Maximizing Speed and Density of Tiled FPGA Overlays via Partitioning

Charles Eric LaForest

J. Gregory Steffan

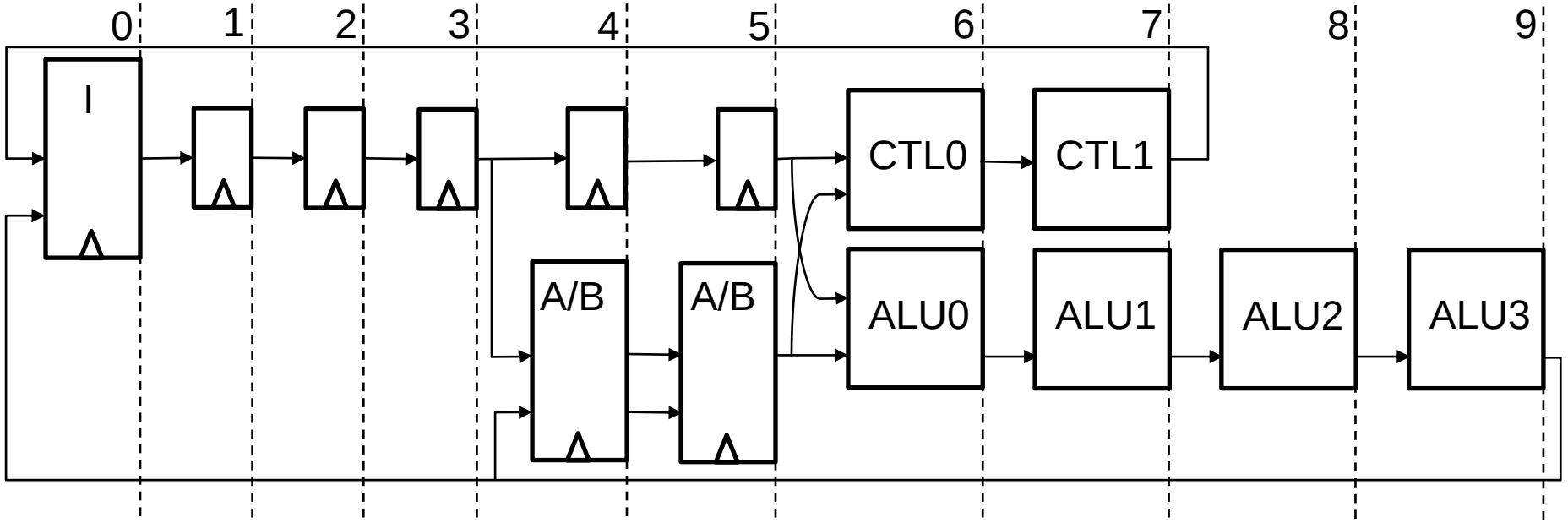
University of Toronto

ICFPT 2013

FPGA Overlay Architectures

- Layer of abstraction over FPGA
  - Easier development
  - Compile software rather than design hardware
  - Typically a soft-processor
- Provides parallelism through “tiling”
  - Multiple Cores
  - Multiple Datapaths (Vector Processors)

Scalar Core: Octavo

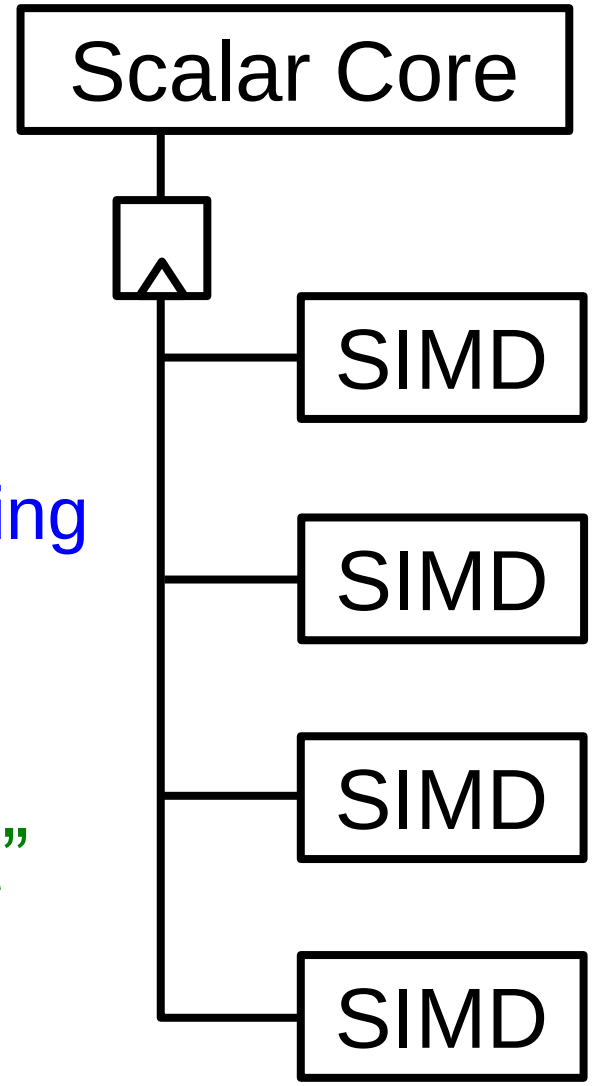


- A soft-processor on Stratix IV
  - 10 stages, 8 threads, 550 MHz in many cases
  - Highly configurable and customizable
- Published at FPGA 2012
  - "*Octavo: an FPGA-Centric Processor Family*"

Tiling Datapaths

- Attach SIMD Lanes
  - Copies of the Scalar Datapath
- Private data memories
- Replicated instruction logic
  - Pipelined distribution and decoding
  - Not the critical path!

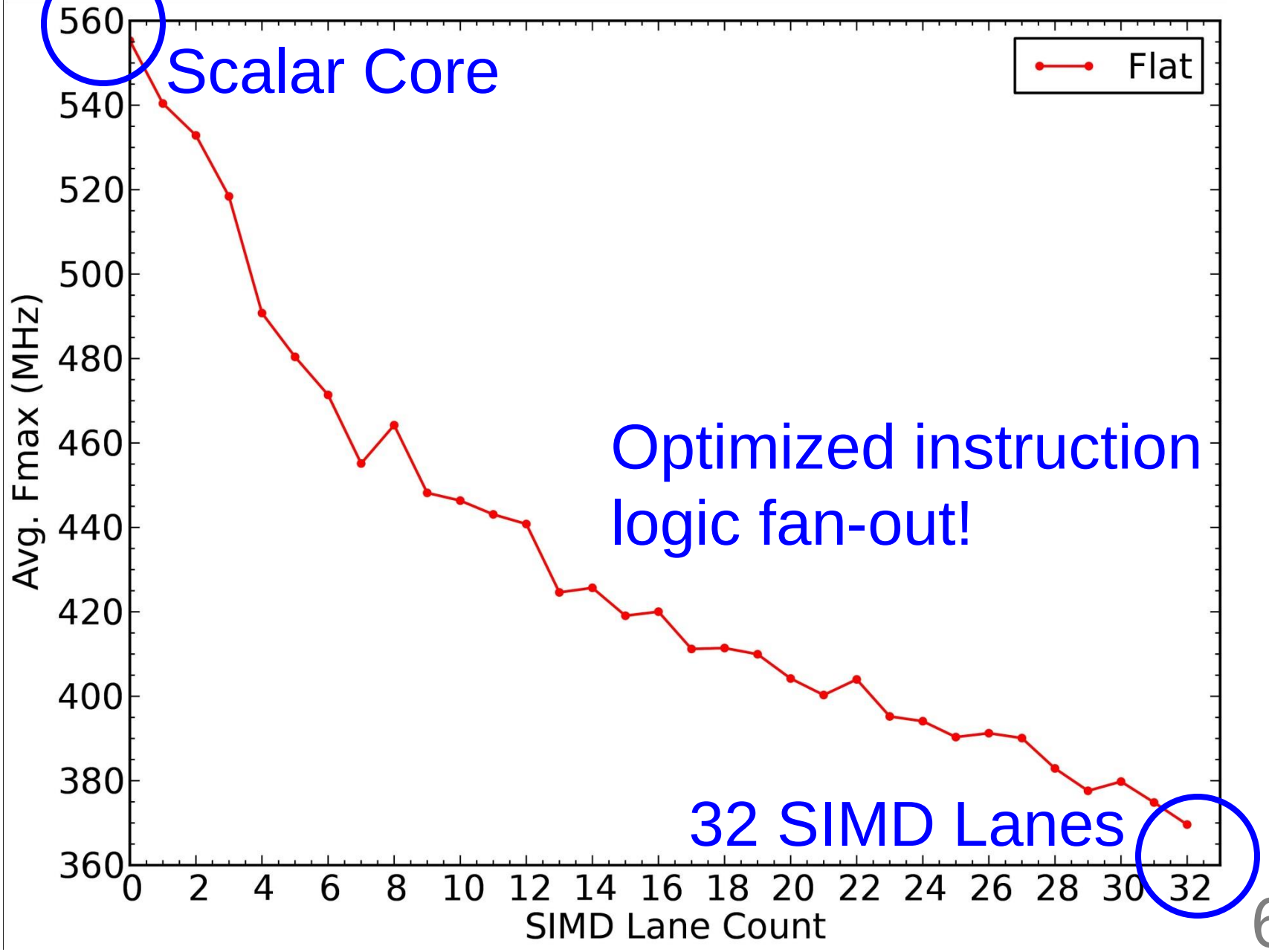
- Intuition: Fmax stays “constant”



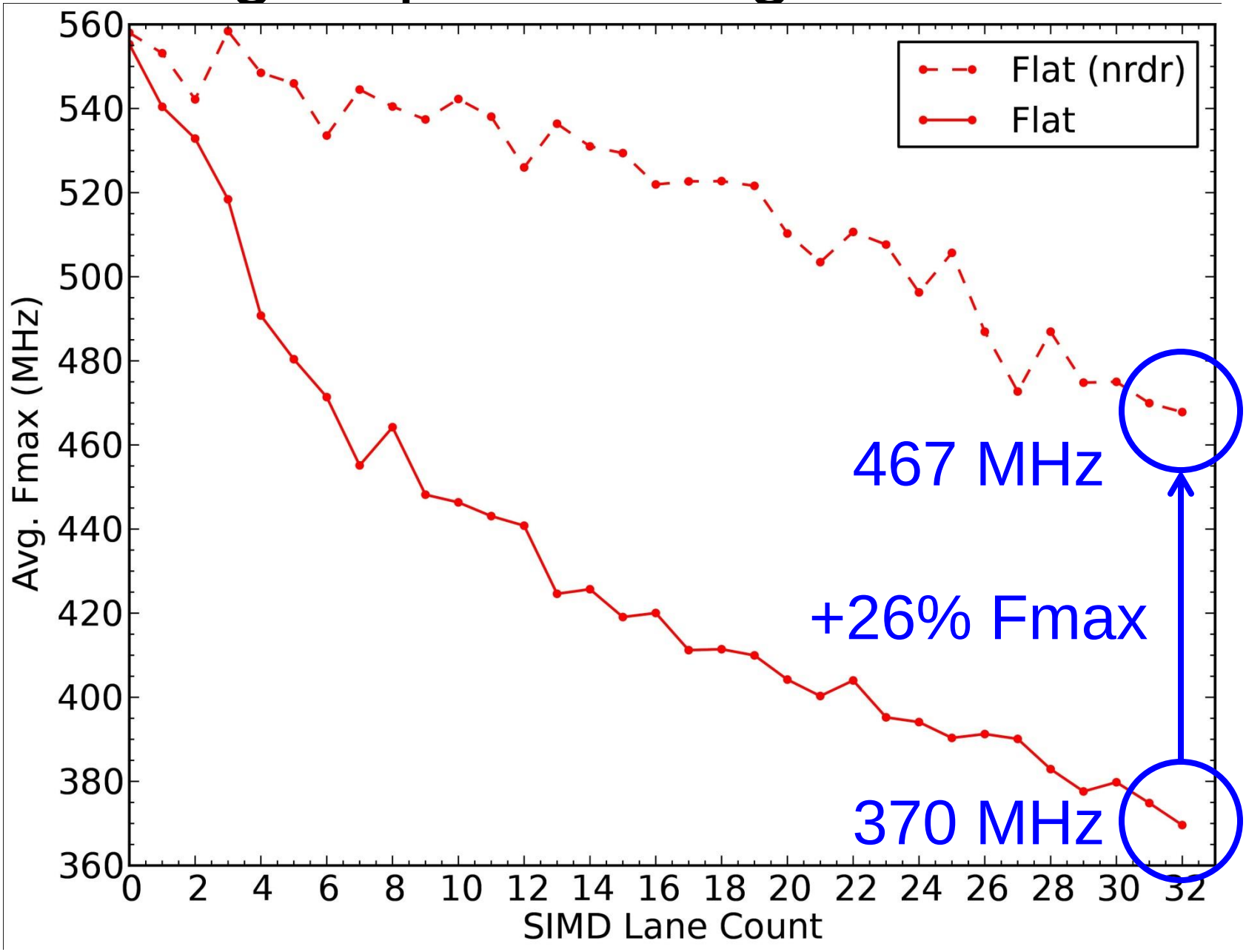
Experimental Framework

- Quartus 12.1 targeting Stratix IV E230
- Test harness to isolate paths to outside
- Synthesize for speed
  - Including full physical synthesis
- Maximum Place & Route effort
- 550 MHz clock target (BRAM Fmax)
- Average results over 10 runs
- Measure area as *equivalent ALMs* (eALMs)

Fmax with Increasing SIMD Lanes

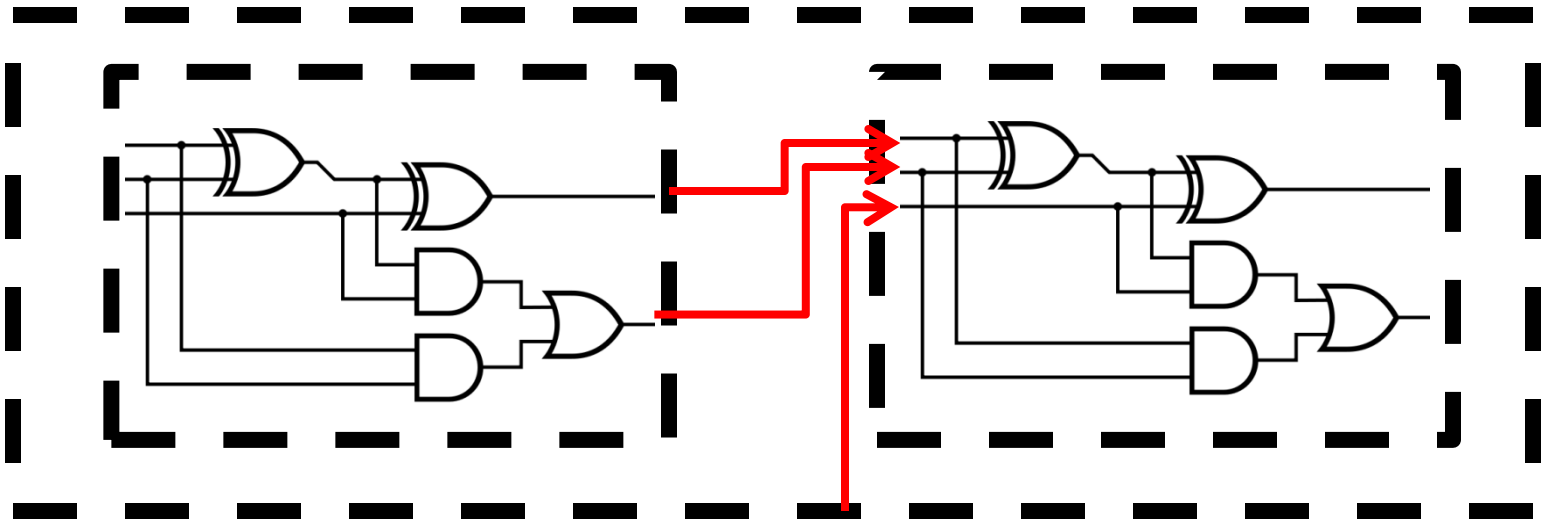


Disabling Duplicate Register Removal

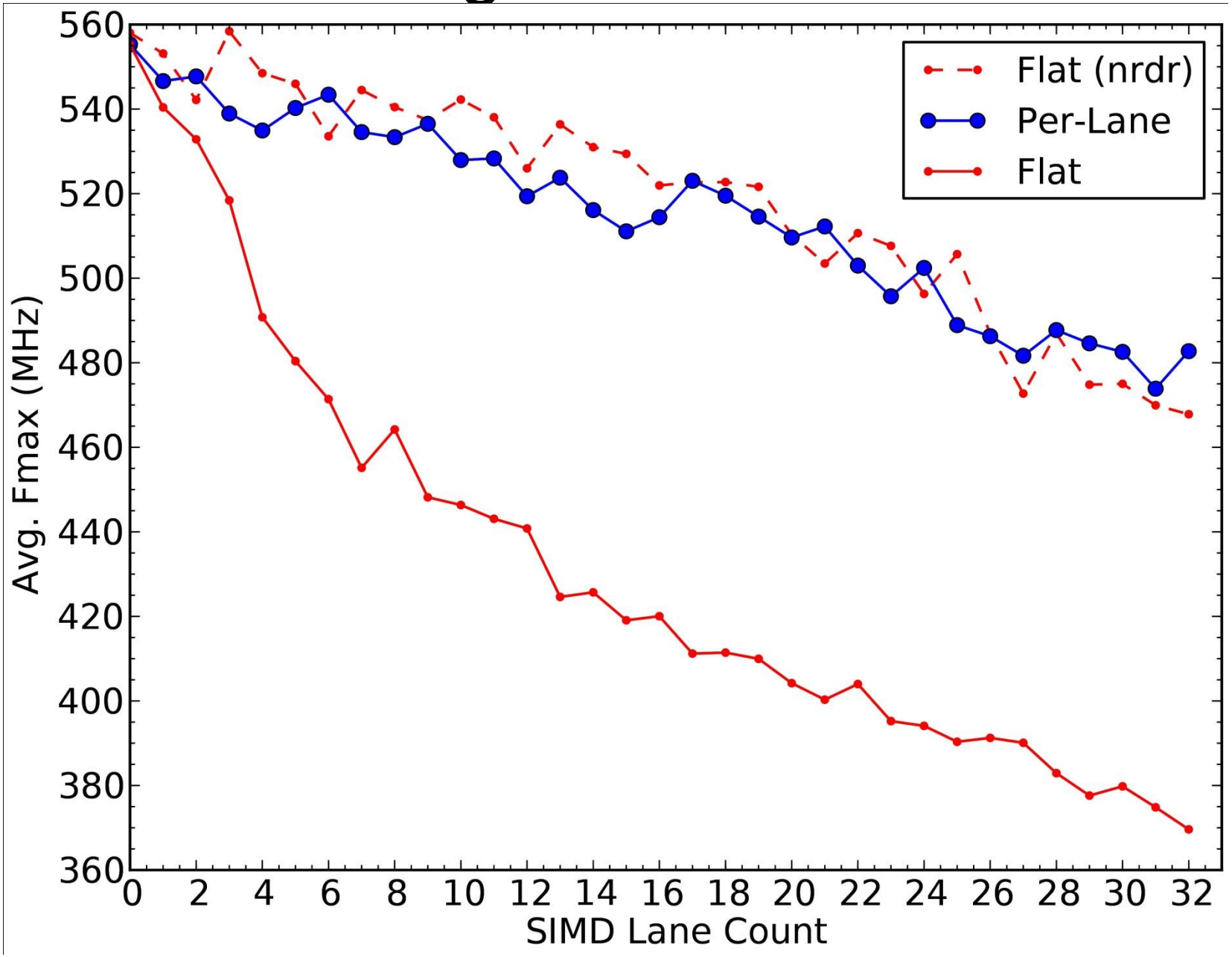


A Better Way: Partitioning

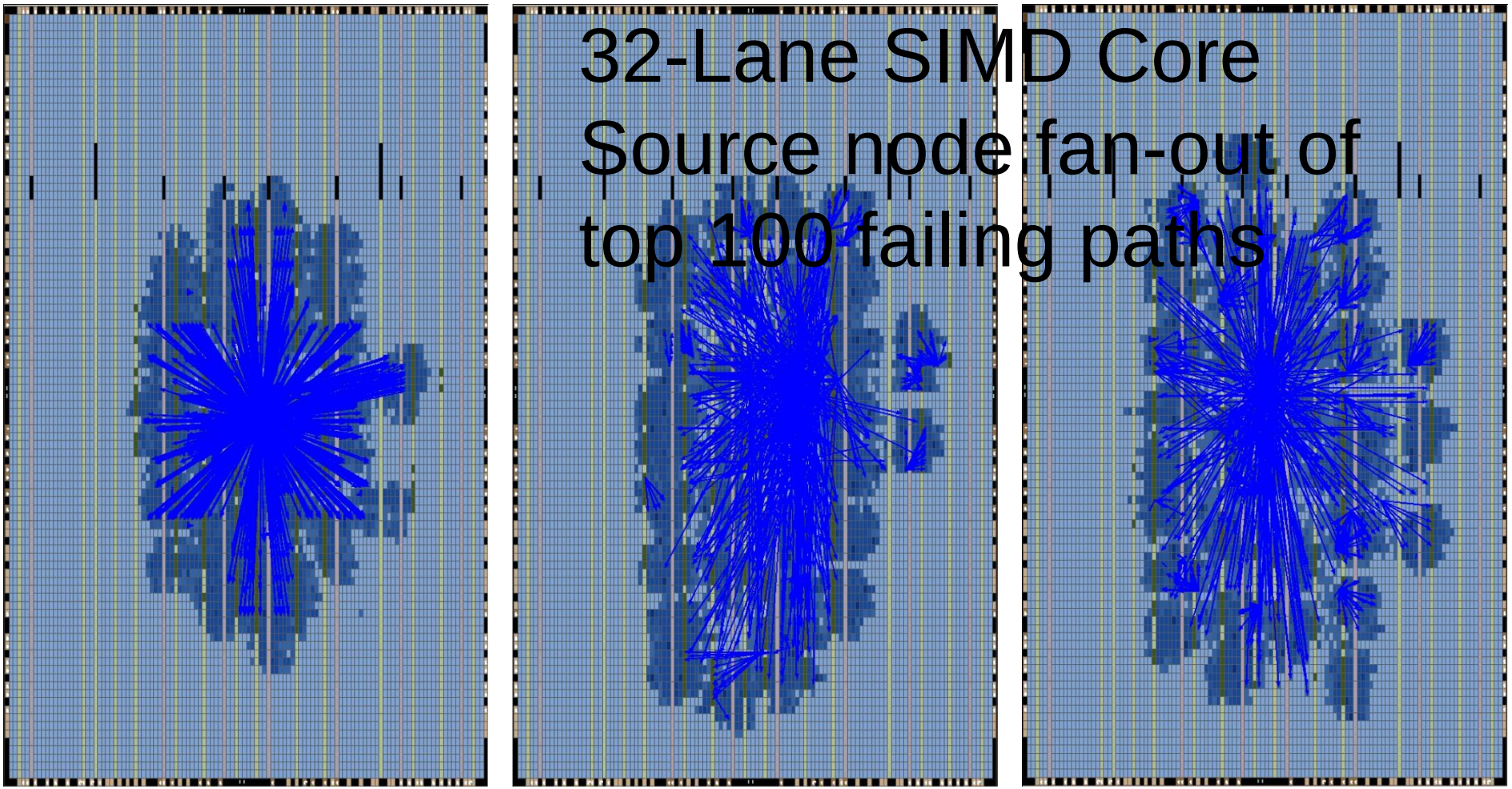
- Logical division of a design
  - Synthesize as separate netlists
  - Generally, optimizations do not cross partitions
    - Register retiming
    - Register de-duplication
    - Boolean simplification



Partitioning Each SIMD Lane

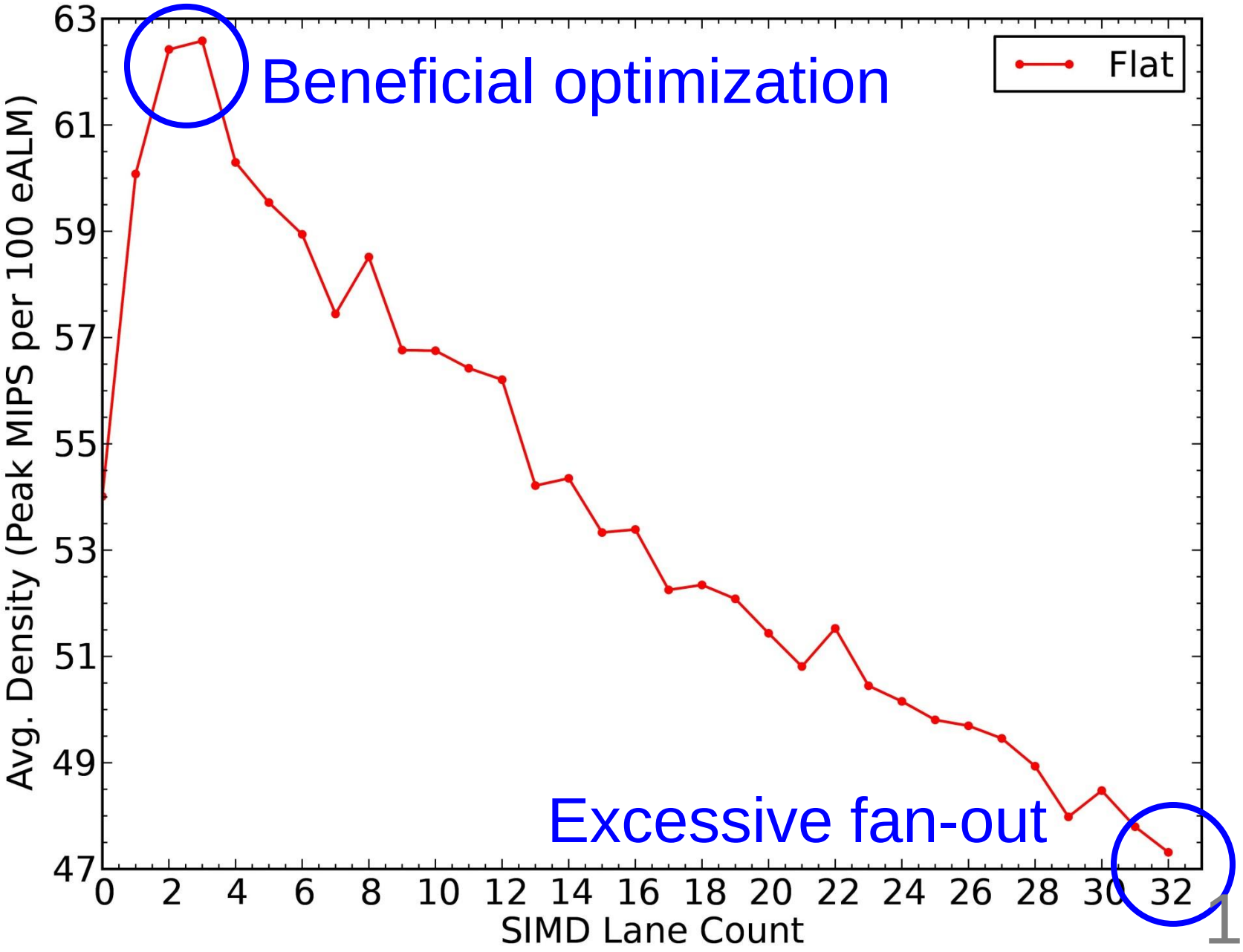


Impact on Critical Paths

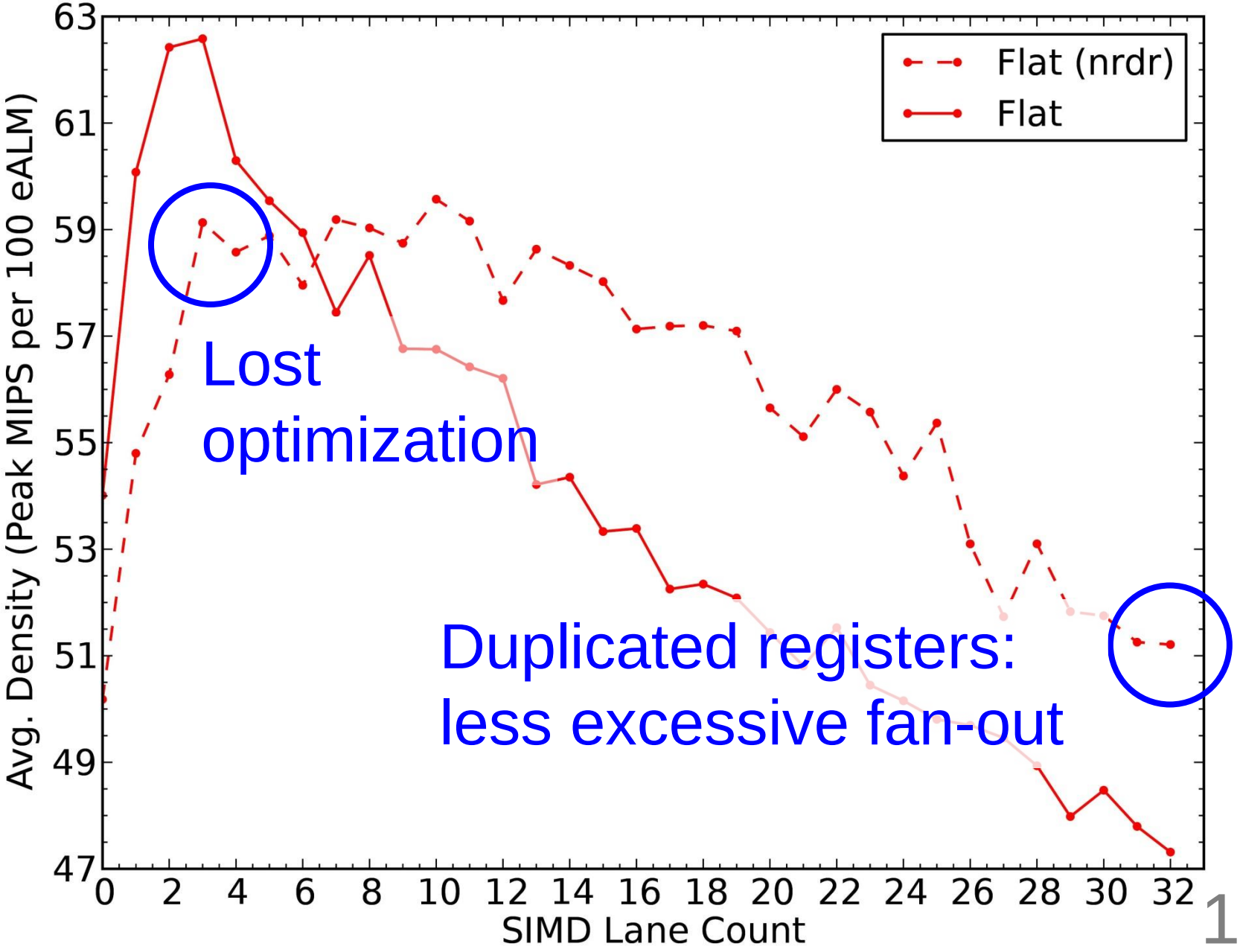


Flat	Duplicate Registers	Partitioned
373 MHz	456 MHz	489 MHz
21 Nodes	43 Nodes	64 Nodes

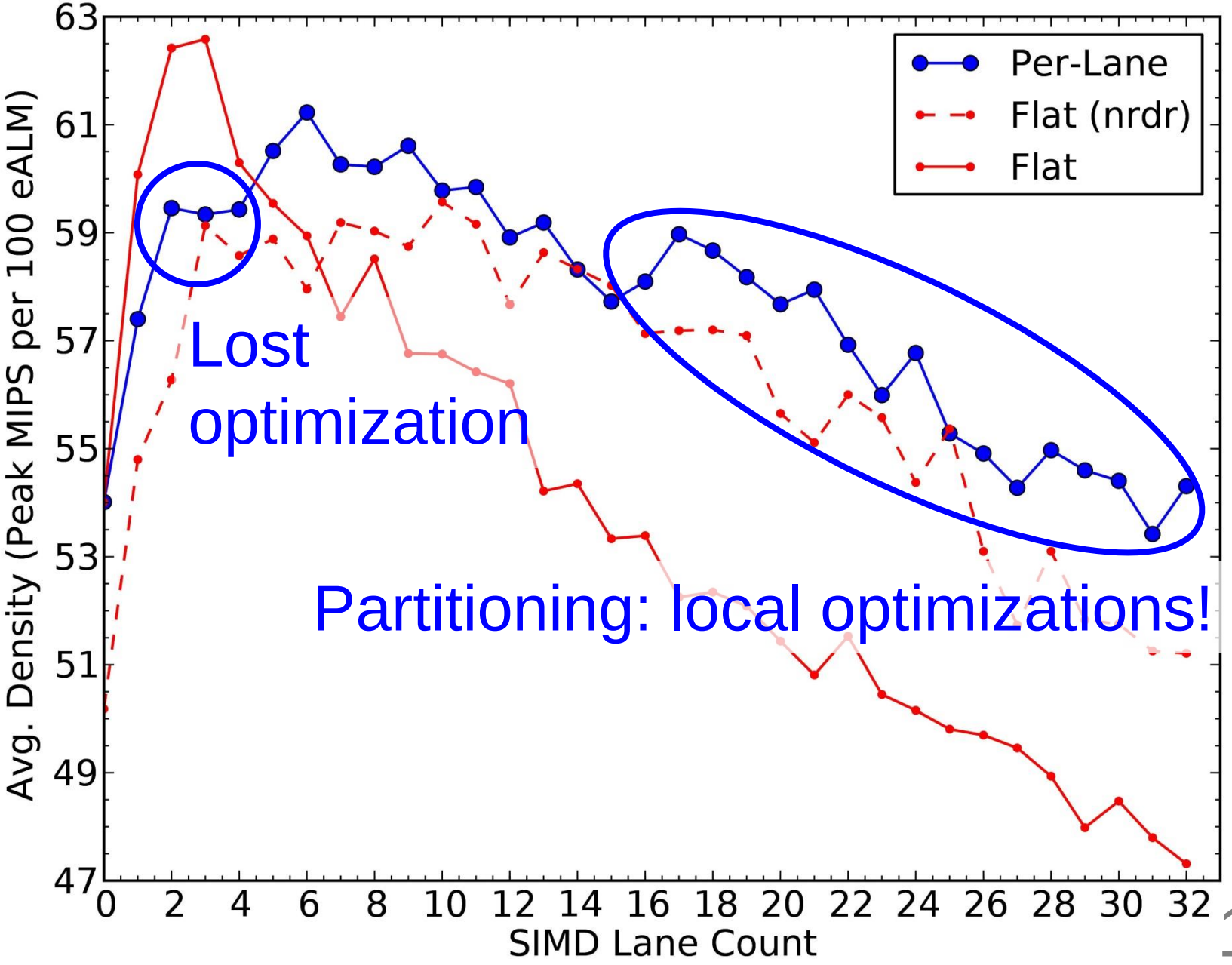
Improving Compute Density



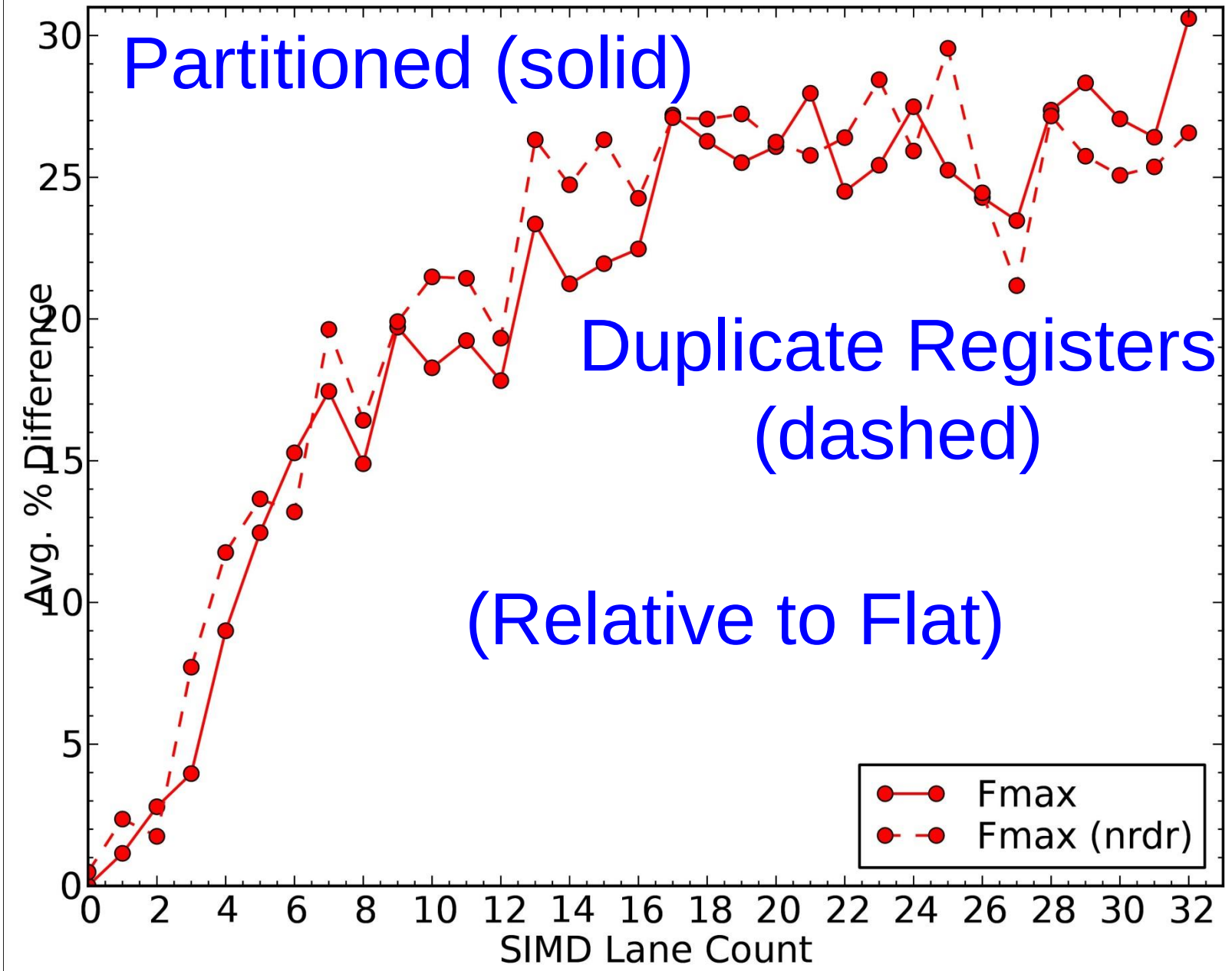
Improving Compute Density



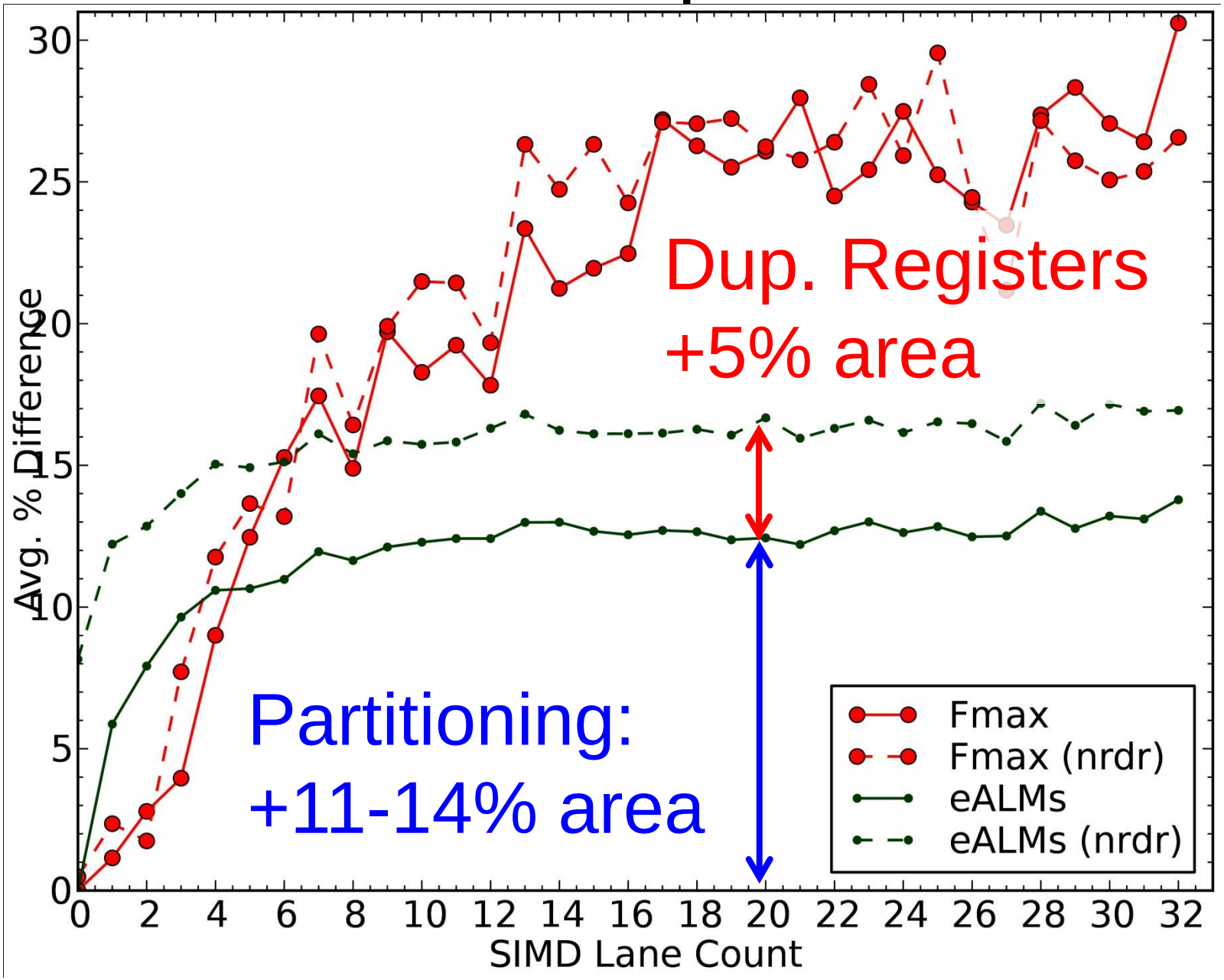
Improving Compute Density



Area Impact


Partitioned (solid)
Duplicate Registers (dashed)
(Relative to Flat)
Avg. % Difference
SIMD Lane Count
0
5
10
15
20
25
30
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32
Fmax
Fmax (nrdr)

Area Impact


Avg. % Difference
SIMD Lane Count
0
5
10
15
20
25
30
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
Dup. Registers
+5% area
Partitioning:
+11-14% area
Fmax
Fmax (nrdr)
eALMs
eALMs (nrdr)

Summary: SIMD Partitioning

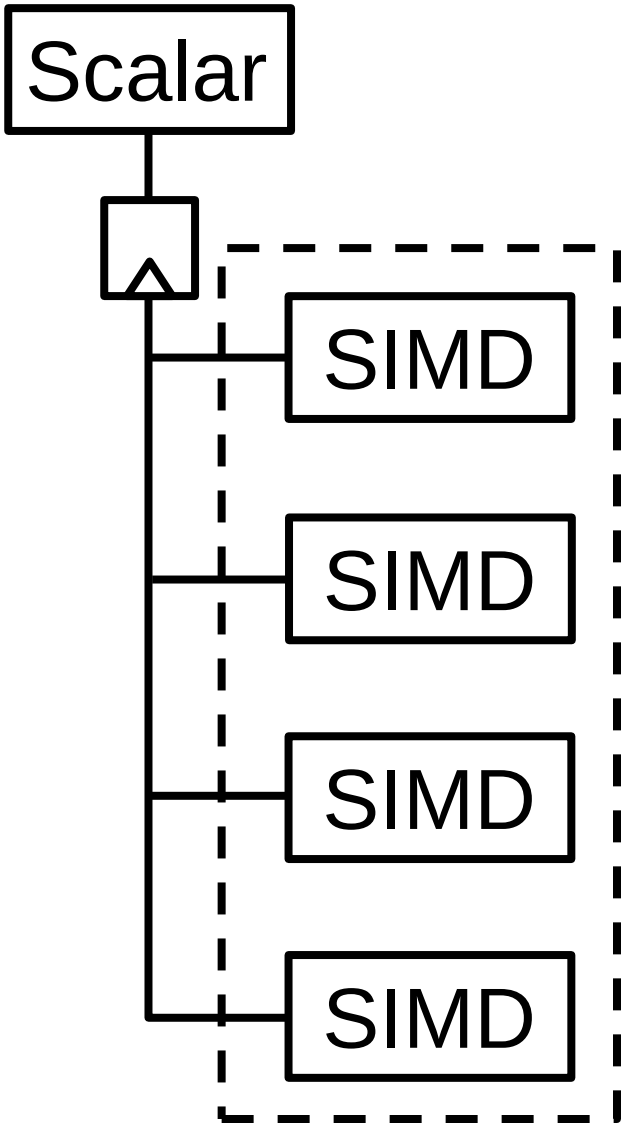
- Scalar Core with 0 to 32 SIMD Lanes
  - Tiling datapaths
- Replicated instruction logic
  - Pipelined distribution and decoding in each Lane
- Placing each Lane in a Partition
  - 32 Lanes: 372 MHz → 482 MHz (+30%)
- Area increase from partitioning: 11-14%
  - Reflects area of preserved replicated logic
- Better to not partition for a few Lanes!
  - 4 lanes or less have better compute density
  - Local optimizations outweigh increased fanout

Partitioning!? Really!?

- Can we get the same results in another way?
- “Layering” replicated logic:
  - Introducing sequential dependency via pipelining
    - Not about breaking critical paths!
    - Isolating instances of replicated logic
  - Prevents optimization of replicated logic
  - Staggers execution of each layer

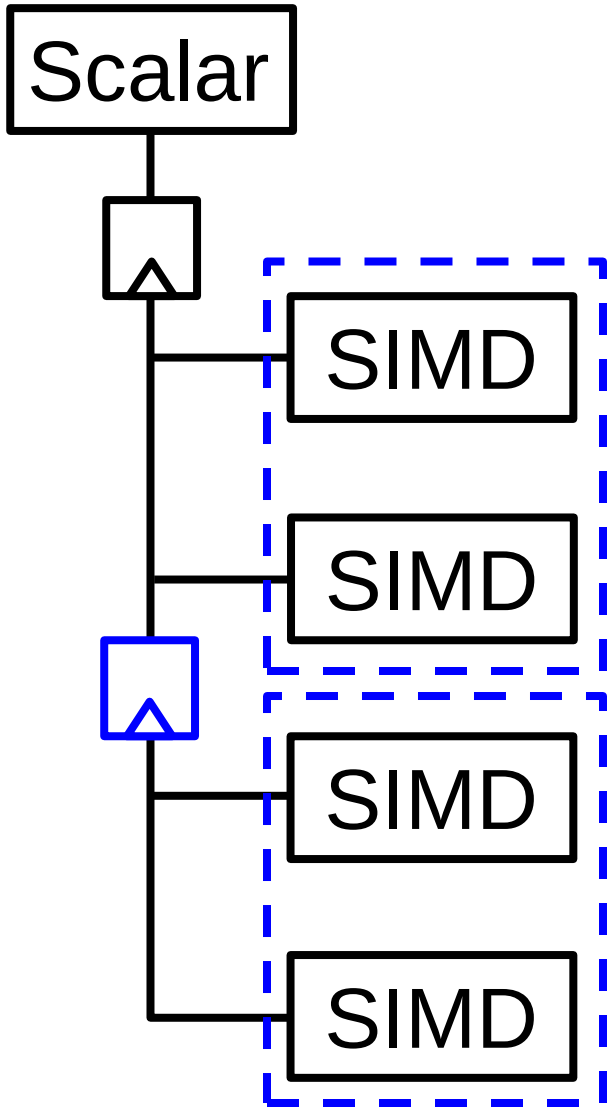
Layering a SIMD Core

1 Layer with 4 Lanes

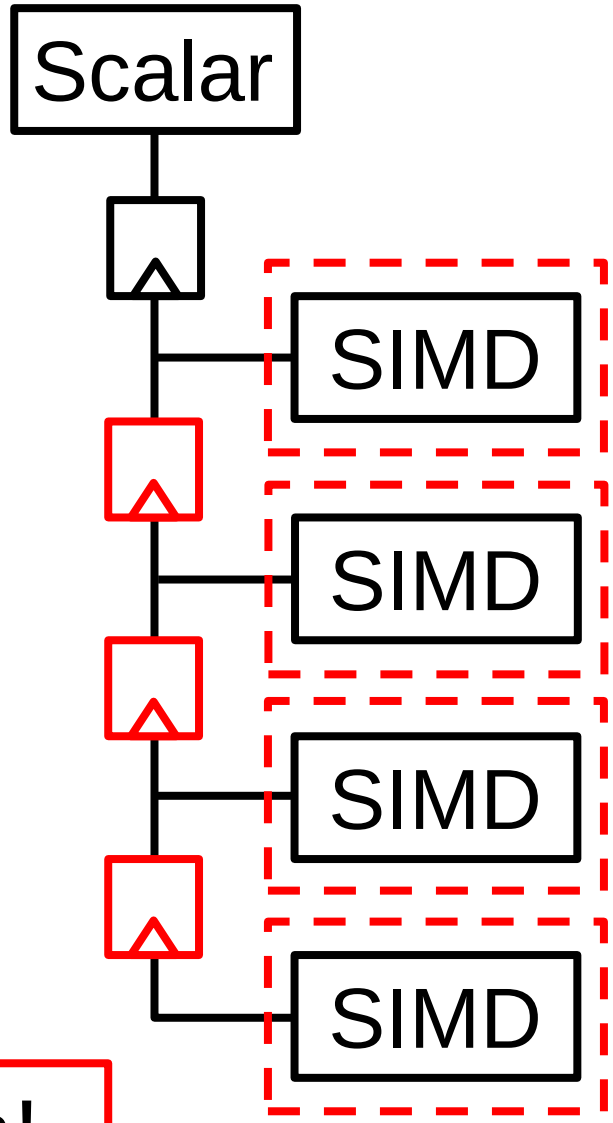


Layering a SIMD Core

2 Layers with 2 Lanes



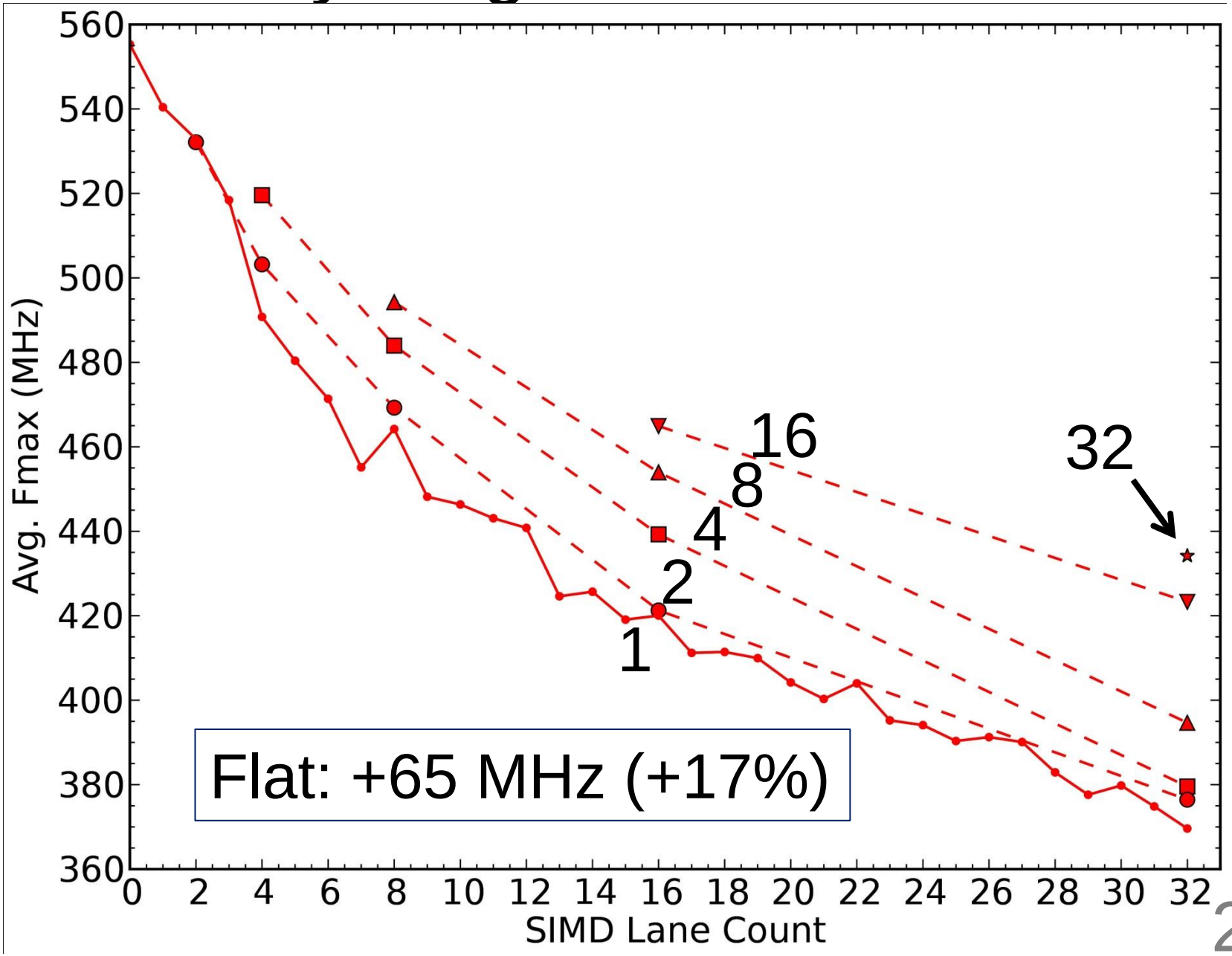
Layering a SIMD Core



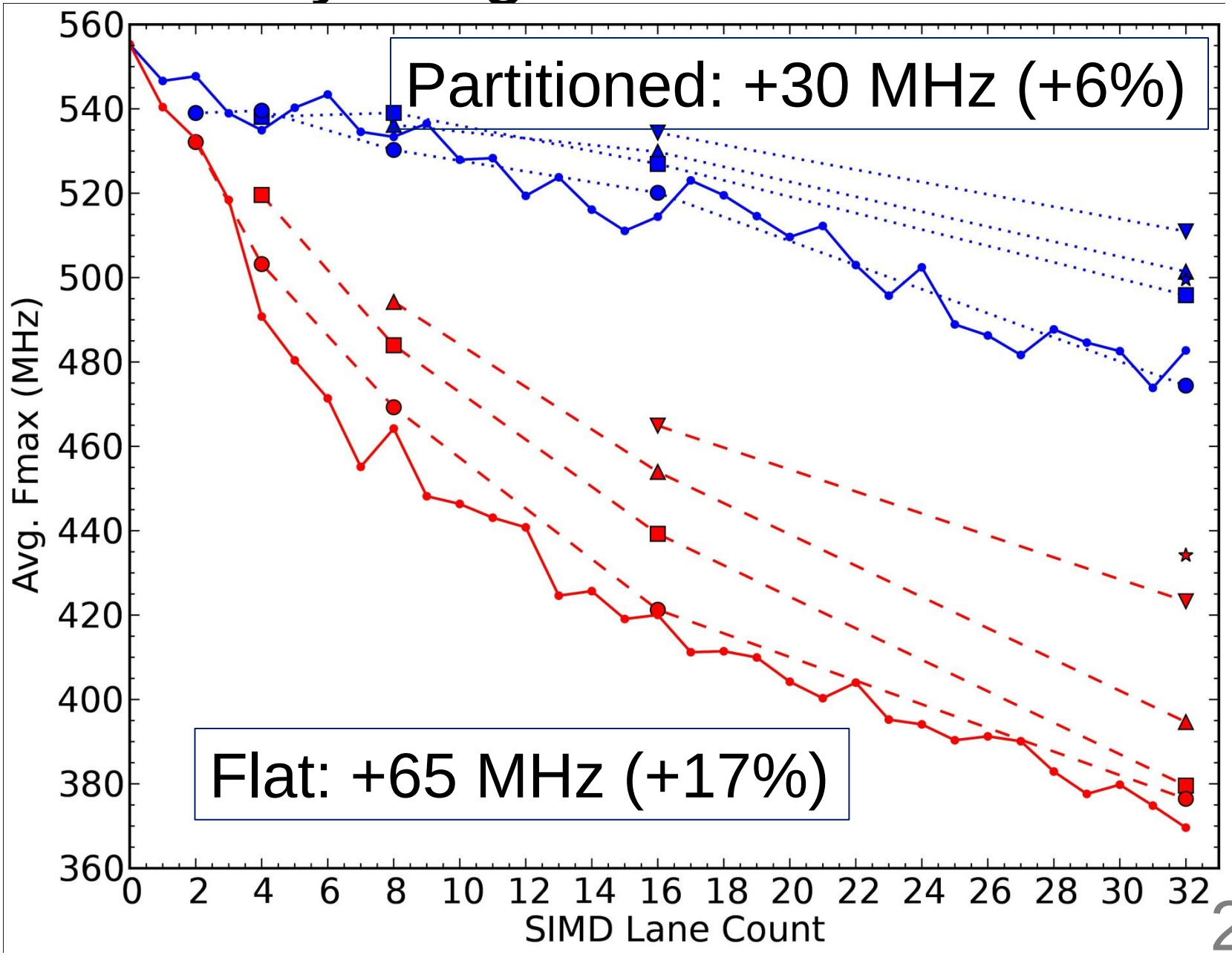
4 Layer with 1 Lane

Staggered SIMD execution!

Layering a SIMD Core

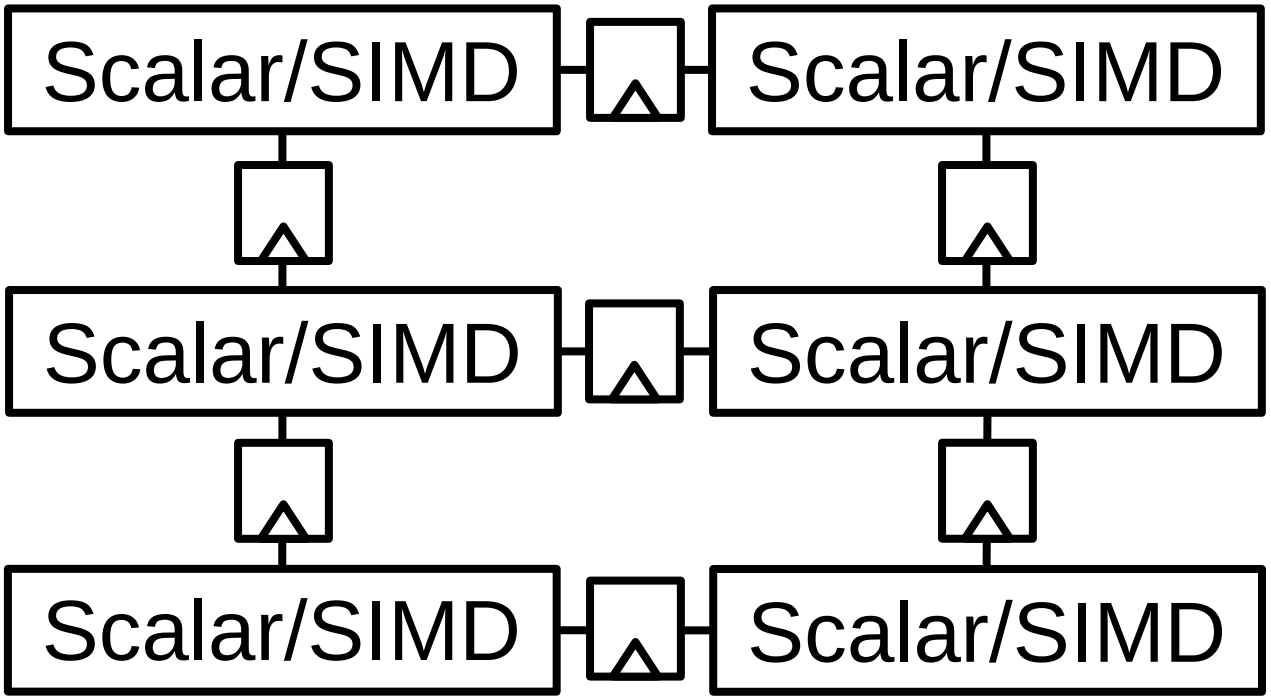


Layering a SIMD Core

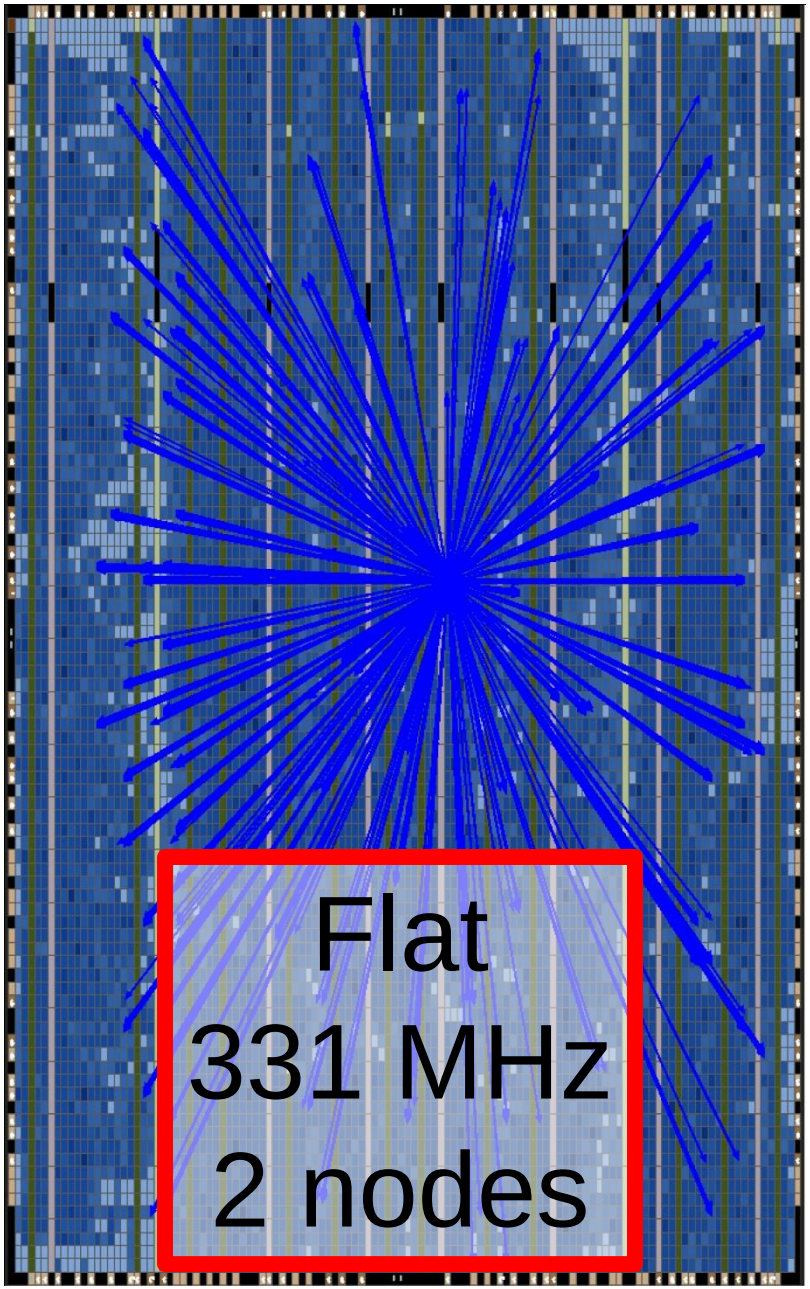


Replicating Entire Processors

- Connect processors in a pipelined Mesh
- Entire processors replicated
- No critical paths between processors
- Intuition: Fmax stays “constant”

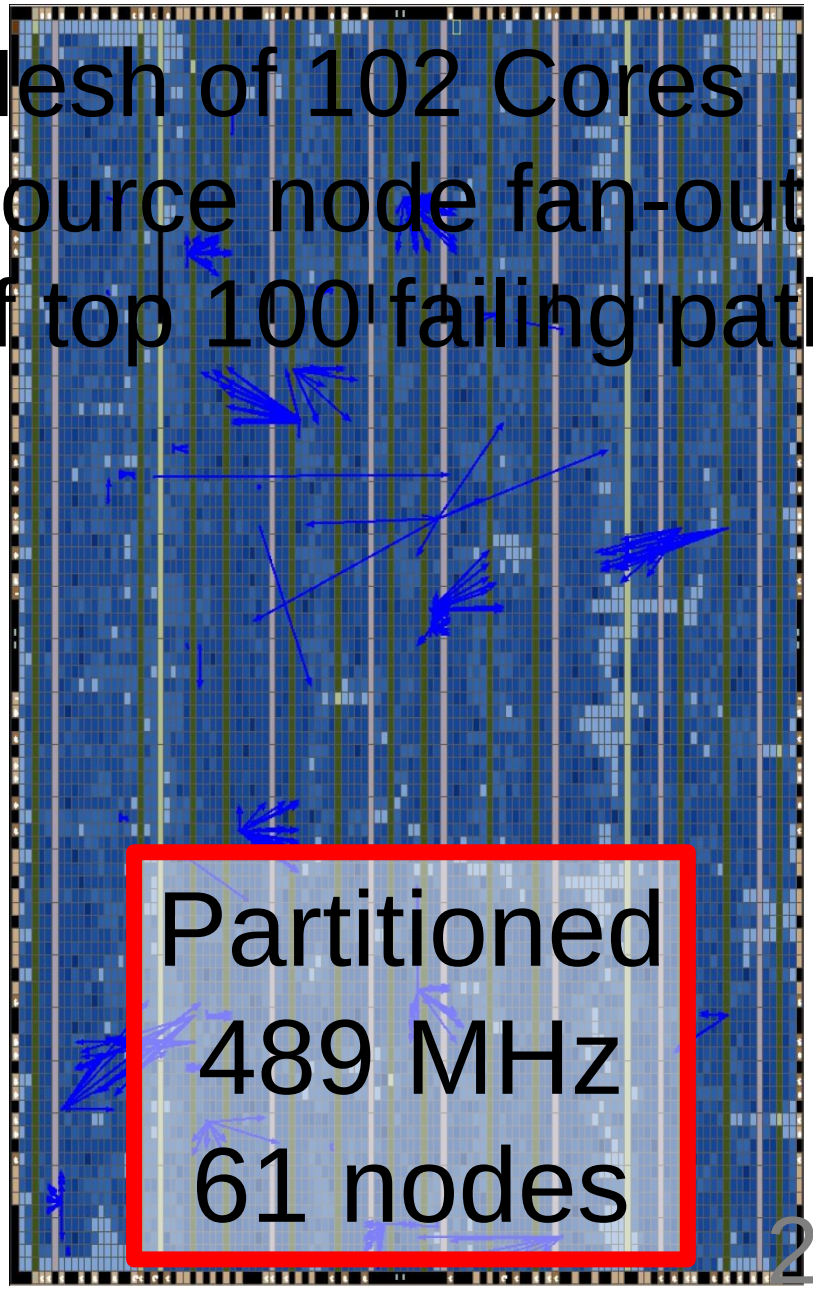


Critical Paths of Meshes



Flat
331 MHz
2 nodes

Mesh of 102 Cores
Source node fan-out
of top 100 failing paths



Partitioned
489 MHz
61 nodes

Partitioning Meshes

- Mesh of 102 Scalar Cores
- Bottleneck: optimized 3-bit counter
  - Round-robin thread counter in each Scalar Core
  - No inputs, but identical and synchronized states
- Placing each Core in a Partition
  - Avg. Fmax: 284 MHz → 437 MHz (54%)
    - Only a 22% Fmax drop over 102x scaling!
- Area increase from partitioning: 0.85%
  - No relation between Fmax and area increases
  - Mysterious 10-11% area overhead from CAD tool
- No significant increase to CAD time!

Summary

- Tiled designs contain replicated logic
  - Forms the critical paths in large tilings
    - Useless optimizations causing excessive fanout
  - Becomes significant at higher speeds
- Partitioning avoids this problem
  - Simpler than per-node management
  - Lower area than disabling duplicate removal
  - Better performance than sequential dependencies
  - Benefit scales with the number of tiles
  - Area increase only proportional to replicated logic
  - No significant change to total CAD time

Further Work

- The CAD tools could automatically…
  - Detect repeated optimizations across modules
    - Tag the replicated logic and/or alert the designer
  - "Restart" optimization
    - Keep performance and save (some) area
    - Only if substantial replicated logic area
  - Partition modules containing replicated logic
- Power Analysis of Partitioning
  - Could go either way…
- CAD tool mysteriously adds area when tiling
  - Main source of density reduction when tiling

Acknowledgements

- Funding:
  - Queen Elizabeth II World Telecom. Congress
  - Walter C. Sumner Foundation
  - Altera
  - NSERC
- Computing:
  - SciNet GPC supercomputer
    - 24 CPU-years of Quartus runs
  - Altera