Approaching Overhead-Free Execution on FPGA Soft-Processors

Charles Eric LaForest
Jason Anderson
J. Gregory Steffan

University of Toronto

ICFPT 2014, Shanghai

Motivation

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

Clip art by Angela Melick, http://www.wastedtalent.ca/

Better Design Processes

- FPGA Overlays (soft-processors)
  - Easy and fast: design system as software
  - Co-design hardware only if necessary
  - Fast overall design cycle
  - Lower performance

Raw Performance Loss

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

CPU Internal Overhead

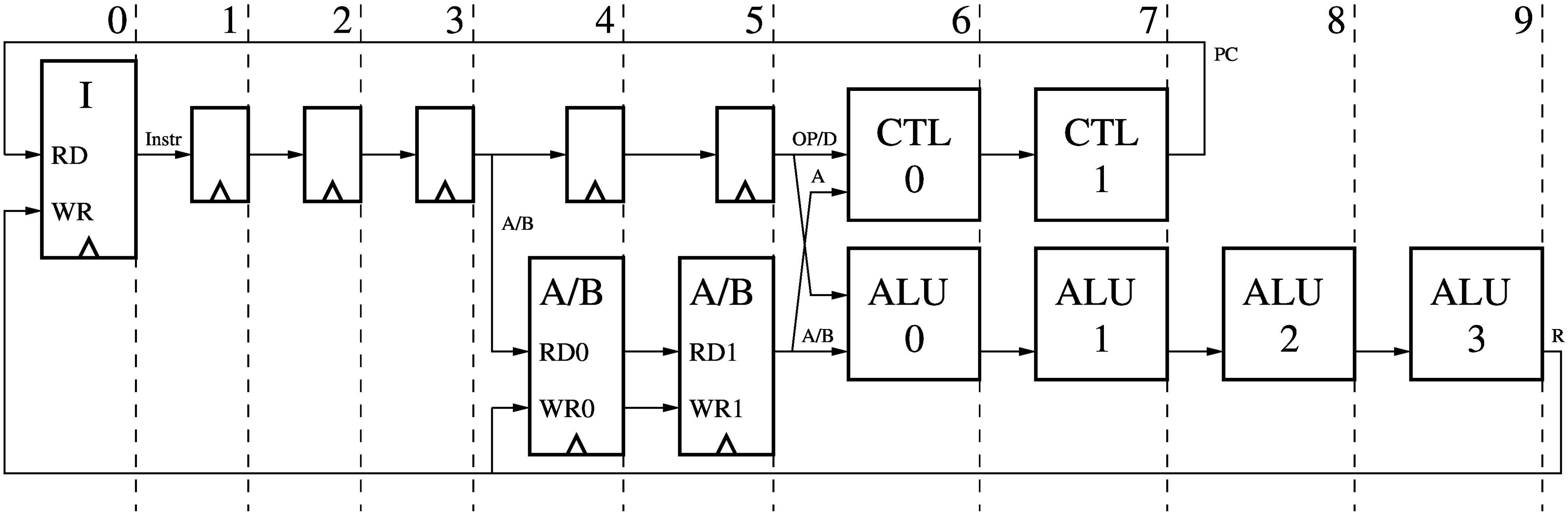
- CPU vs. custom hardware
  - Sequential excution vs. Spatial parallelism
  - Address/Loop calculations vs. Counters
  - Branching vs. Multiplexers
    - FSMs

Reducing CPU Overhead

- CPU pipelining and multi-threading
  - Raw speed increase, but no effect on overhead
- Loop unrolling
  - Code bloat
  - Regular code/data
- Code vectorizing
  - Challenging
  - Regular code/data

A Partial Solution: Octavo

“Octavo: An FPGA-Centric Processor Family”, FPGA 2012



- Exceeds 500 MHz on Stratix IV (550 MHz max!)
- 8 threads (fixed round-robin dispatch)
- Easily extensible with hardware accelerators

Enabling Overhead-Free Execution

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

Enabling Overhead-Free Execution

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

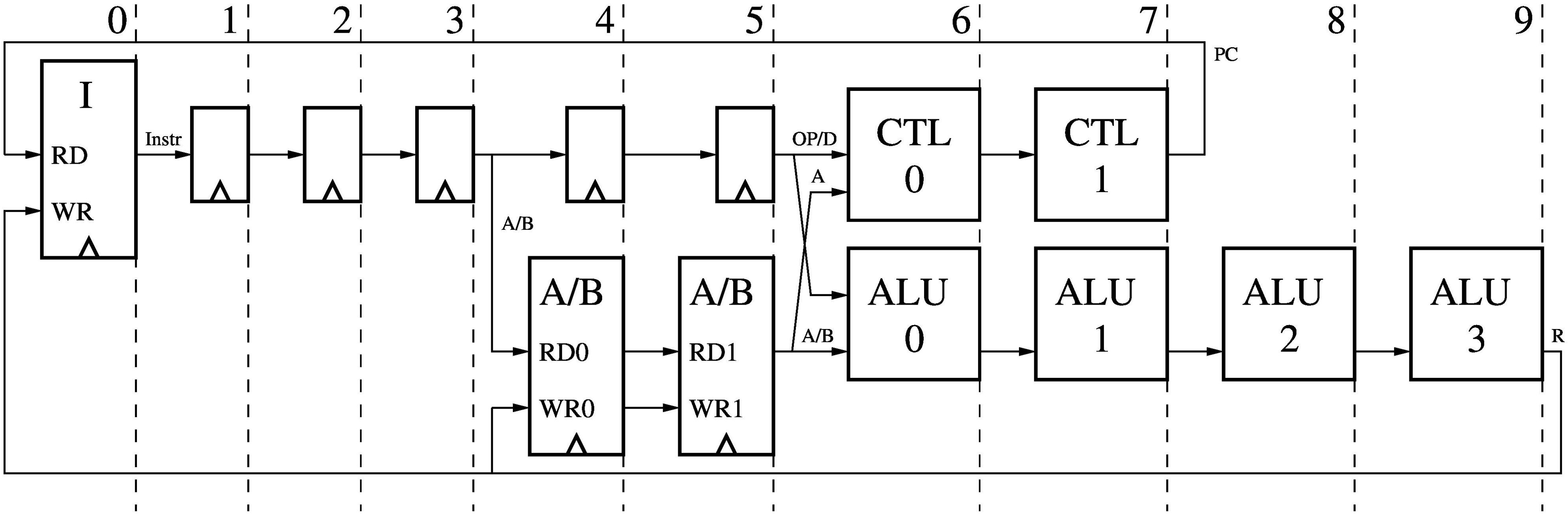
- Flow-control (folded, cancelling, multi-way)
- Addressing (indirect with post-increment)
- Useful work

Parallel Sub-Programs in Octavo

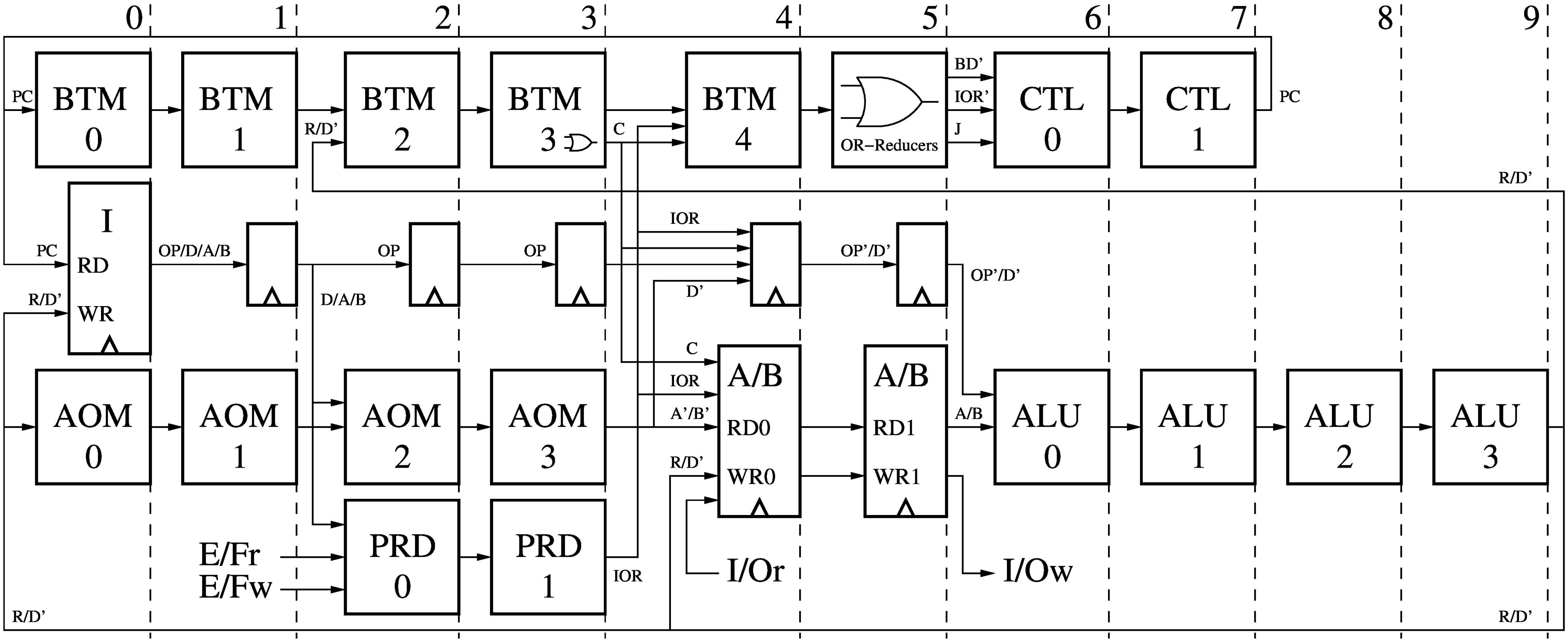
```
outer:  ADD seed_ptr, ptr_init, 0
inner:  LW  temp, seed_ptr
        MUL temp, temp, 3 ; BEVNn even ; BLTZn outer
        ADD temp, temp, 1 ; JMP output
even:   SRA temp, temp, 1
output: SW  temp, seed_ptr
        SW  temp, OUTPUT  ; JMP inner
```

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

Original Octavo Soft-Processor

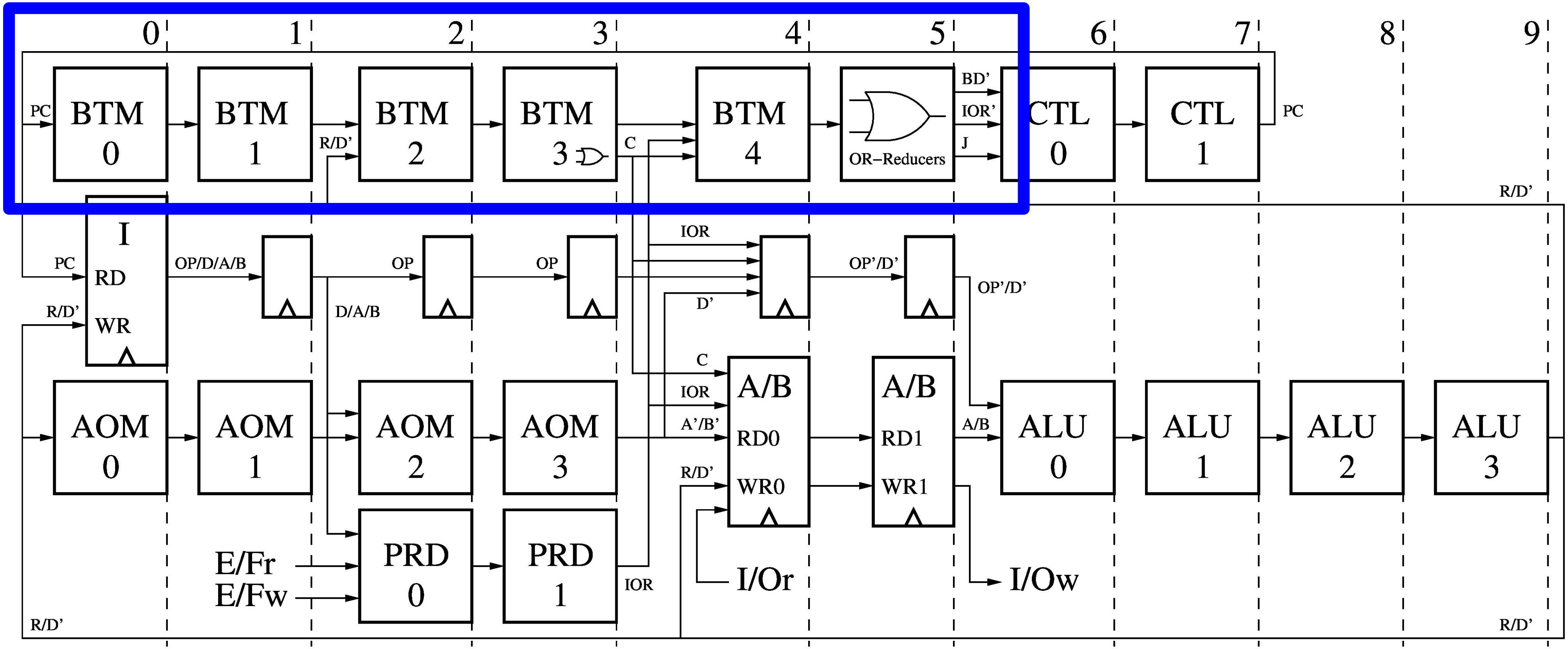


Reduced-Overhead Octavo



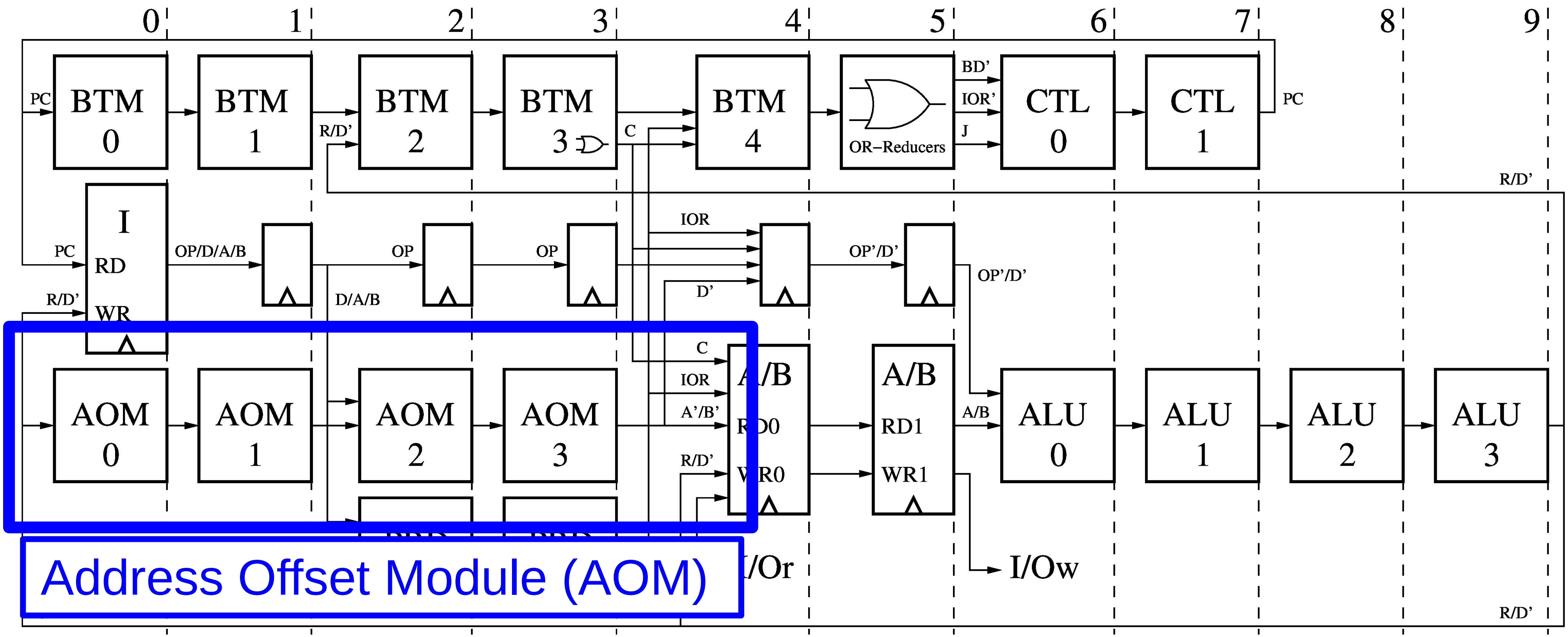
Reduced-Overhead Octavo

Branch Trigger Module (BTM)



(Branches not in fetched instructions!)

Reduced-Overhead Octavo



Address Offset Module (AOM)

(One entry for each instruction operand)

AOM and BTM Entries

- Each AOM entry: one pointer
- Each BTM entry: one branch

AOM and BTM Entries

- Each AOM entry: one pointer
- Each BTM entry: one branch

- Currently: up to 4 pointers and 8 branches
  - Per thread! (32 pointers and 64 branches total)
  - While still reaching 500 MHz peak on Stratix IV

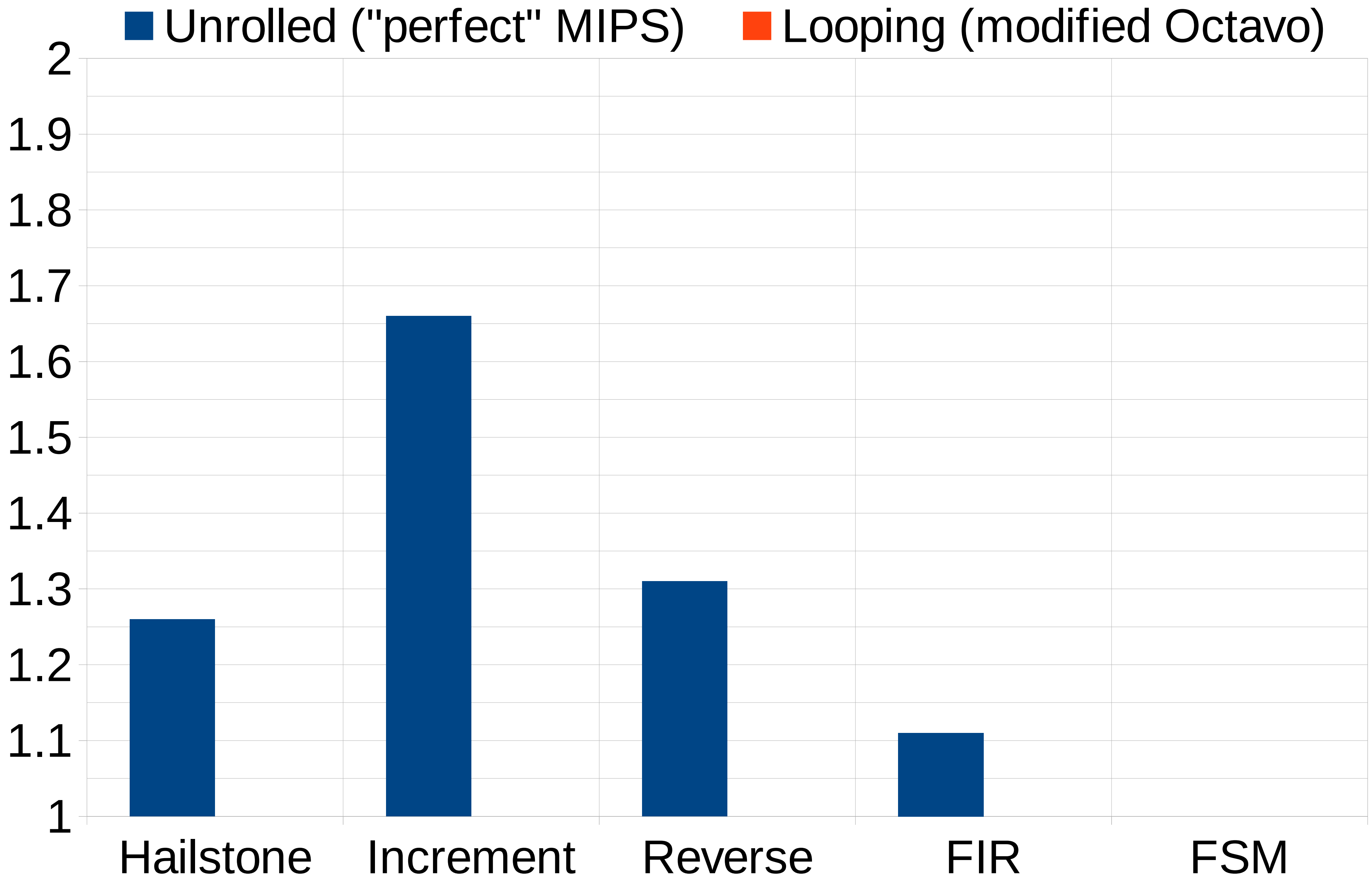
AOM and BTM Entries

- Each AOM entry: one pointer
- Each BTM entry: one branch

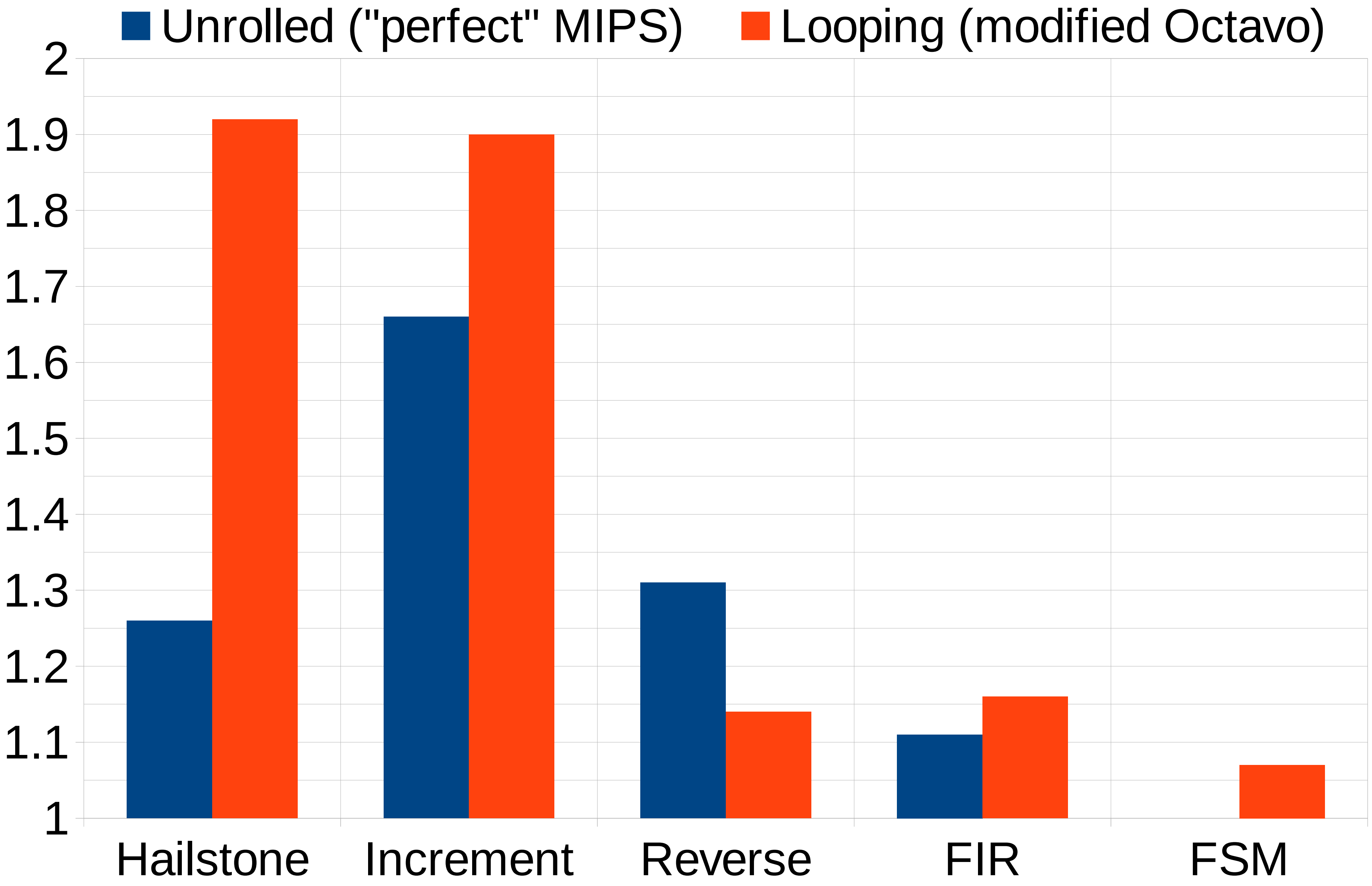
- Currently: up to 4 pointers and 8 branches
  - Per thread! (32 pointers and 64 branches total)
  - While still reaching 500 MHz peak on Stratix IV

- Benchmarking: 2 pointers and 4 branches
  - Reaches 495 MHz avg., 510 MHz peak
  - Shows behaviour with partial AOM/BTM support

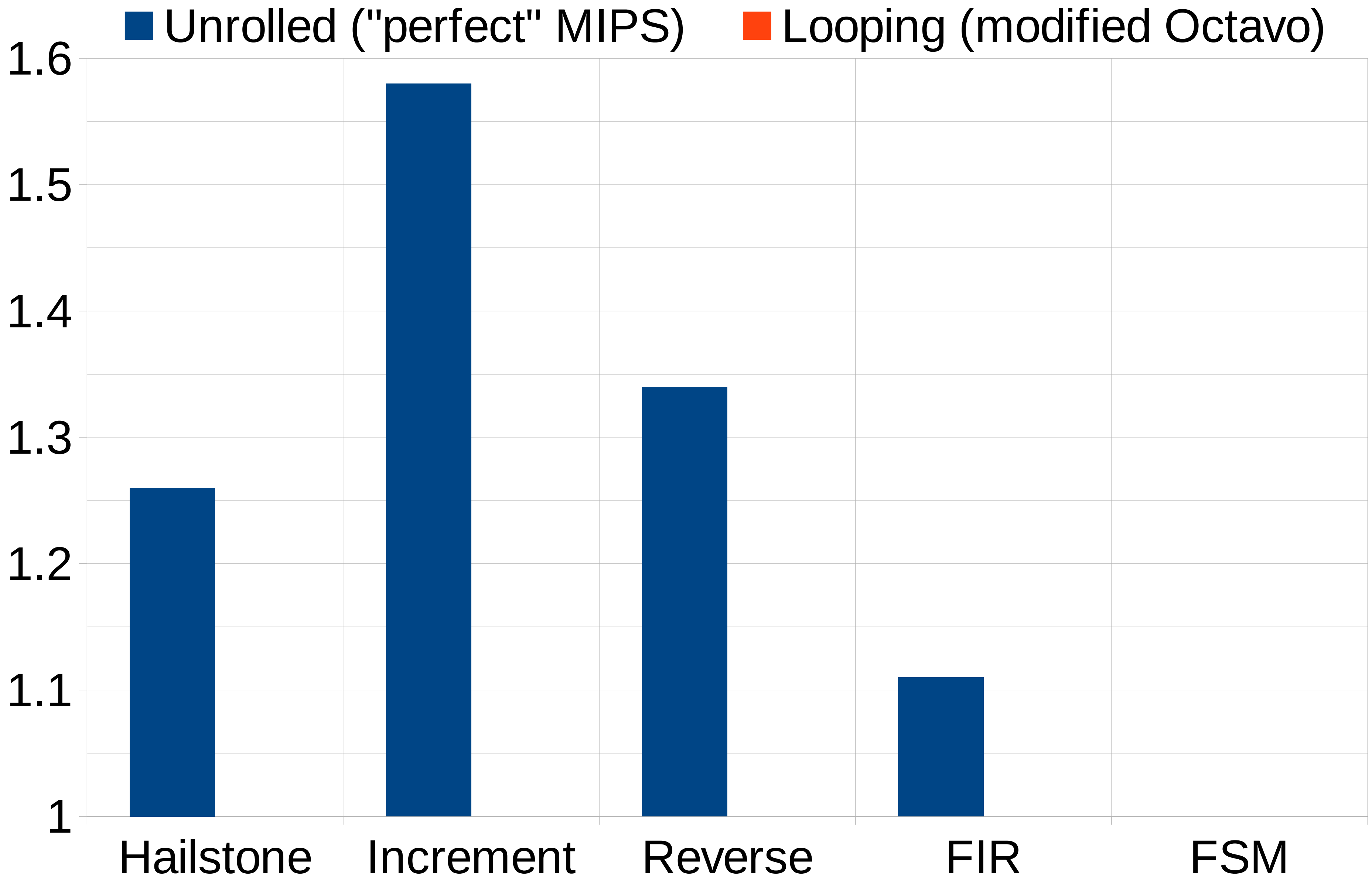
Benchmark Speedup



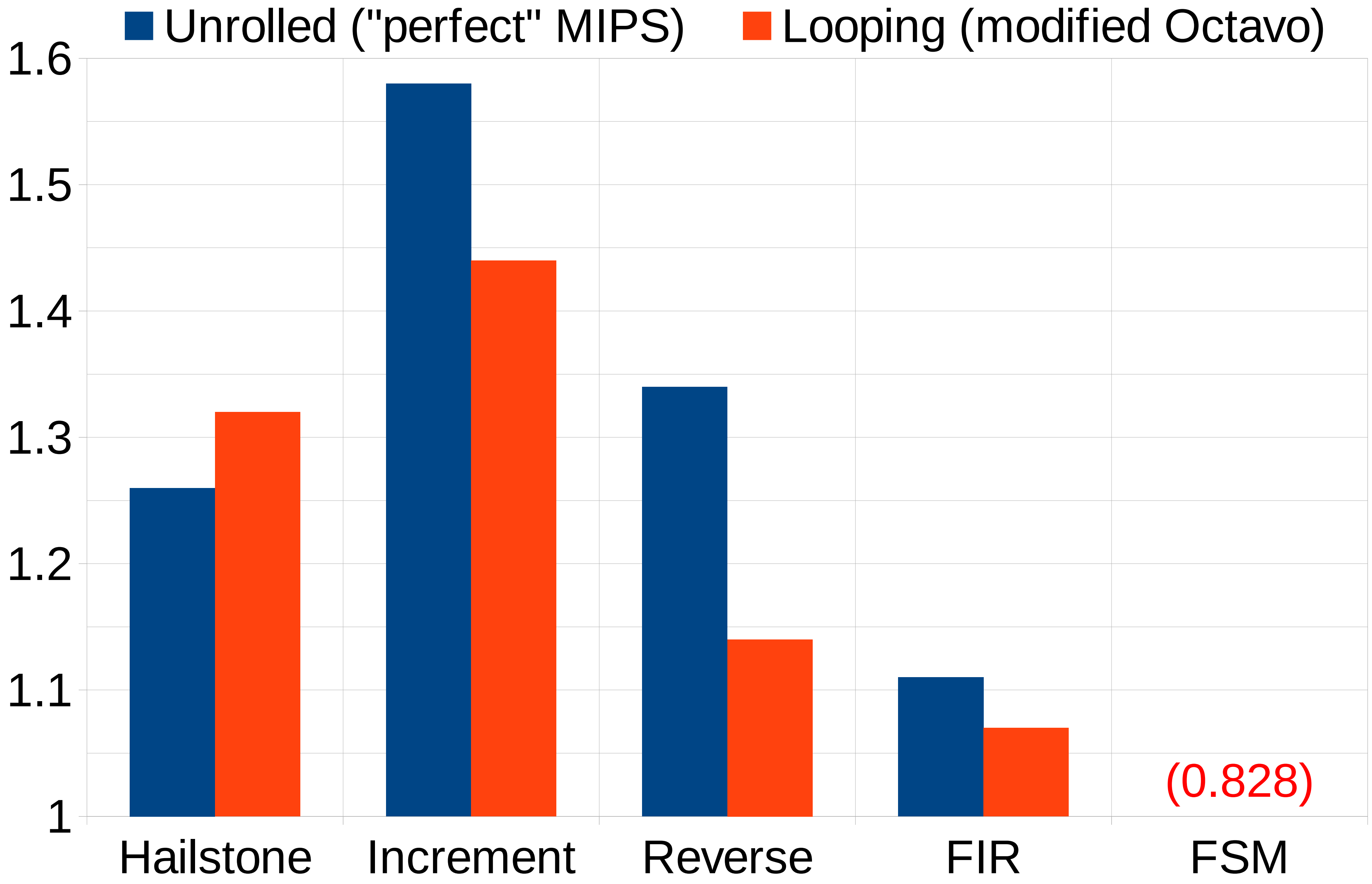
Benchmark Speedup



Benchmark Efficiency Increase



Benchmark Efficiency Increase



Future Improvements

- BTM: additional branch conditions
  - Programmable loop counters

Future Improvements

- BTM: additional branch conditions
  - Programmable loop counters

- AOM: extend pointer increments
  - Negative steps
  - Strided and modulo addressing

Future Improvements

- BTM: additional branch conditions
  - Programmable loop counters
- AOM: extend pointer increments
  - Negative steps
  - Strided and modulo addressing
- Both: improve area usage
  - More efficient use of internal memories

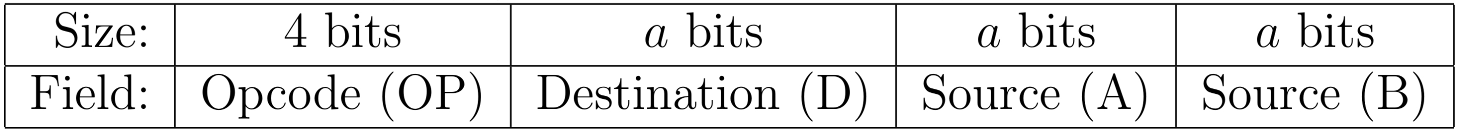
Ongoing Work

https://github.com/laforest/Octavo

Clip art by Angela Melick, http://www.wastedtalent.ca/

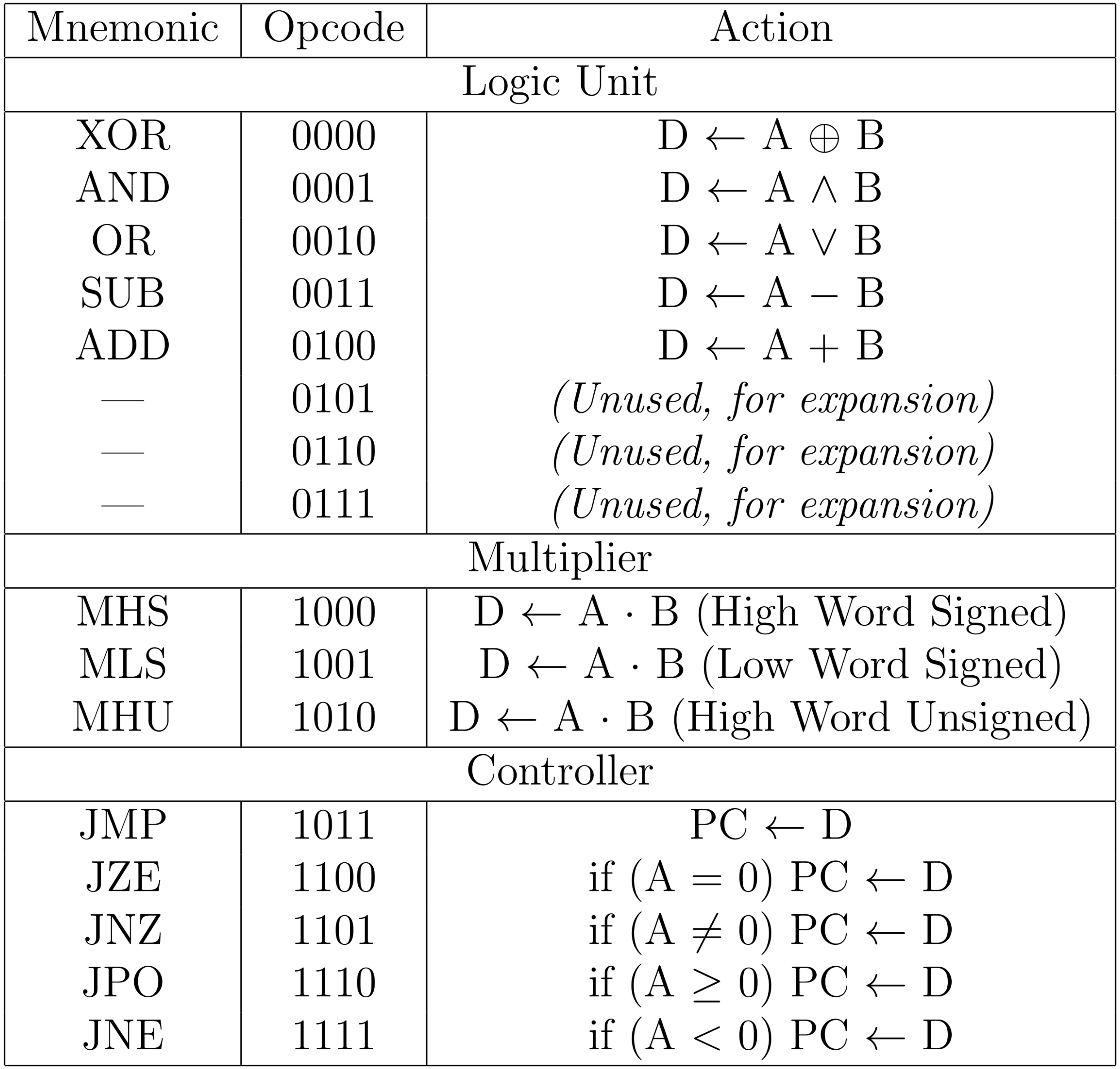
Extra Slides

Table 3.1: Octavo's Instruction Word Format.



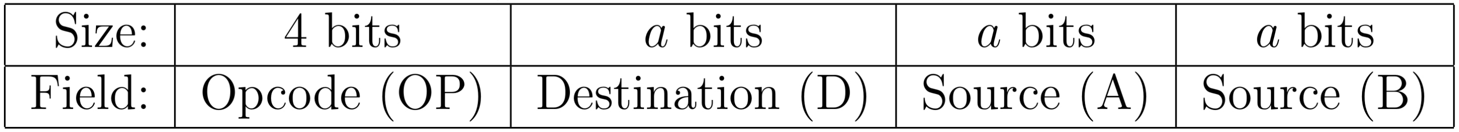
Size:	4 bits	a bits	a bits	a bits
Field:	Opcode (OP)	Destination (D)	Source (A)	Source (B)

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



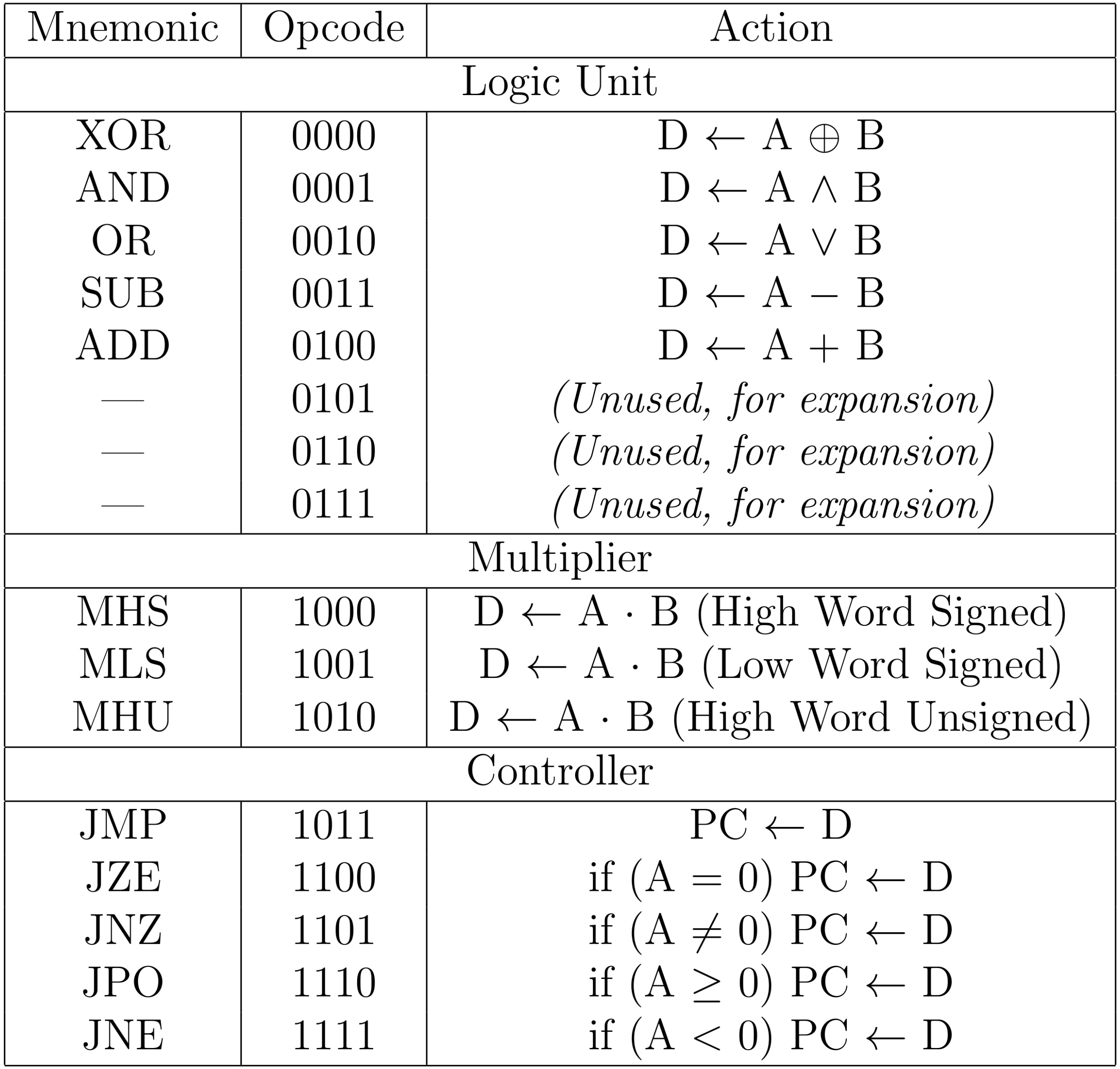
Mnemonic	Opcode	Action
	Logic Unit	
XOR	0000	$D \leftarrow A \oplus B$
AND	0001	$D \leftarrow A \wedge B$
OR	0010	$D \leftarrow A \vee B$
SUB	0011	$D \leftarrow A - B$
ADD	0100	$D \leftarrow A + B$
—	0101	*(Unused, for expansion)*
—	0110	*(Unused, for expansion)*
—	0111	*(Unused, for expansion)*
	Multiplier	
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.



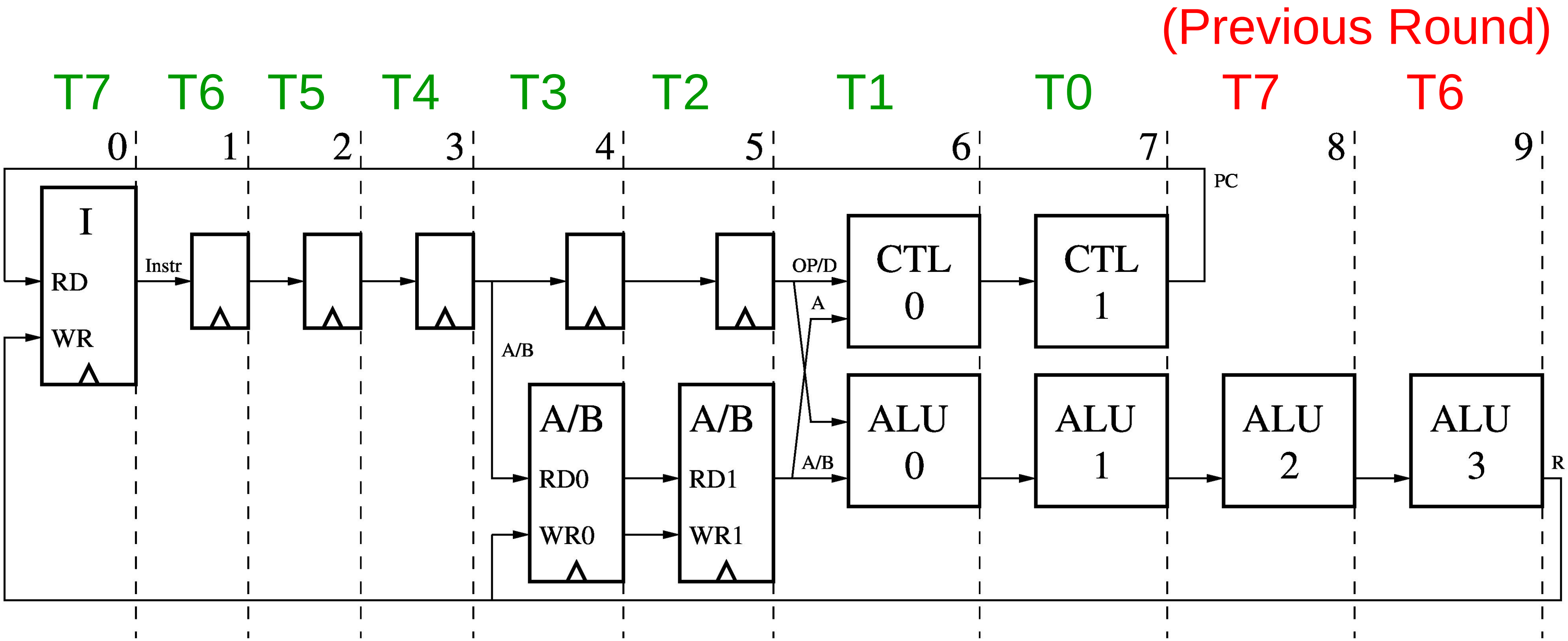
Size:	4 bits	a bits	a bits	a bits
Field:	Opcode (OP)	Destination (D)	Source (A)	Source (B)

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



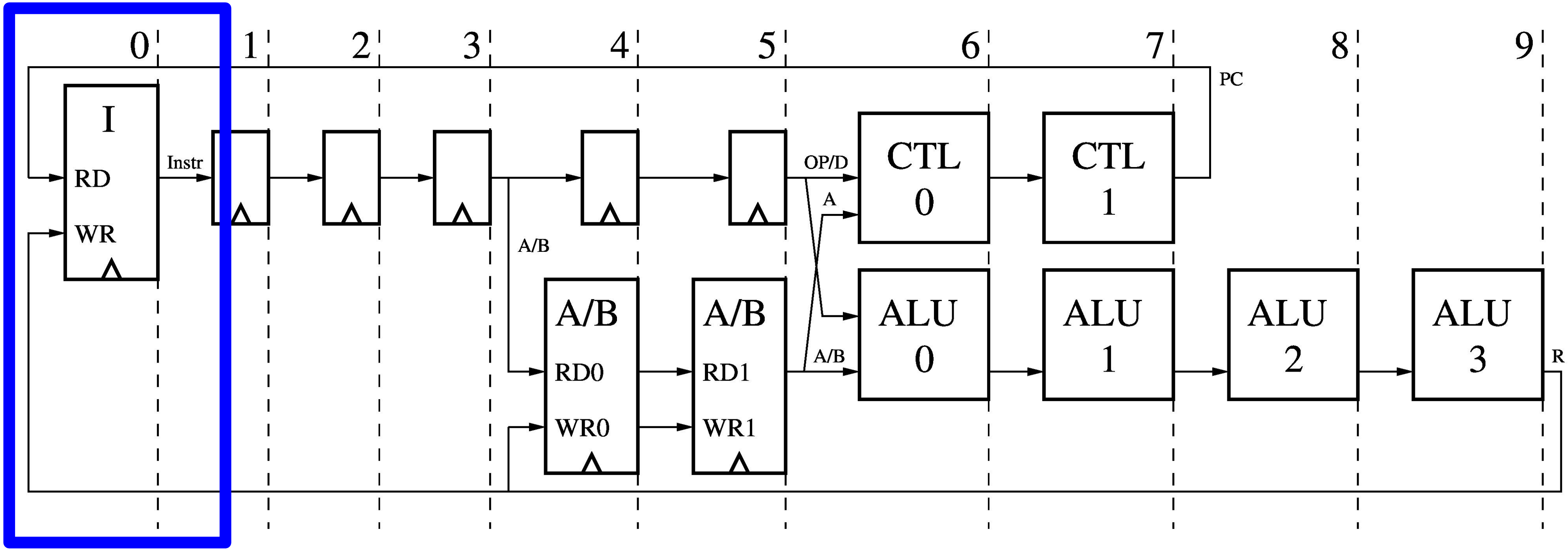
Mnemonic	Opcode	Action
Logic Unit		
XOR	0000	D ← A ⊕ B
AND	0001	D ← A ∧ B
OR	0010	D ← A ∨ B
SUB	0011	D ← A − B
ADD	0100	D ← A + B
—	0101	*(Unused, for expansion)*
—	0110	*(Unused, for expansion)*
—	0111	*(Unused, for expansion)*
Multiplier		
MHS	1000	D ← A · B (High Word Signed)
MLS	1001	D ← A · B (Low Word Signed)
MHU	1010	D ← A · B (High Word Unsigned)
Controller		
JMP	1011	PC ← D
JZE	1100	if (A = 0) PC ← D
JNZ	1101	if (A ≠ 0) PC ← D
JPO	1110	if (A ≥ 0) PC ← D
JNE	1111	if (A < 0) PC ← D

Octavo Soft-Processor

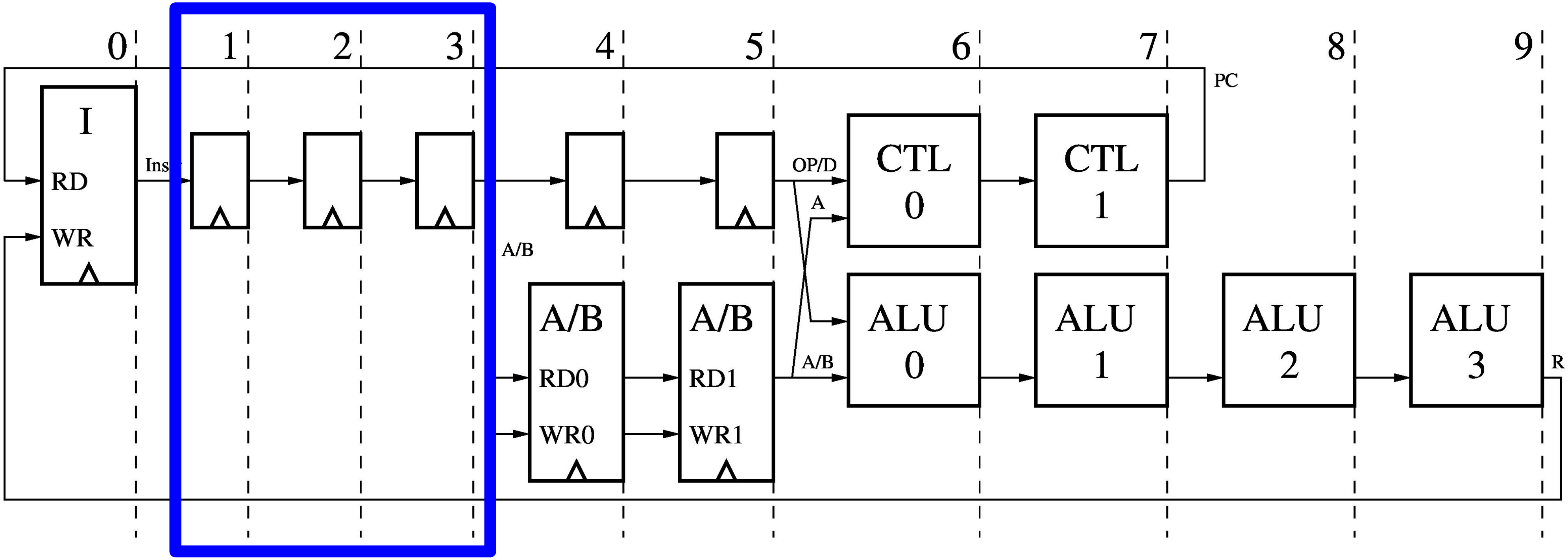


- Reaches 550 MHz on Stratix IV FPGA
- 8 threads (fixed round-robin)
- 1024 36-bit integer words for each I/A/B memory

Instruction Memory

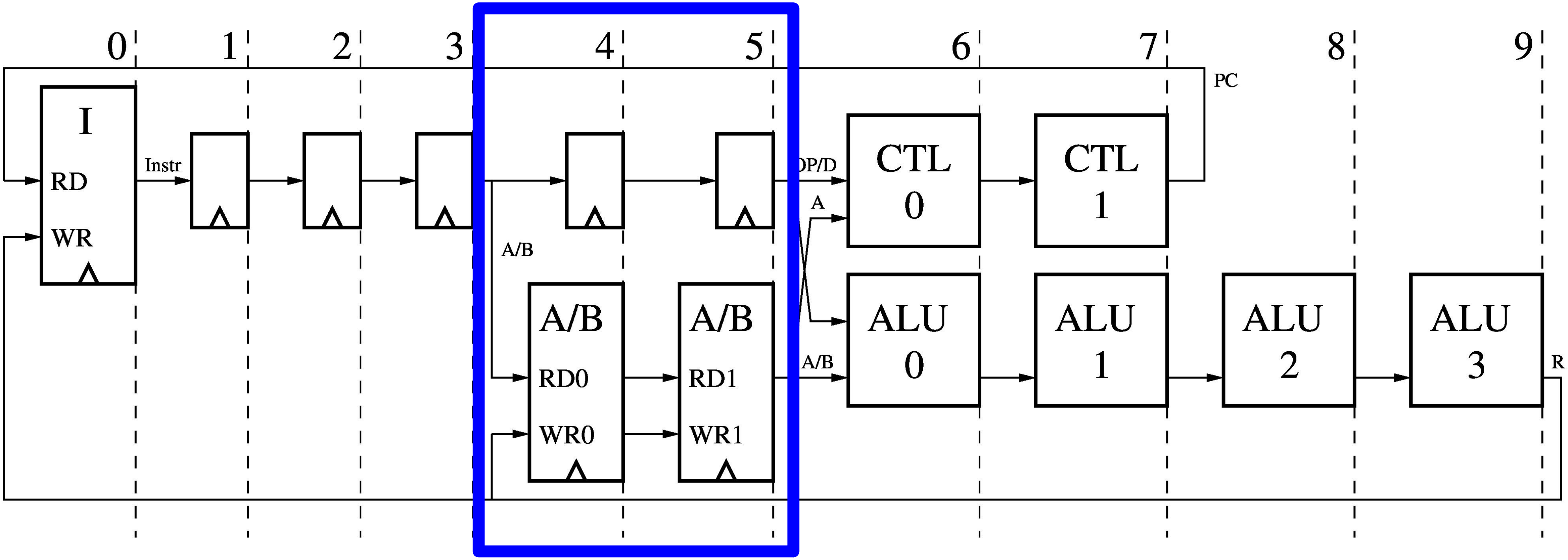


Empty Pipeline Stages



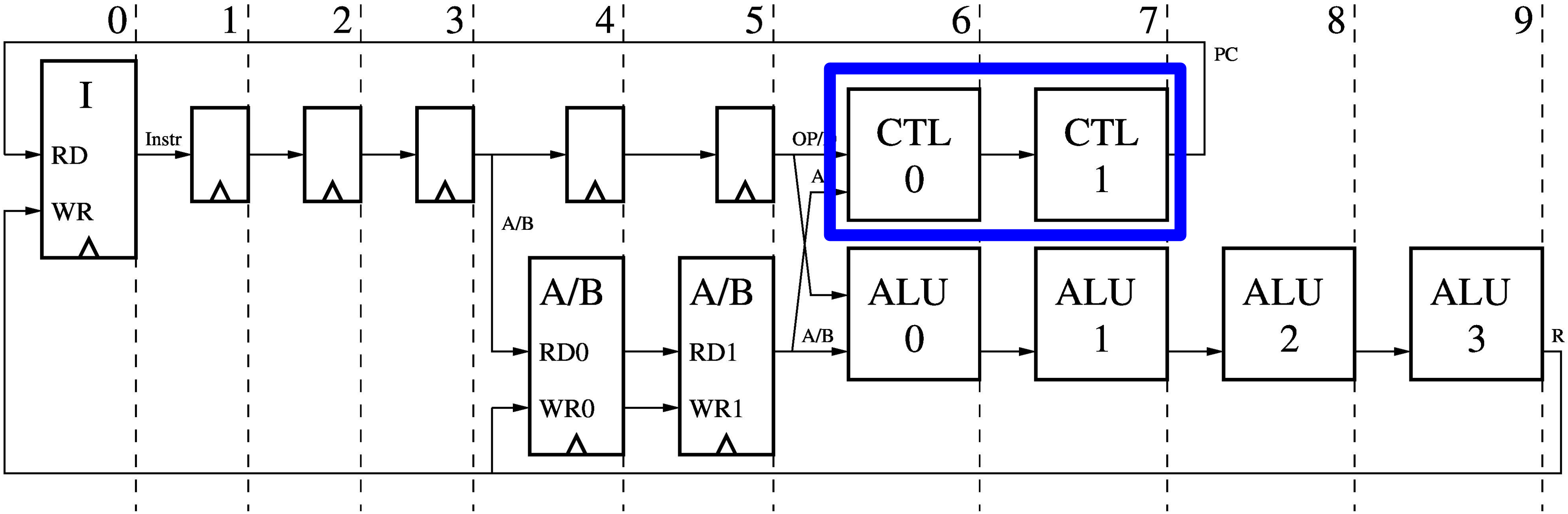
- Necessary for high frequency operation
- Used for special functions later...

A and B Data Memories



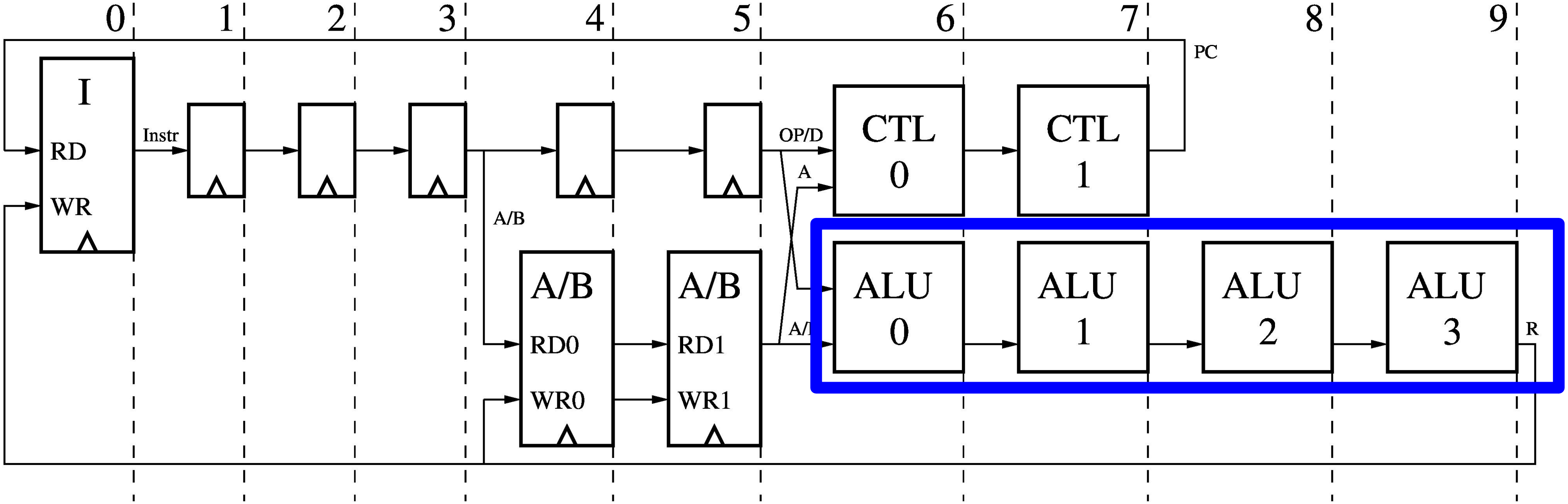
- Memory-mapped I/O ports
- Can attach custom hardware to ports

Controller



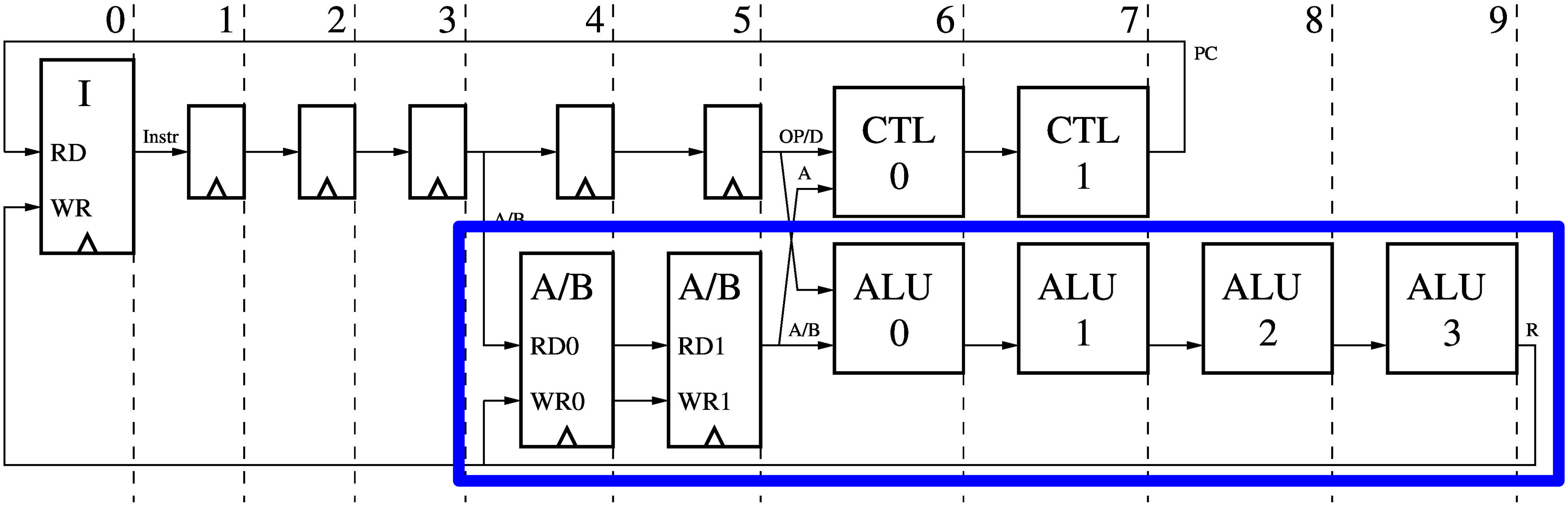
- Computes next PC for each thread (8 Pcs)
- Calculates jumps and branches

ALU



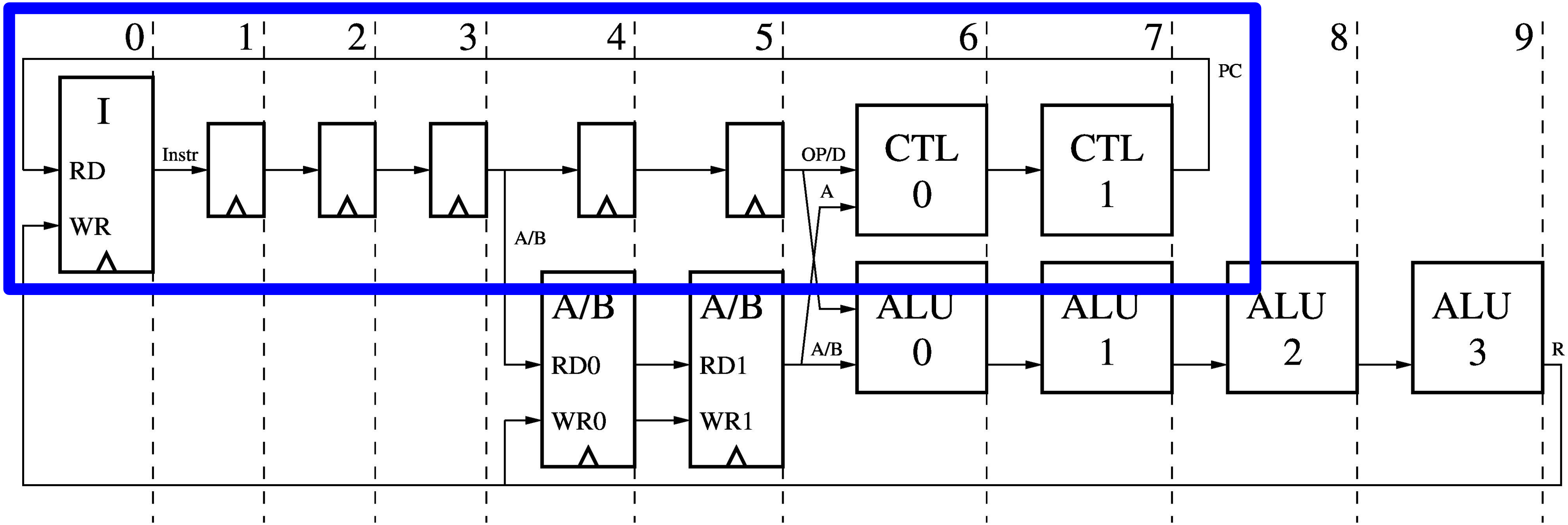
- Calculates ADD, XOR, MUL, etc...
- Output written to all memories

Data Path



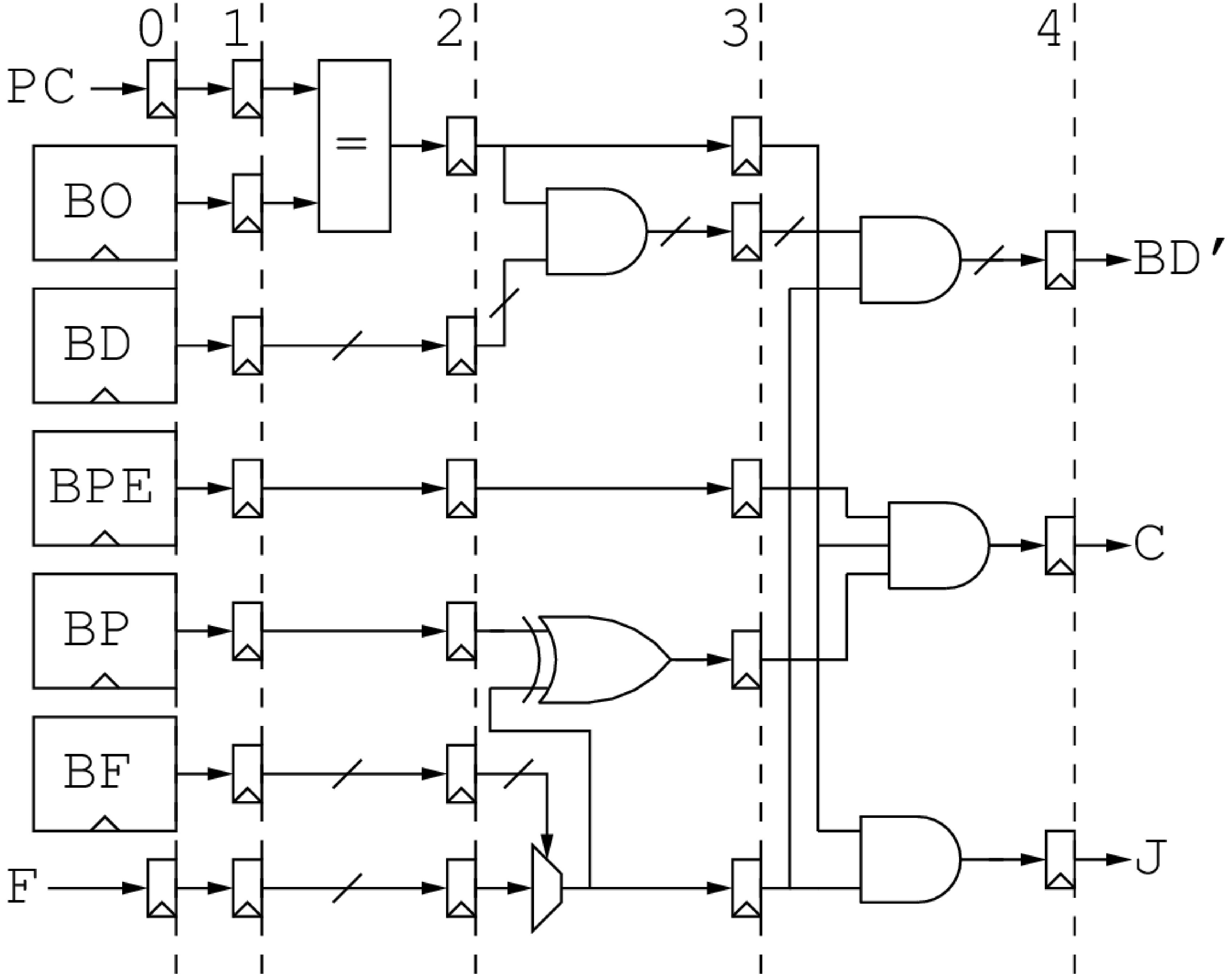
- 8 stages (2 read, 4 compute, 2 write)

Control Path

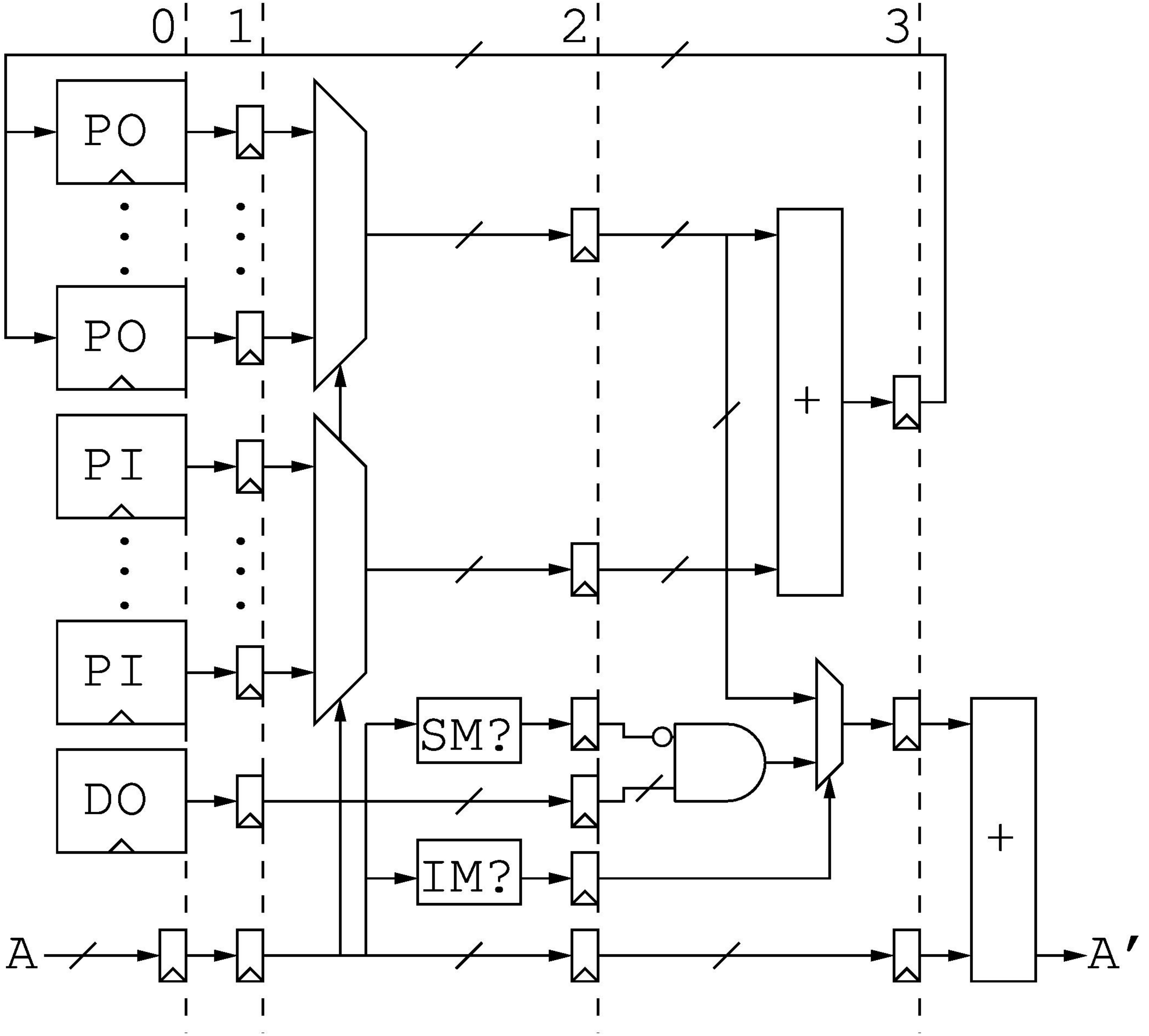


- 8 stages to match Data Path
- Offset due to empty stages (1,2,3)
- 1-cycle RAW hazard from ALU to Instr. Mem.

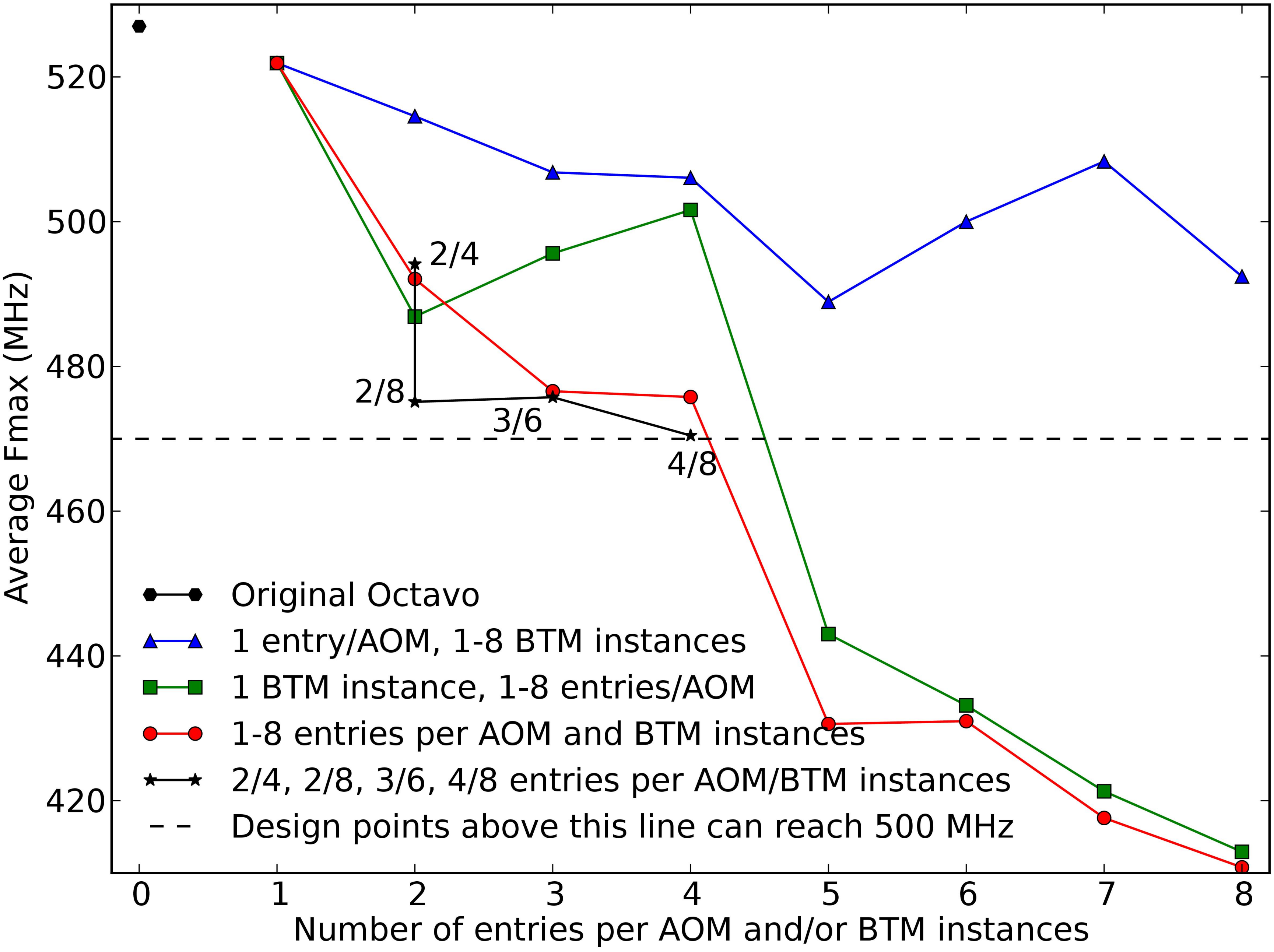
Branch Trigger Module



Address Offset Module



AOM/BTM Configurations


Average Fmax (MHz)
Number of entries per AOM and/or BTM instances
2/4
2/8
3/6
4/8
Original Octavo
1 entry/AOM, 1-8 BTM instances
1 BTM instance, 1-8 entries/AOM
1-8 entries per AOM and BTM instances
2/4, 2/8, 3/6, 4/8 entries per AOM/BTM instances
Design points above this line can reach 500 MHz