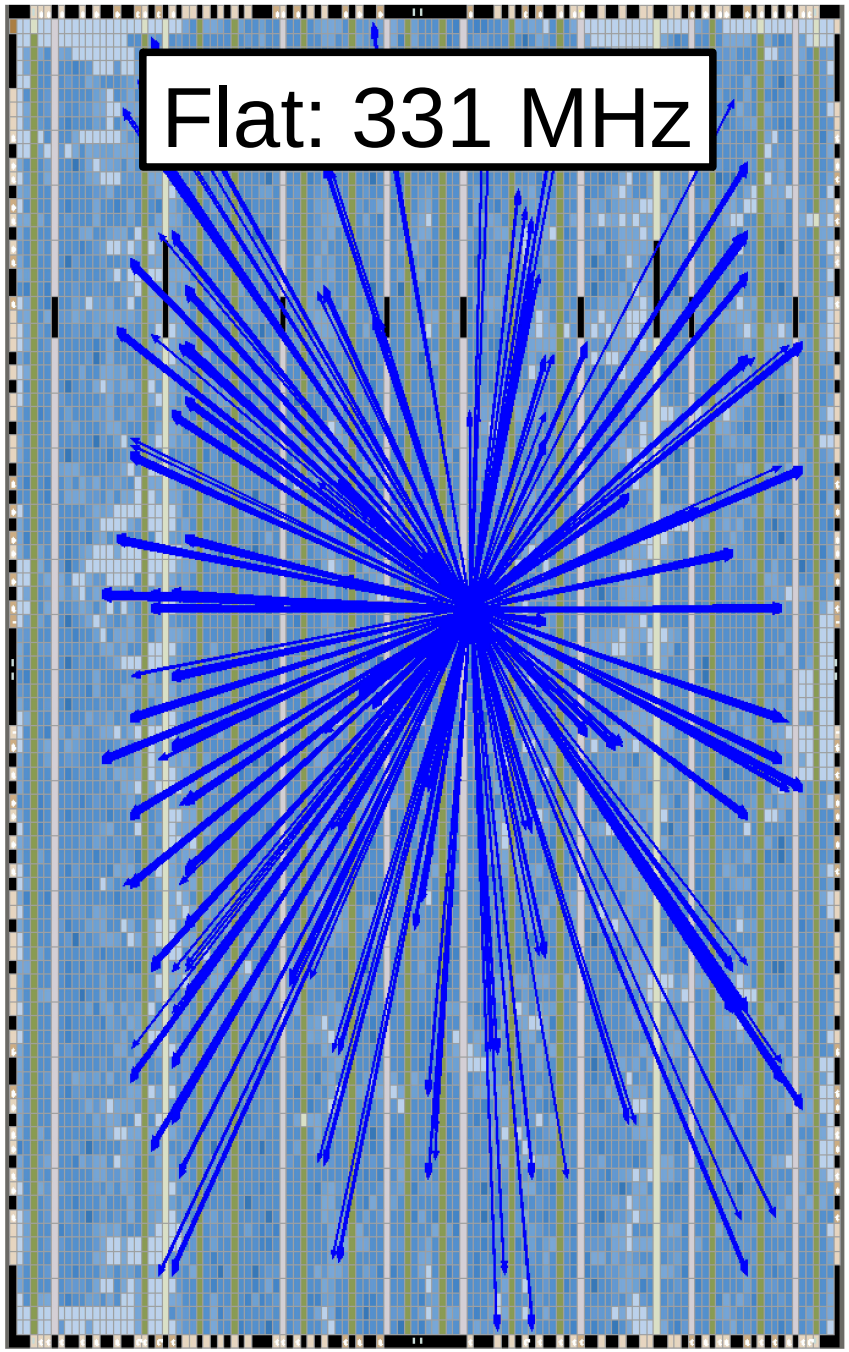
Meshes with 32 Datapaths Total



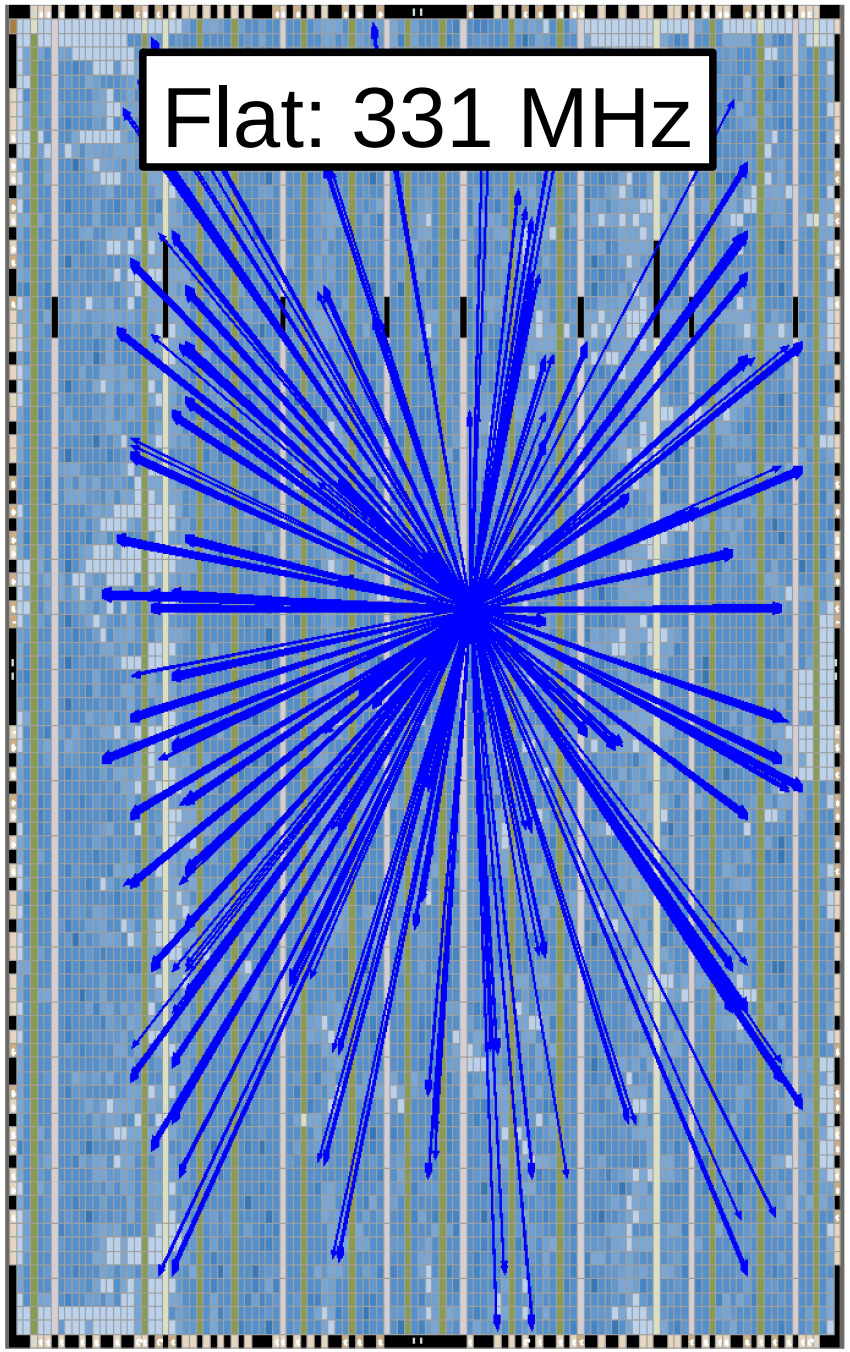
Mesh of 102 Octavo Cores

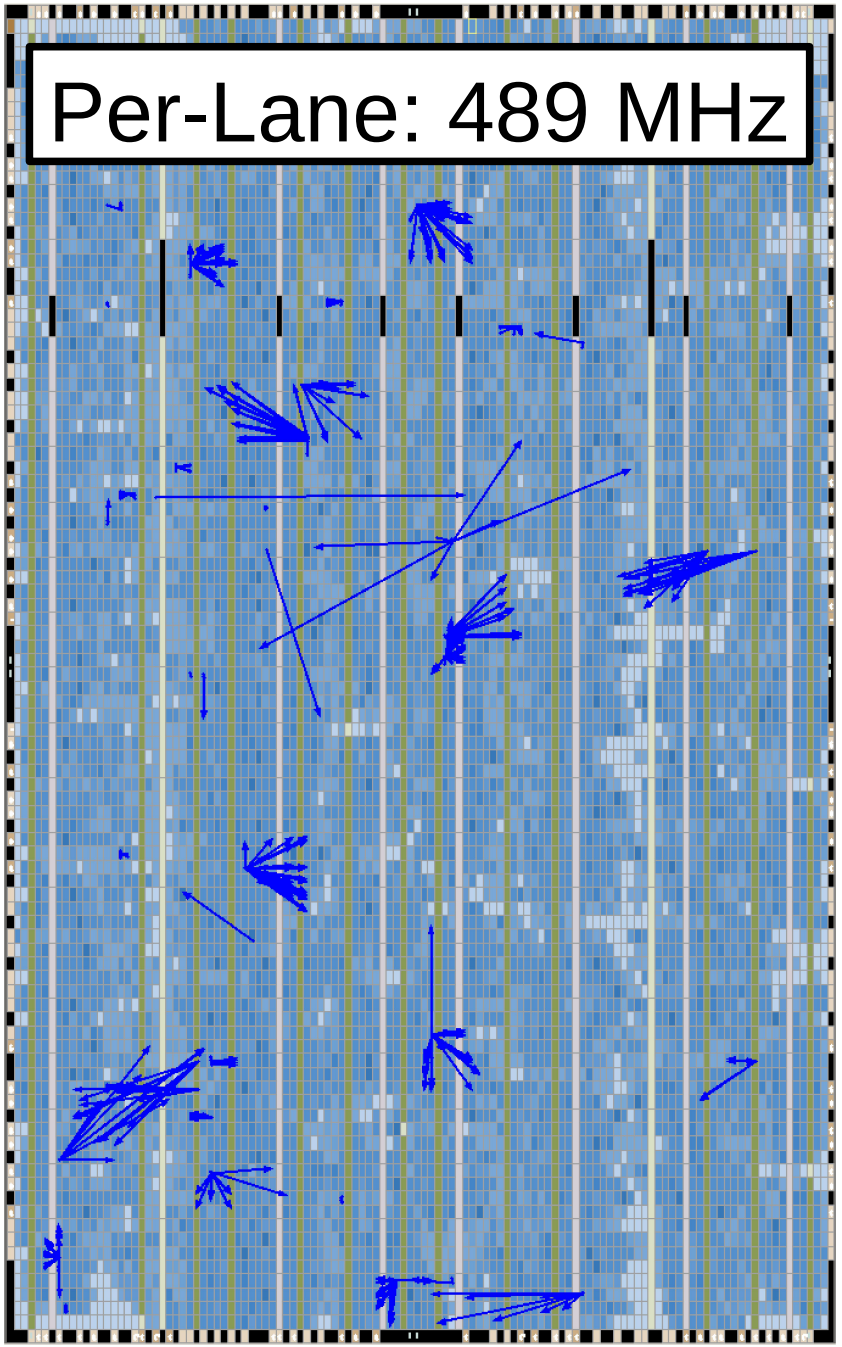


Mesh of 102 Scalar Octavos (17x6)



Mesh of 102 Scalar Octavos (17x6)





Overhead-Free Execution

- Problems
  - Speedup ultimately limited by execution overhead
  - Addressing and flow-control (per thread)
  - Worsened by hardware assistance

Overhead-Free Execution

- Problems
  - Speedup ultimately limited by execution overhead
  - Addressing and flow-control (per thread)
  - Worsened by hardware assistance

- Solutions
  - Extract overhead as “sub-programs” (per thread)
  - Execute them in parallel along the pipeline
  - Decreases Fmax 6.1%, increases area 73%*

Sequential Sub-Programs in MIPS

```
outer: seed_ptr = ptr_init
inner: temp = MEM[seed_ptr]
       if (temp < 0):
           goto outer
       temp2 = temp & 1
       if (temp2 == 1):
           temp = (temp * 3) + 1
       else:
           temp = temp / 2
       MEM[seed_ptr] = temp
       seed_ptr += 1
       OUTPUT = temp
       goto inner
```

- Flow-control
- Addressing
- Useful work

Sequential Sub-Programs in Octavo

```
outer:  ADD seed_ptr, ptr_init, 0
inner:  LW  temp, seed_ptr
        BLTZn outer, temp
        BEVNn even,  temp
        MUL temp, temp, 3
        ADD temp, temp, 1
        JMP output
even:   SRA temp, temp, 1
output: SW  temp, seed_ptr
        ADD seed_ptr, seed_ptr, 1
        SW  temp, OUTPUT
        JMP inner
```

- Flow-control
- Addressing
- Useful work

Removing Flow-Control Overhead

```
outer:  ADD seed_ptr, ptr_init, 0
inner:  LW  temp, seed_ptr
        BLTZn outer, temp
        BEVNn even,  temp
        MUL temp, temp, 3
        ADD temp, temp, 1
        JMP output
even:   SRA temp, temp, 1
output: SW  temp, seed_ptr
        ADD seed_ptr, seed_ptr, 1
        SW  temp, OUTPUT
        JMP inner
```

- Flow-control
- Addressing
- Useful work

Parallel Sub-Programs in Octavo

```
outer:  ADD seed_ptr, ptr_init, 0
inner:  LW  temp, seed_ptr
        BLTZn outer, temp
        BEVNn even,  temp
        MUL temp, temp, 3
        ADD temp, temp, 1
        JMP output
even:   SRA temp, temp, 1
output: SW  temp, seed_ptr
        ADD seed_ptr, seed_ptr, 1
        SW  temp, OUTPUT
        JMP inner
```

- Flow-control
- Addressing
- Useful work

Parallel Sub-Programs in Octavo

```
outer:  ADD seed_ptr, ptr_init, 0
inner:  LW  temp, seed_ptr
        MUL temp, temp, 3 ; BEVNn even ; BLTZn outer
        ADD temp, temp, 1 ; JMP output
even:   SRA temp, temp, 1
output: SW  temp, seed_ptr
        SW  temp, OUTPUT  ; JMP inner
```

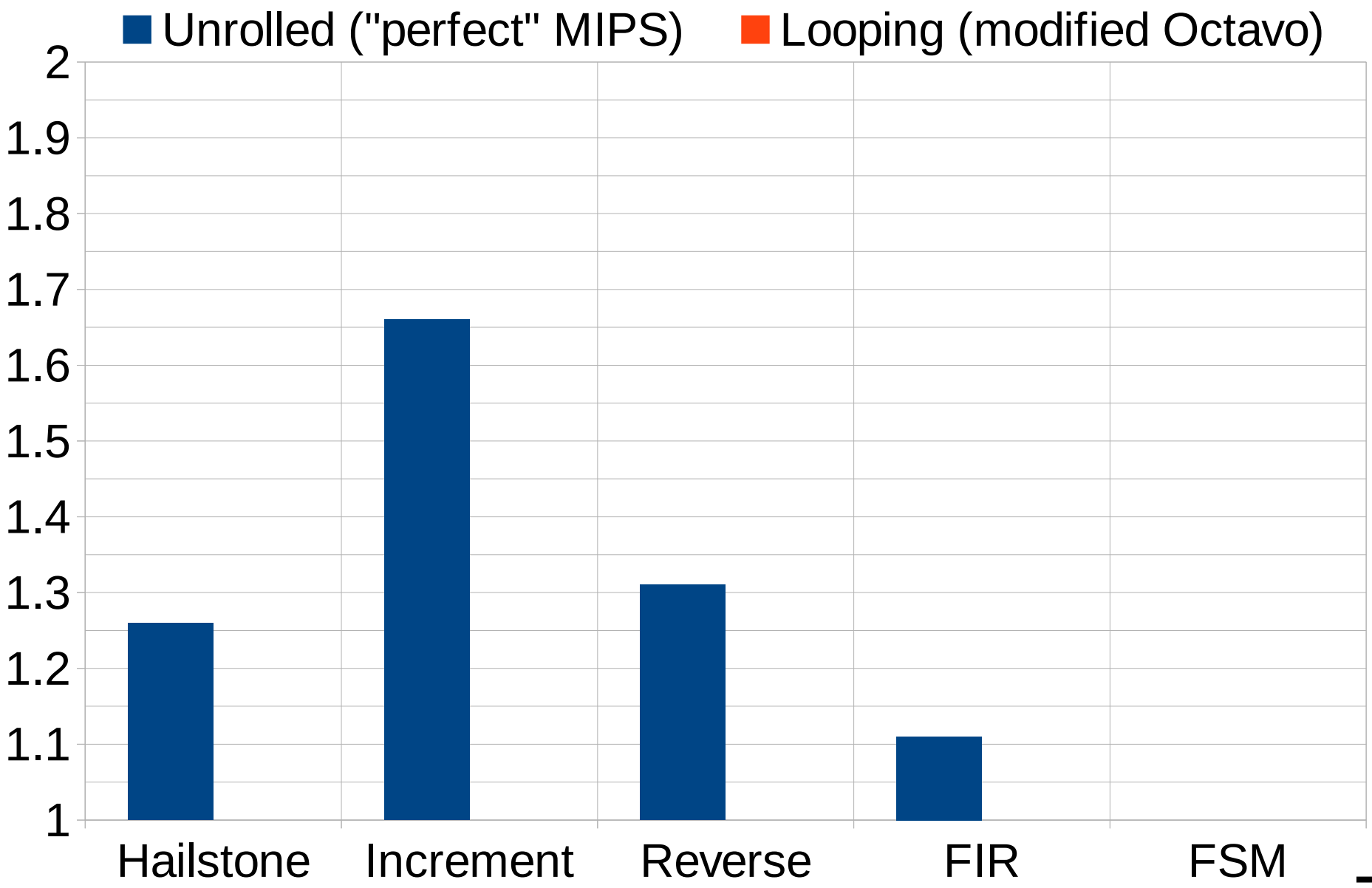
- Flow-control (folded, cancelling, multi-way)
- Addressing (indirect with post-increment)
- Useful work

Parallel Sub-Programs in Octavo

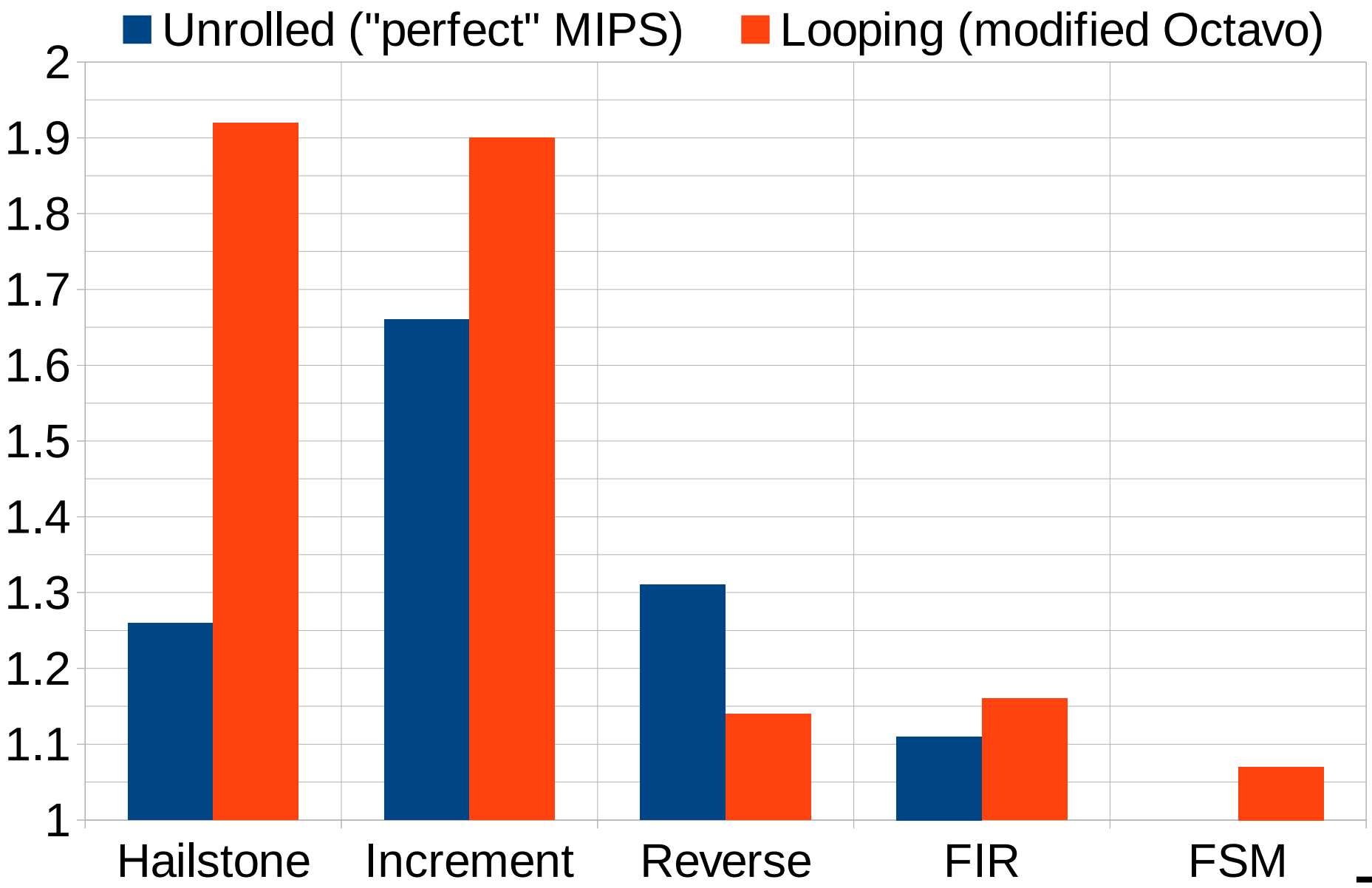
```
outer:  ADD seed_ptr, ptr_init, 0
inner:  LW  temp, seed_ptr
        MUL temp, temp, 3 ; BEVNn even ; BLTZn outer
        ADD temp, temp, 1 ; JMP output
even:   SRA temp, temp, 1
output: SW  temp, seed_ptr
        SW  temp, OUTPUT  ; JMP inner
```

- Flow-control (folded, cancelling, multi-way)
- Addressing (indirect with post-increment)
- Useful work

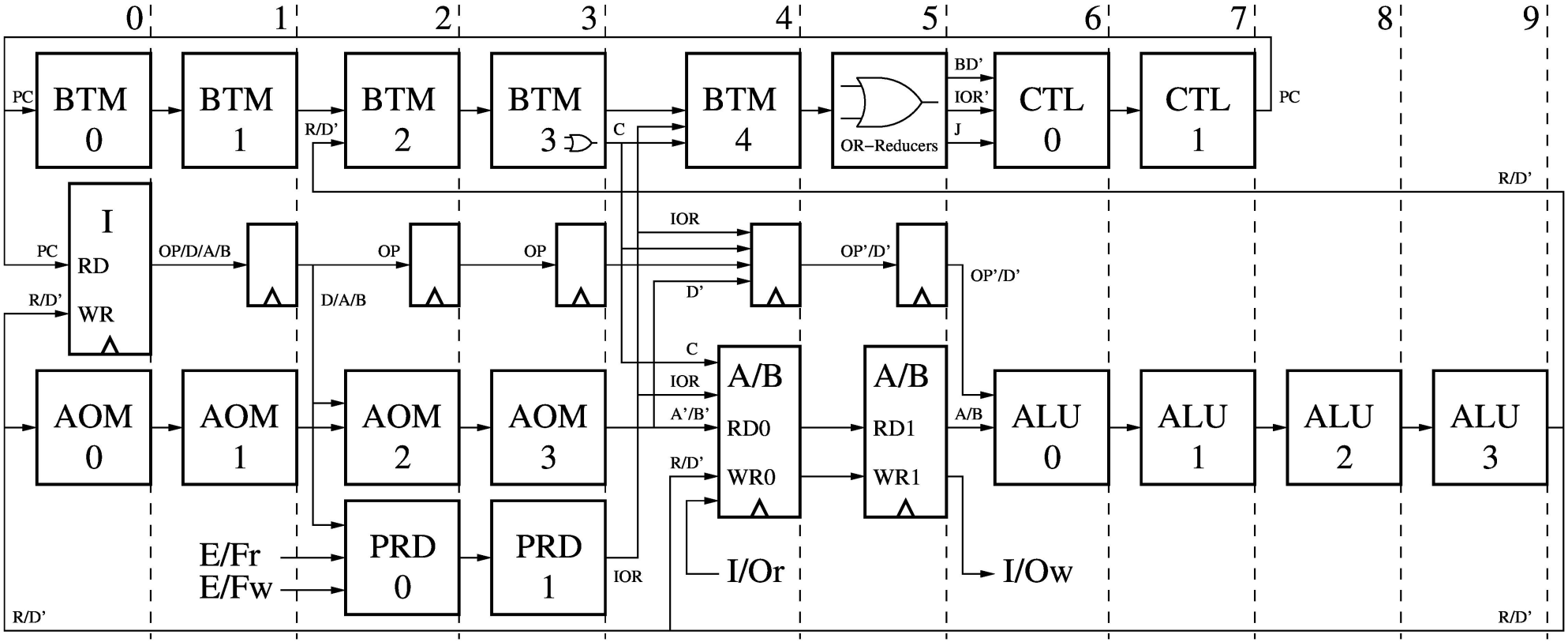
Speedups


Unrolled ("perfect" MIPS)
Looping (modified Octavo)
2
1.9
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1
1
Hailstone
Increment
Reverse
FIR
FSM

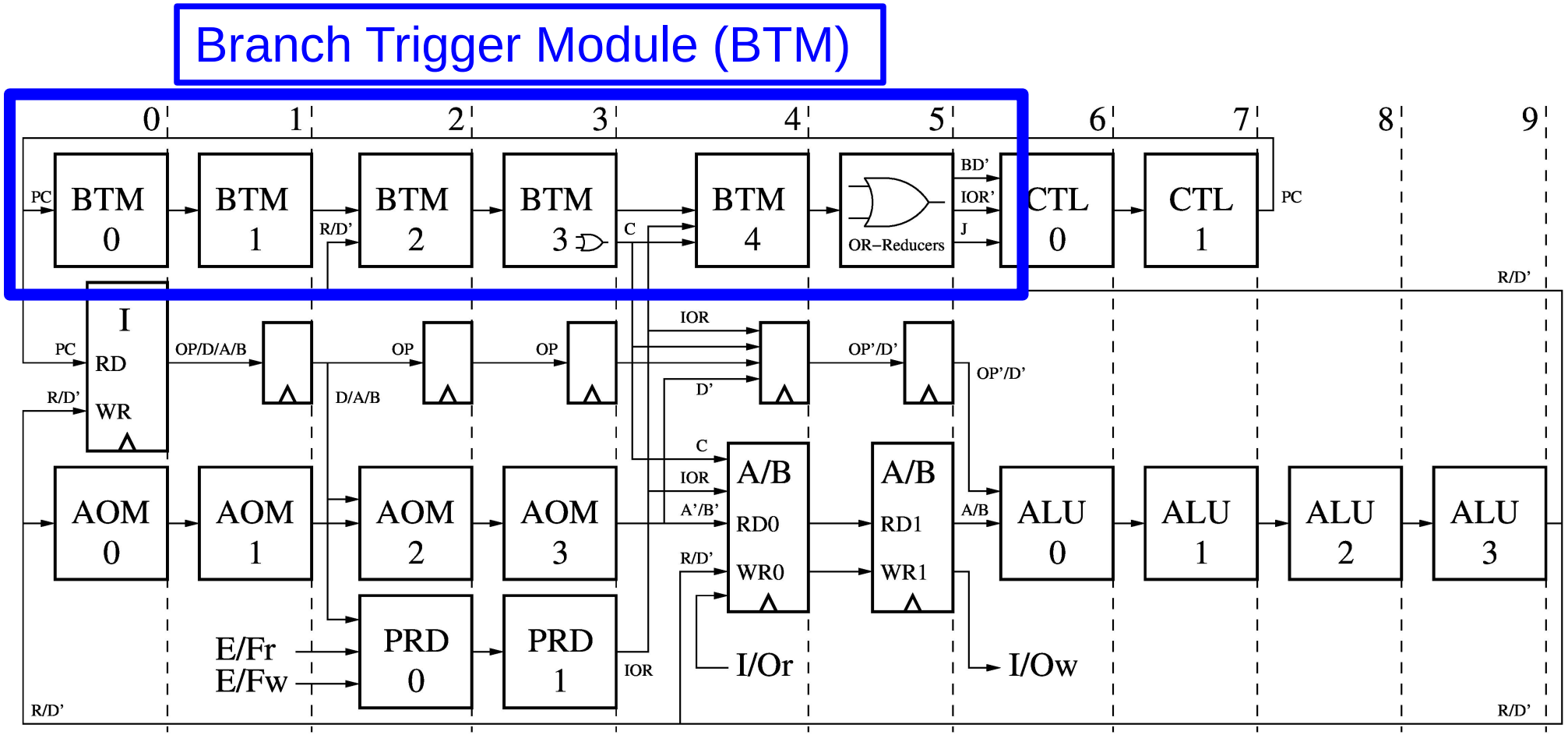
Speedups


Unrolled ("perfect" MIPS)
Looping (modified Octavo)
2
1.9
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1
1
Hailstone
Increment
Reverse
FIR
FSM

Reduced-Overhead Octavo

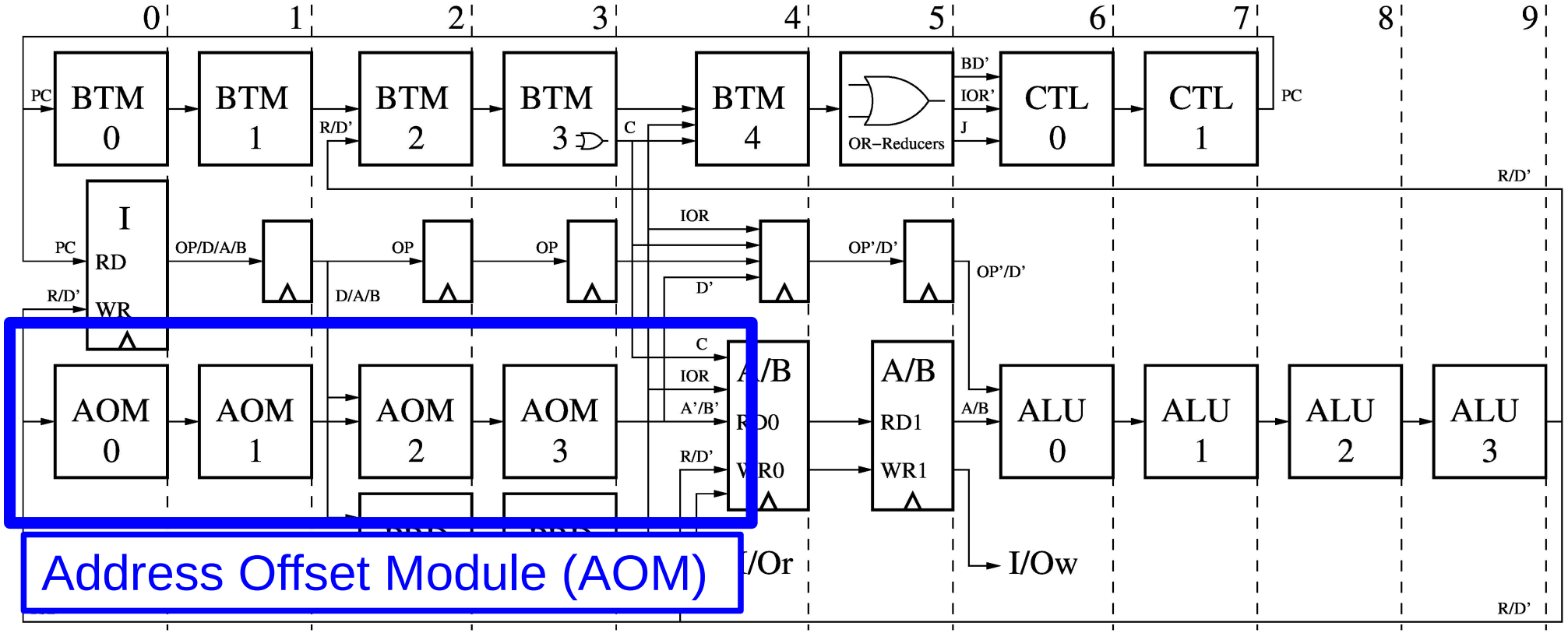


Reduced-Overhead Octavo



(Branches not in fetched instructions!)

Reduced-Overhead Octavo



(One entry for each instruction operand)

Benchmarking

- Compares FPGA design processes:
  - Octavo (Multi-Threaded Soft-Processor)

Benchmarking

- Compares FPGA design processes:
  - Octavo (Multi-Threaded Soft-Processor)
  - MXP (Soft Vector Processor)
    - A. Severance, J. Edwards, H. Omidian, G. Lemieux, "*Soft Vector Processors with Streaming Pipelines*", FPGA 2014

Benchmarking

- Compares FPGA design processes:
  - Octavo (Multi-Threaded Soft-Processor)
  - MXP (Soft Vector Processor)
    - A. Severance, J. Edwards, H. Omidian, G. Lemieux, "*Soft Vector Processors with Streaming Pipelines*", FPGA 2014
  - LegUp (plain C HLS)
    - A. Canis, J. Choi, M. Aldham, V. Zhang, A. Kammoona, T. Czajkowski, S. D. Brown, J. H. Anderson, "*LegUp: An Open-source High-level Synthesis Tool for FPGA-based Processor/Accelerator Systems*", TRETS, Sept. 2013

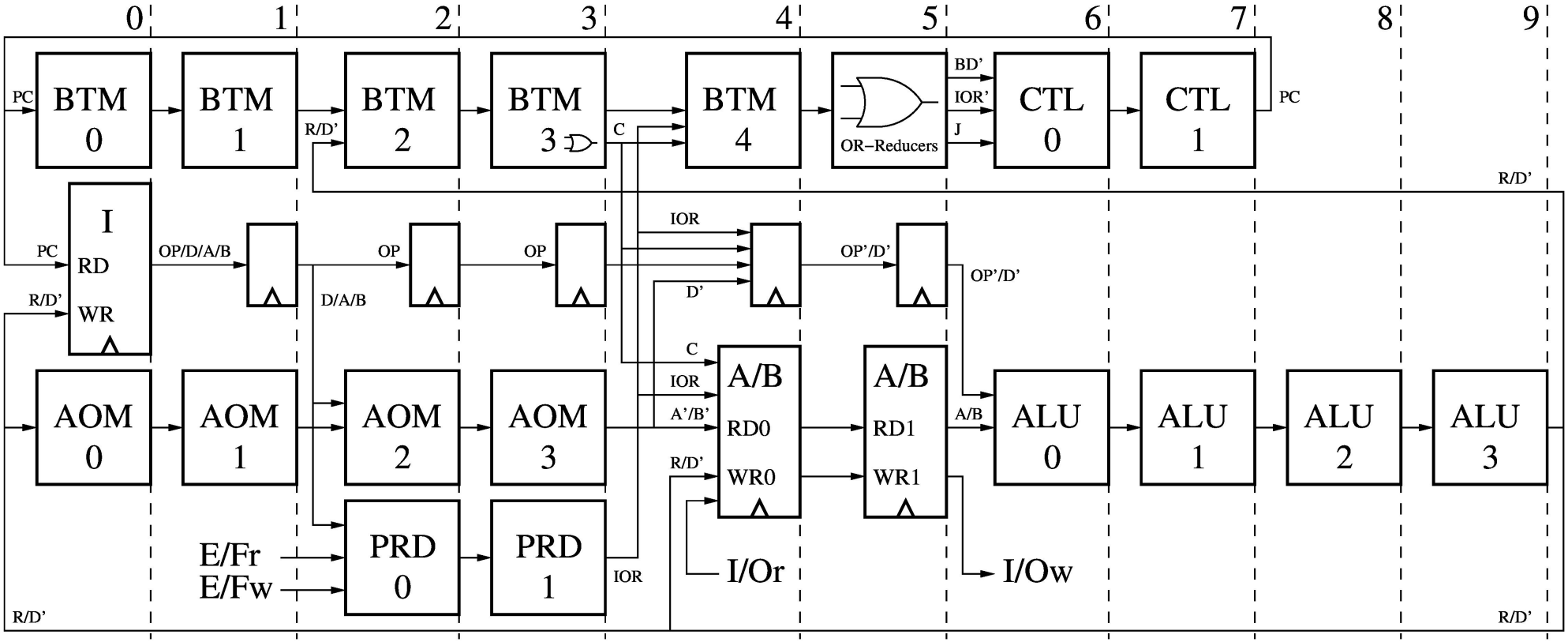
Benchmarking

- Compares FPGA design processes:
  - Octavo (Multi-Threaded Soft-Processor)
  - MXP (Soft Vector Processor)
    - A. Severance, J. Edwards, H. Omidian, G. Lemieux, "*Soft Vector Processors with Streaming Pipelines*", FPGA 2014
  - LegUp (plain C HLS)
    - A. Canis, J. Choi, M. Aldham, V. Zhang, A. Kammoona, T. Czajkowski, S. D. Brown, J. H. Anderson, "*LegUp: An Open-source High-level Synthesis Tool for FPGA-based Processor/Accelerator Systems*", TRETS, Sept. 2013
  - Verilog (hand-optimized HDL for speed)

Benchmarking

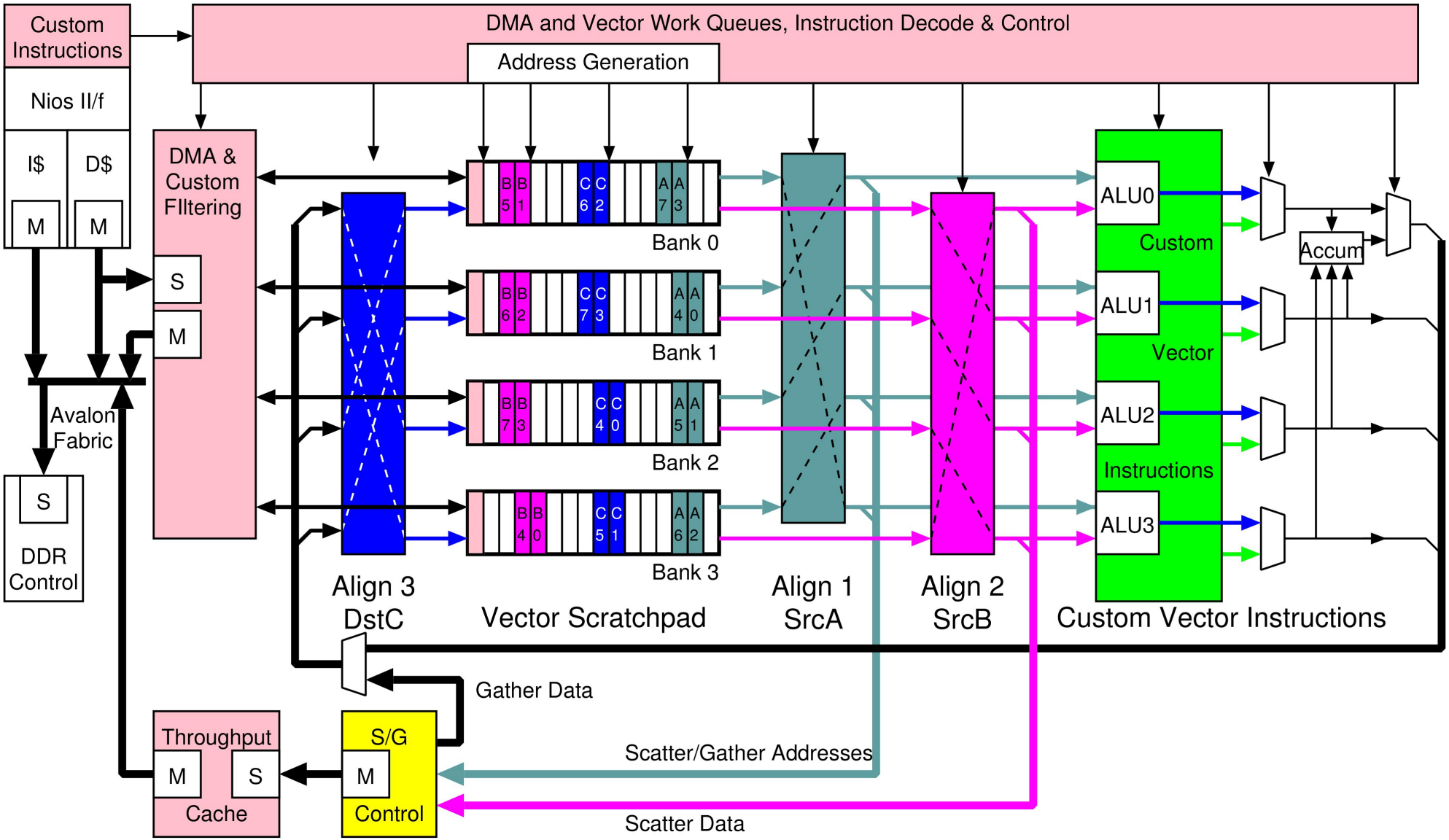
- Compares FPGA design processes:
  - Octavo (Multi-Threaded Soft-Processor)
  - MXP (Soft Vector Processor)
    - A. Severance, J. Edwards, H. Omidian, G. Lemieux, "*Soft Vector Processors with Streaming Pipelines*", FPGA 2014
  - LegUp (plain C HLS)
    - A. Canis, J. Choi, M. Aldham, V. Zhang, A. Kammoona, T. Czajkowski, S. D. Brown, J. H. Anderson, "*LegUp: An Open-source High-level Synthesis Tool for FPGA-based Processor/Accelerator Systems*", TRETS, Sept. 2013
  - Verilog (hand-optimized HDL for speed)
- All results relative to a Scalar Octavo core

Octavo



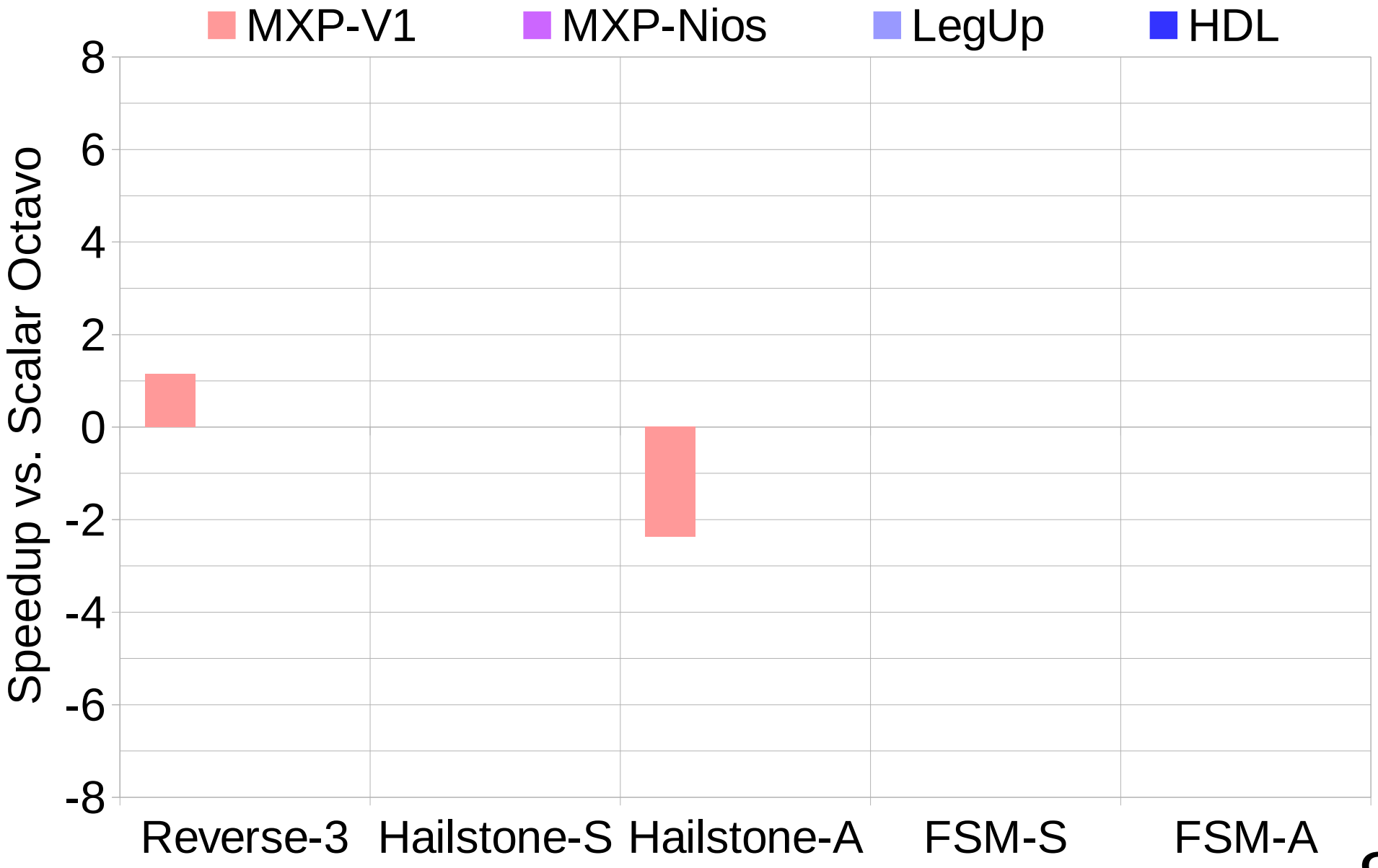
- 1 to 32 SIMD Datapaths total (-L1 to -L32)
- Each Datapath has Accelerators on I/O Ports:
  - Accumulator
  - Array Reversal Channel

MXP

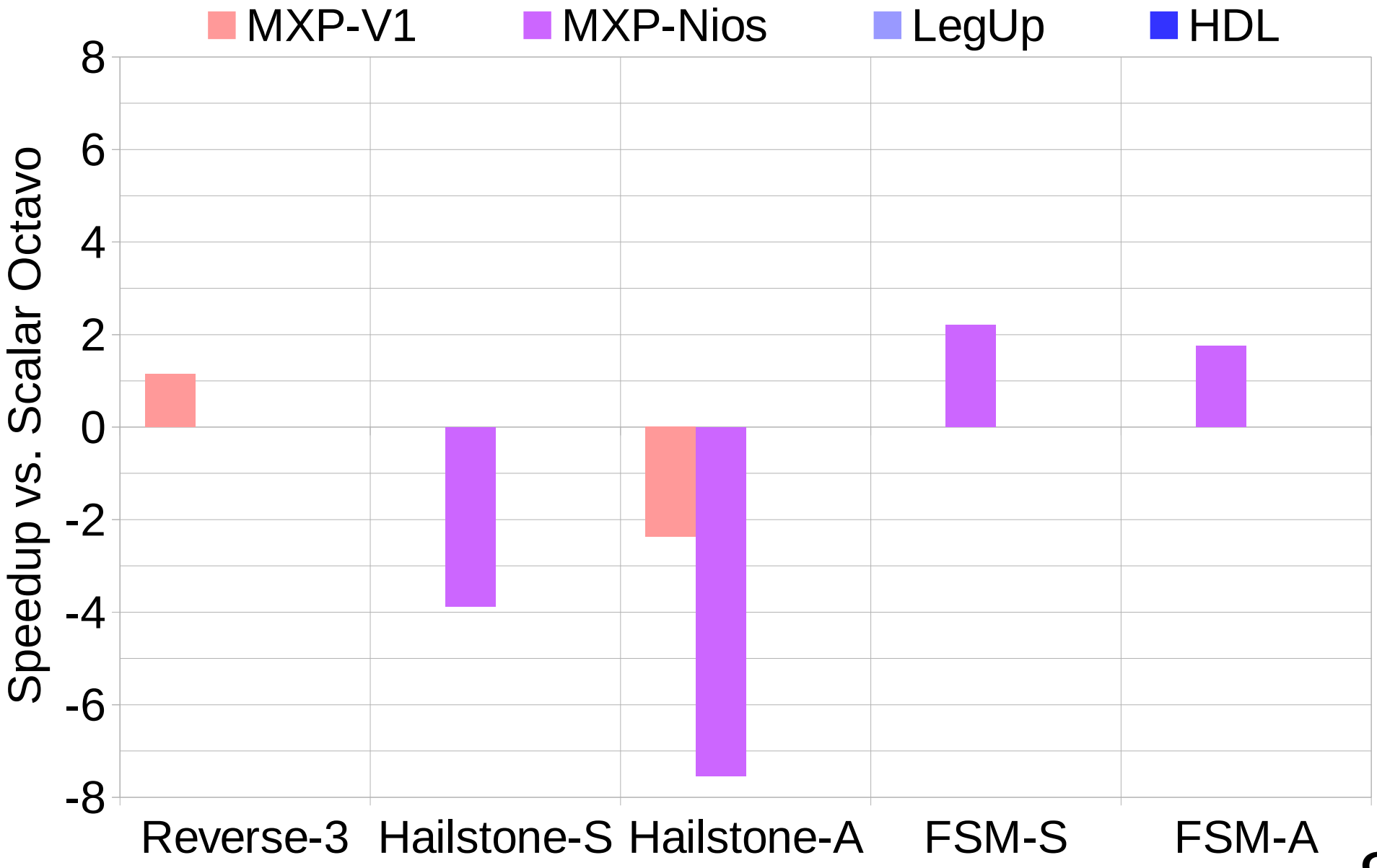


- -V1 to -V32 Vector Lanes, with table-lookups

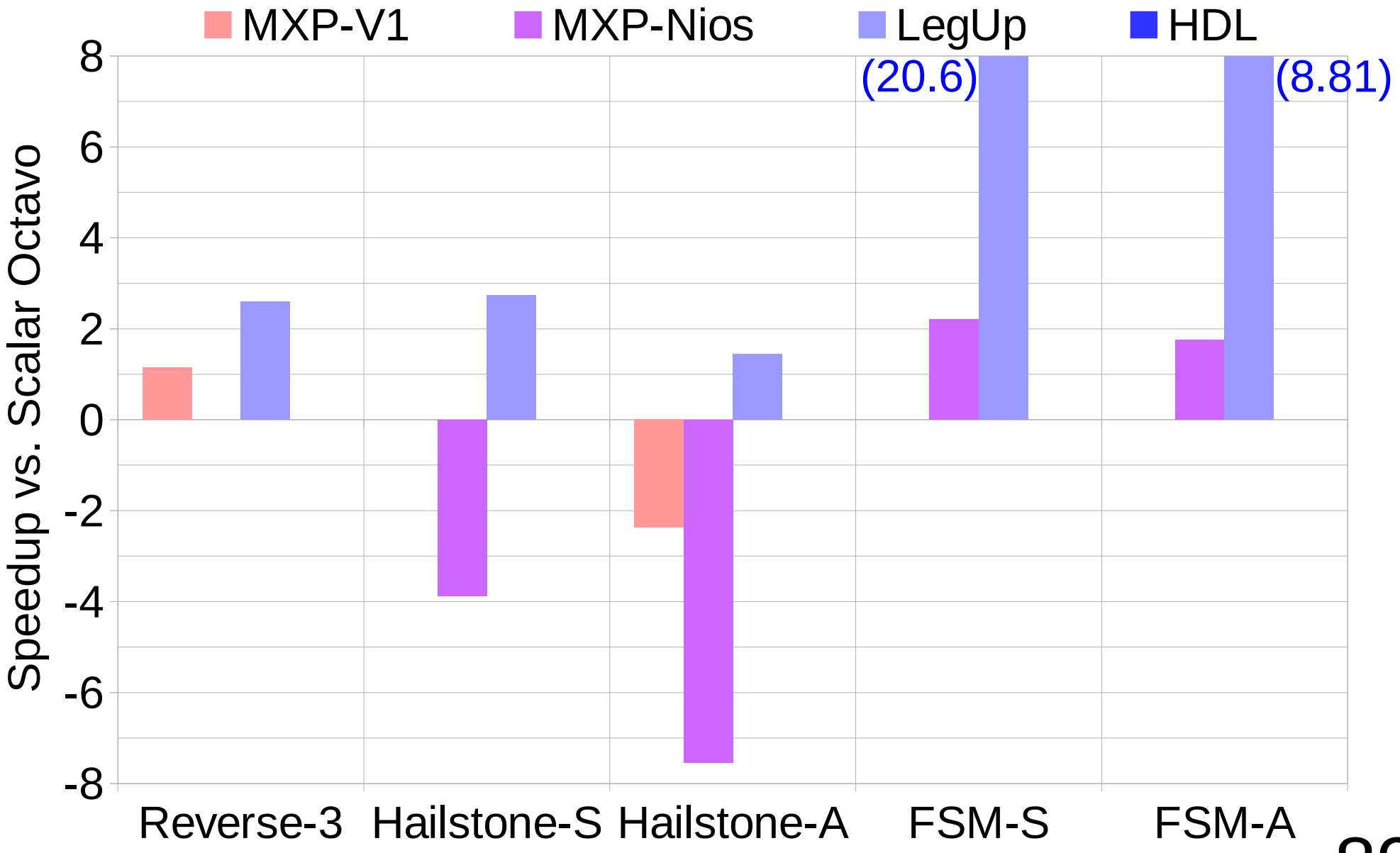
Sequential Speedup



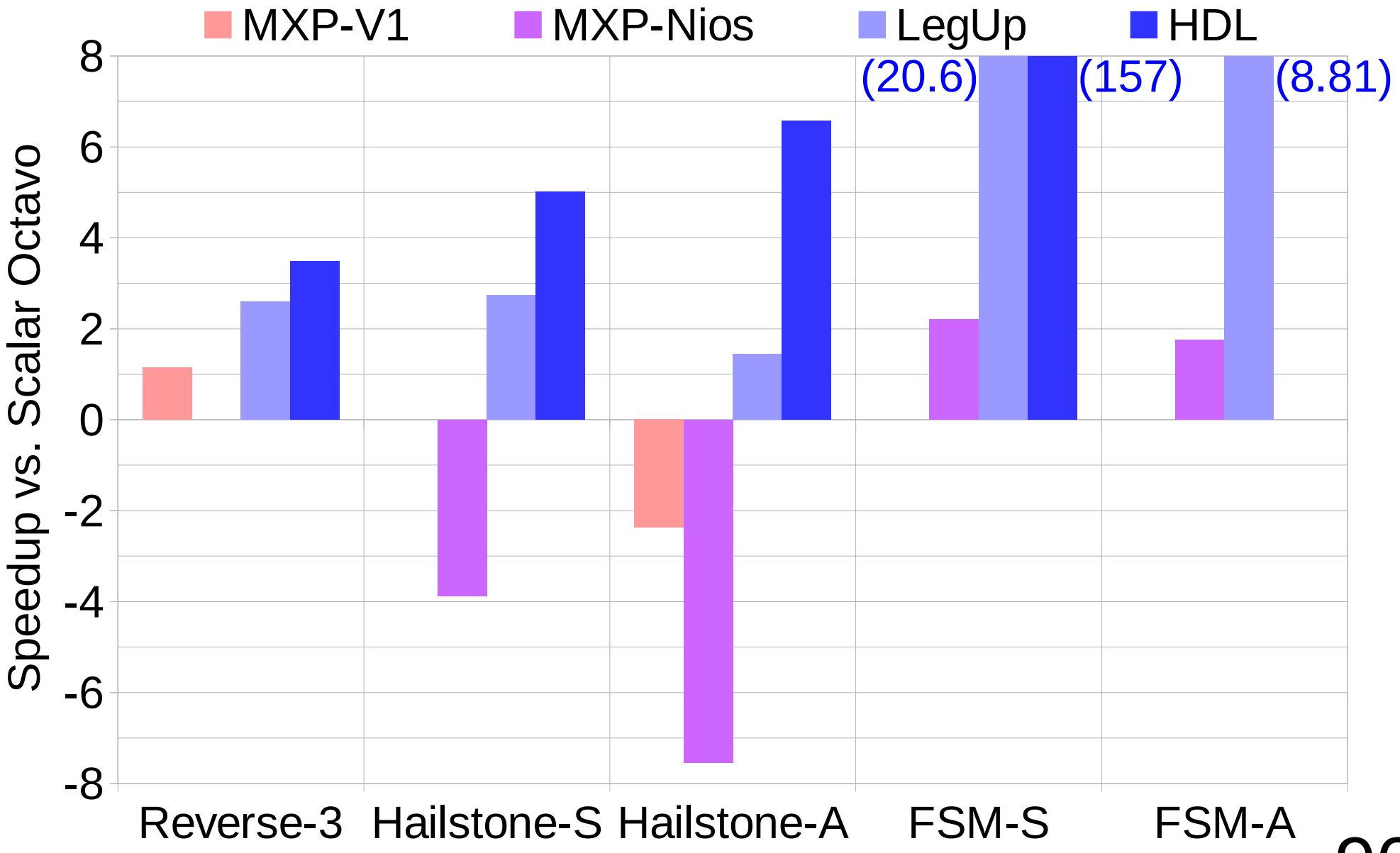
Sequential Speedup



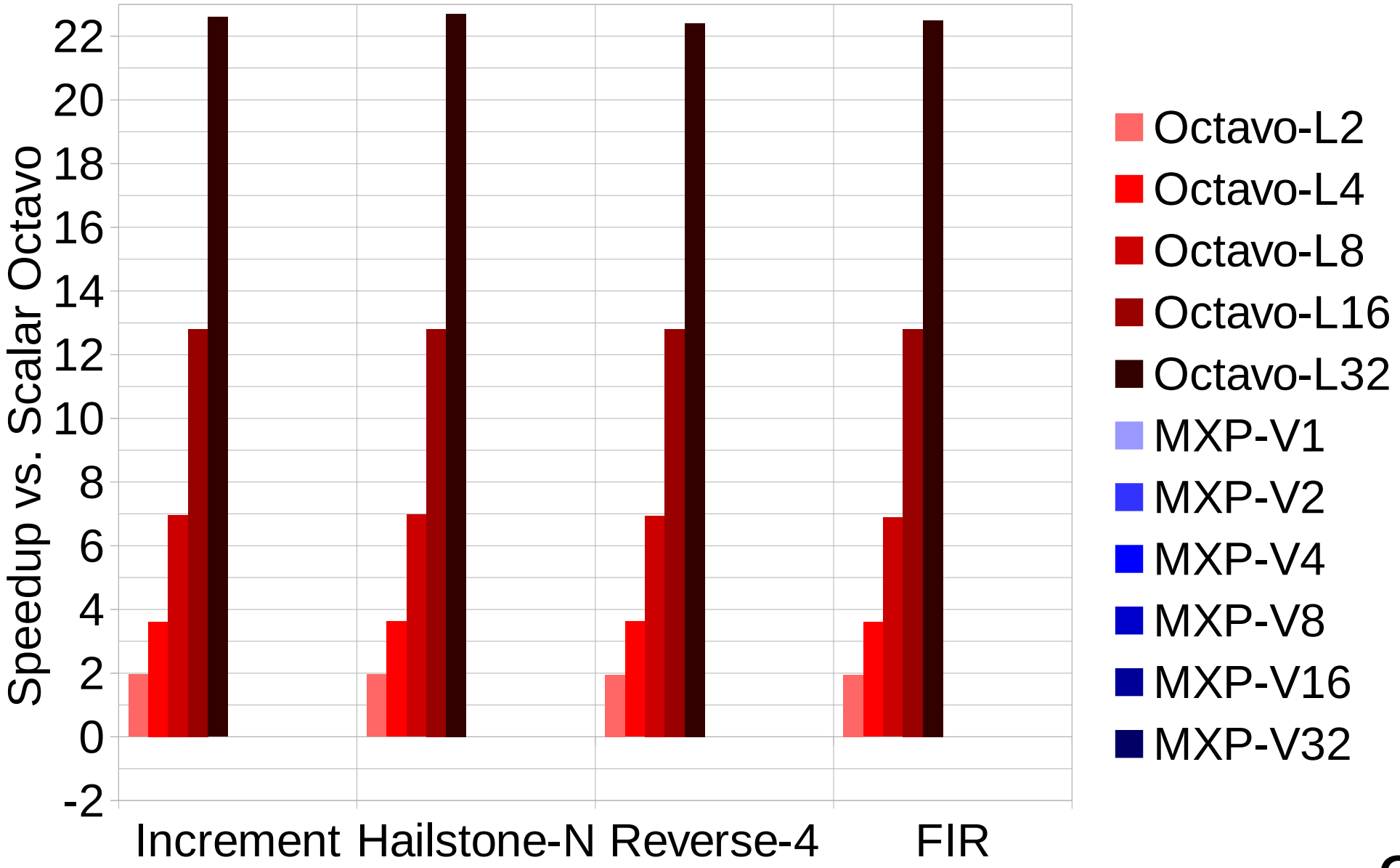
Sequential Speedup



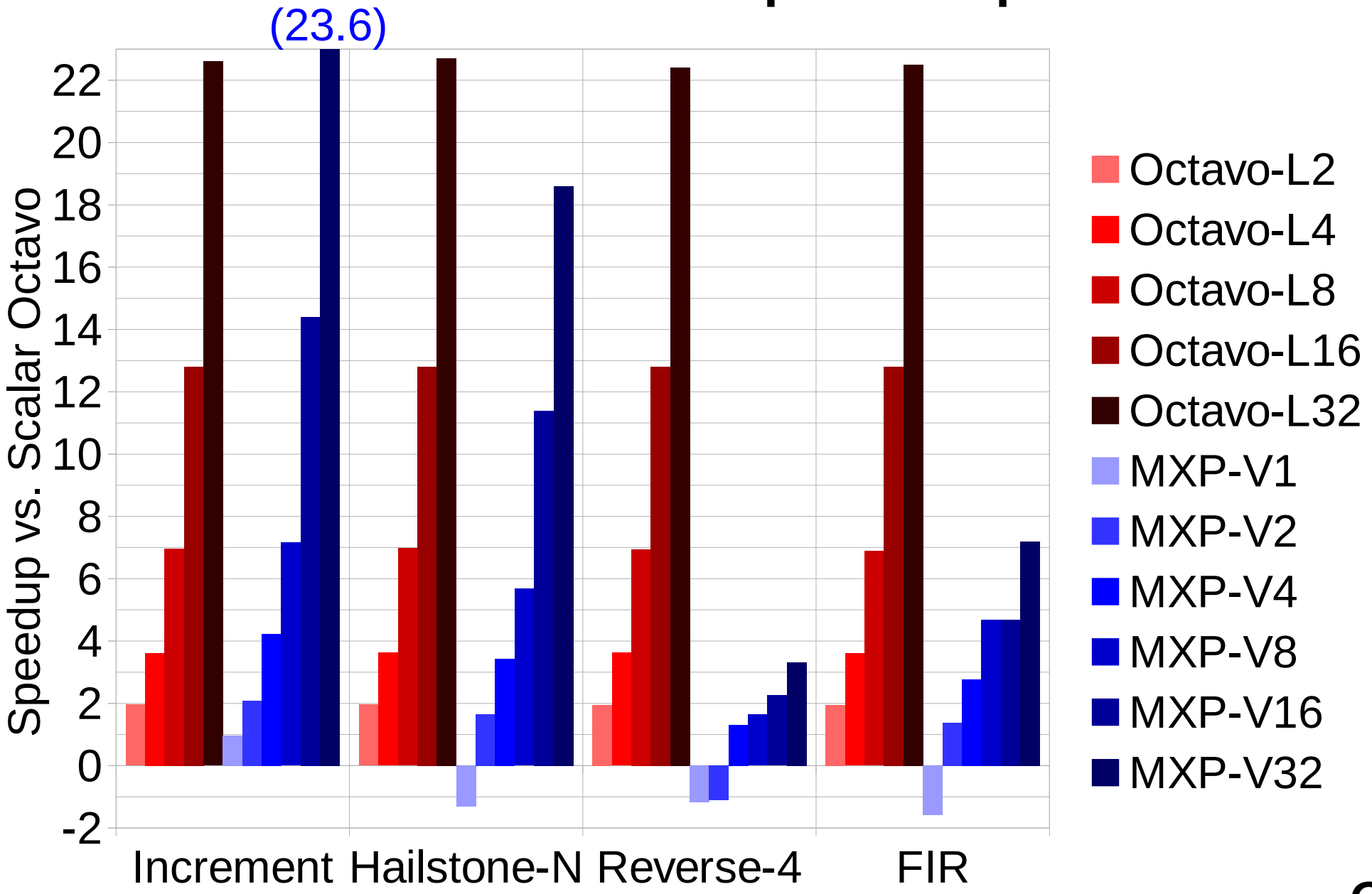
Sequential Speedup



Parallel Speedup



Parallel Speedup



Summary of Contributions

- ***How do we improve overlay performance?***

Summary of Contributions

- ***How do we improve overlay performance?***

1. Octavo: 500+ MHz soft-processor

   - Operates at >90% of absolute maximum on Stratix IV FPGA

   - C. E. LaForest, J. G. Steffan, "*Octavo: An FPGA-Centric Processor Family*", FPGA 2012

Summary of Contributions

- ***How do we improve overlay performance?***

1. Octavo: 500+ MHz soft-processor
   - Operates at >90% of absolute maximum on Stratix IV FPGA
   - C. E. LaForest, J. G. Steffan, "*Octavo: An FPGA-Centric Processor Family*", FPGA 2012
2. Preserved performance under scaling
   - e.g.: Fmax -22% over 102x scaling
   - C. E. LaForest, J. G. Steffan, "M*aximizing Speed and Density of Tiled FPGA Overlays via Partitioning*", ICFPT 2013

Summary of Contributions

- ***How do we improve overlay performance?***

1. Octavo: 500+ MHz soft-processor
   - Operates at >90% of absolute maximum on Stratix IV FPGA
   - C. E. LaForest, J. G. Steffan, "*Octavo: An FPGA-Centric Processor Family*", FPGA 2012
2. Preserved performance under scaling
   - e.g.: Fmax -22% over 102x scaling
   - C. E. LaForest, J. G. Steffan, "M*aximizing Speed and Density of Tiled FPGA Overlays via Partitioning*", ICFPT 2013
3. Overlap execution overhead with useful work
   - Better speedup than loop unrolling on "perfect" MIPS CPU
   - C. E. LaForest, J. H. Anderson, J. G. Steffan, "*Approaching Overhead-Free Execution on FPGA Soft-Processors*", ICFPT 2014

Future Work

- Programming Support

Future Work

- Programming Support
- Further reduce execution overhead

Future Work

- Programming Support
- Further reduce execution overhead
- Increase resource diversity

Future Work

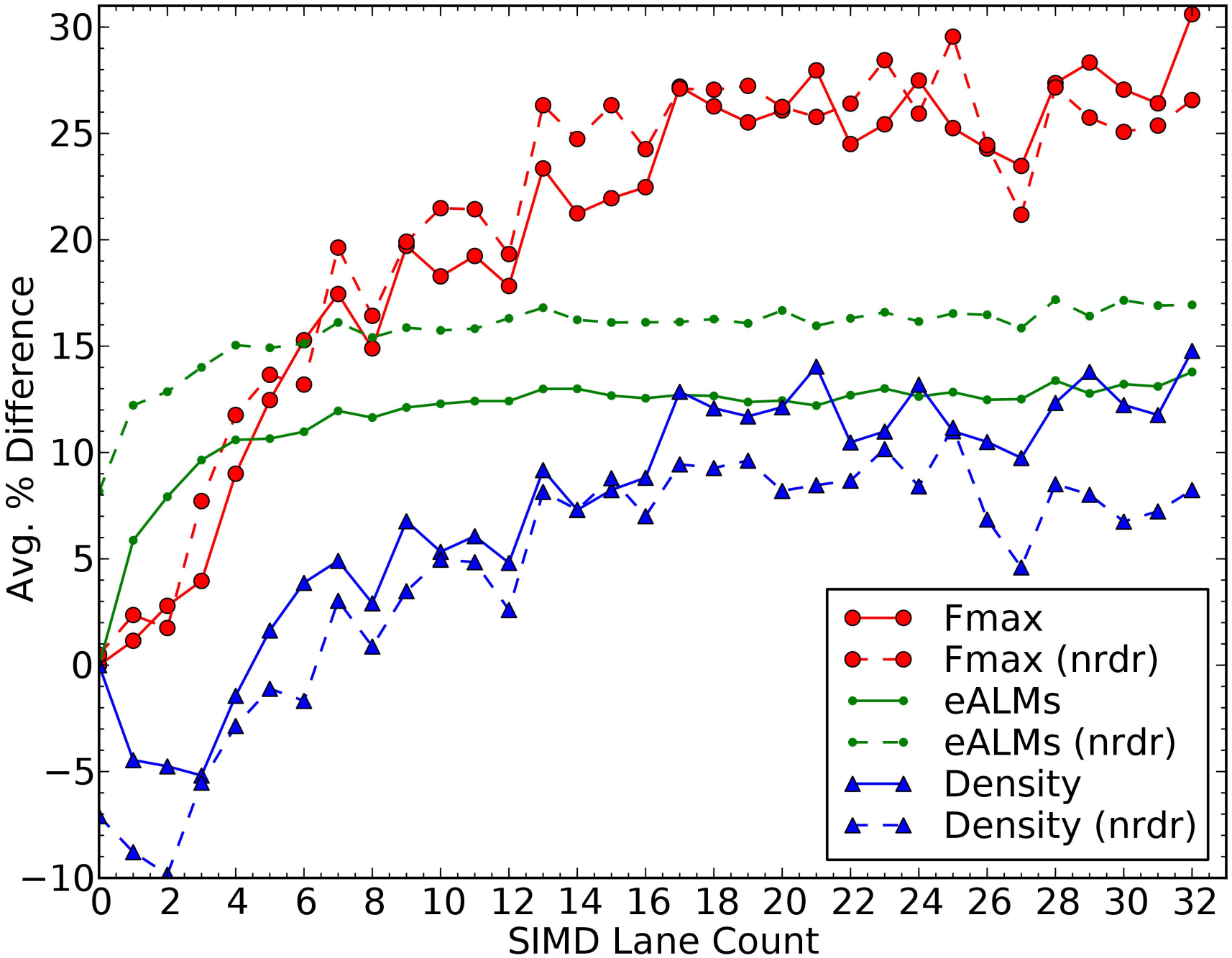
- Programming Support
- Further reduce execution overhead
- Increase resource diversity
- Increase efficiency
  - Non-branching code
  - Internal Memory bandwidth
  - Bit-level Parallelism
  - ALU Utilization
  - Measure/Reduce Power

Future Work

https://github.com/laforest/Octavo

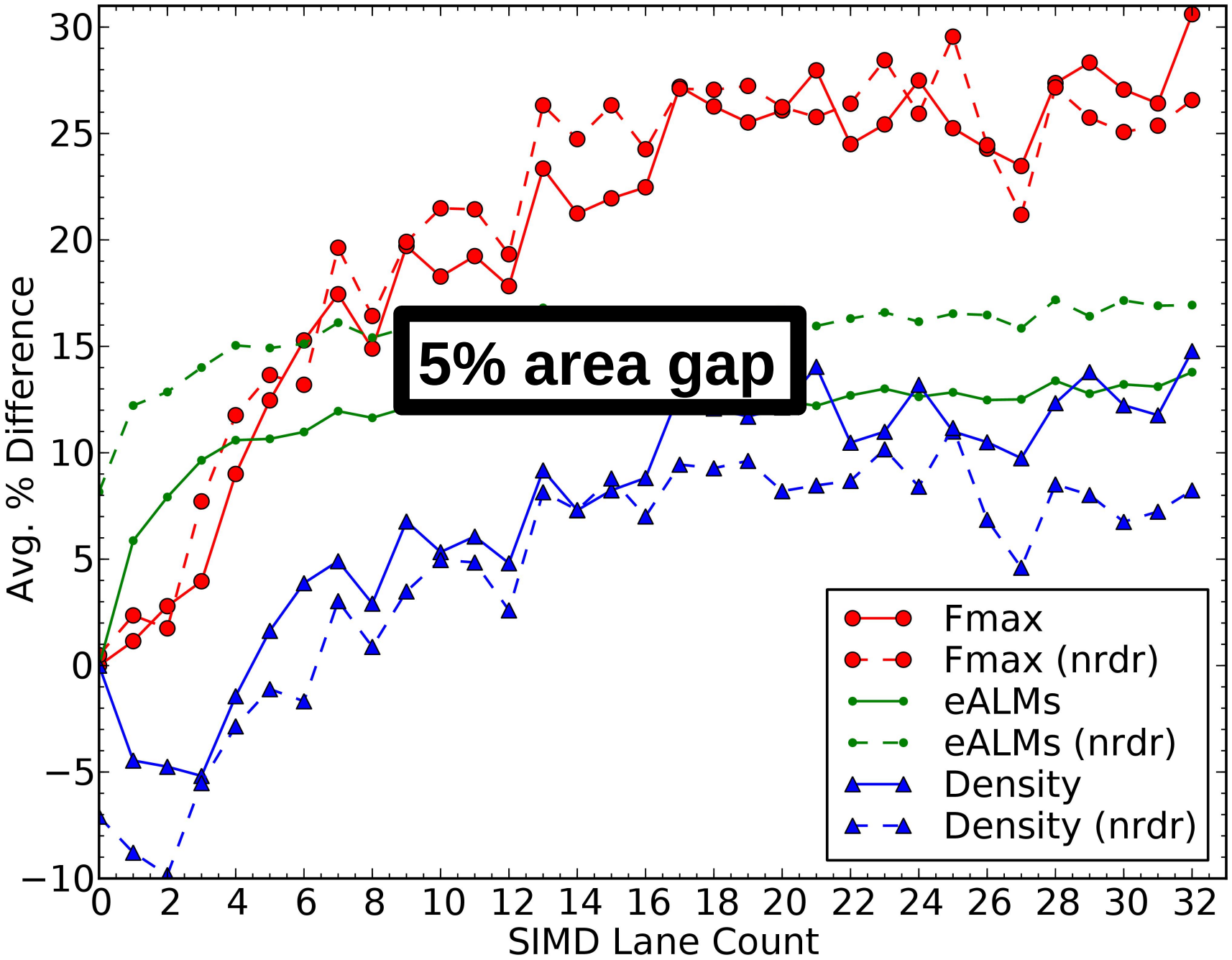
Extra Slides

Impact on Area



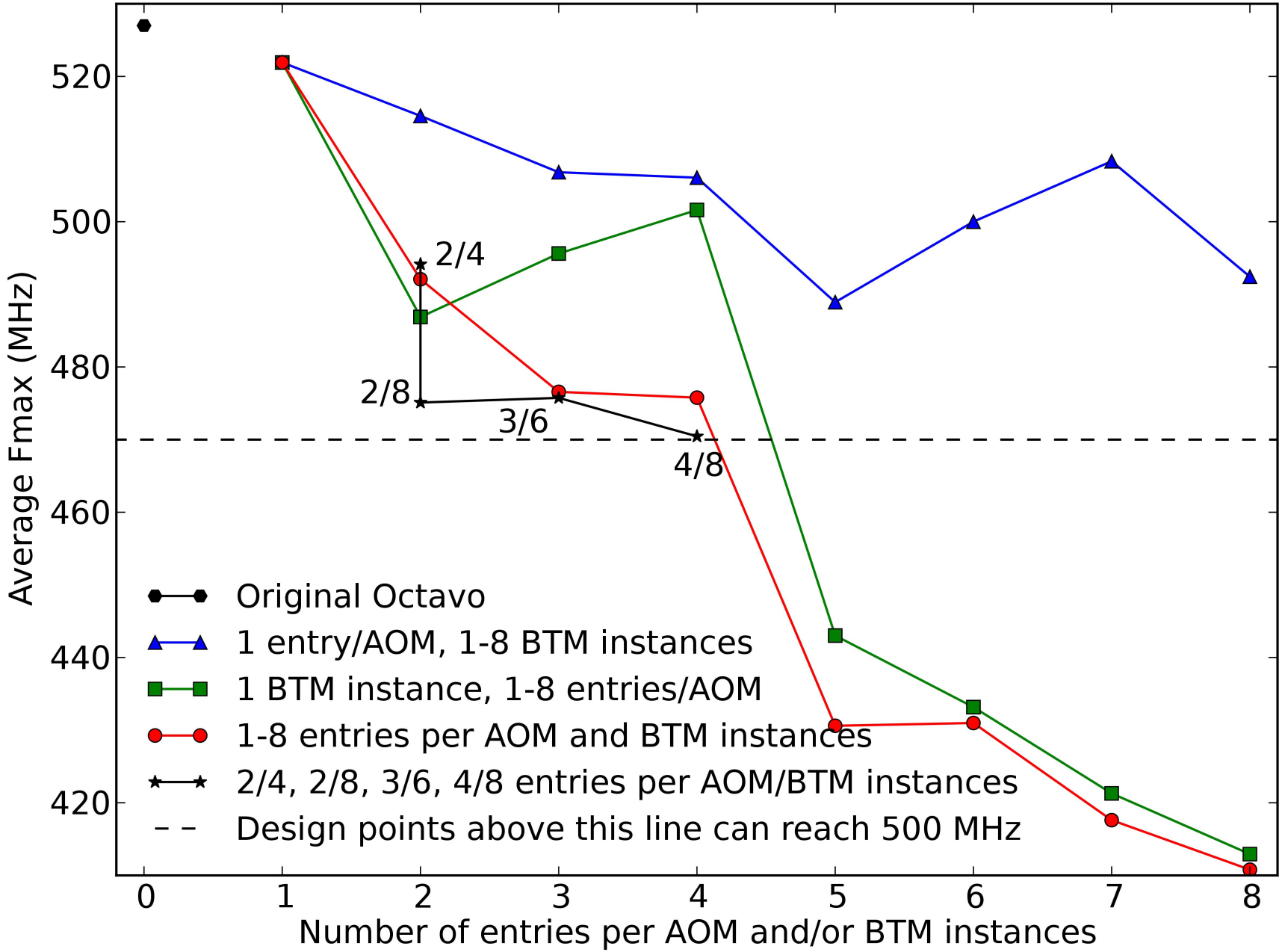
.03

Impact on Area

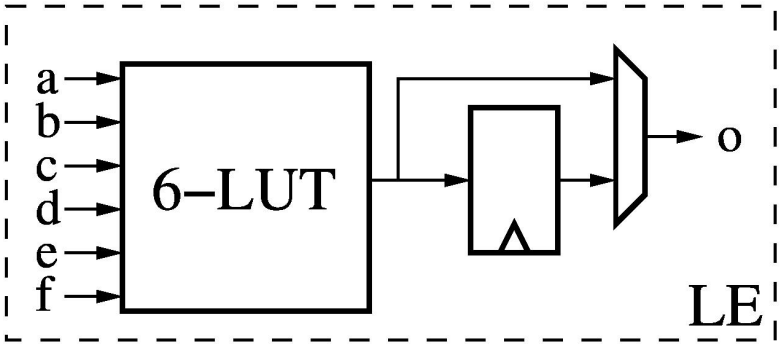


.04

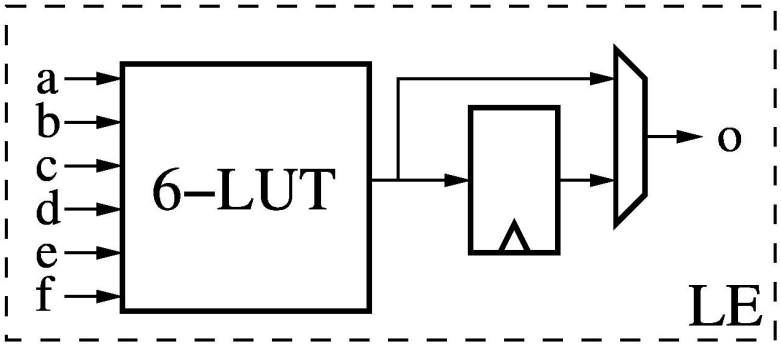
AOM/BTM Configurations

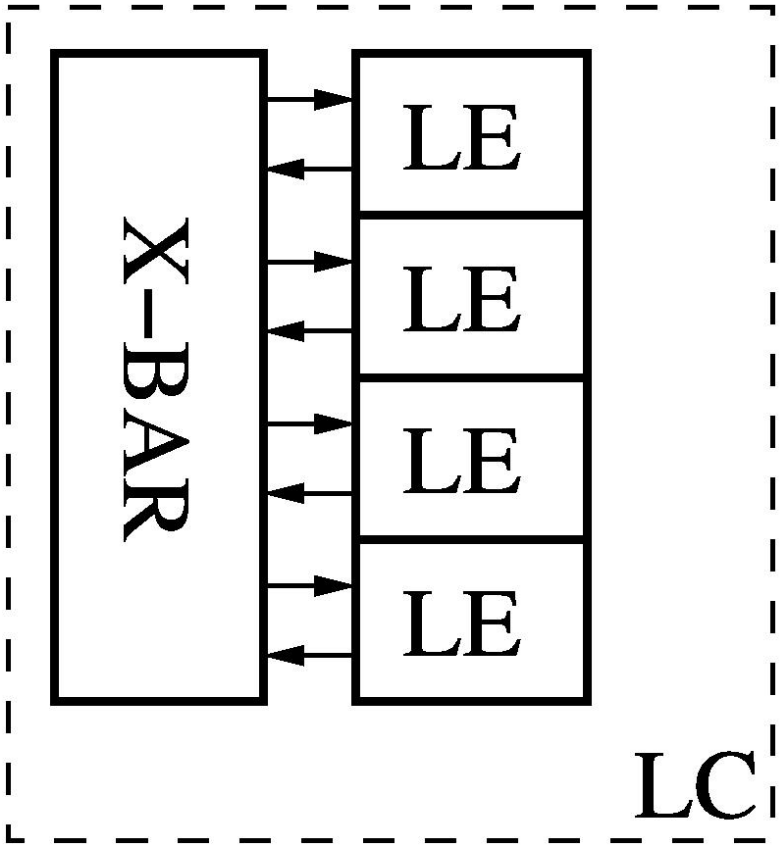


Logic Element (LE)

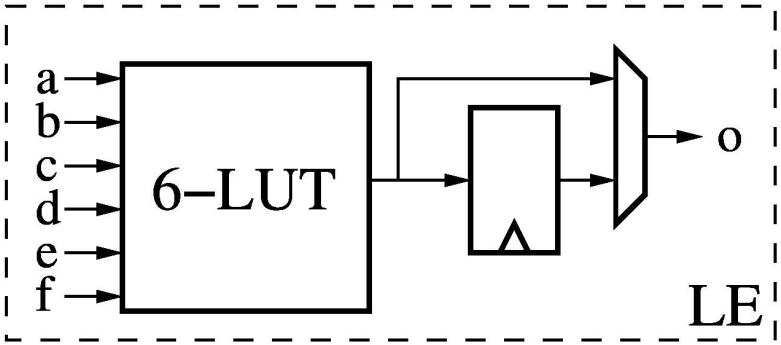


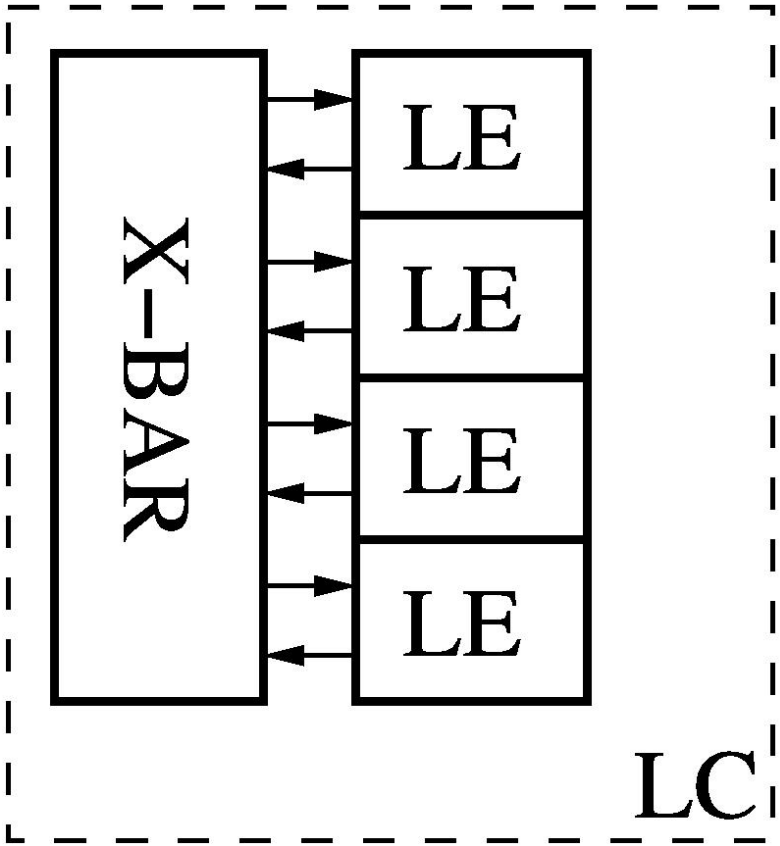
Logic Cluster (LC)

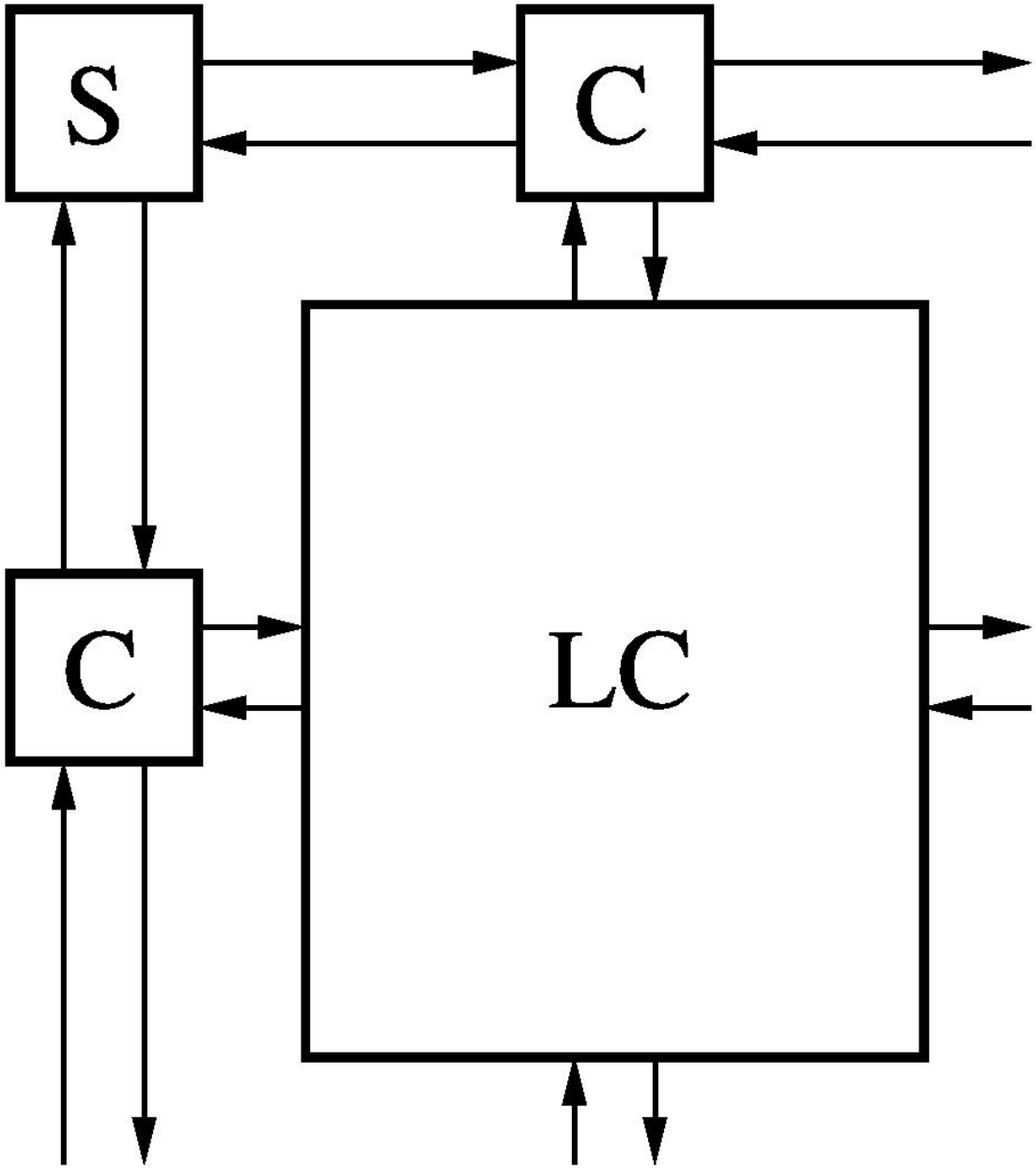




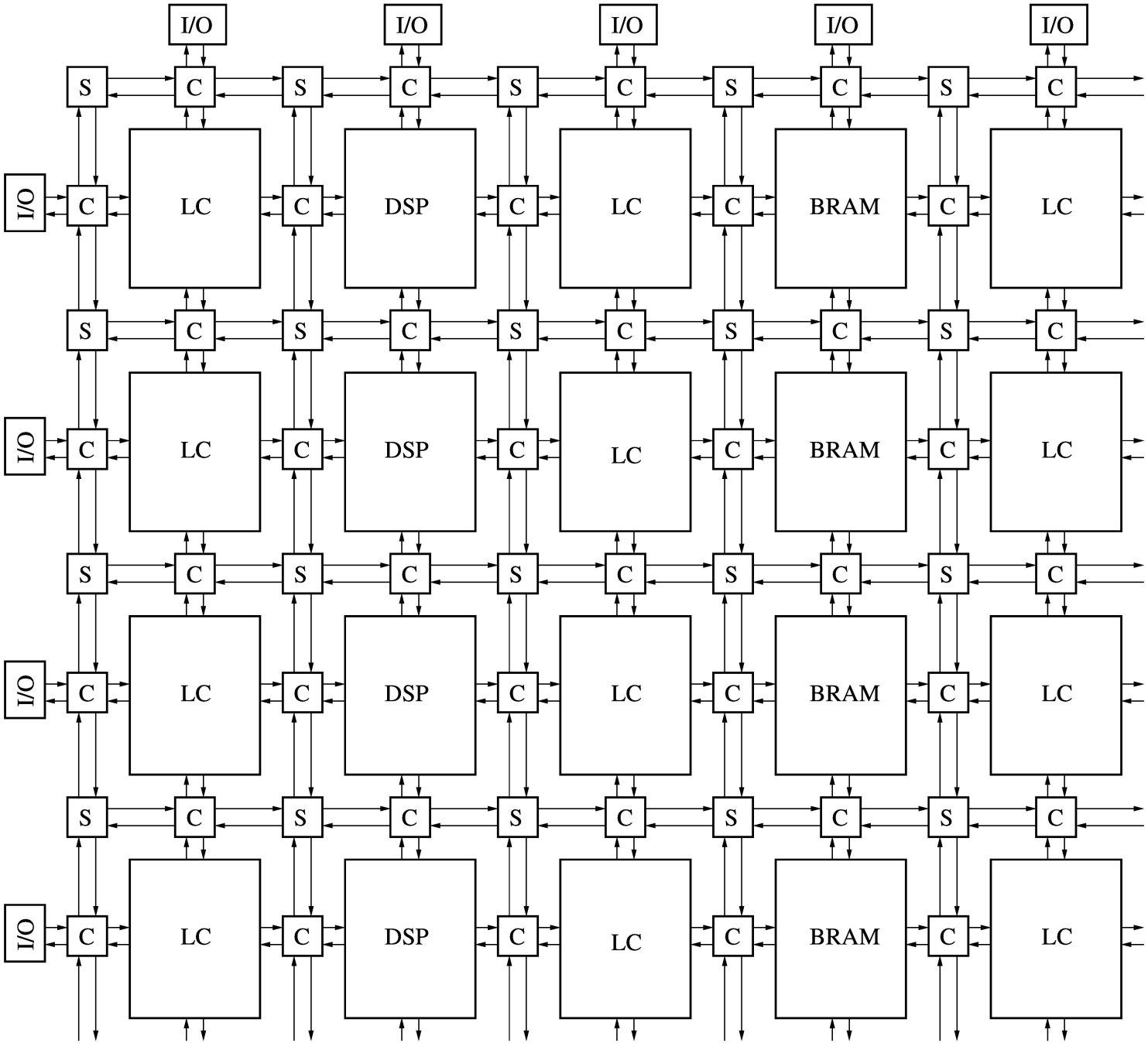
Switch Boxes and Connection Blocks



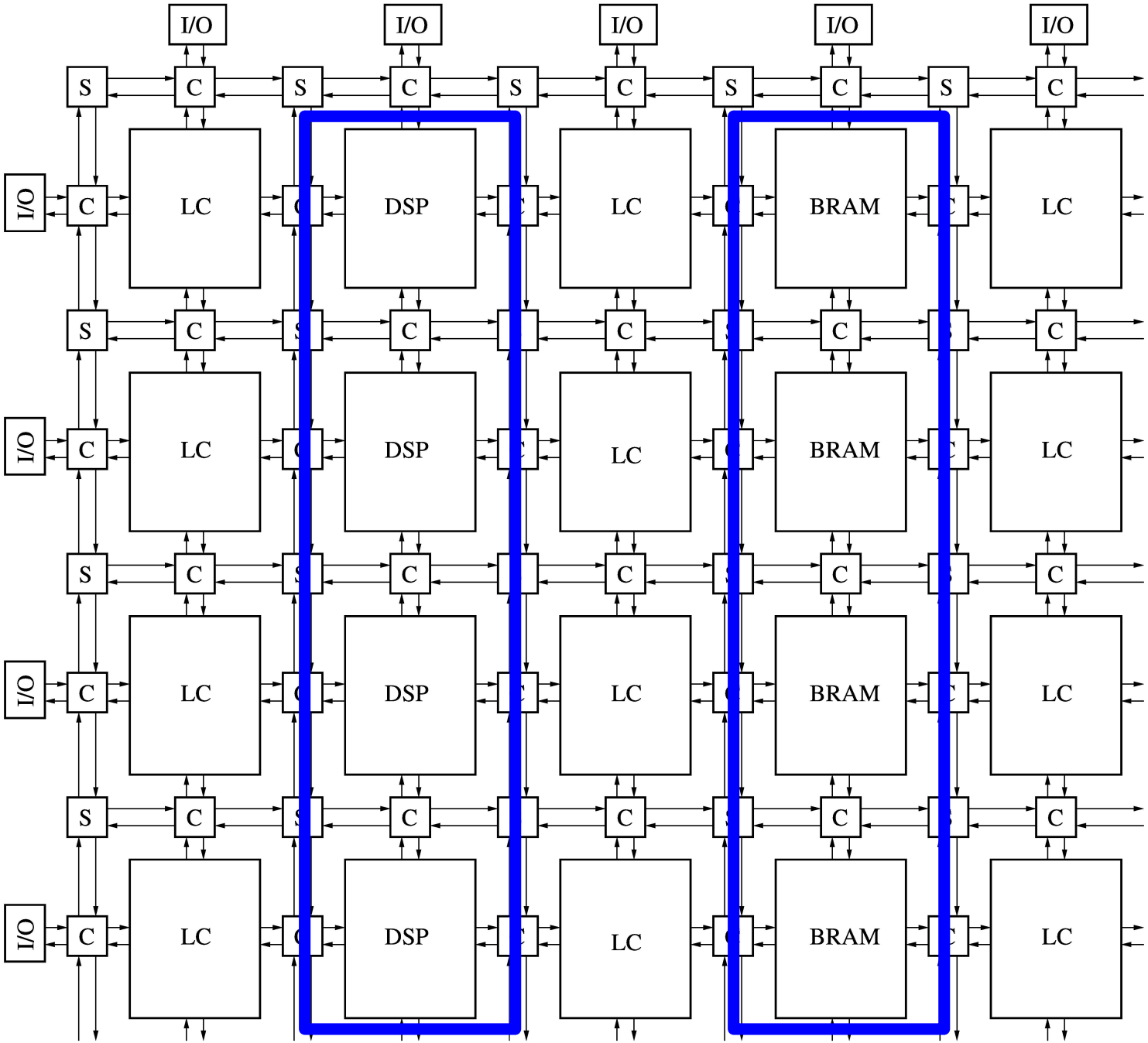




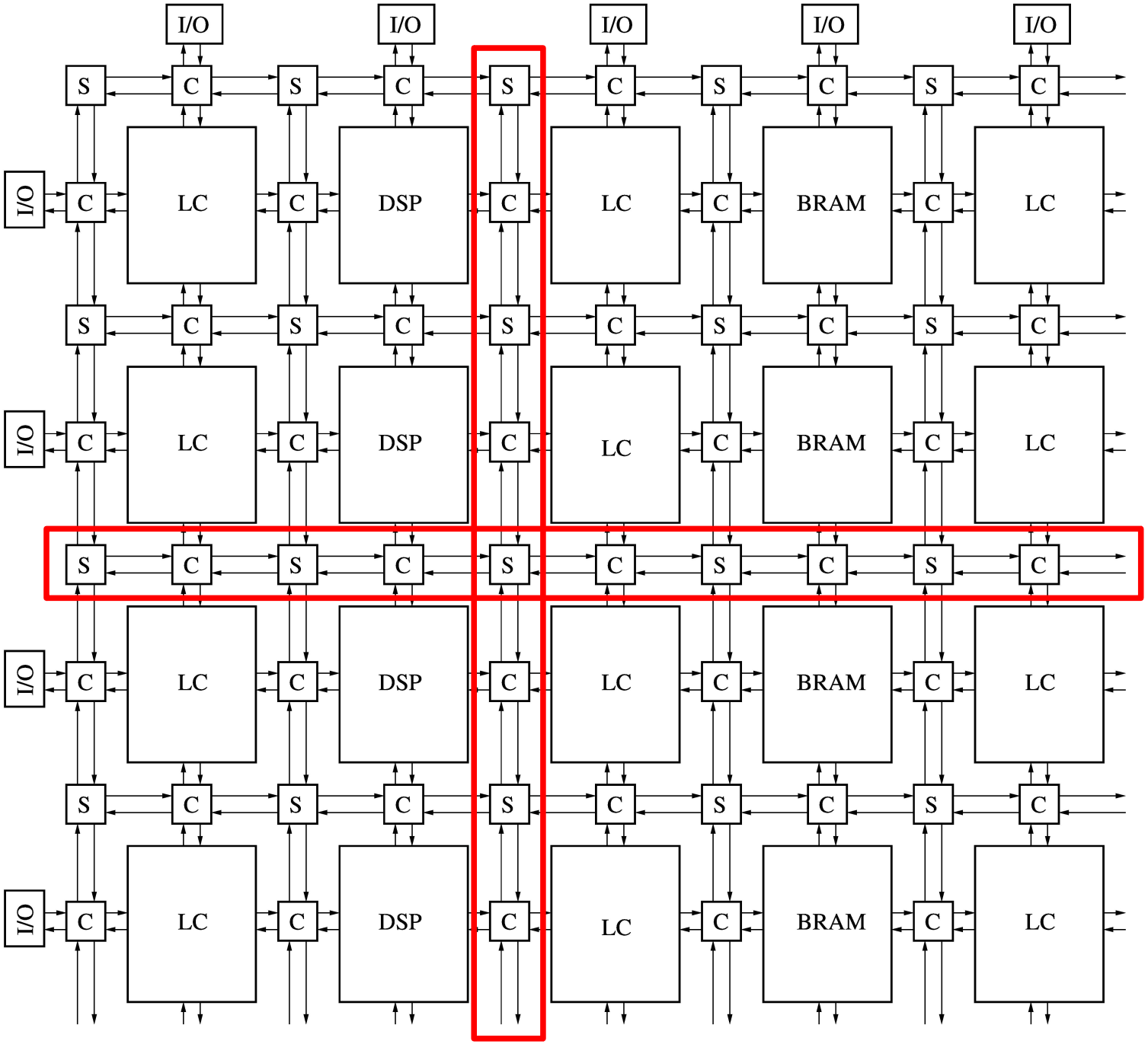
Generic Island-Style FPGA



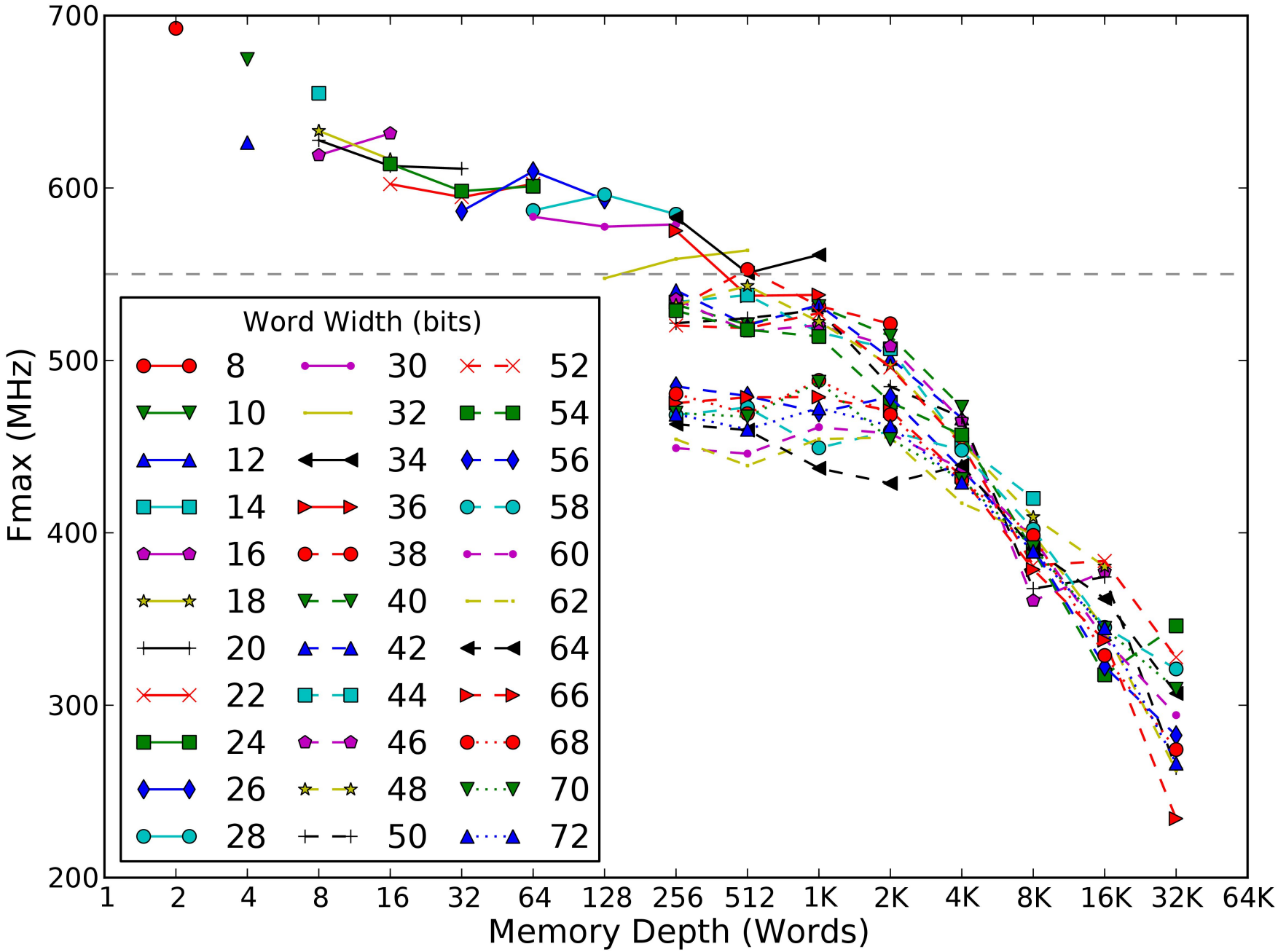
Hard Blocks (RAM, DSP)



Interconnect Delay Dominates



Fmax vs. Memory Depth



Fmax vs. Memory Depth

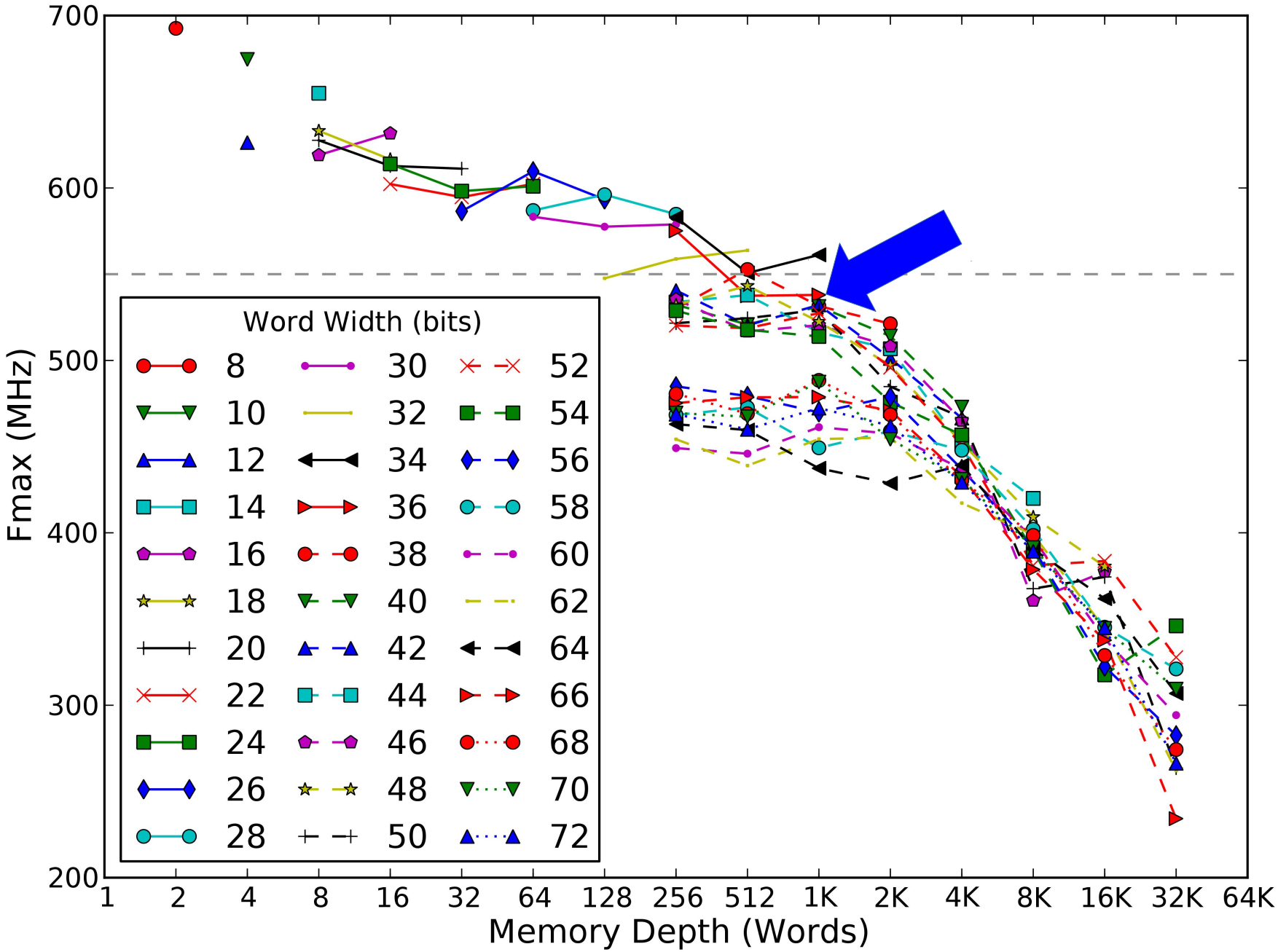
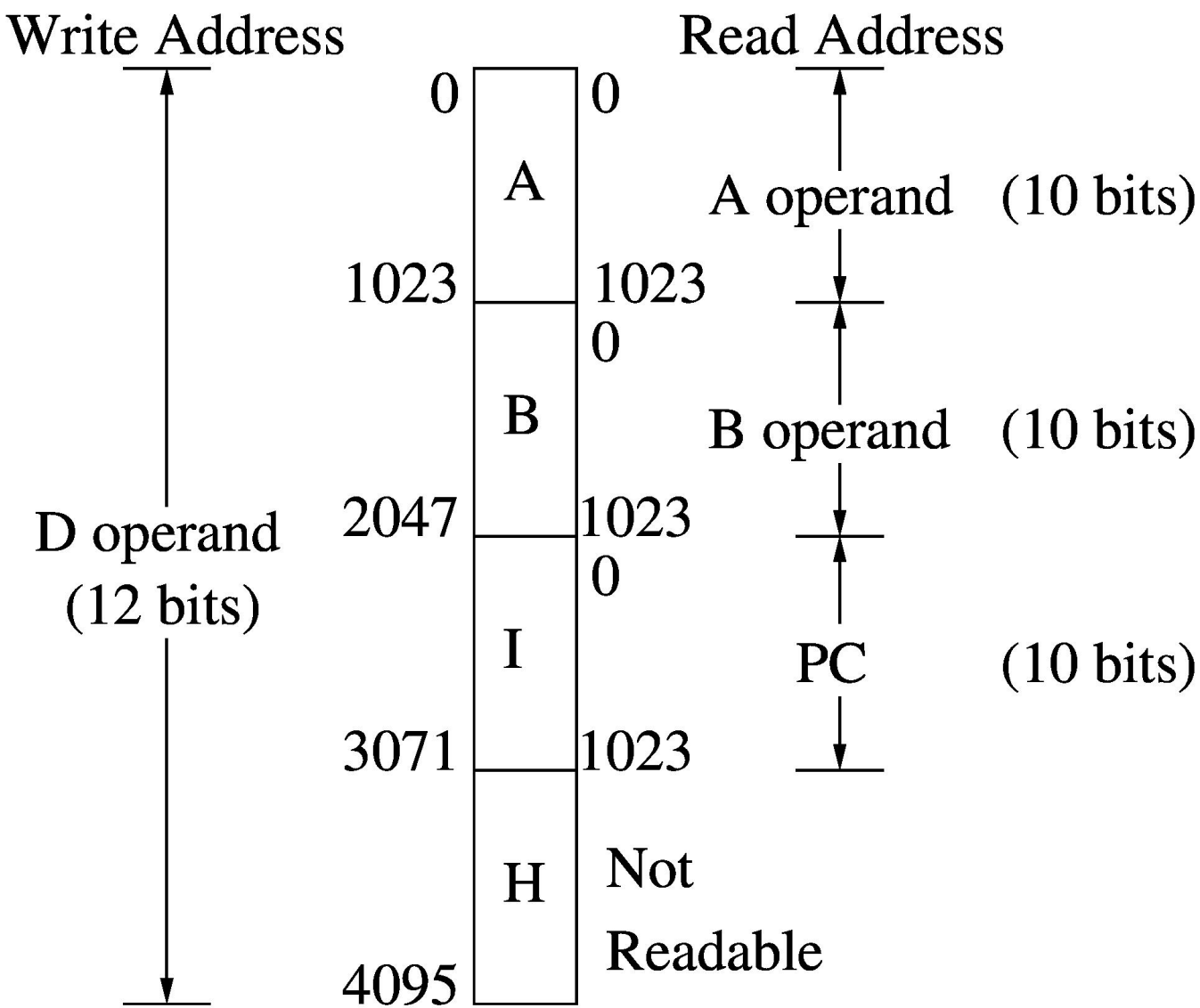
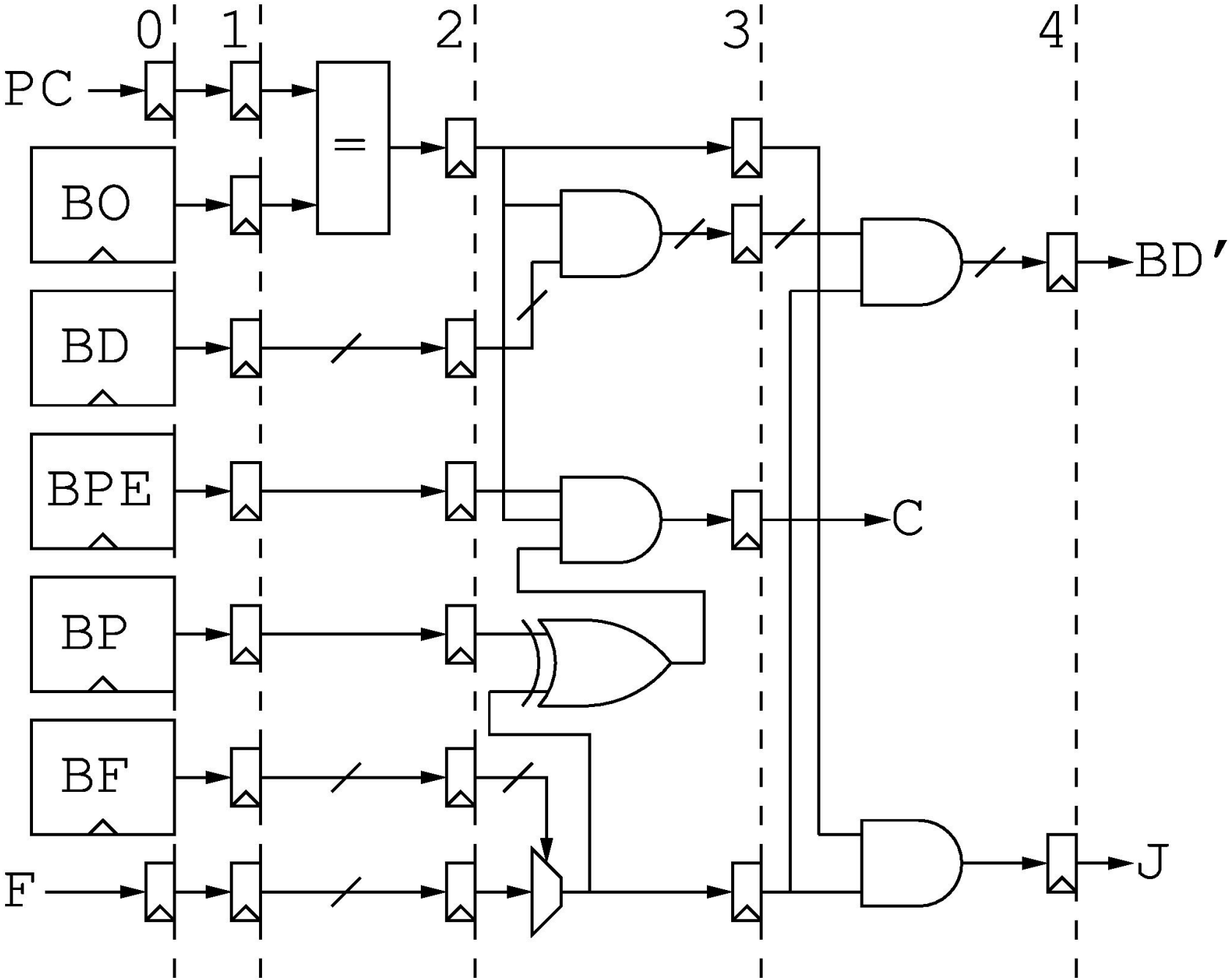


Table 5.1: Octavo's Instruction Word Format with Extended Write Address Space

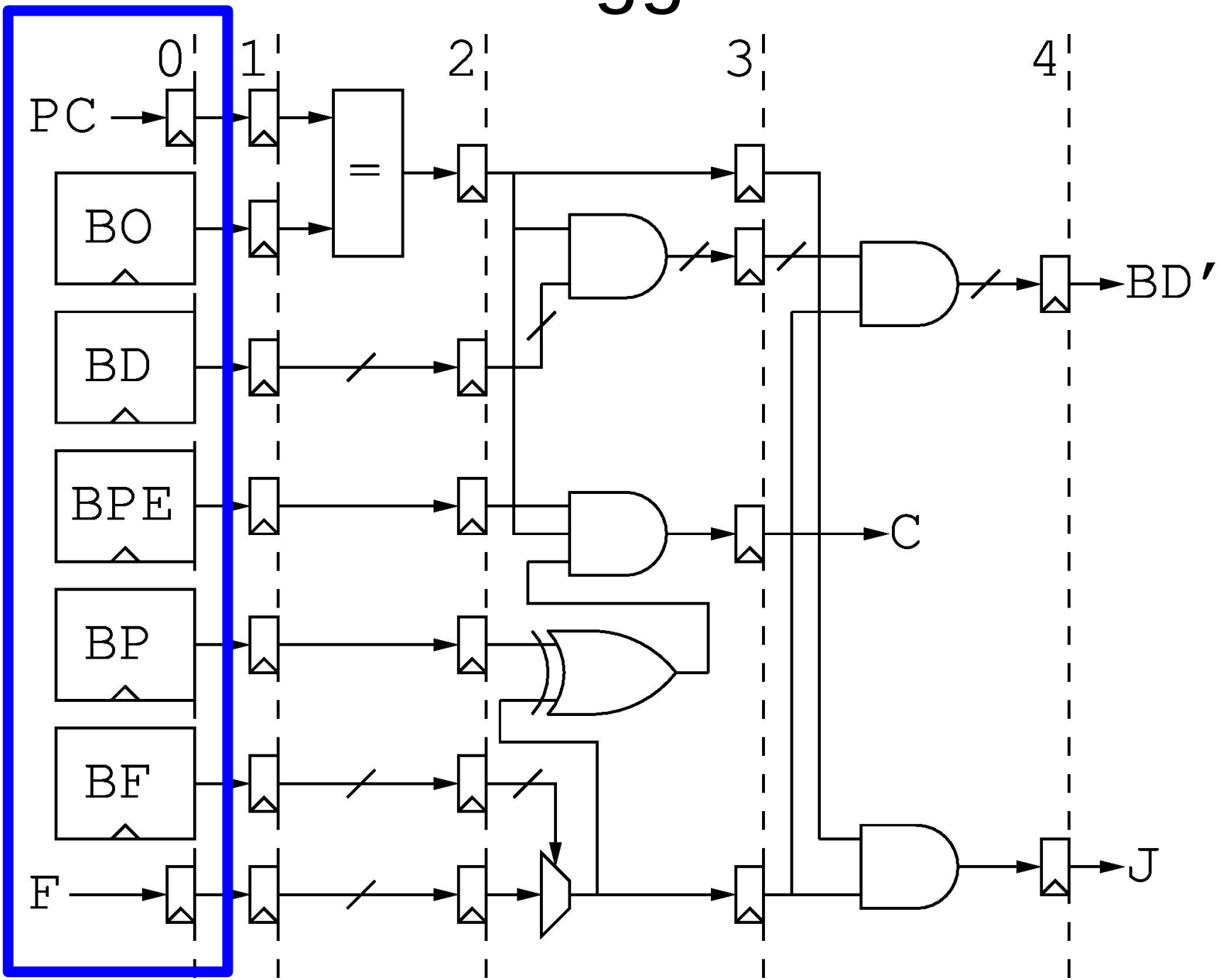
Size:	4 bits	12 bits	10 bits	10 bits
Field:	Opcode (OP)	Destination (D)	Source (A)	Source (B)



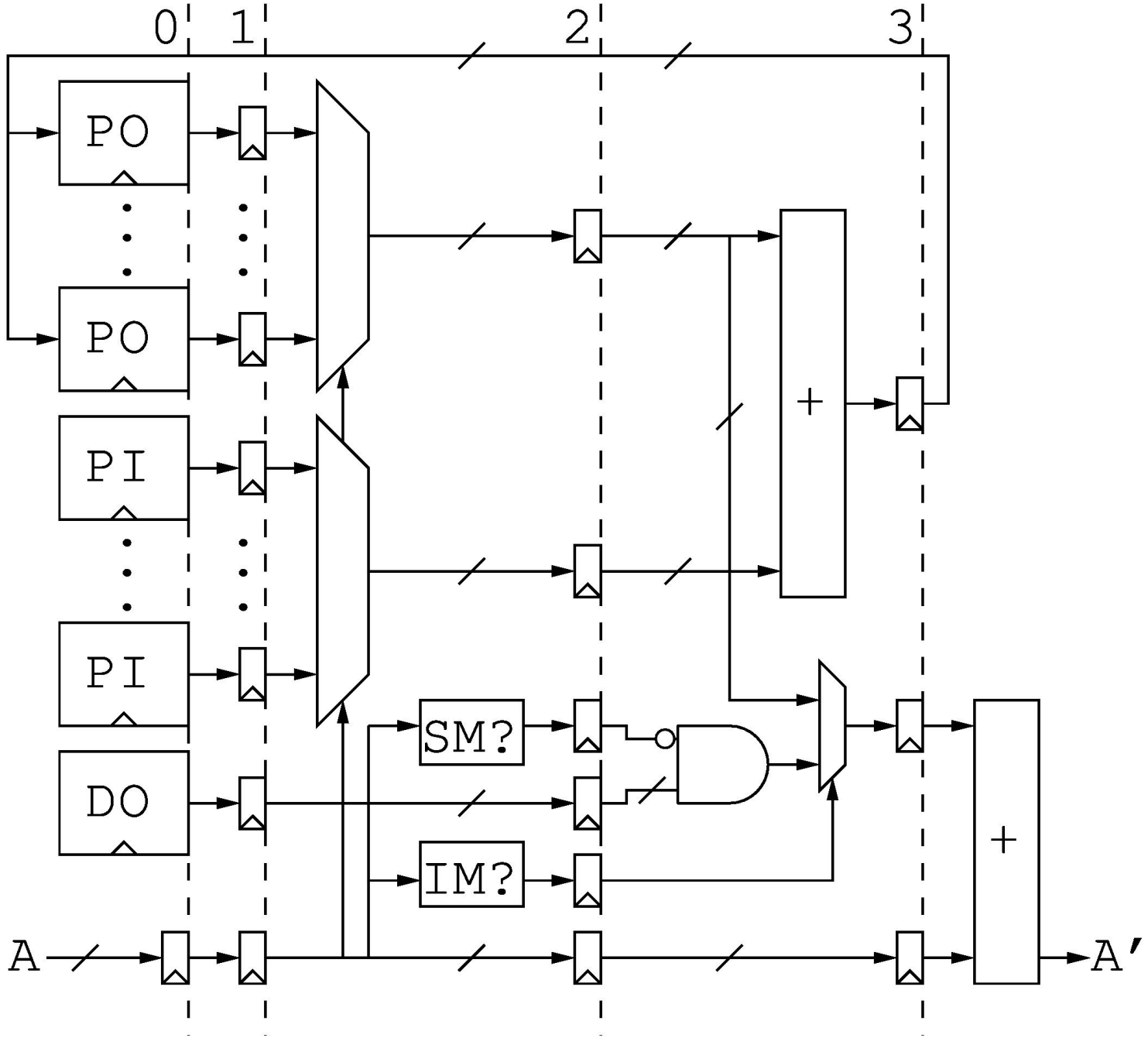
Branch Trigger Module



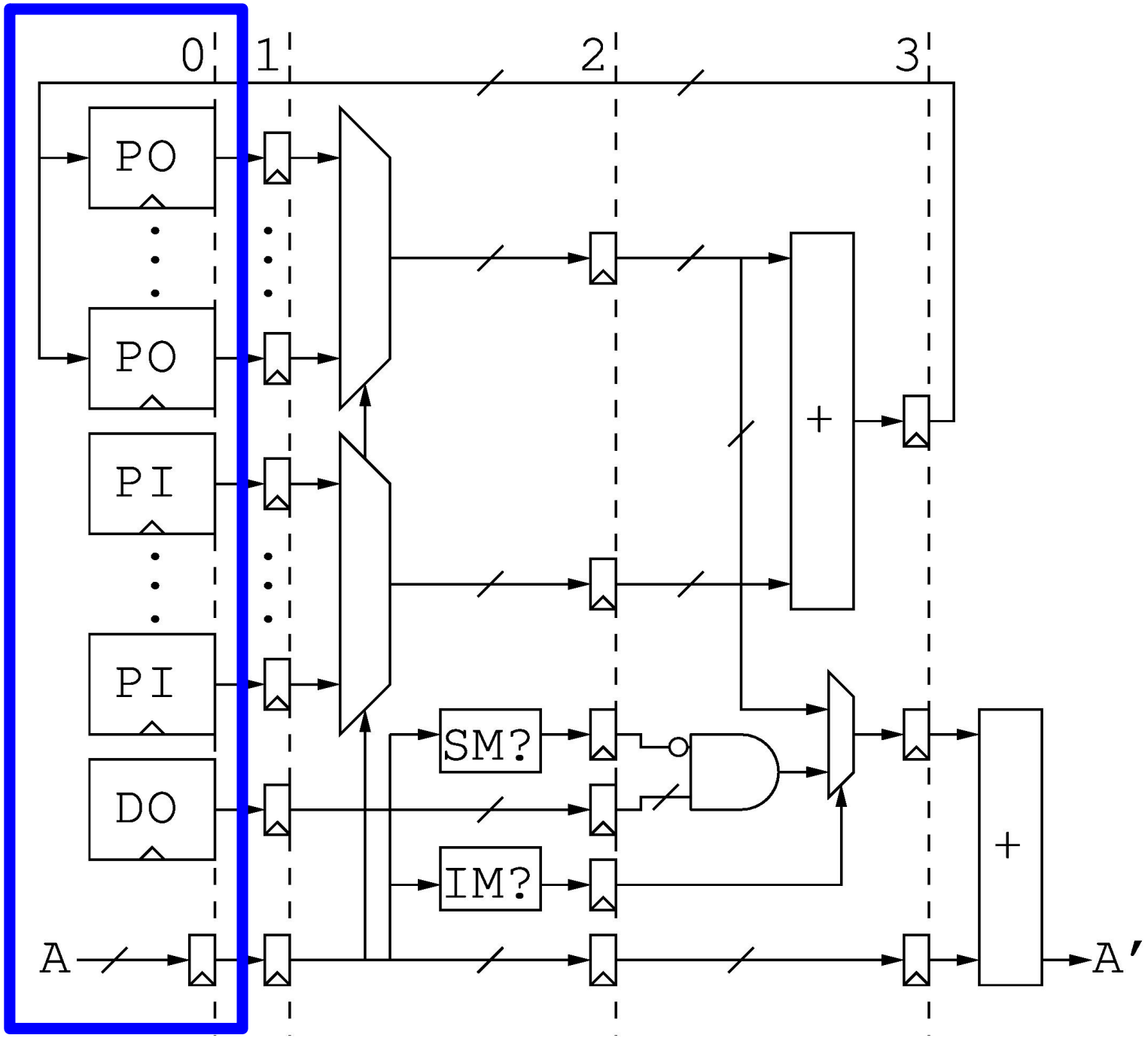
Branch Trigger Module



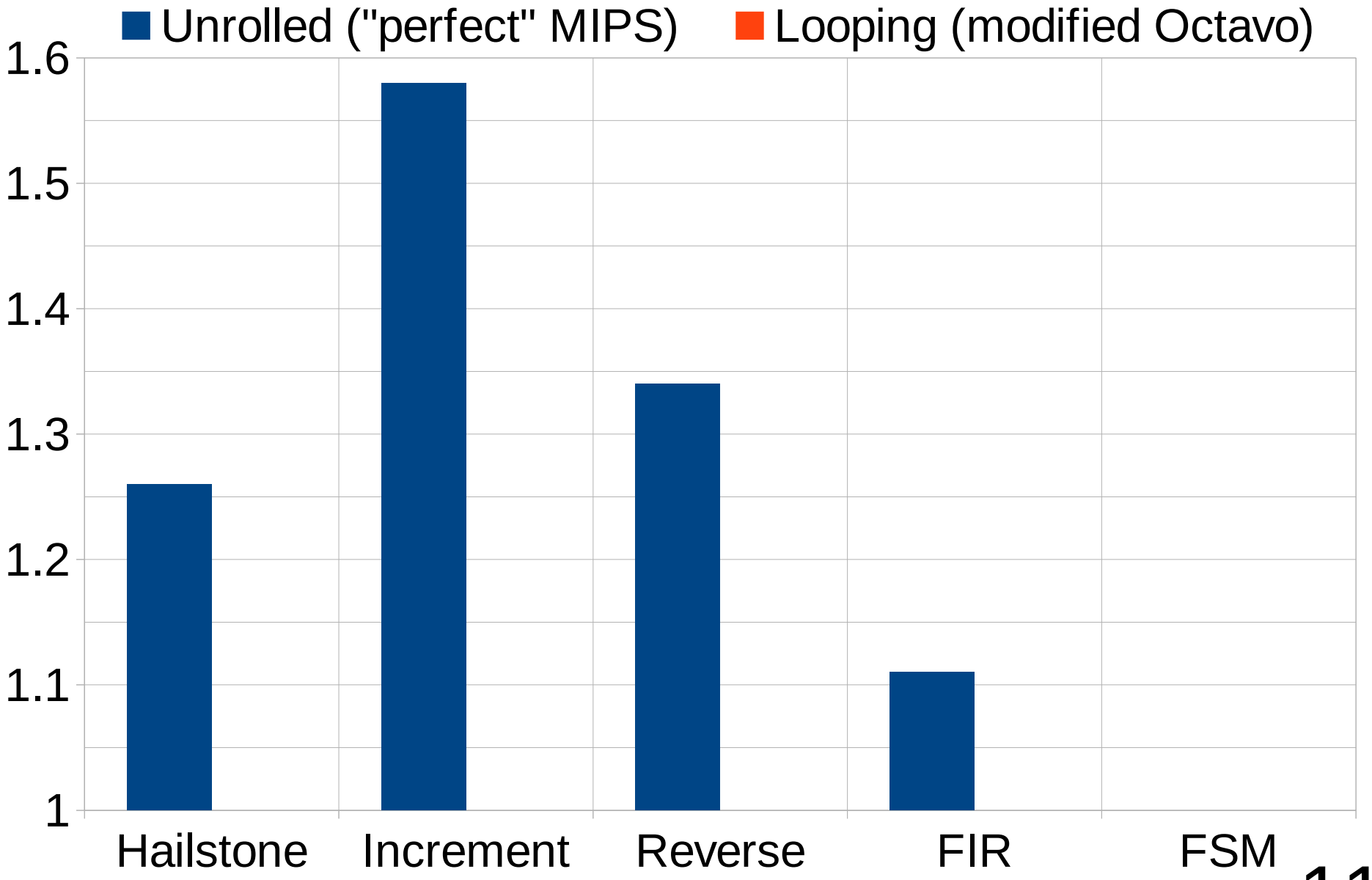
Address Offset Module



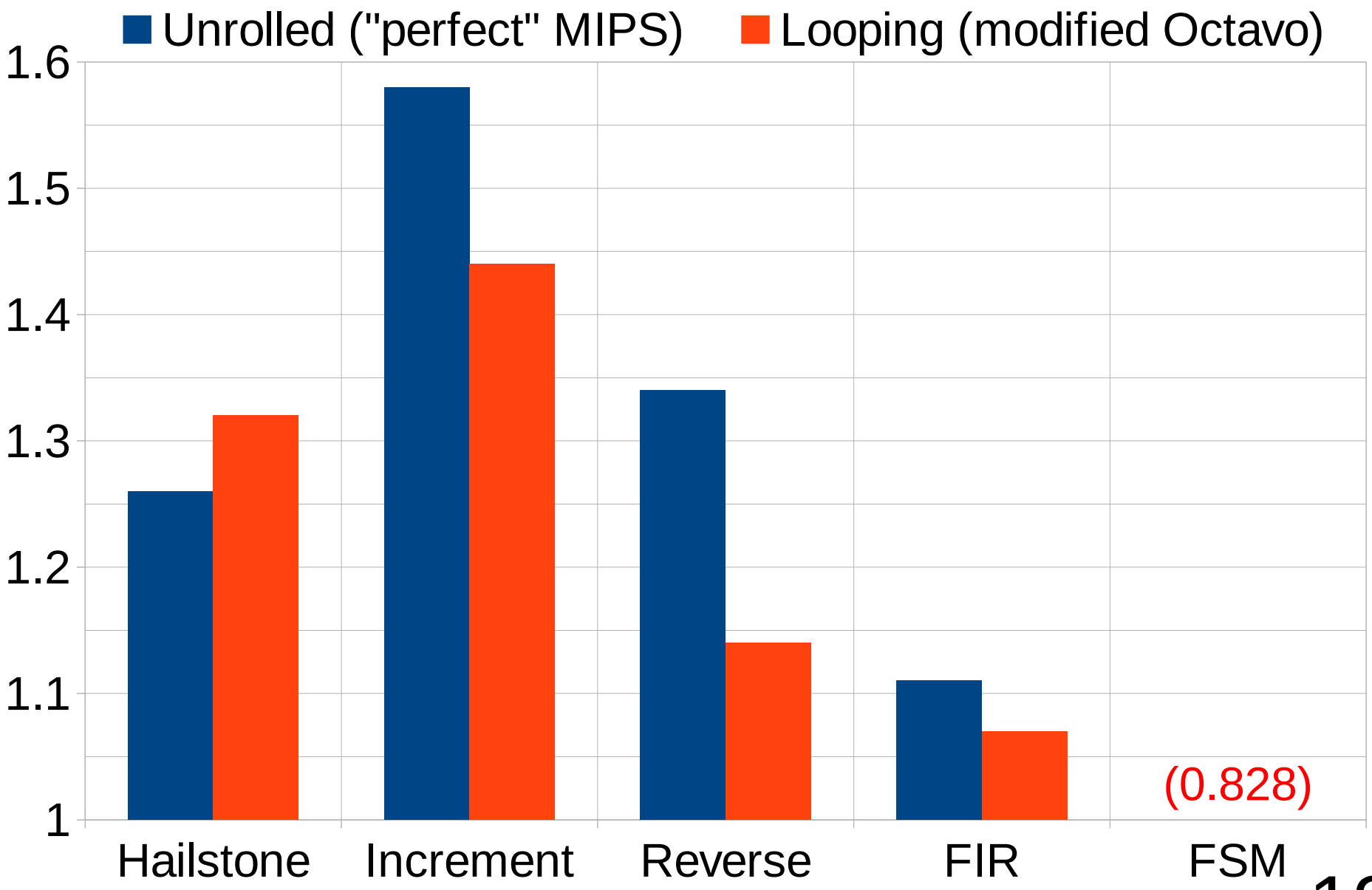
Address Offset Module



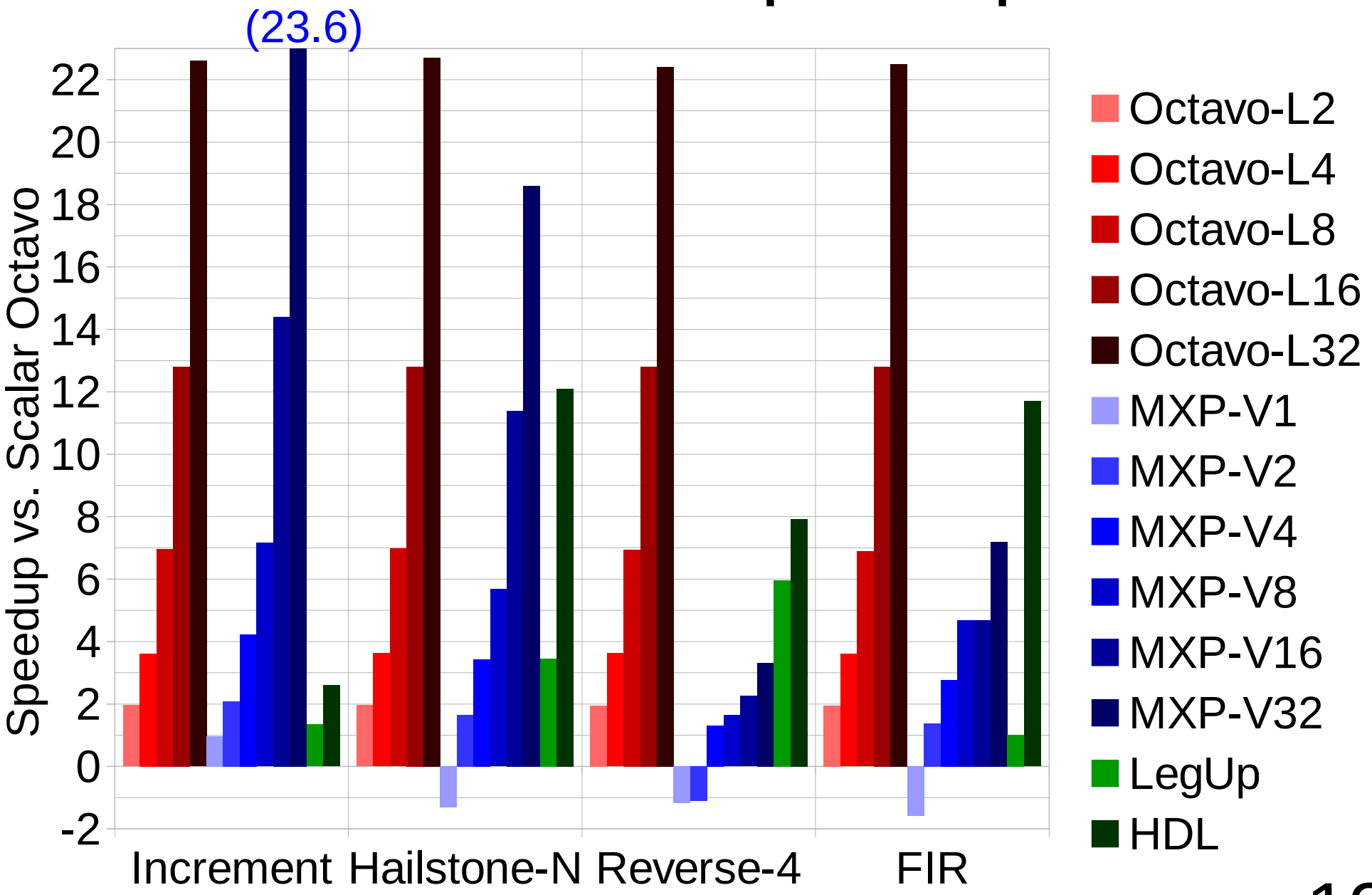
Efficiency Increase



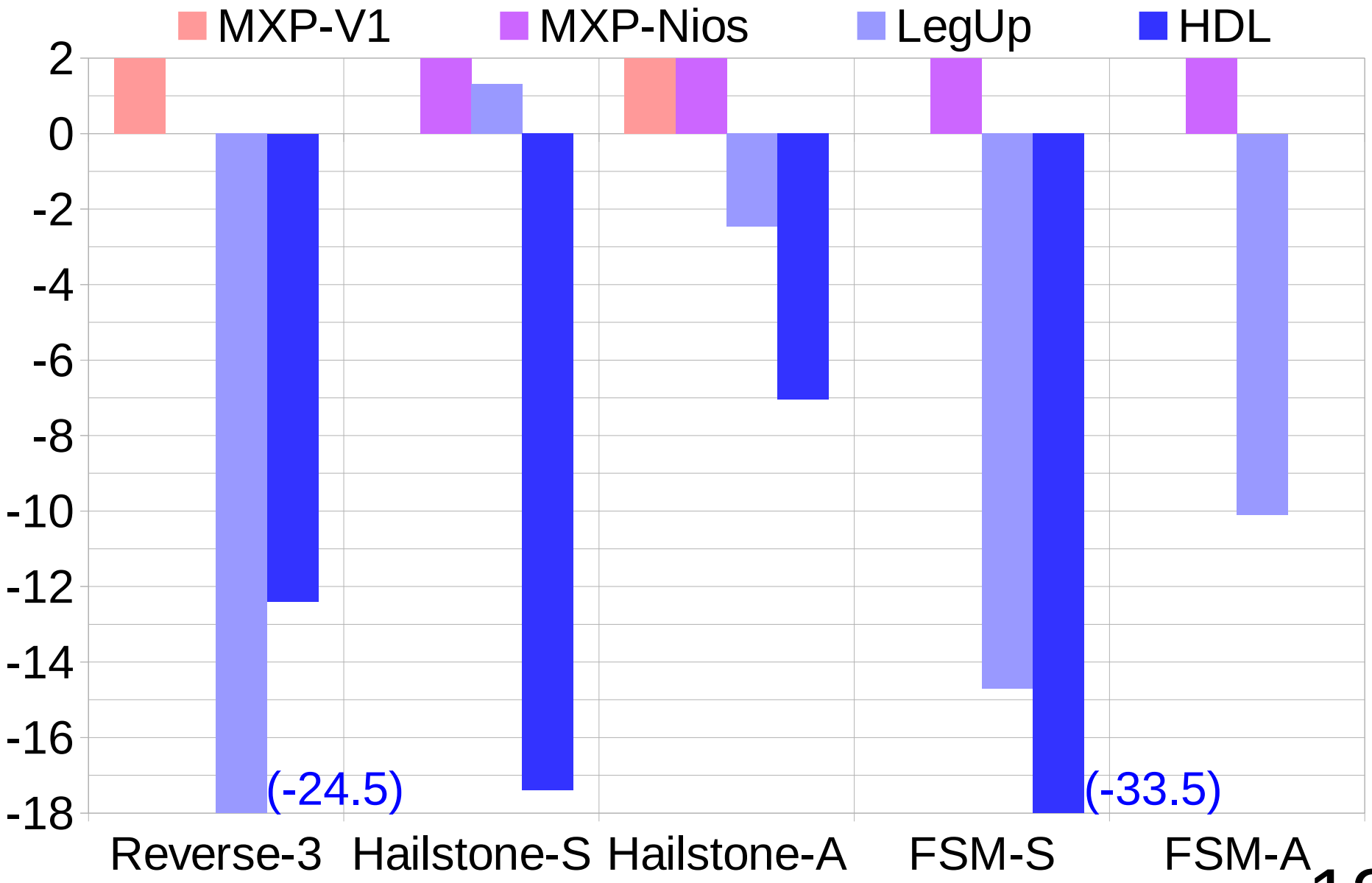
Efficiency Increase



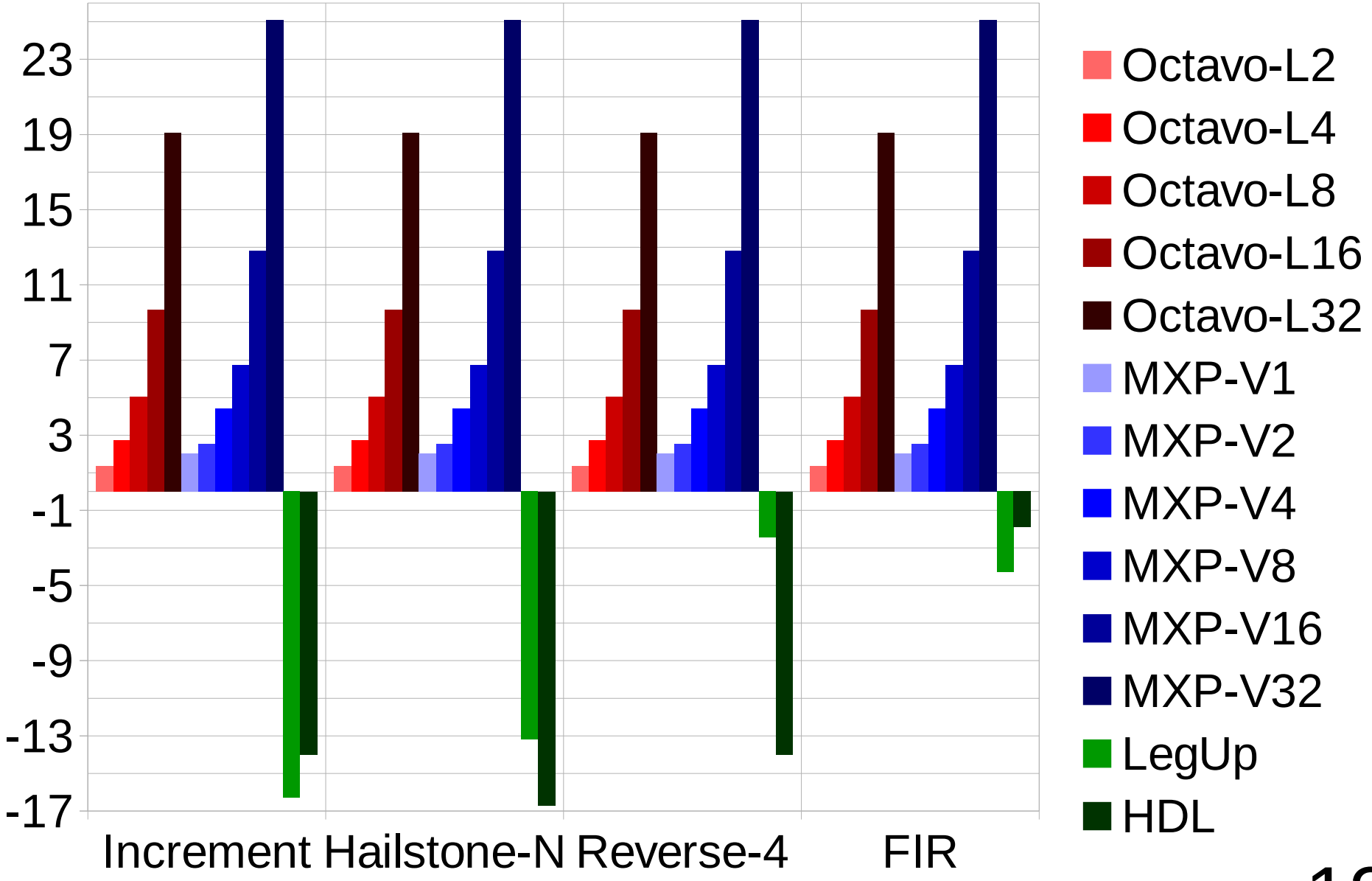
Parallel Speedup



Sequential Area Ratios



Parallel Area Ratios



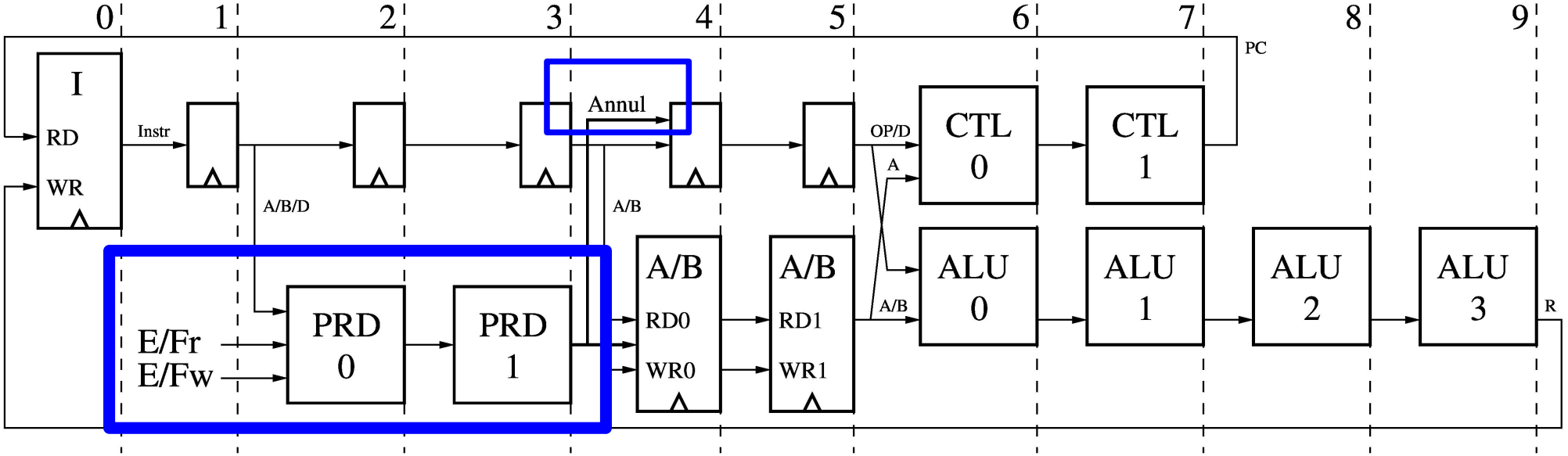
Planning for Larger Systems

- Problems
  - Inefficient use of memory (overlapping)
  - Wasting I/O ports on low-traffic control hardware
  - Software busy-wait loops for I/O

Planning for Larger Systems

- Problems
  - Inefficient use of memory (overlapping)
  - Wasting I/O ports on low-traffic control hardware
  - Software busy-wait loops for I/O
- Solutions
  - Extending the write address space (2 spare bits)
  - Predicating instructions on I/O port readiness
  - Decreases Fmax 4.5%, increases area 5.5%

Instruction I/O Predication



- Empty/Full bit on each read/write I/O port
- Annul instruction if not all addressed ports ready
- Re-issue instruction next thread cycle