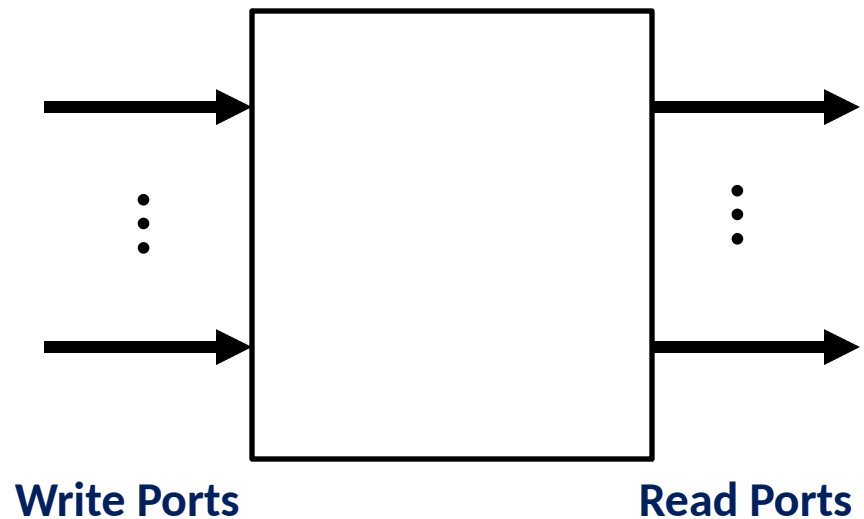# Multi-ported Memories for FPGAs via XOR
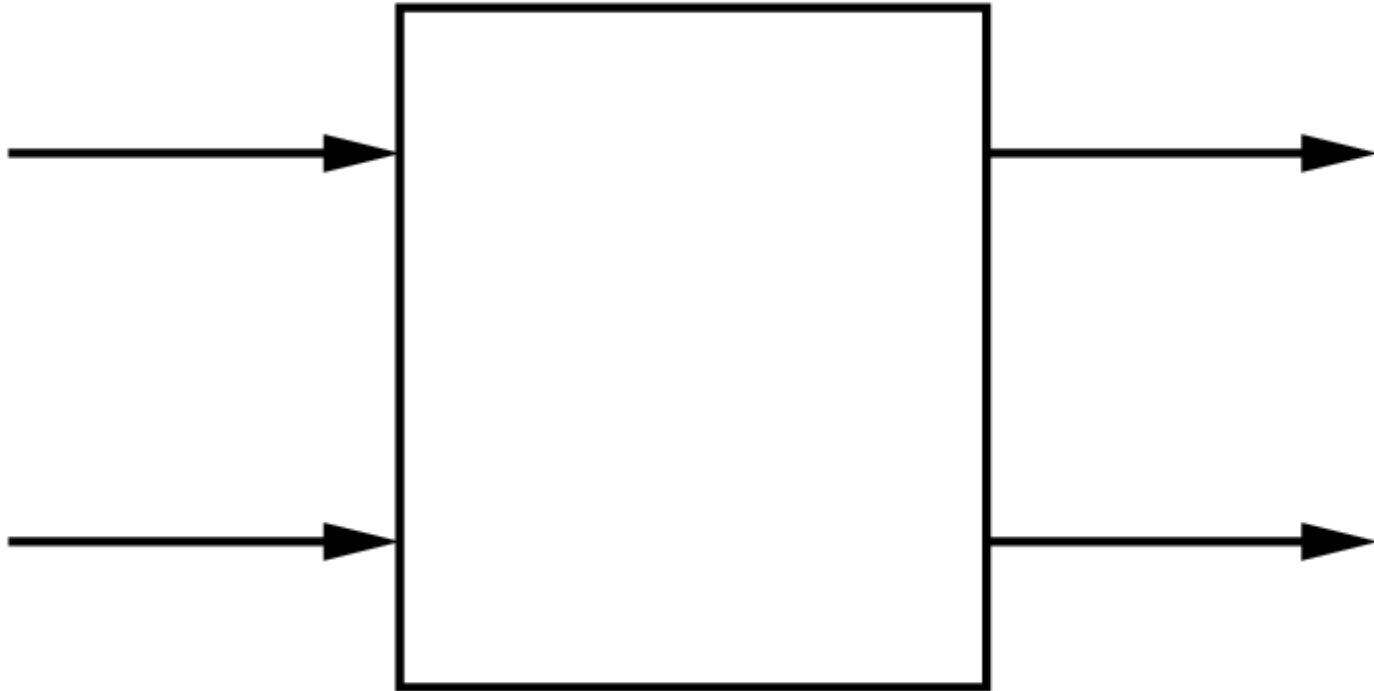
Eric LaForest, Ming Liu, Emma Rapati, and Greg Steffan
ECE, University of Toronto

Multi-Ported Memories (MPM)

- **MPM: Memory with more than 2 ports**

- **Many uses:**
  - register files
  - queues/buffers

- **FPGA BRAMs:**
  - have only 2 ports

- **Building MPMs:**
  - multiple BRAMs
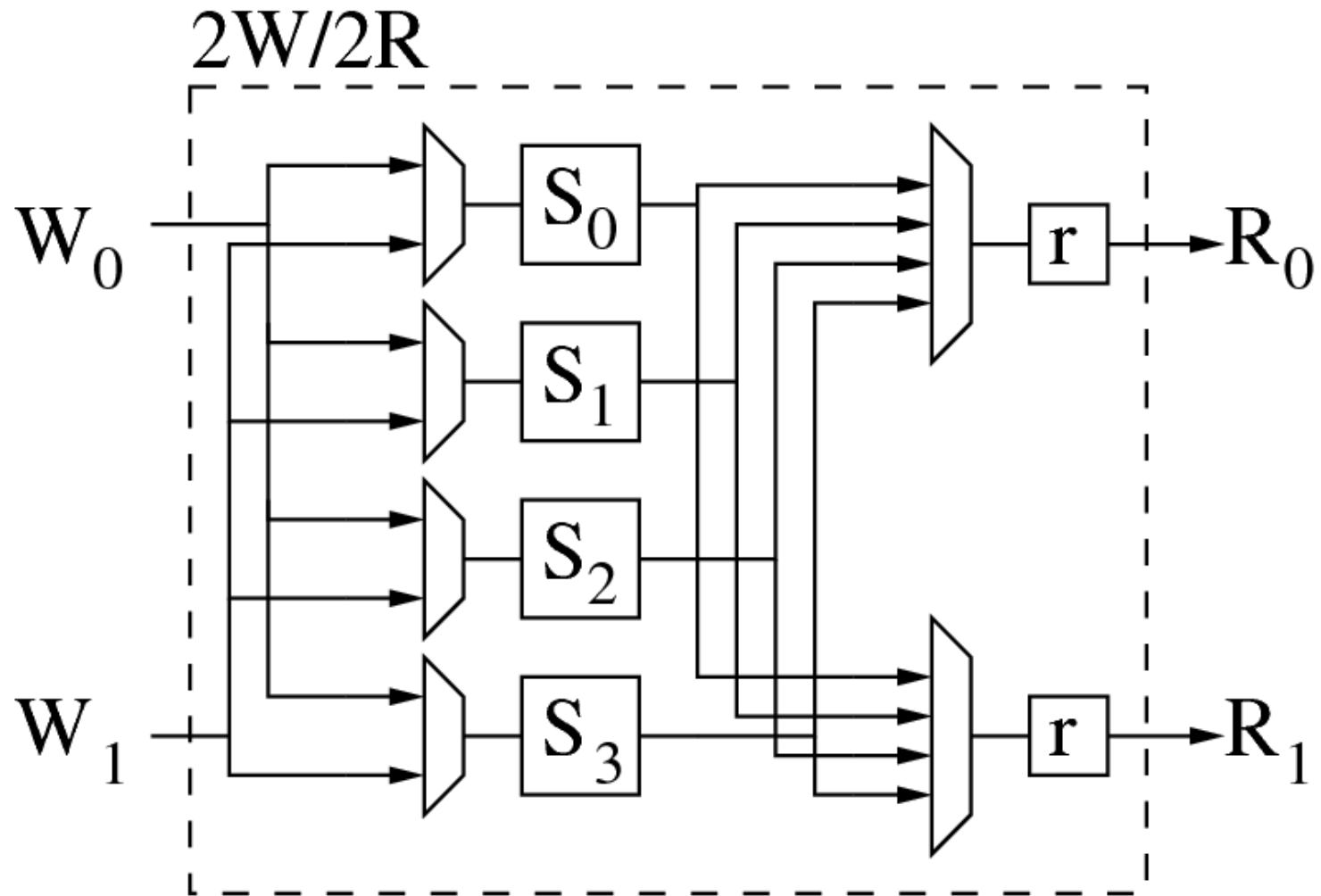 - logic elements (ALMs/LEs)
  - clever combinations

Write Ports
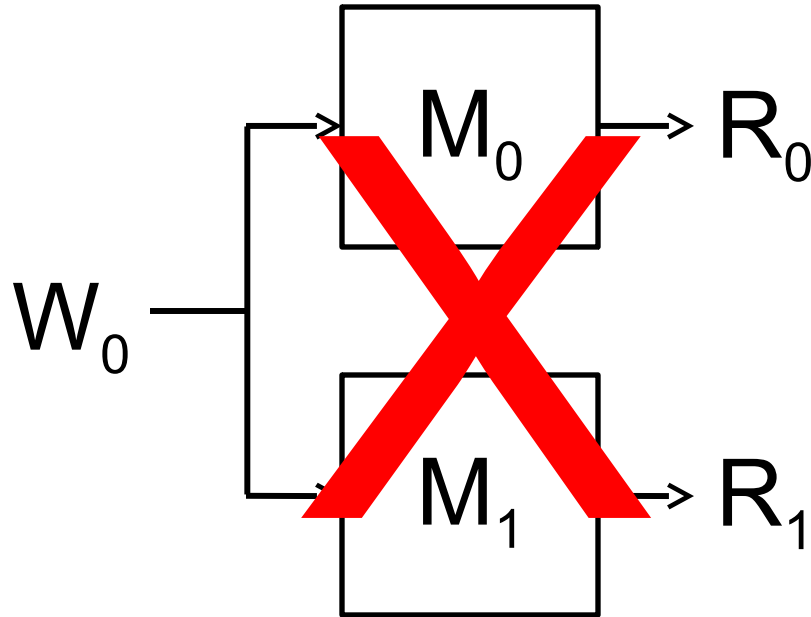
Read Ports

2

Example: 2W/2R MPM

How can we build this?

2W/2R: Pure-ALMs/LEs

Scales very poorly with memory depth ▋▋

1W/2R: Replicated BRAMS

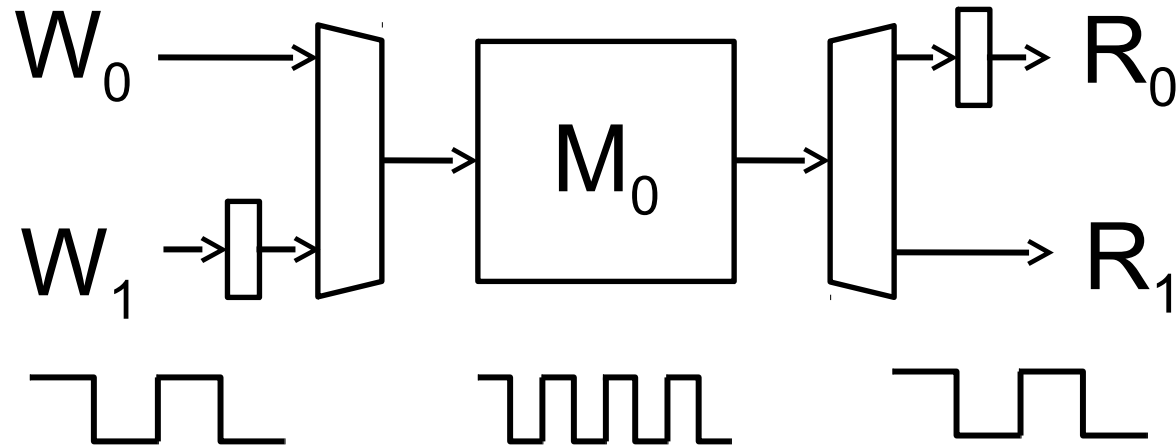**Fairly efficient ⏪**   **Only one write port ⏸**

2W/2R Banked BRAMs
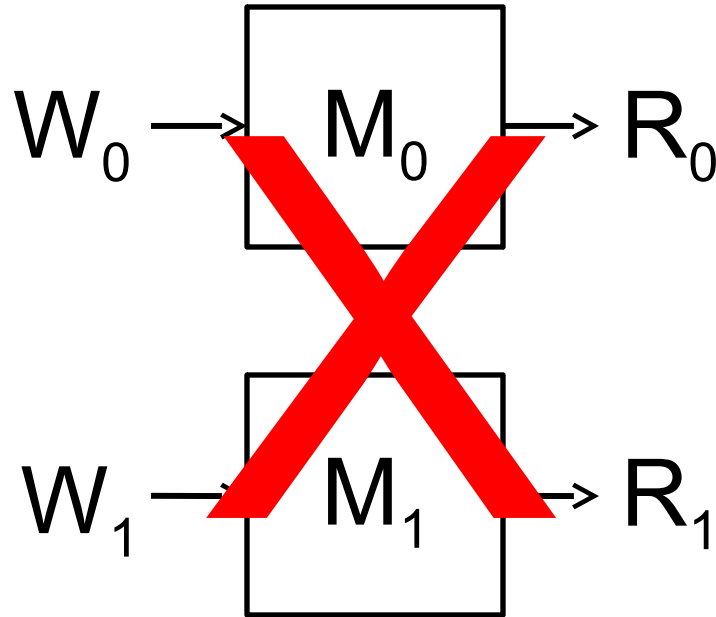
Multiple write ports ◀◀   **Fragmented data** ❚❚
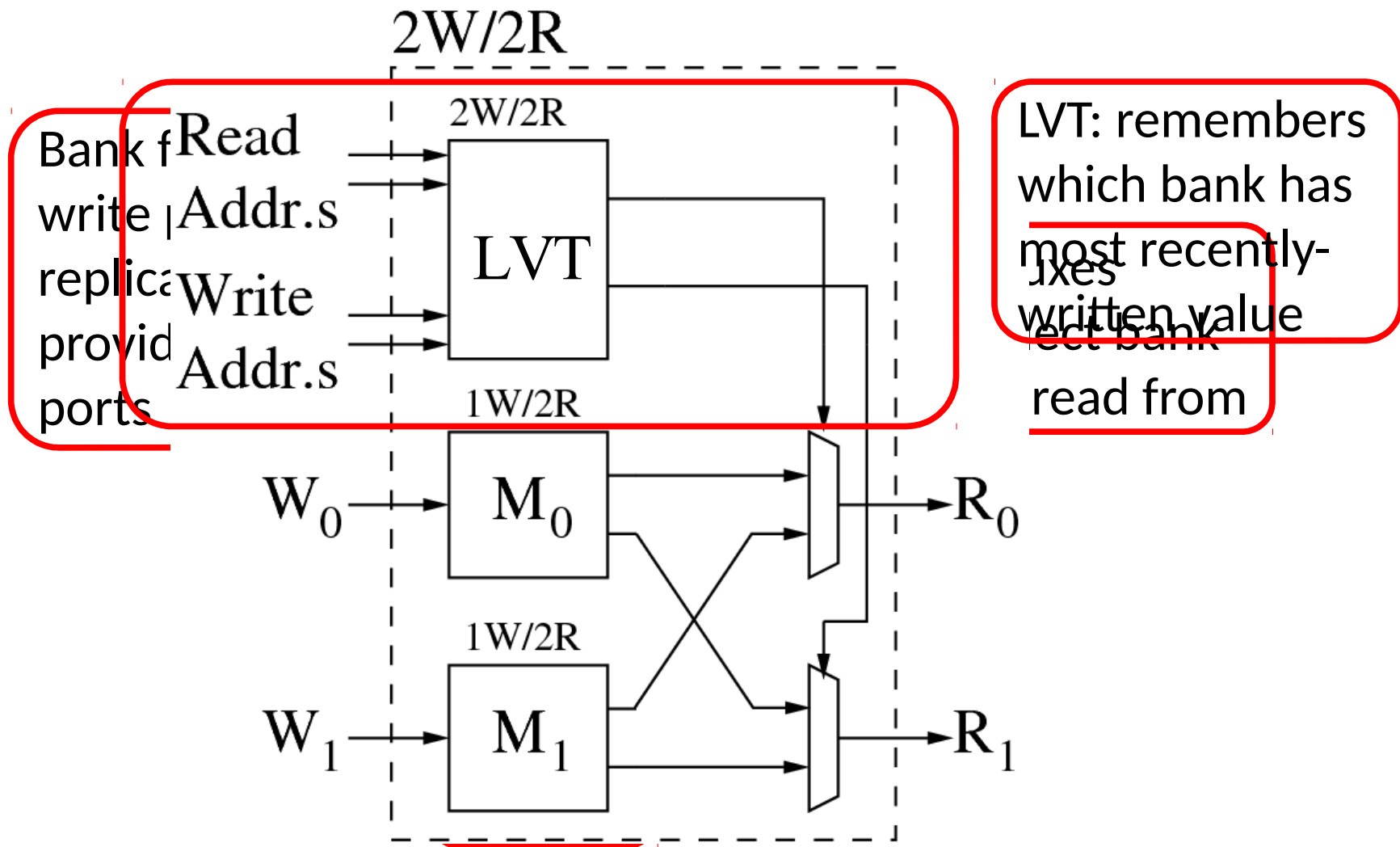
2W/2R "Multipumping"

No fragmentation ◀◀   **Divides clock speed** ▮▮

Review:
The Live Value Table (LVT) Approach (FPGA'10)

Efficient Multi-Ported Memories for FPGAs, Eric LaForest and J. Gregory Steffan, International Symposium on Field-Programmable Gate Arrays, Monterey, CA, February, 2010.

LVT-Based MPM

Bank f[Read] write [Addr.s] replica[Write] provid[Addr.s] ports

2W/2R

2W/2R

LVT

1W/2R

$W_0 \rightarrow M_0$

$\rightarrow R_0$

1W/2R

$W_1 \rightarrow M_1$

$\rightarrow R_1$

LVT: remembers which bank has most recently-written value

uxes lect bank read from

9

LVT-Based MPM

2W/2R **Many ALMs!**

Read
Addr.s

2W/2R

LVT

Write
Addr.s

**Significant
Multiplexing!**

1W/2R

W_0 M_0 R_0

1W/2R

W_1 M_1 R_1
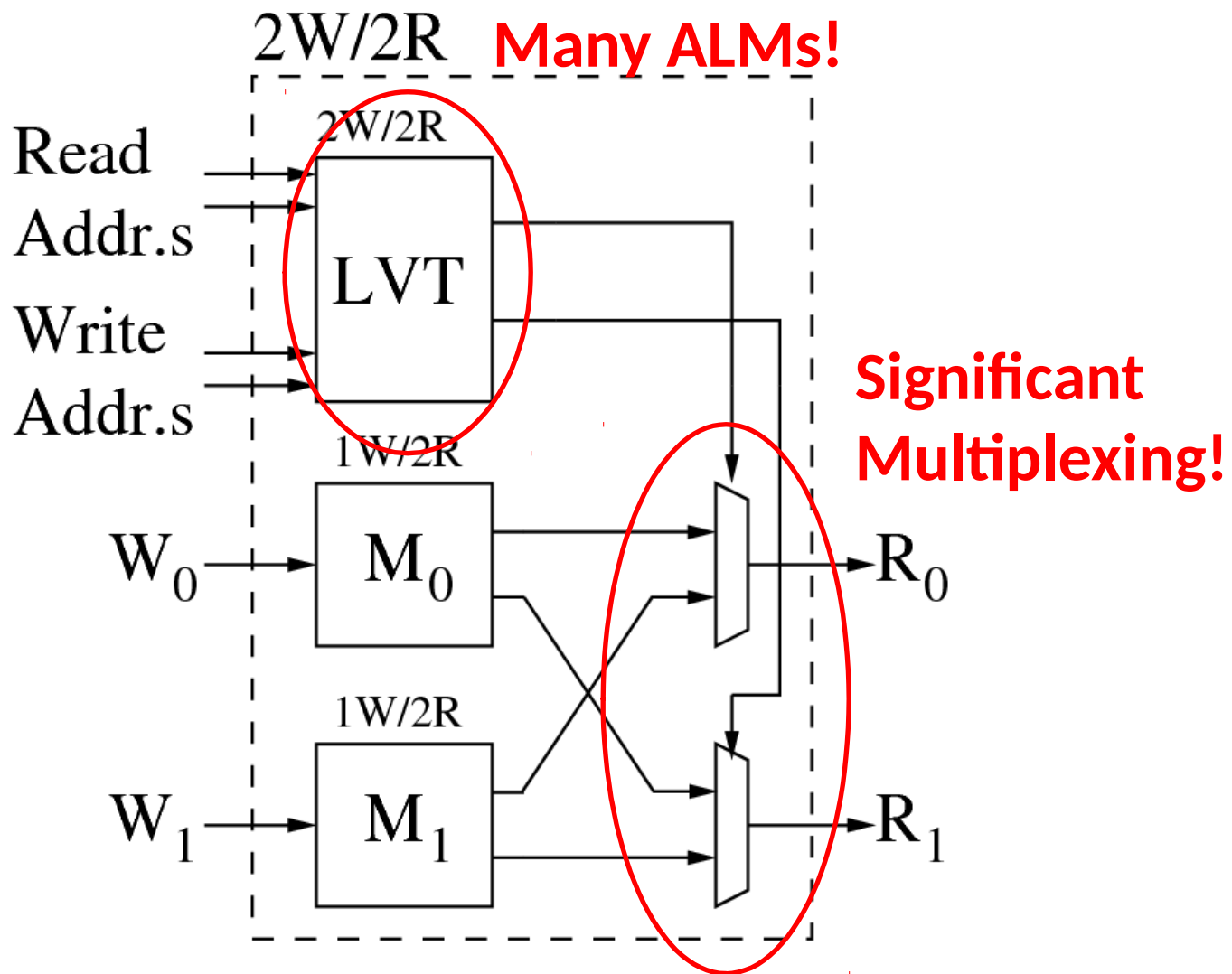
Punchline: LVT is a big freq win, but...

An XOR Approach

XOR

- **XOR basics:**
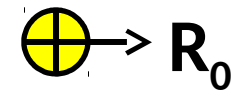
  $A \oplus 0 = A$

  $B \oplus B = 0$

- **Implication:**

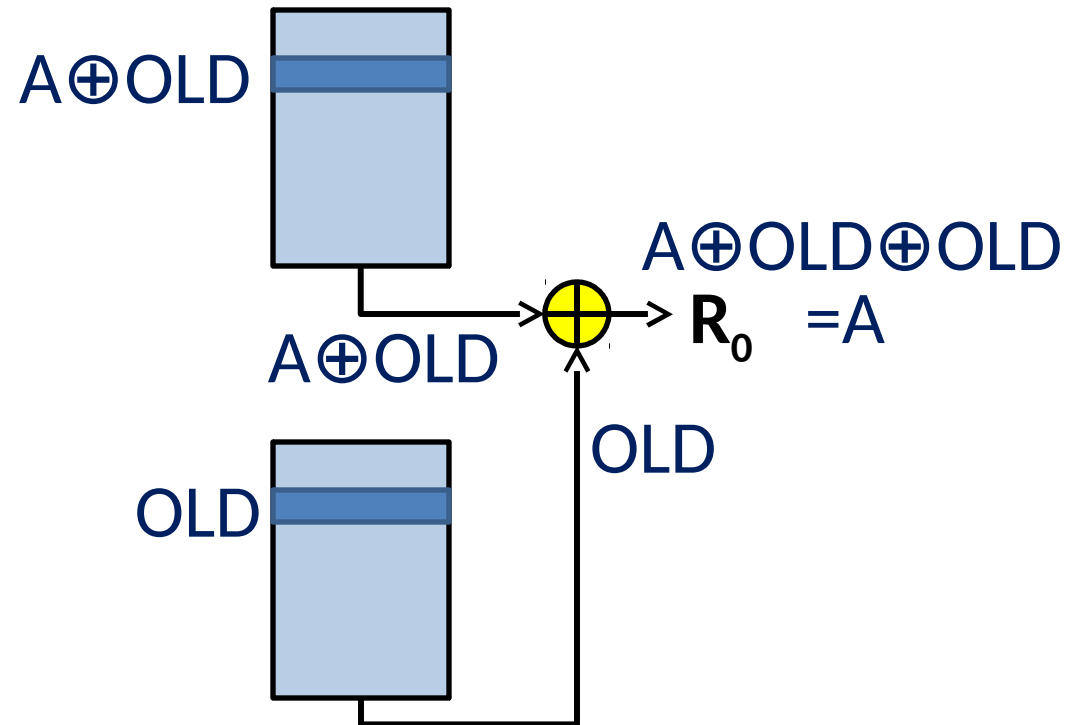 $A \oplus B \oplus B = A$

Can we exploit XOR to build better MPMs?

Intuition: avoid LVT-table, multiplexing

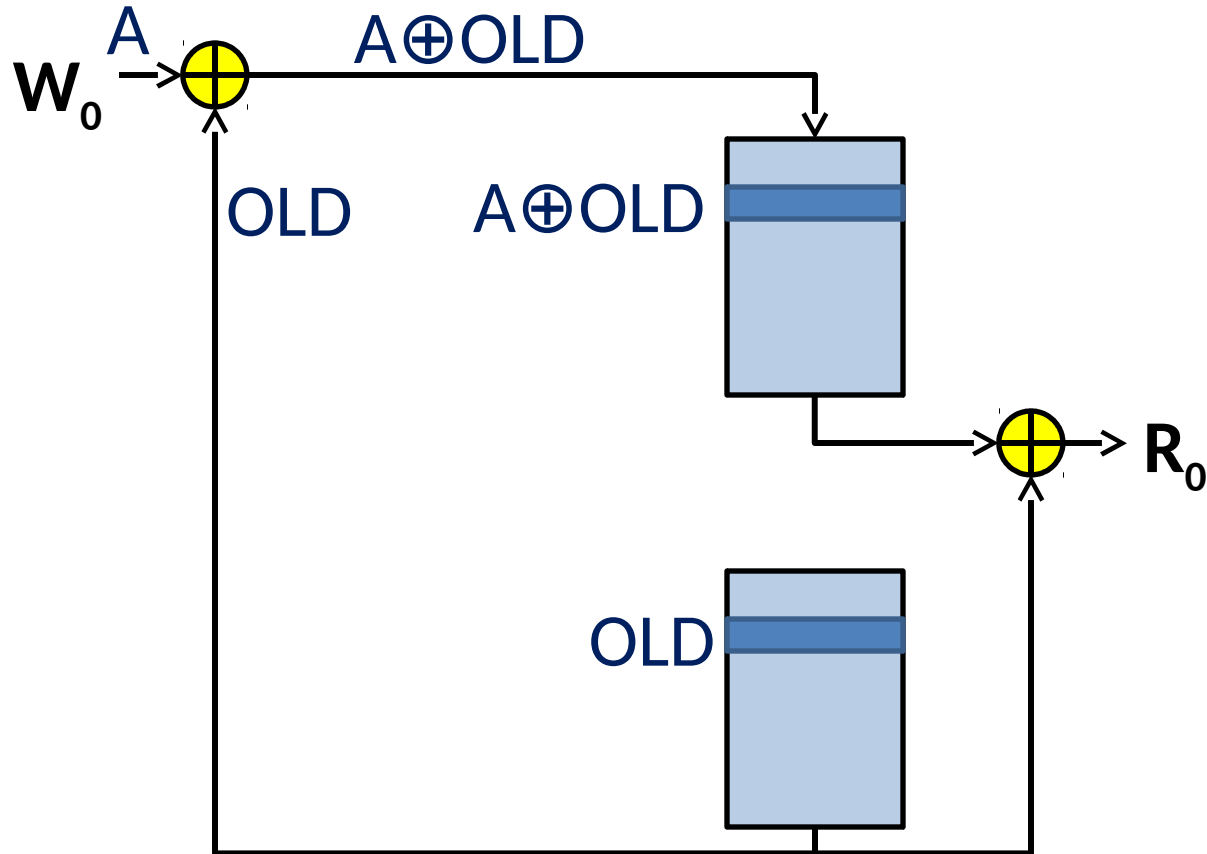# 2W/2R XOR Design

$\oplus \rightarrow R_0$

Goal: a read is only an XOR operation

2W/2R XOR Design

A⊕OLD

A⊕OLD⊕OLD
R_0 =A

A⊕OLD

OLD

OLD

Focus on one location for now

2W/2R XOR Design

XOR new value with old value

2W/2R XOR Design

W_0 → ⊕

R_0
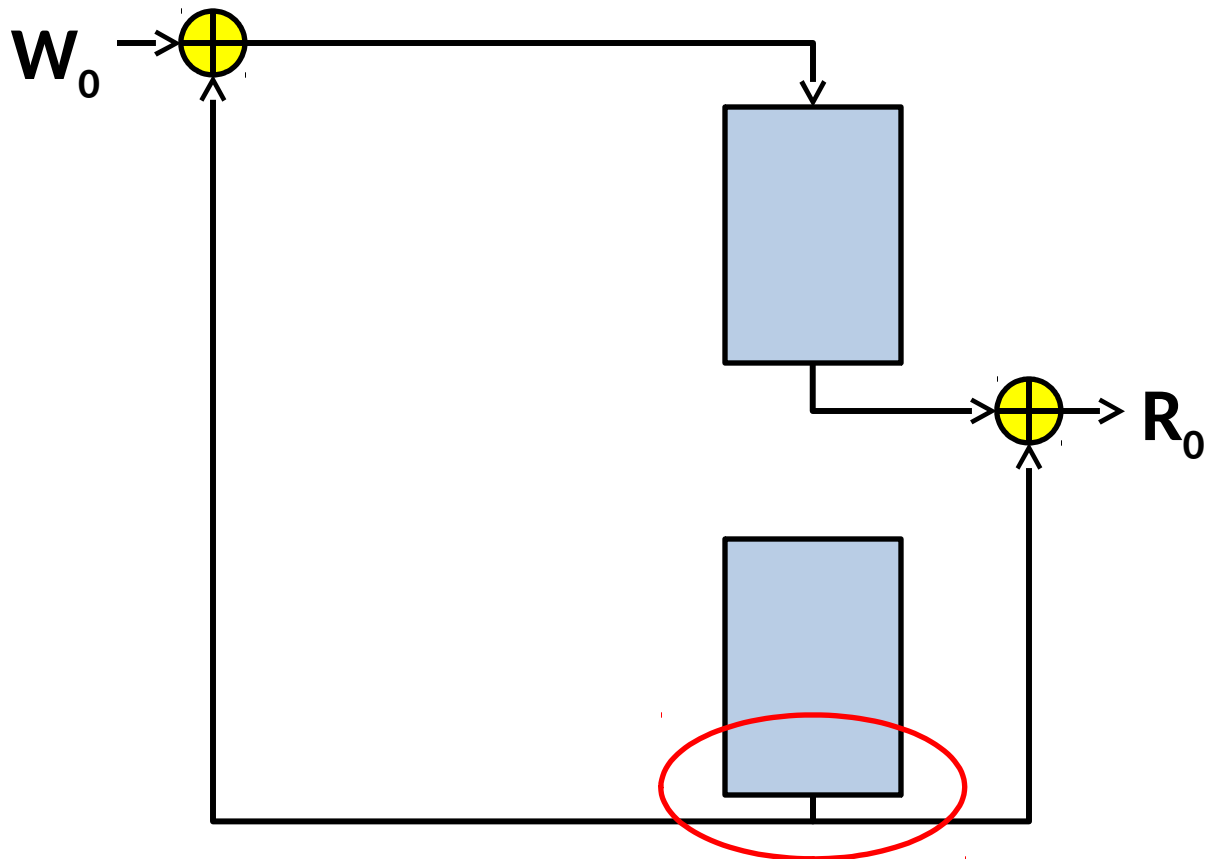
Support multiple locations, two write ports

2W/2R XOR Design
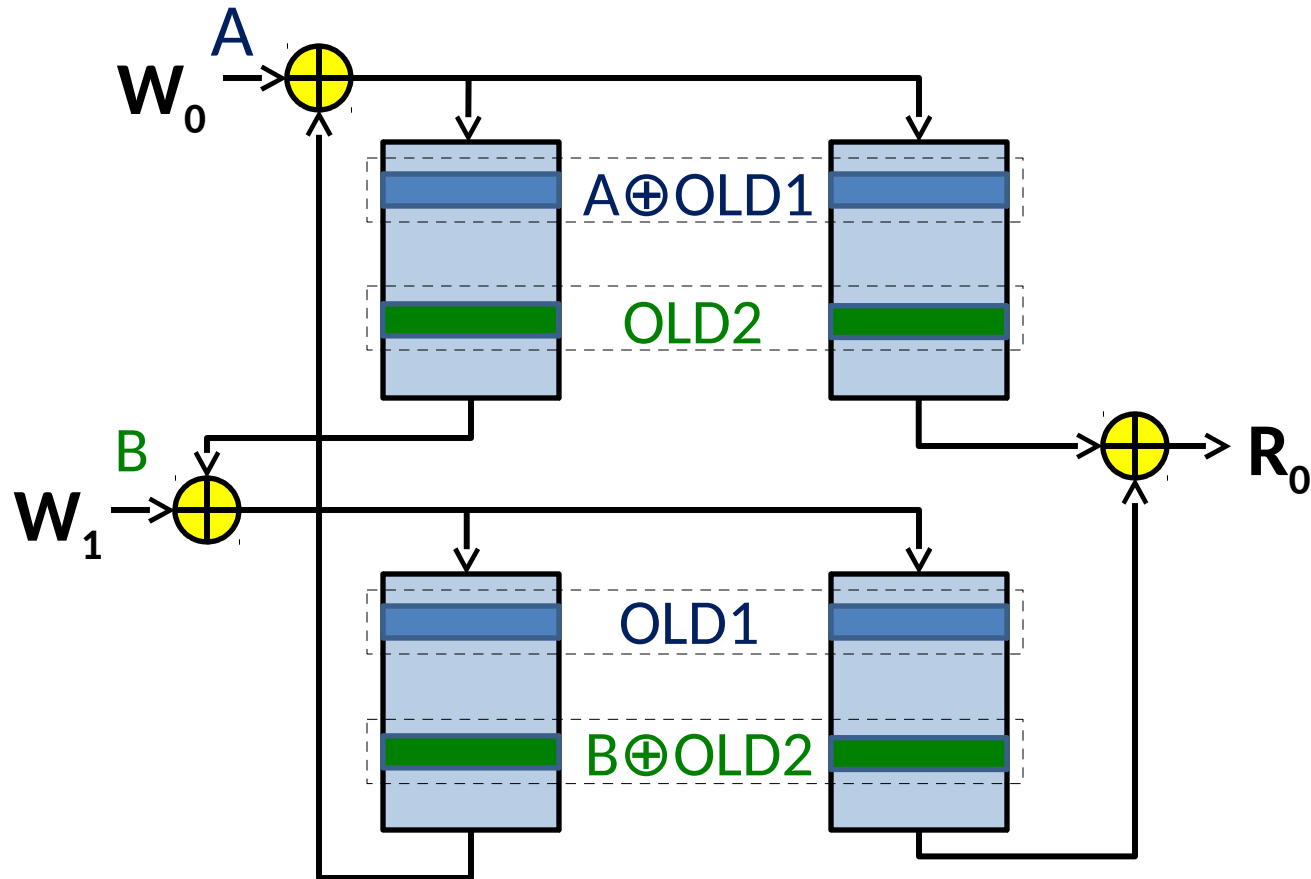
Most-recently-written bank holds new value XOR old(s)

2W/2R XOR Design
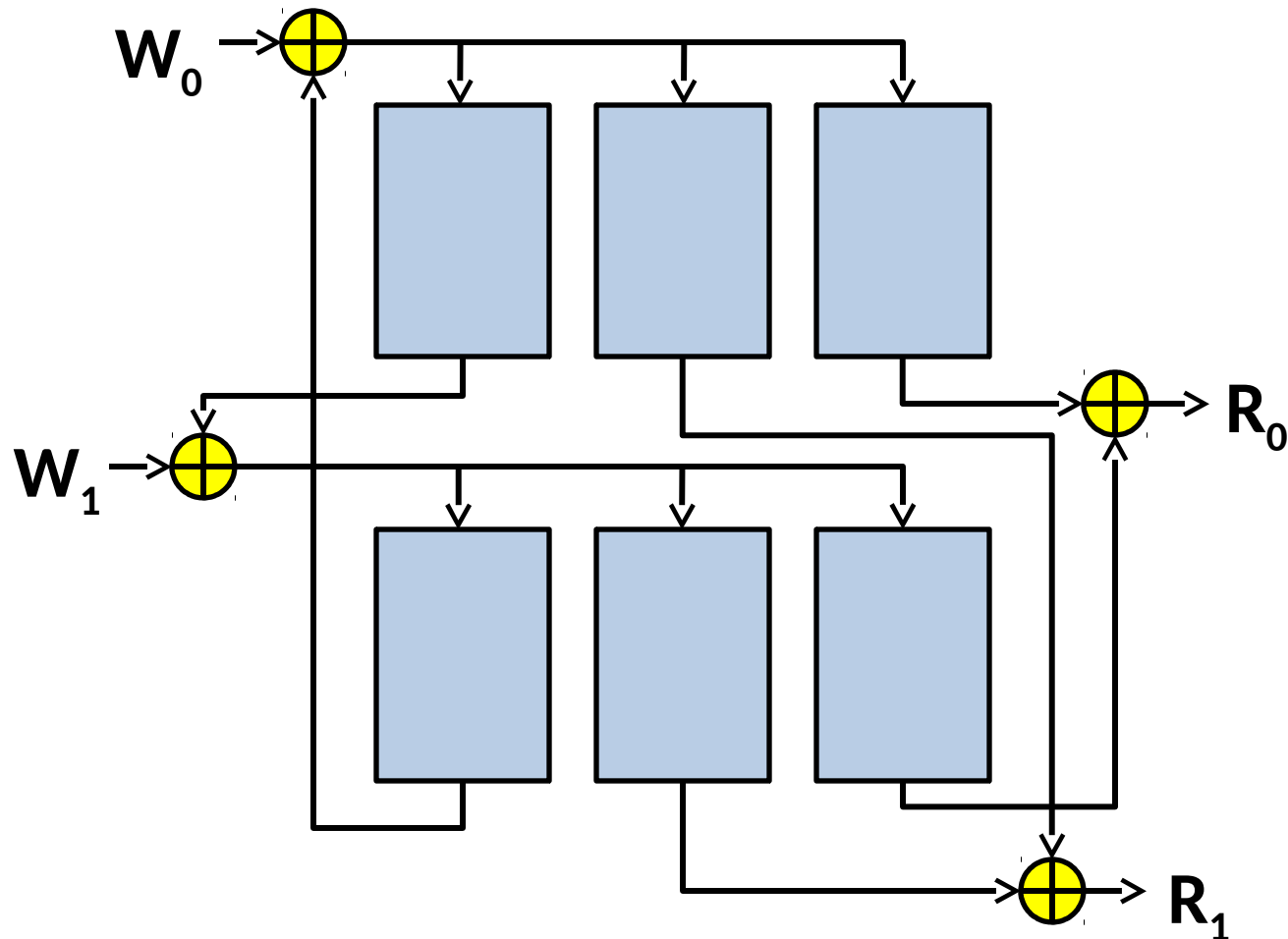
Add support for second read port---done! (almost)

2W/2R XOR Design
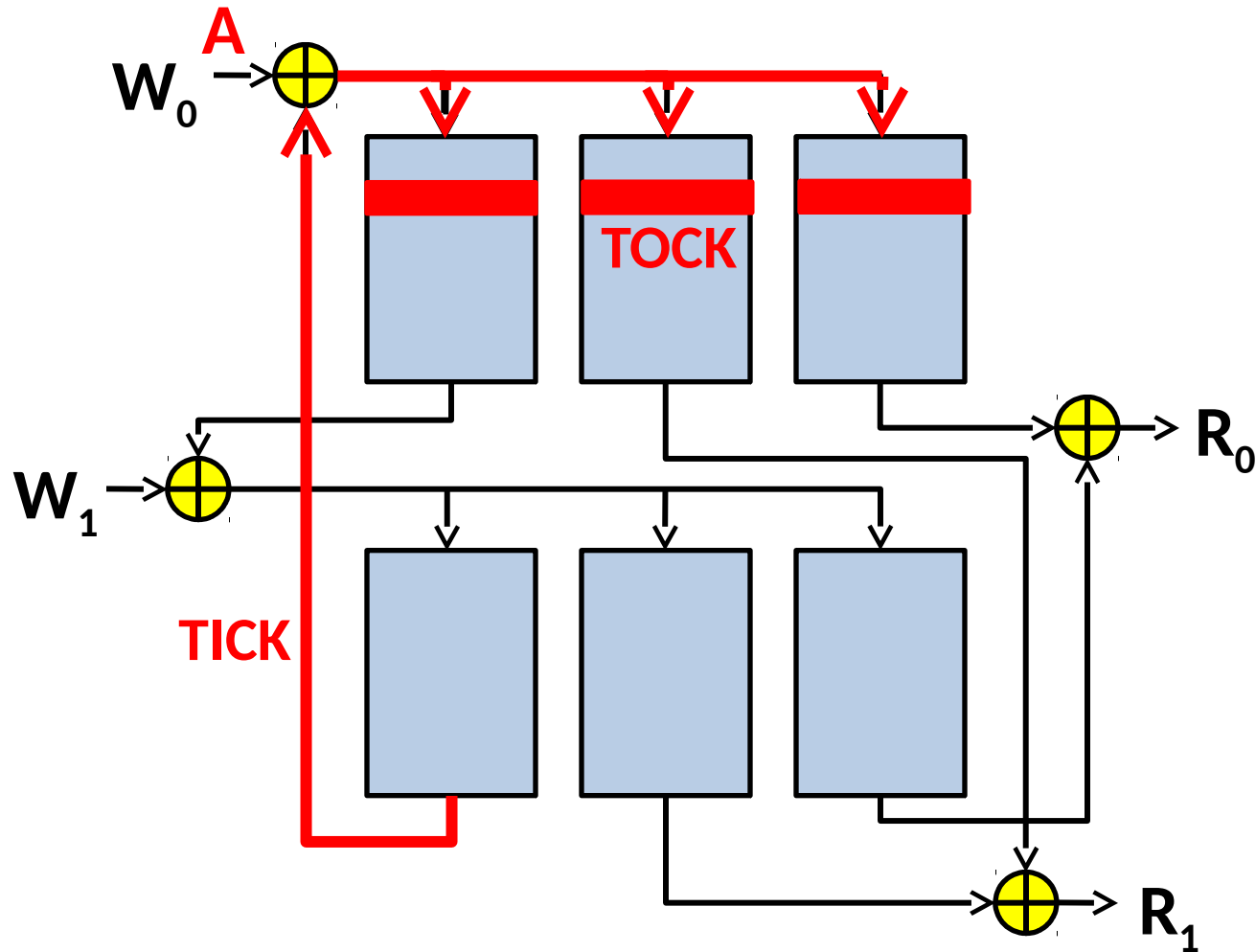
Writing requires reading: hence 2 cycles to write!
Solution: need pipelining to avoid stalling

2W/2R XOR Design

What if read a location one cycle after written?
Solution: bypass with forwarding logic

Generalized XOR Design
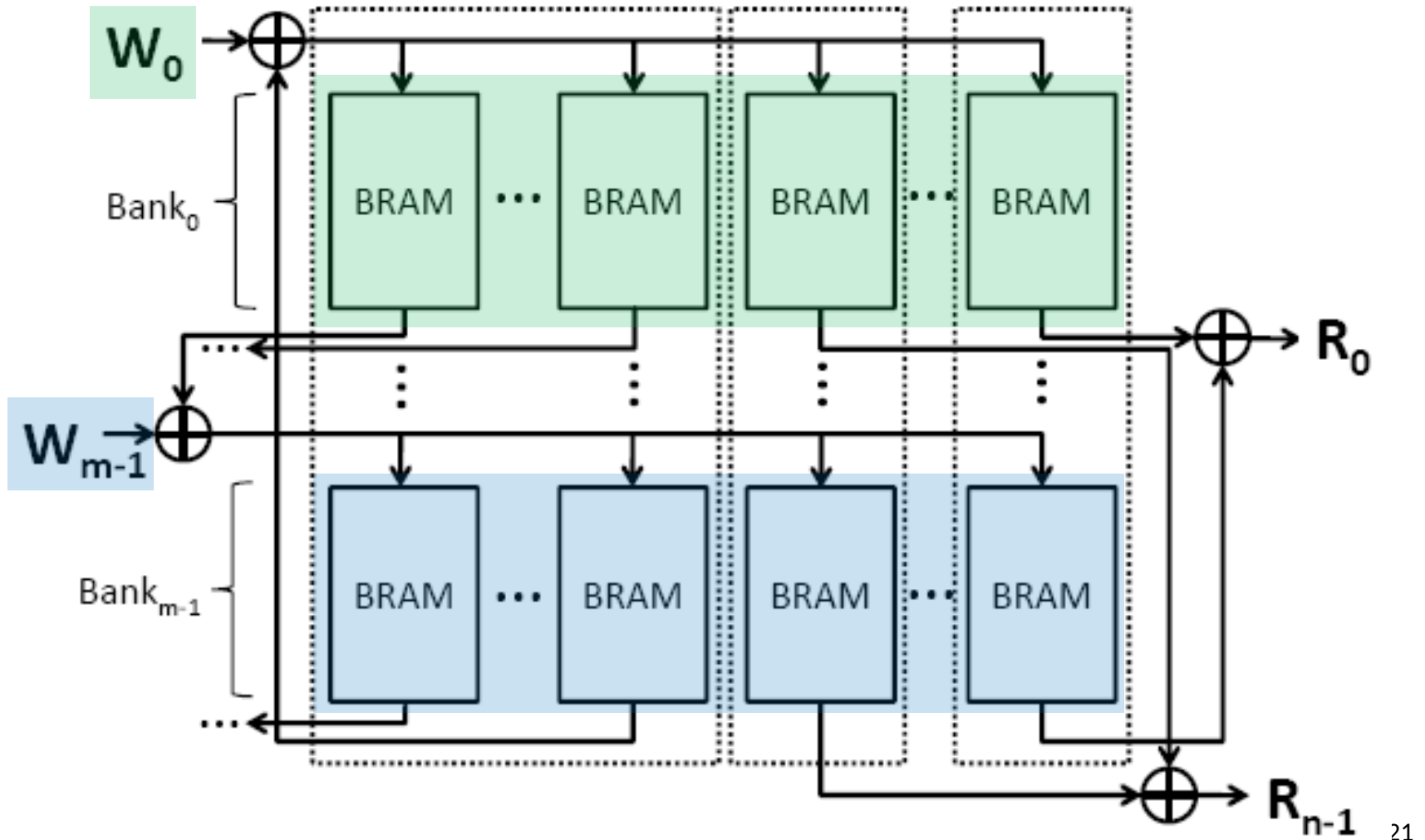
Generalized XOR Design
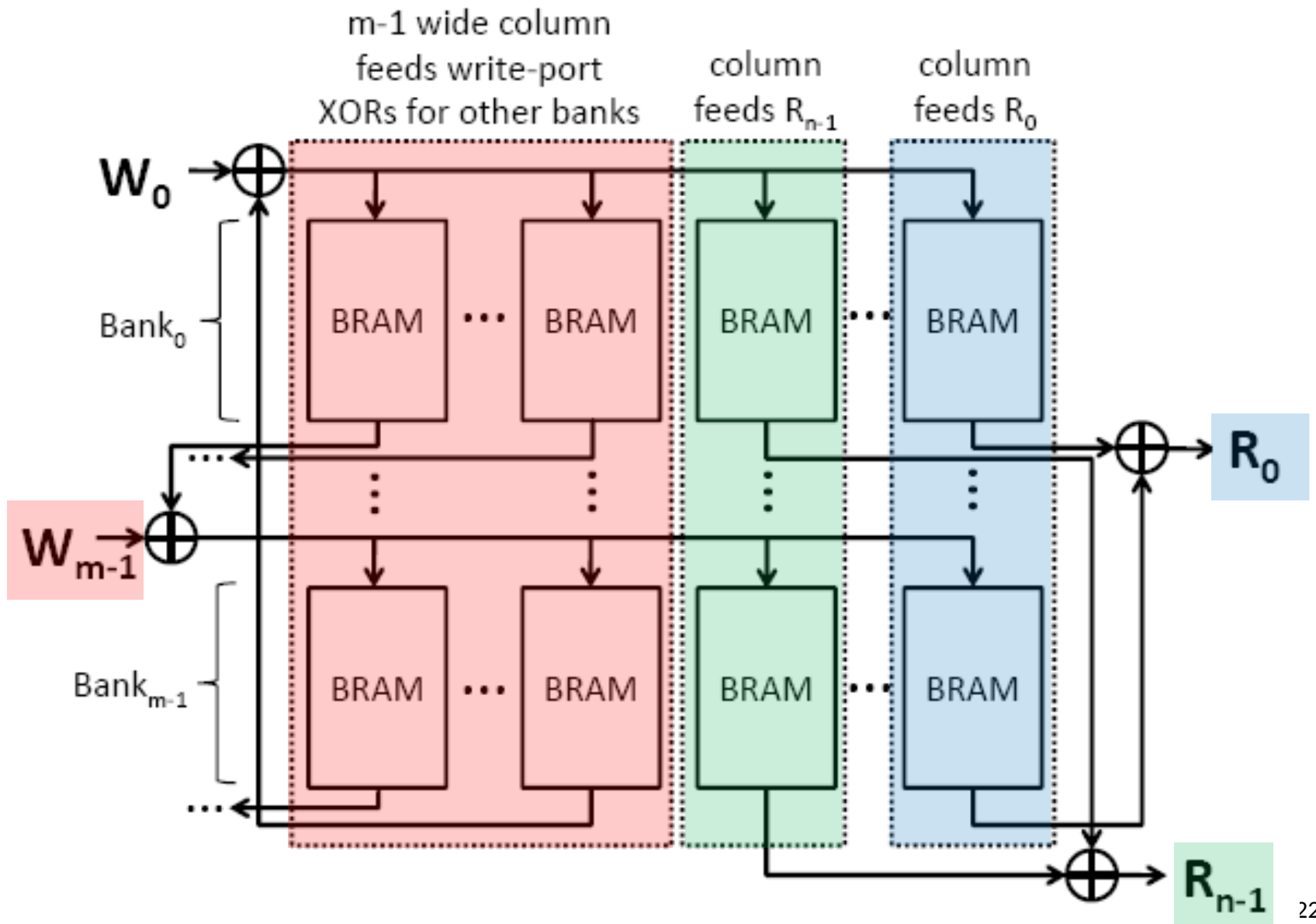
LVT vs XOR

Methodology
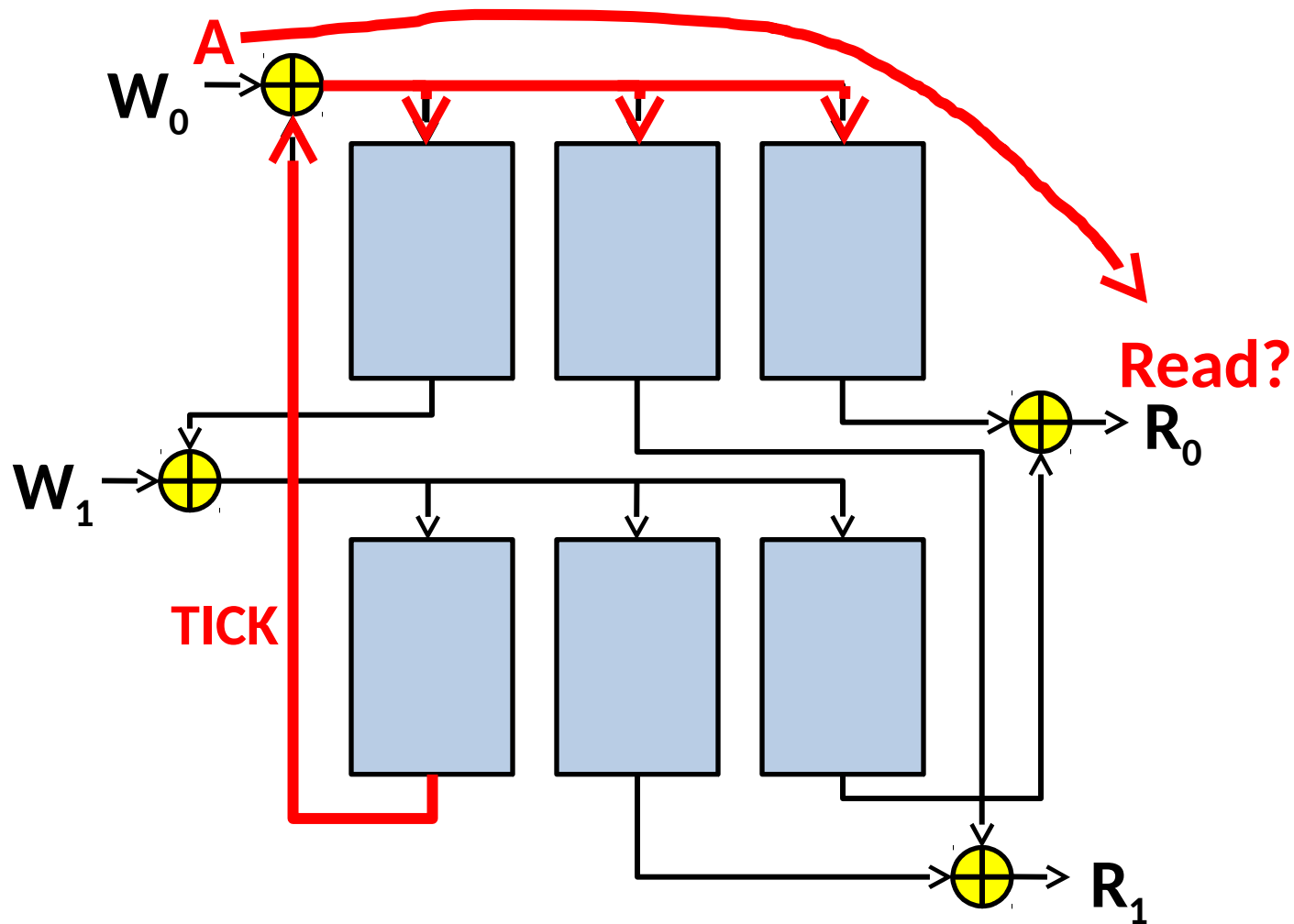
Use Quartus 10.0 to target Stratix IV

- Favor speed over area, optimize
- Average over 10 seeds to get Fmax

Measure area as *Total Equivalent Area (TEA)*

- Expresses area in a single unit (ALMs)
- 1 M9K == 28.7ALMs **

Measure a large design space

- Depth: 32-8192 memory locations
- Ports: 2W/4R, 4W/8R, 8W/16R

** H. Wong, J. Rose and V. Betz, "Comparing FPGA vs. Custom CMOS and the Impact on Processor Microarchitecture," *ACM Int. Symp.on FPGAs*, 2011

Example Layout: 8192-deep 2W/4R
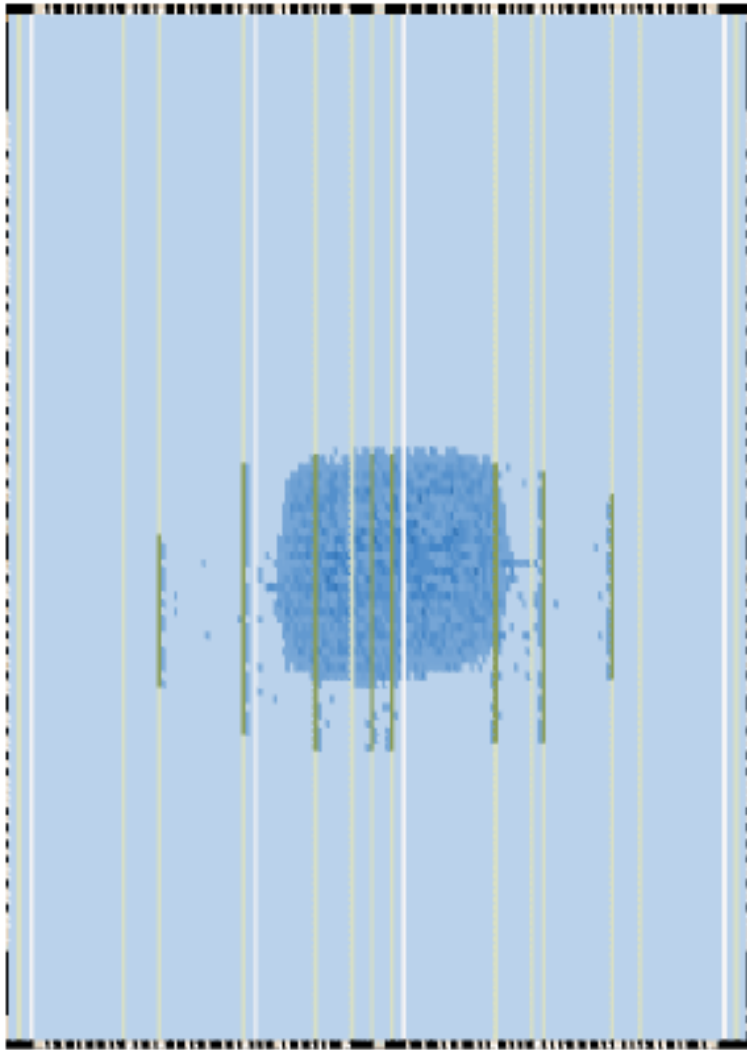
LVT
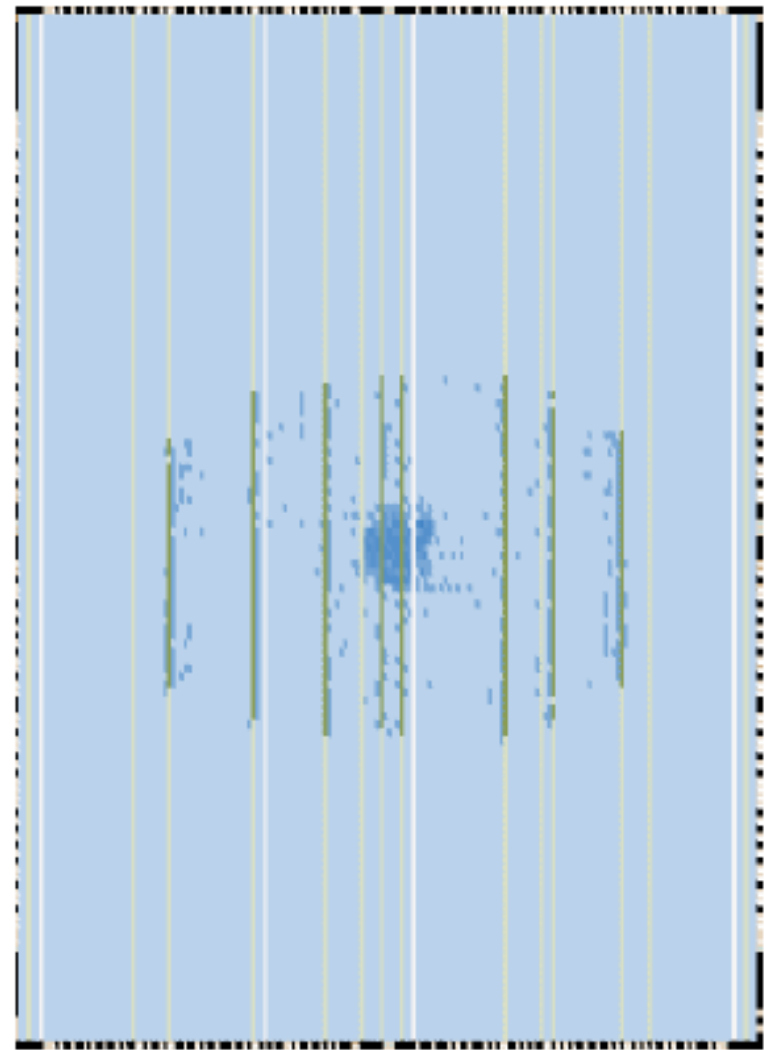
XOR

Significant resource diversity!

2W/4R
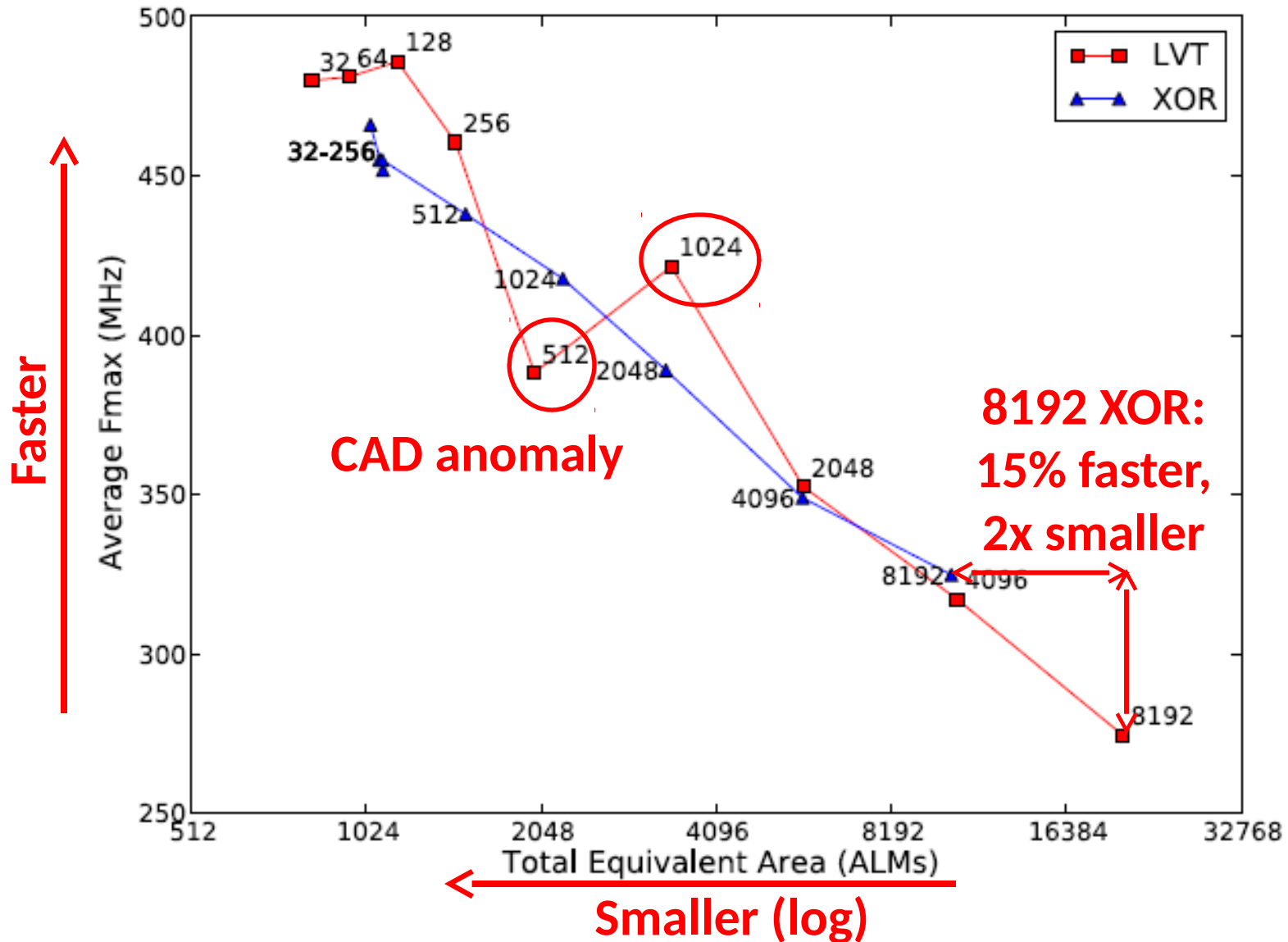
LVT better for small designs, XOR better for large

Navigating the Design Space (2W/4R)

Depth	Design that minimizes:		
	Delay	ALMs	BRAMs
32	Equal	LVT	LVT
64	LVT	LVT	LVT
128	LVT	XOR	LVT
256	Equal	XOR	LVT
512	XOR	XOR	LVT
1024	Equal	XOR	LVT
2048	XOR	XOR	LVT
4096	XOR	XOR	LVT
8192	XOR	XOR	LVT

Which is best? That depends…

Summary

2W/4R

Depth	Design that minimizes:		
	Delay	ALMs	BRAMS
32	Equal	LVT	LVT
64	LVT	LVT	LVT
128	LVT	XOR	LVT
256	Equal	XOR	LVT
512	XOR	XOR	LVT
1024	Equal	XOR	LVT
2048	XOR	XOR	LVT
4096	XOR	XOR	LVT
8192	XOR	XOR	LVT

4W/8R

Depth	Design that minimizes:		
	Delay	ALMs	BRAMS
32	LVT	LVT	LVT
64	LVT	LVT	LVT
128	LVT	Equal	LVT
256	Equal	XOR	LVT
512	Equal	XOR	LVT
1024	XOR	XOR	LVT
2048	XOR	XOR	LVT
4096	XOR	XOR	LVT

8W/16R

Depth	Design that minimizes:		
	Delay	ALMs	BRAMS
32	LVT	LVT	LVT
64	LVT	LVT	LVT
128	LVT	Equal	LVT
256	Equal	XOR	LVT
512	LVT	XOR	LVT
1024	XOR	XOR	LVT

Use LVT when:
- **want to minimize BRAMs**
  - **building <= 128 depth**

else use XOR, i.e. when:
- **>= 256 & spare BRAMS**

28

Conclusions

- **Use LVT when**
  - building up to 128-entry designs
  - you want to minimize BRAM usage

- **Use XOR when**
  - building 256-entry or larger designs
  - you want to minimize ALM usage

- **Interesting Library/Generator?**
  - help the designer automatically navigate this space

- **Further work**
  - Exploring "true-dual-port" mode, stalls, power
  - Results on other vendor's FPGAs