Scalable and Deterministic Timing-Driven Parallel Placement for FPGAs

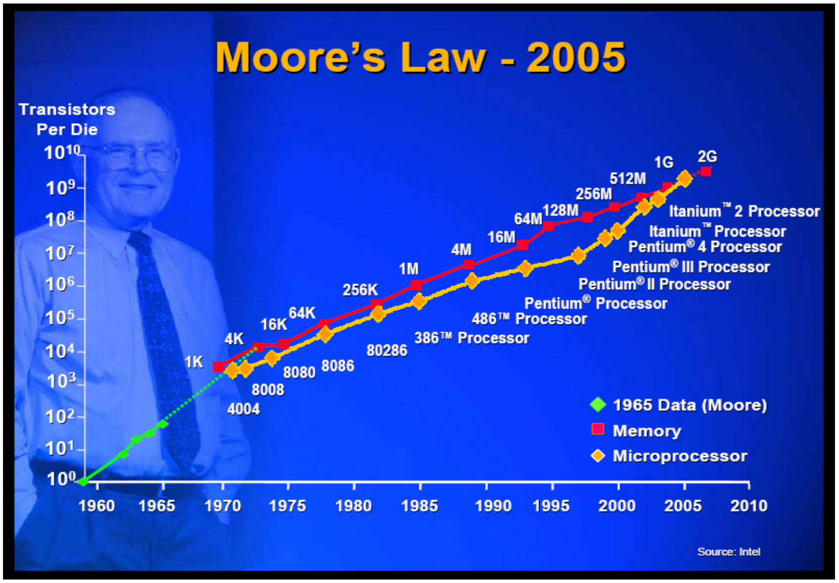
Chris Wang
Guy Lemieux

University of British Columbia

FPGA 2011, March 1st, 2011
Monterey, California

Motivation/Background

- Technology scaling
- FPGA synthesis runtime
  - **not** scaling
- Runtime crisis



Moore's Law - 2005

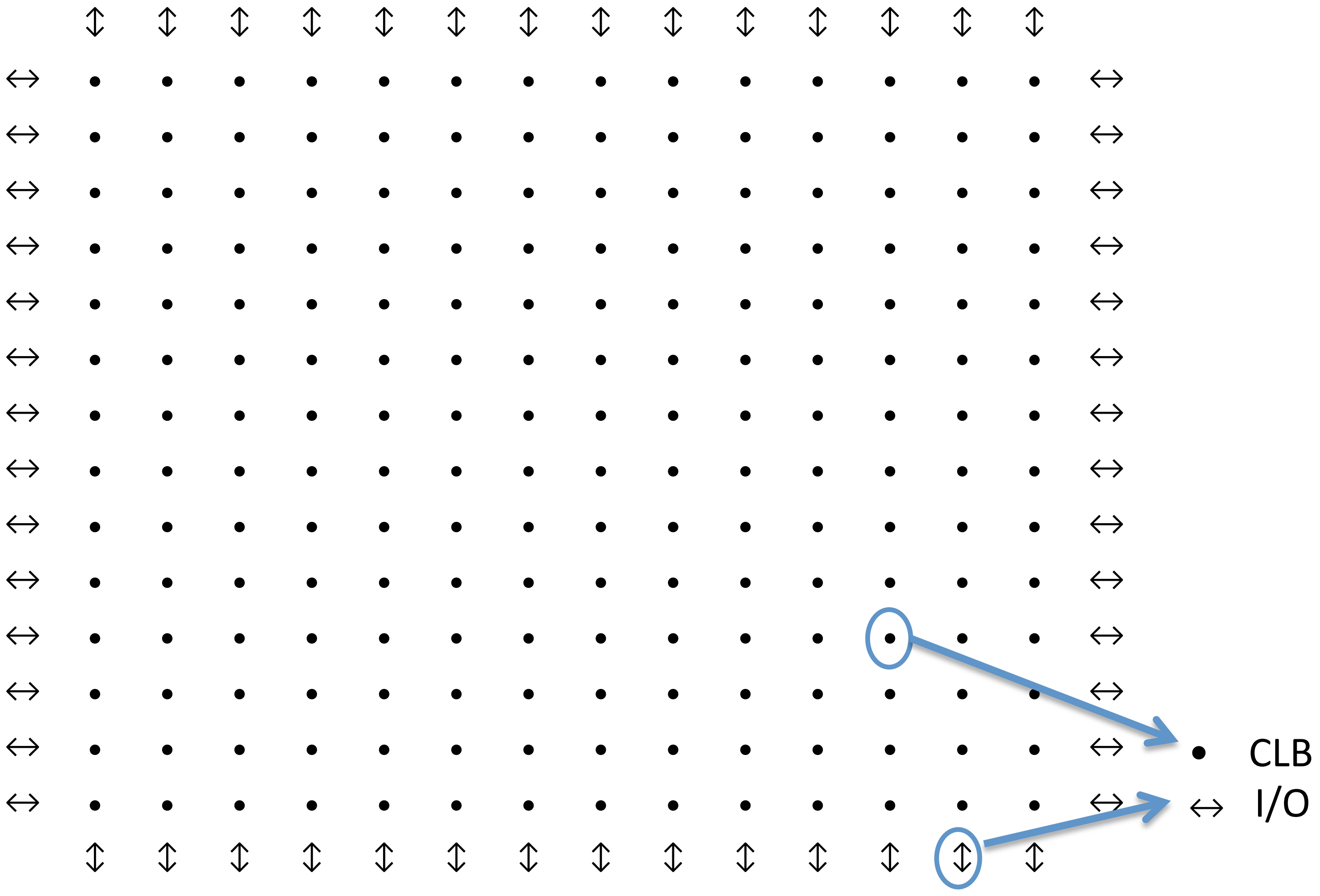
Source: Intel

- 2.2x speedup on 4 processors [Ludwin, FPGA '08]
- A distributed annealing algorithm implemented on a FPGA [Wrighton, FPGA '03]
  – Expanding neighbourhood region, self-hosted on a MPPA [Smecher, FPT '09]

Contributions

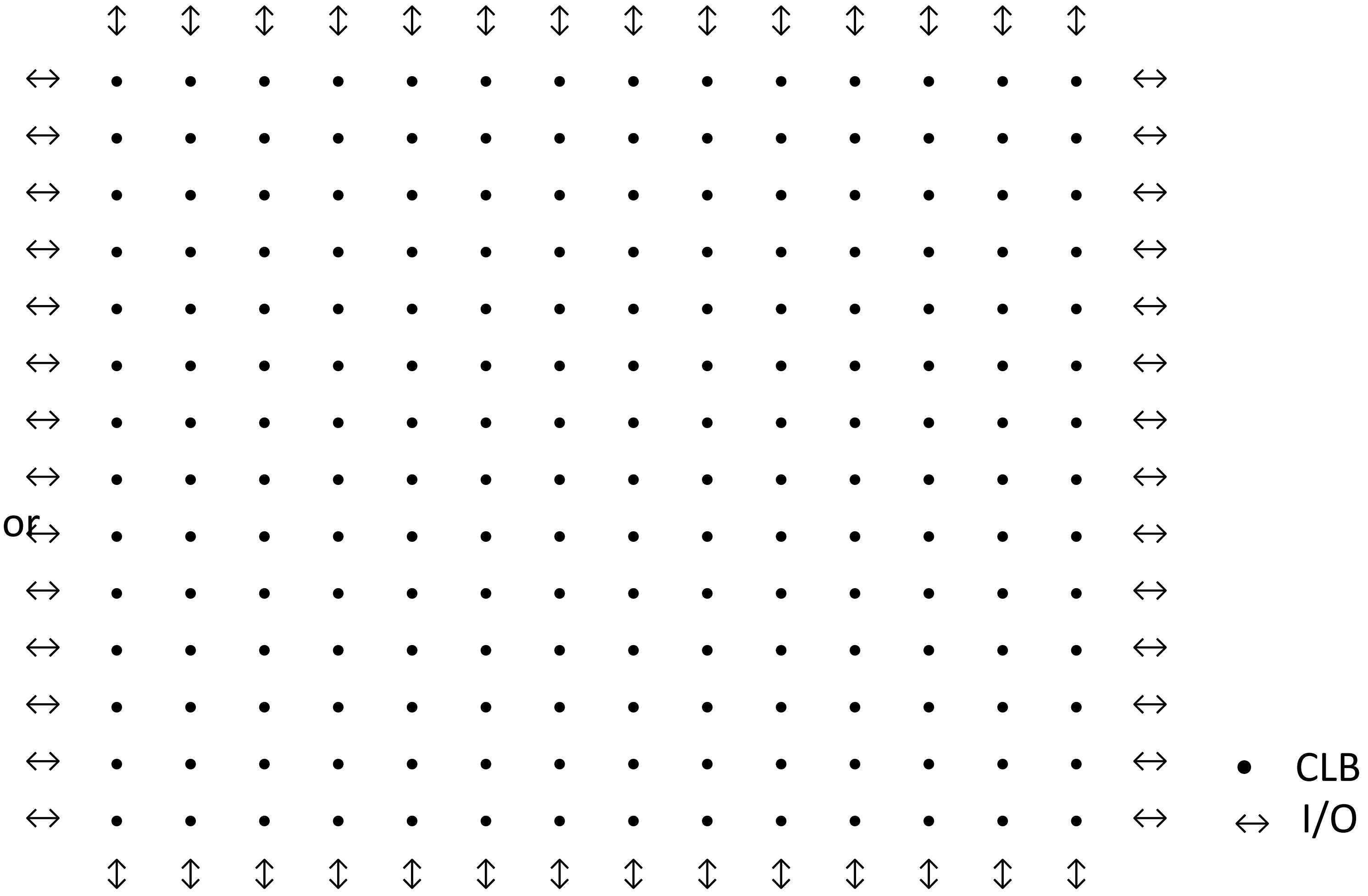
- Parallel Placement on Multicore CPUs
  - Implemented in VPR5.0.2 using pthreads
- Deterministic
  - Result reproducible when same # of threads used
- Scalability
  - More threads→ more speedup
  - QoR is conserved
- Timing-Driven

FPGA



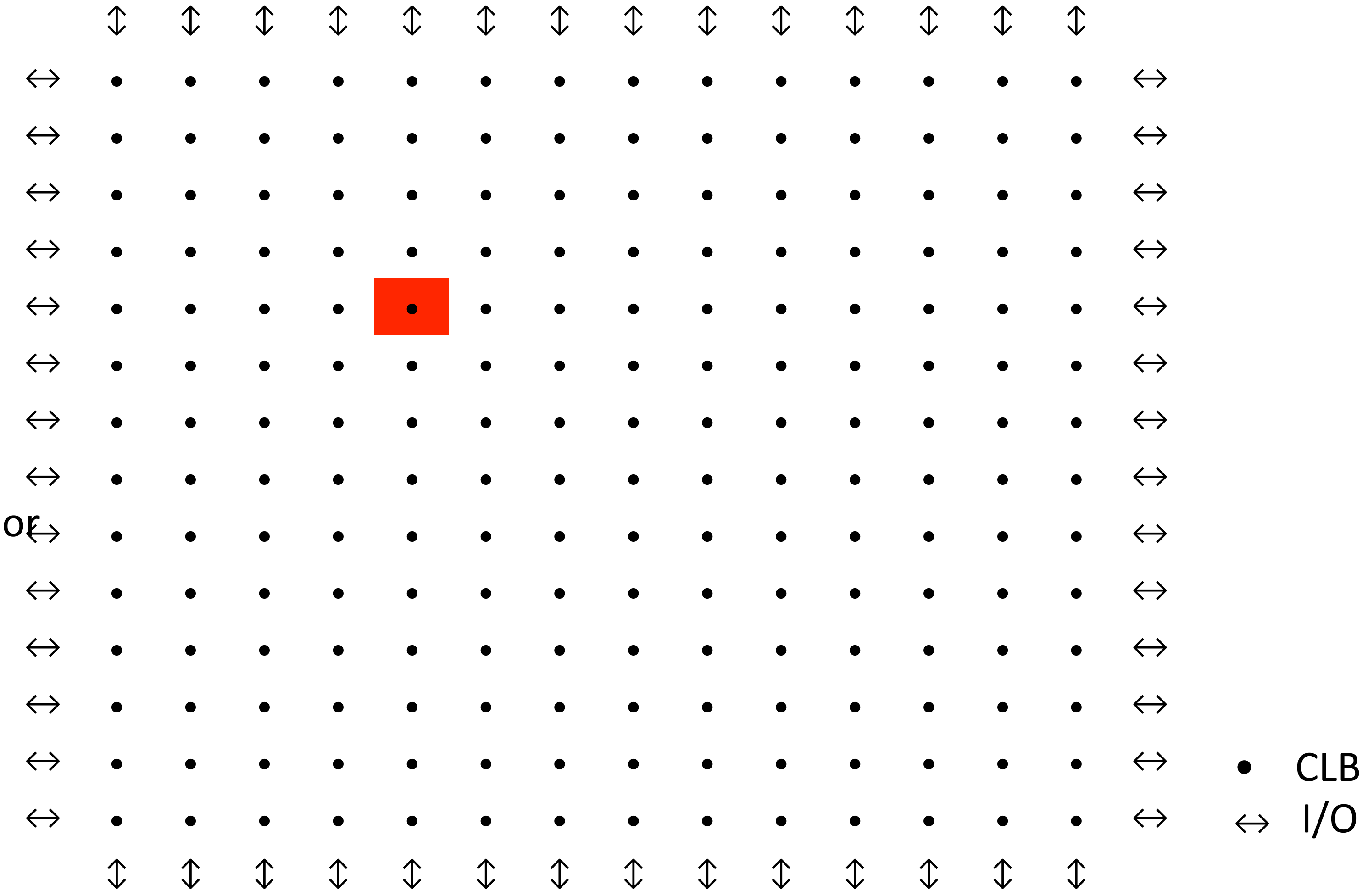
The algorithm

1. Identify block
2. Rand[0,100] > THRESHOLD?
3. Generate a random neighbor to swap with
4. Evaluate swap
5. Move to next block



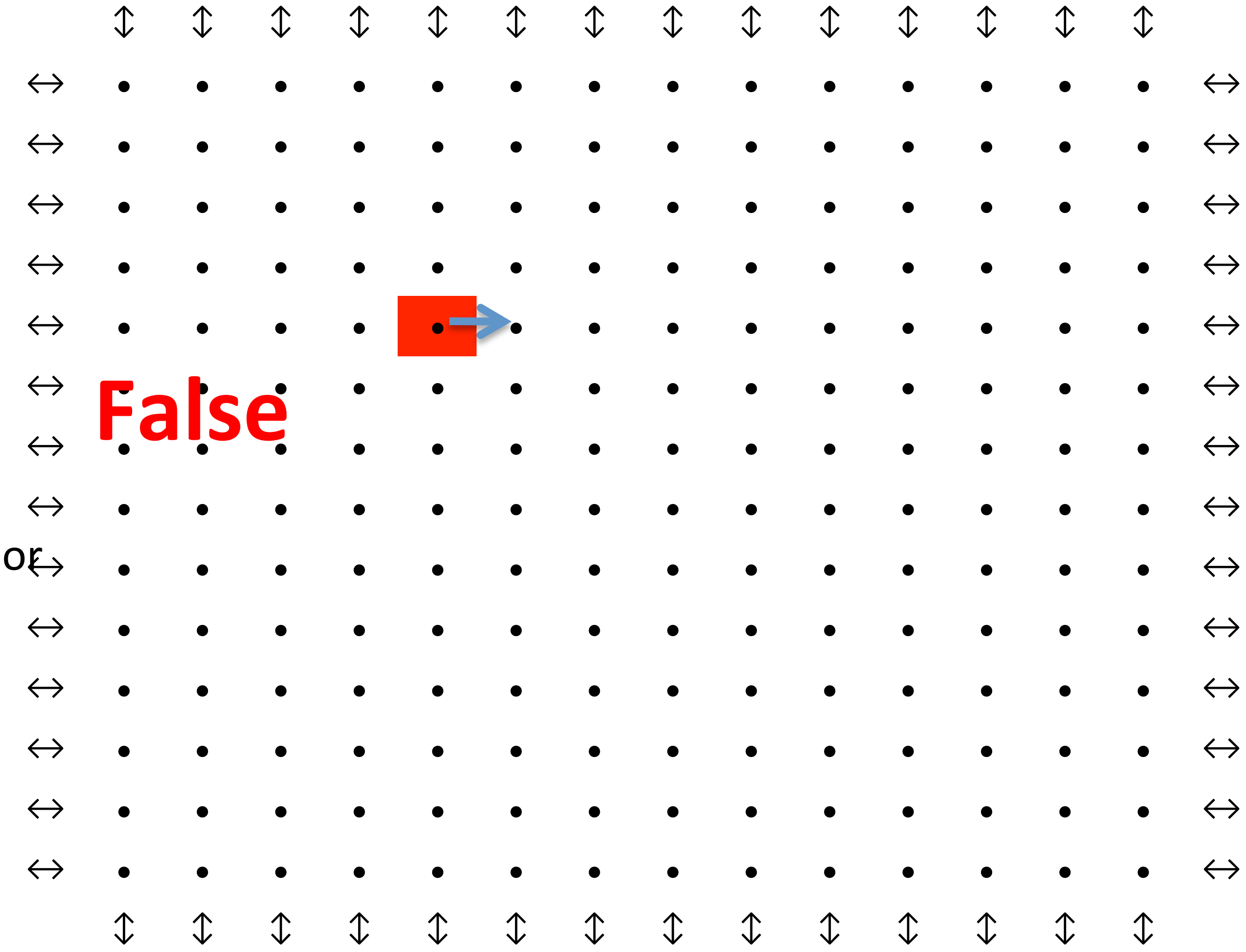
The algorithm

1. **Identify block**
2. Rand[0,100] > THRESHOLD?
3. Generate a random neighbor to swap with
4. Evaluate swap
5. Move to next block



The algorithm

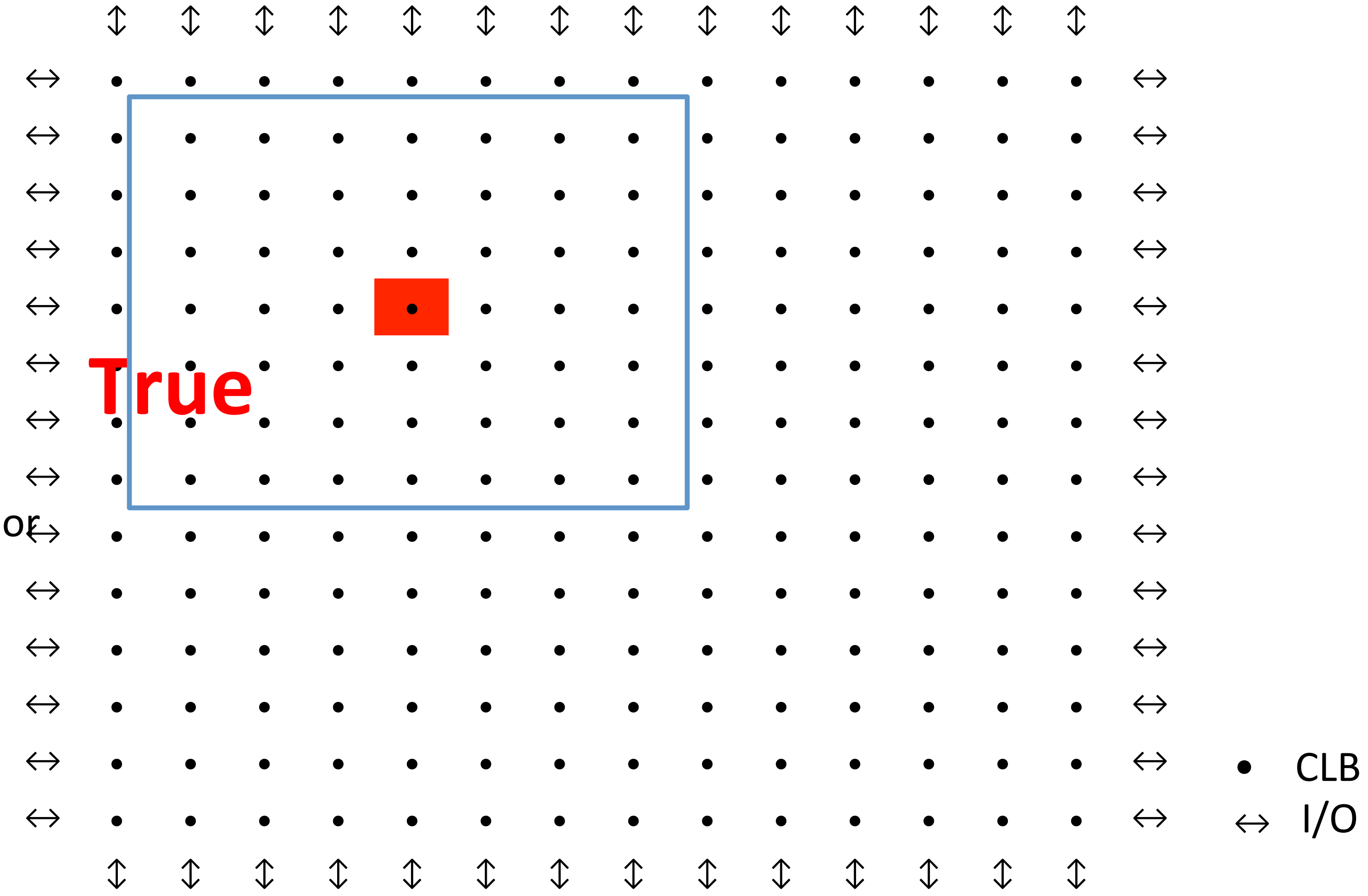
1. Identify block
2. **Rand[100] > THR OLD?**
3. Gene a rand eighbor to s ith
4. Eval swap
5. Move to next block



- • CLB
- ↔ I/O

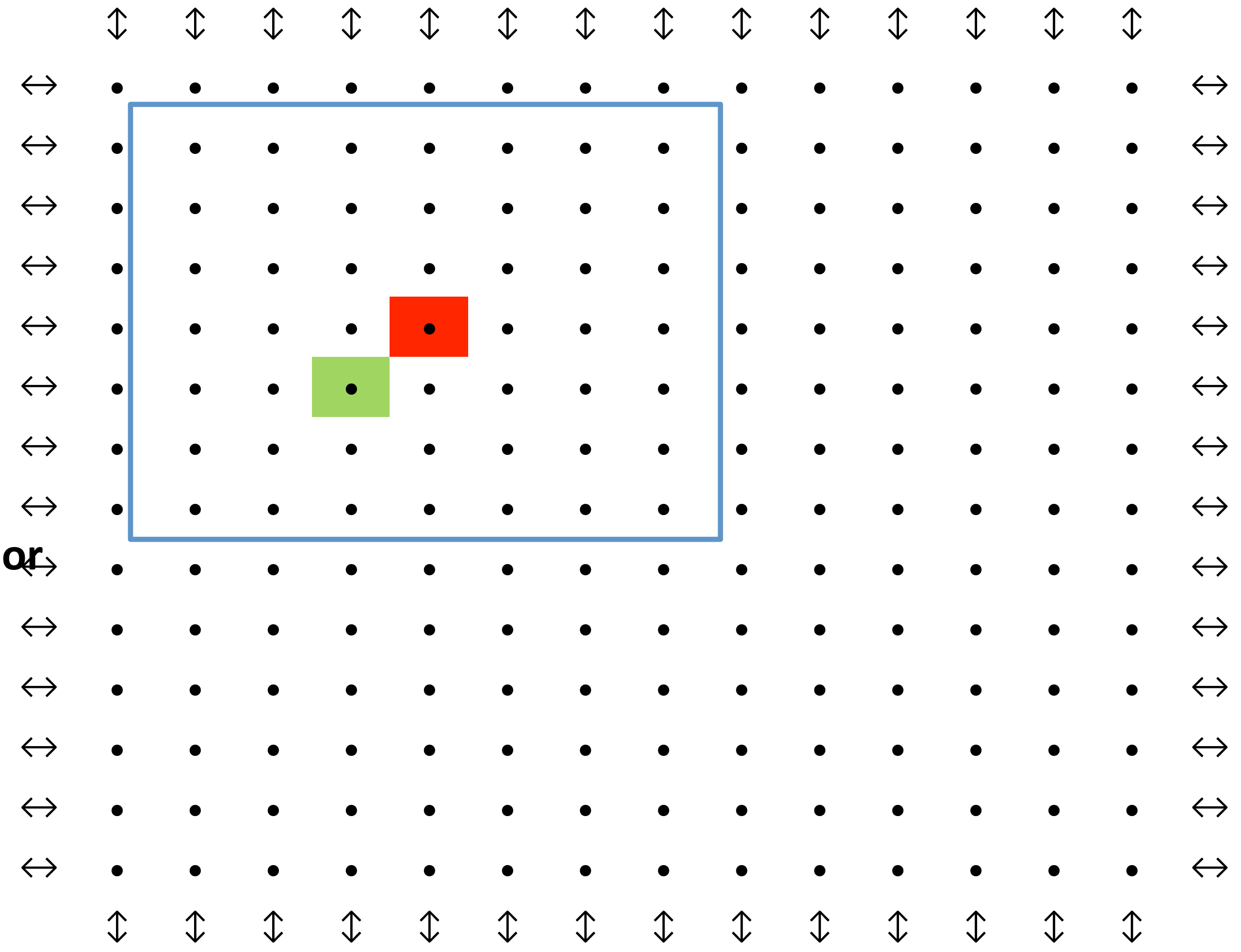
The algorithm

1. Identify block
2. **Rand[0,100] > THRESHOLD?**
3. Generate a random neighbor to swap with
4. Evaluate swap
5. Move to next block



The algorithm

1. Identify block
2. Rand[0,100] > THRESHOLD?
3. **Generate a random neighbor to swap with**
4. Evaluate swap
5. Move to next block

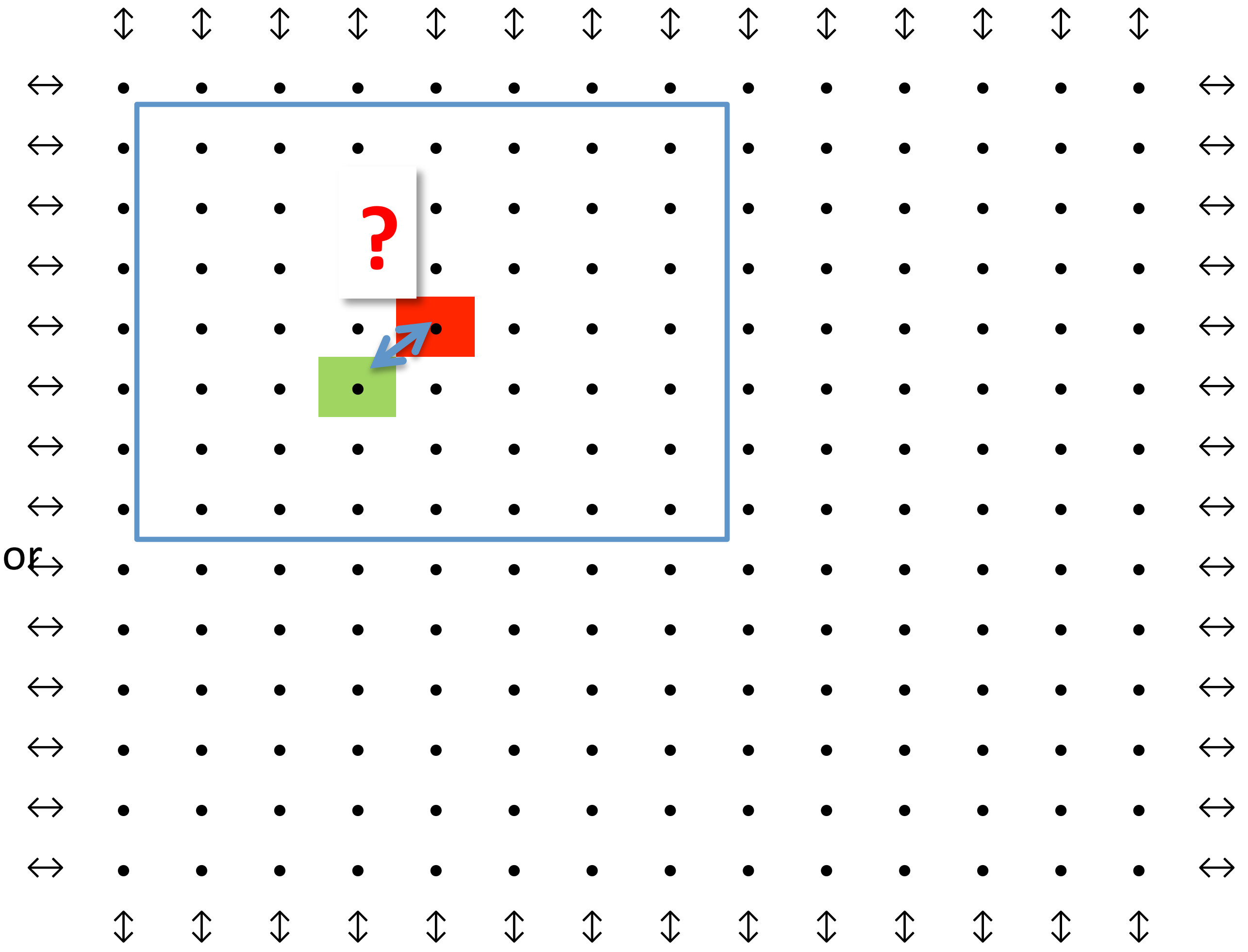


● CLB

↔ I/O

The algorithm

1. Identify block
2. Rand[0,100] > THRESHOLD?
3. Generate a random neighbor to swap with
4. **Evaluate swap**
5. Move to next block

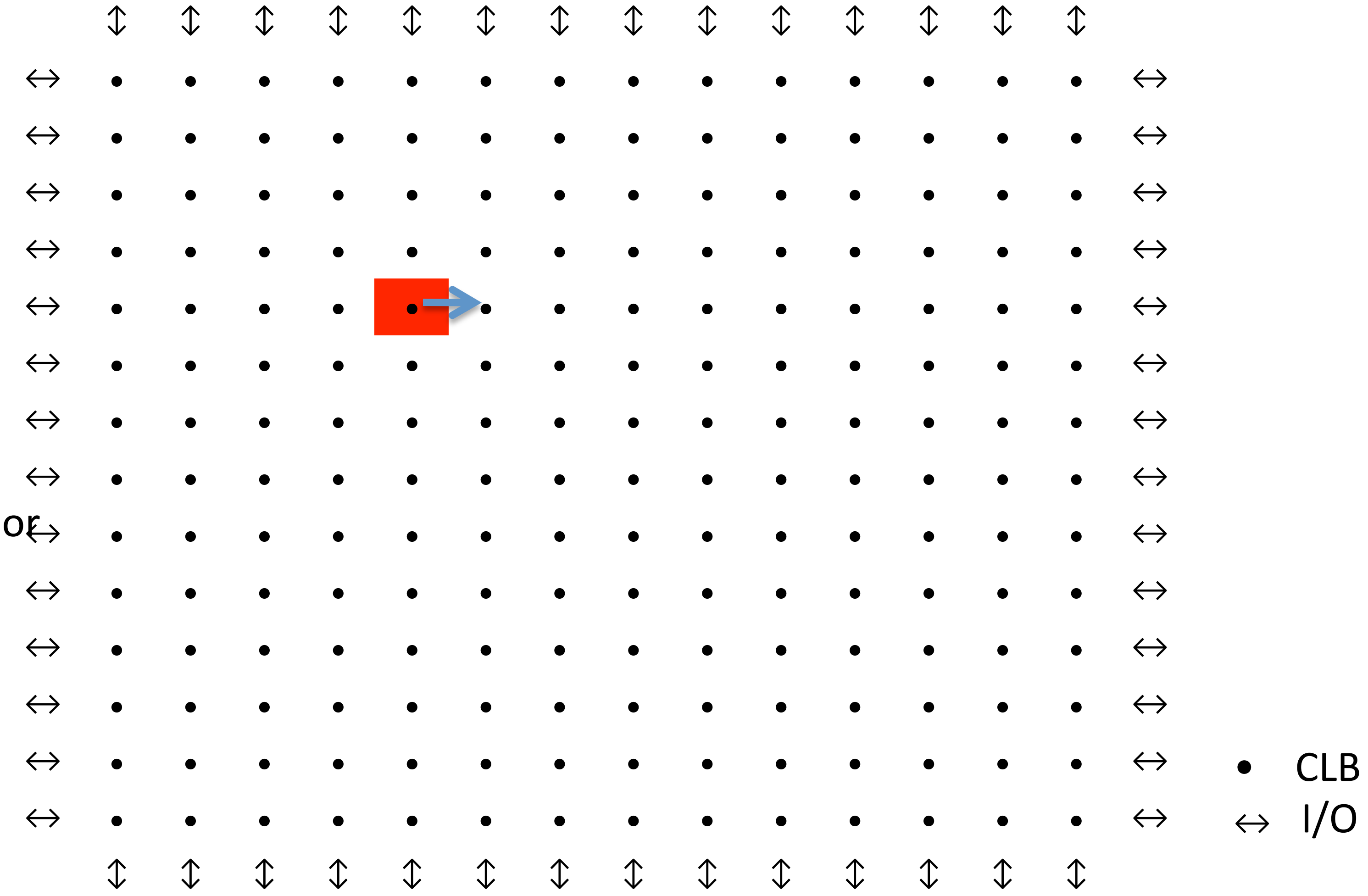


● CLB

↔ I/O

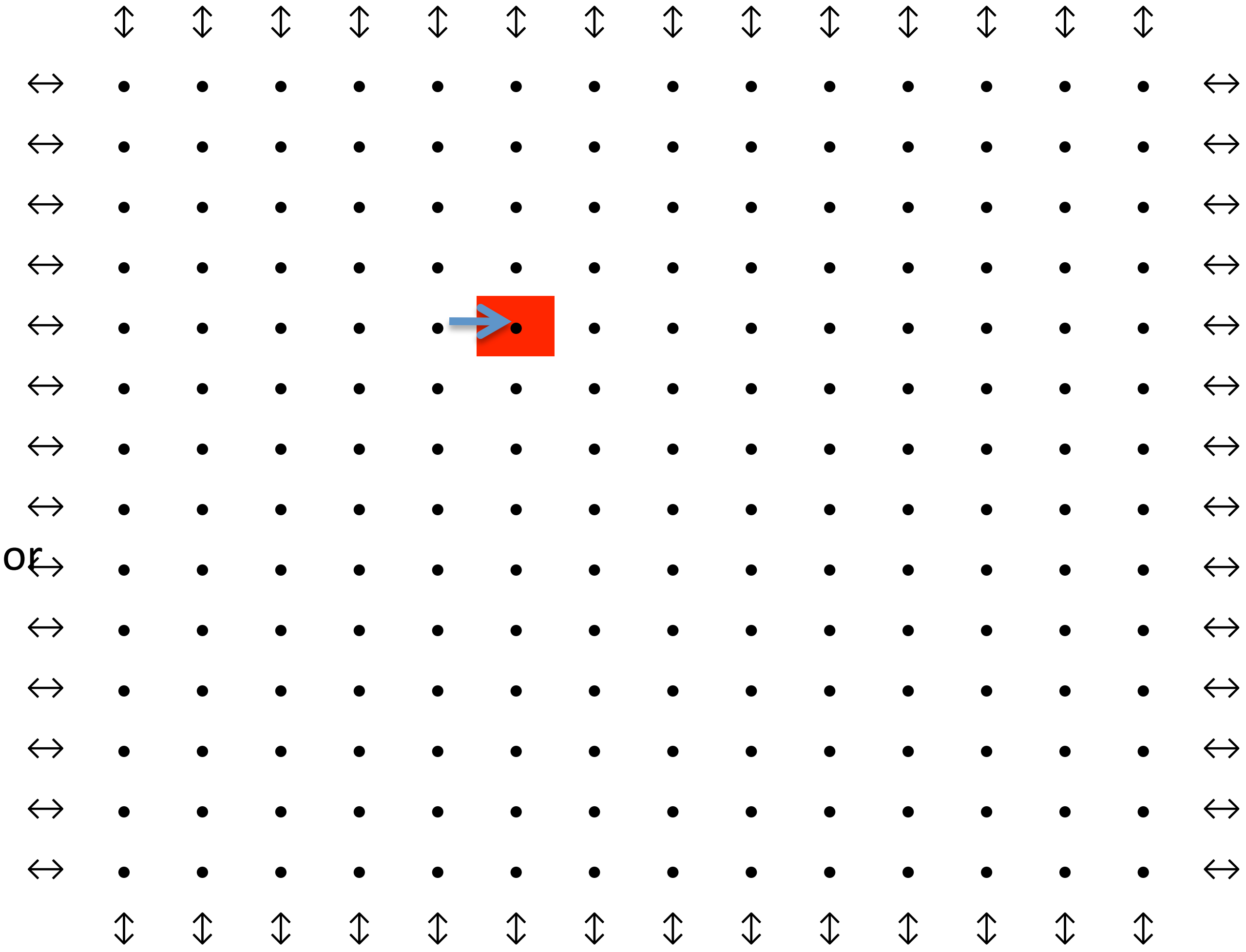
The algorithm

1. Identify block
2. Rand[0,100] > THRESHOLD?
3. Generate a random neighbor to swap with
4. Evaluate swap
5. **Move to next block**



The algorithm

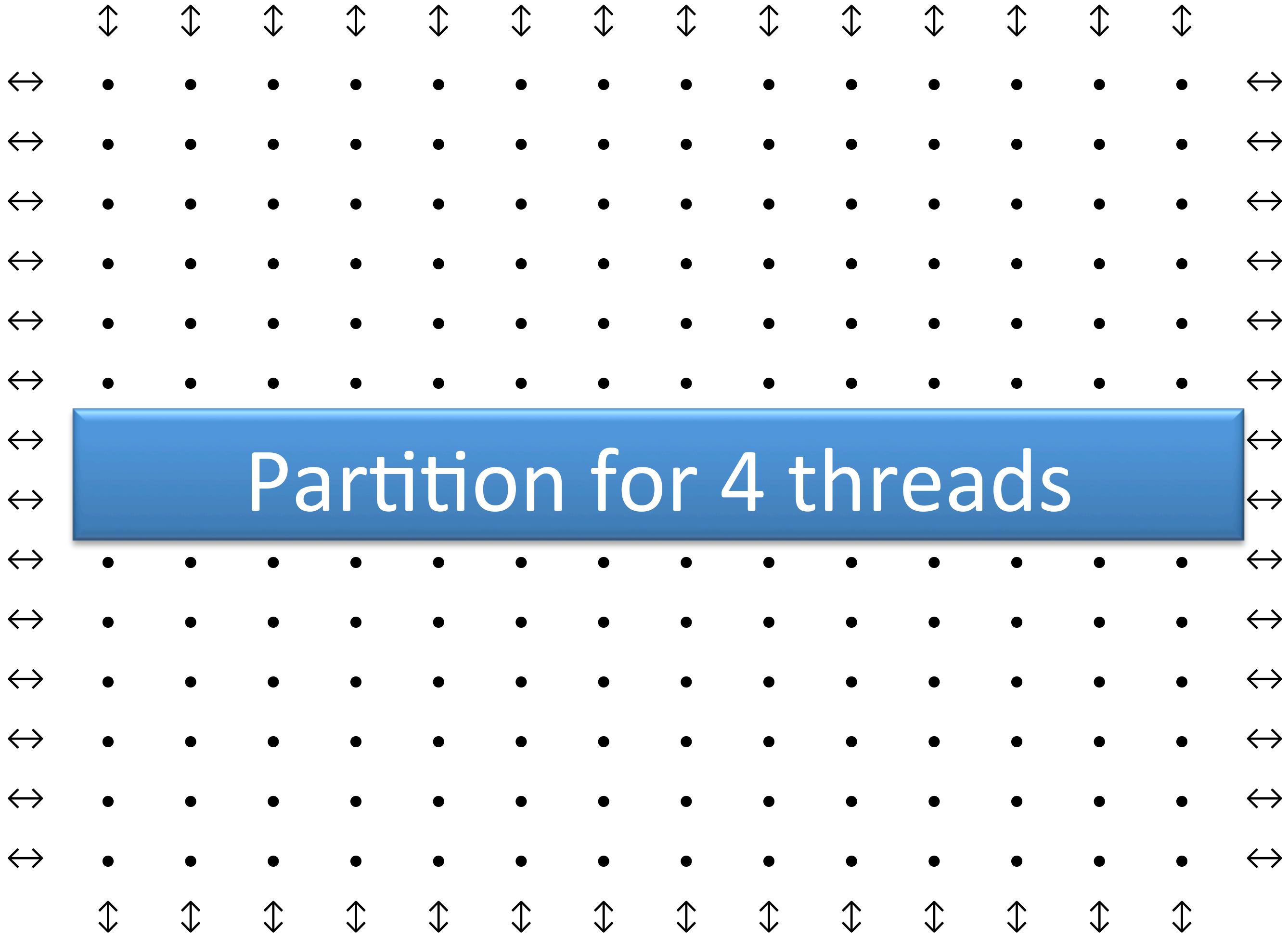
1. Identify block
2. Rand 100] > THR OLD?
3. Gene e a rand neighbor to sw with
4. Evalu e swap
5. **Move to next block**



● CLB

↔ I/O

The algorithm – Grid Partition

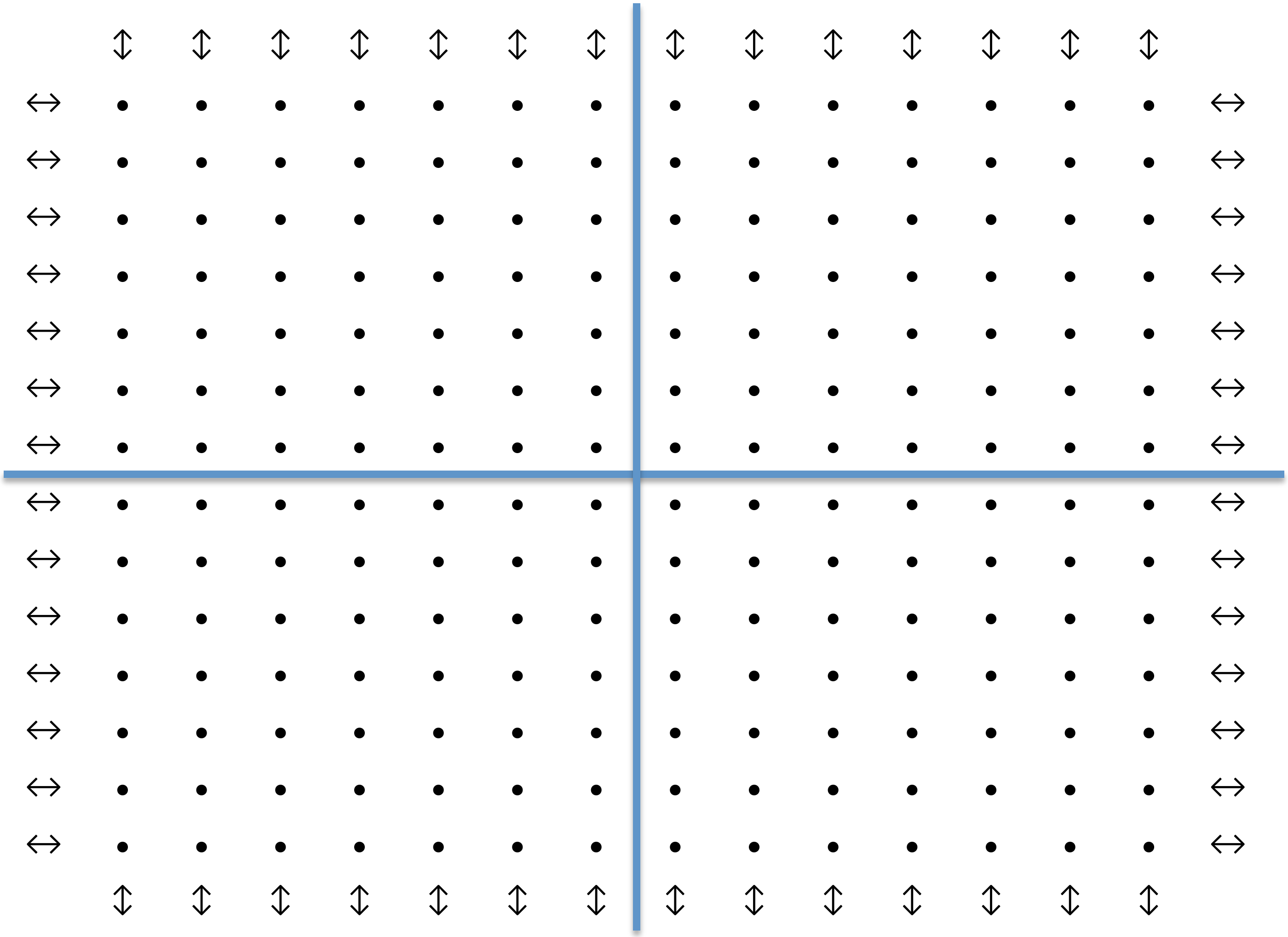


Partition for 4 threads

● CLB

↔ I/O

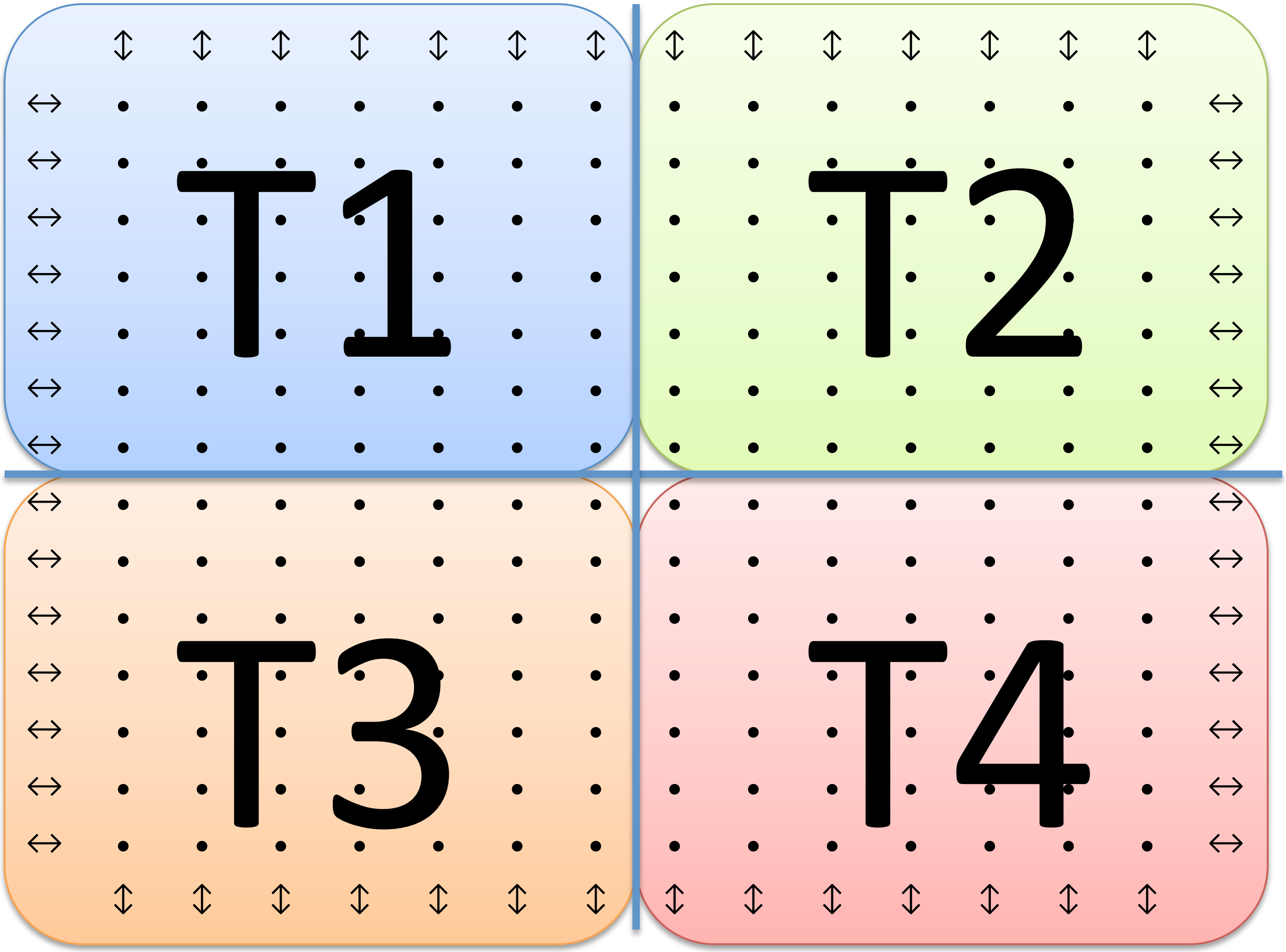
The algorithm – Grid Partition



● CLB

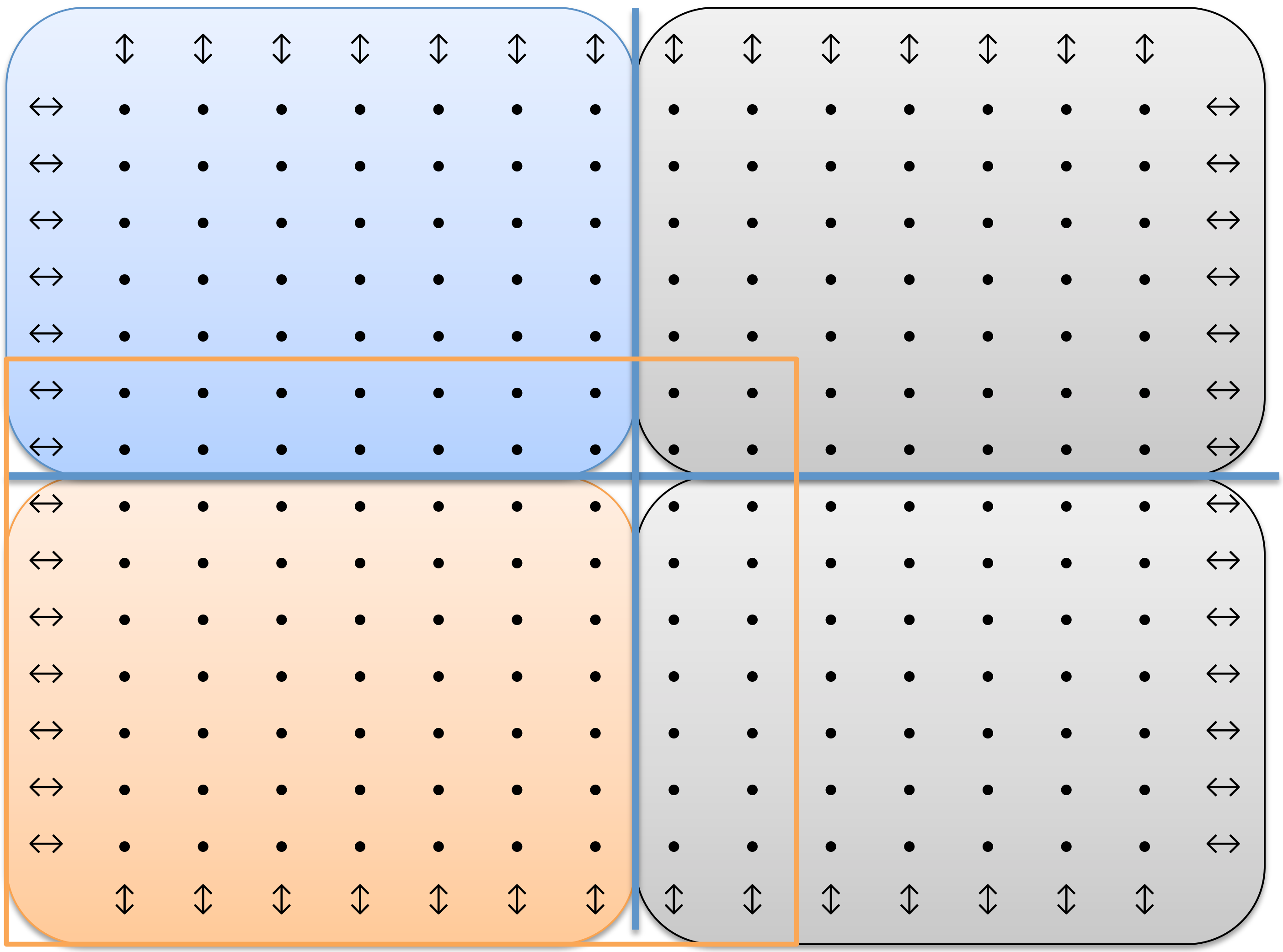
↔ I/O

The algorithm – Grid Partition



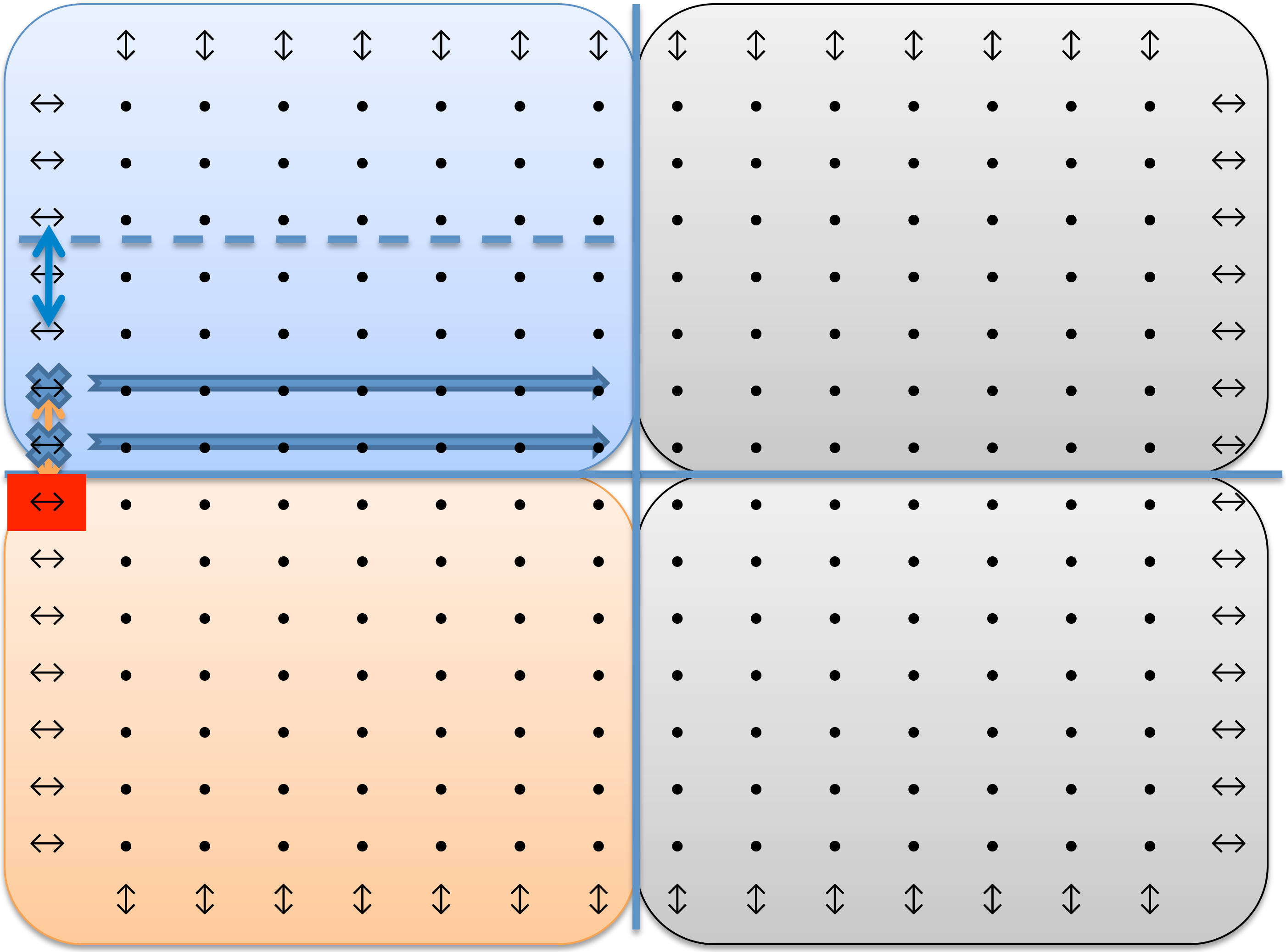
● CLB
↔ I/O

The algorithm – Grid Partition

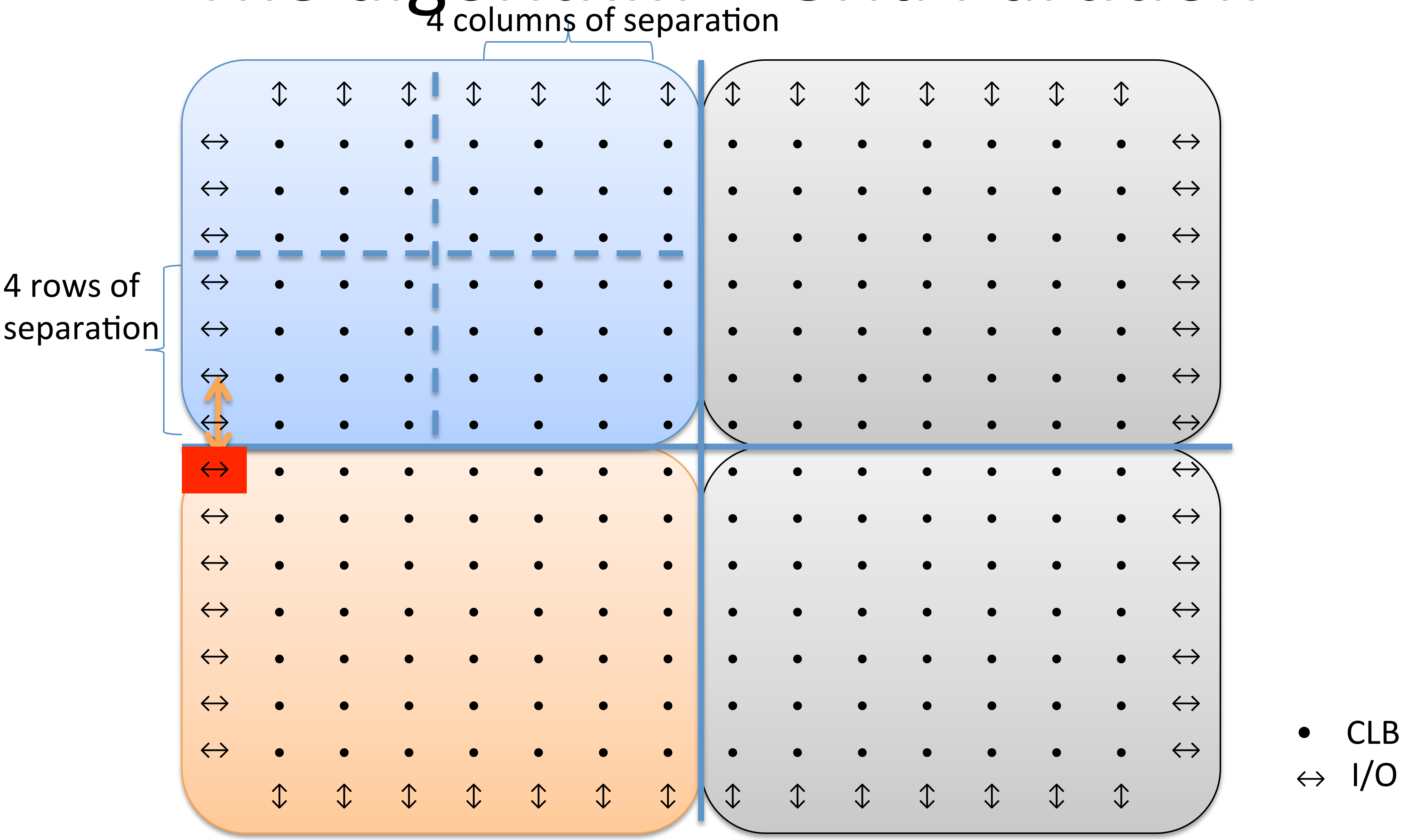


● CLB
↔ I/O

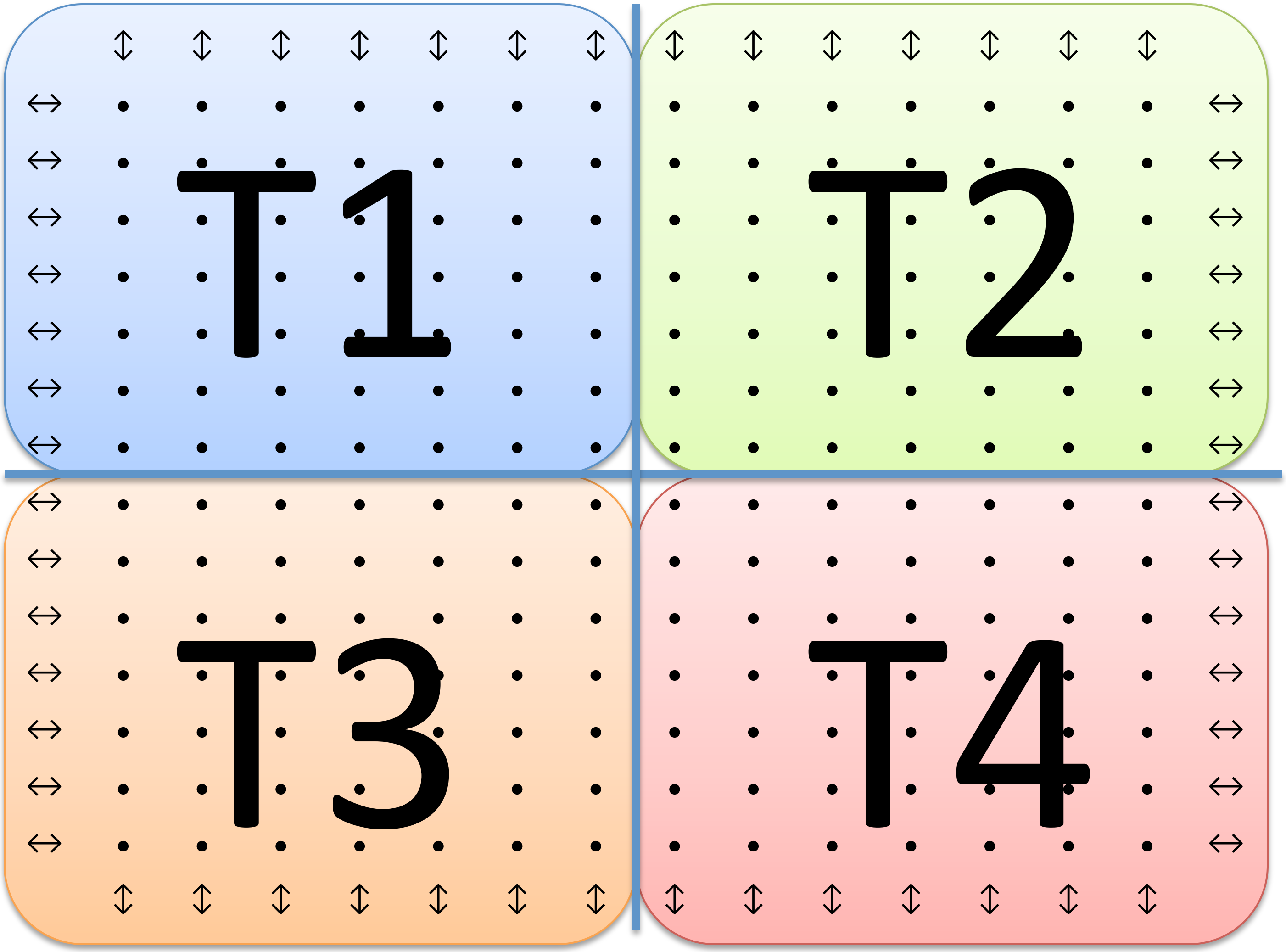
The algorithm – Grid Partition



The algorithm – Grid Partition

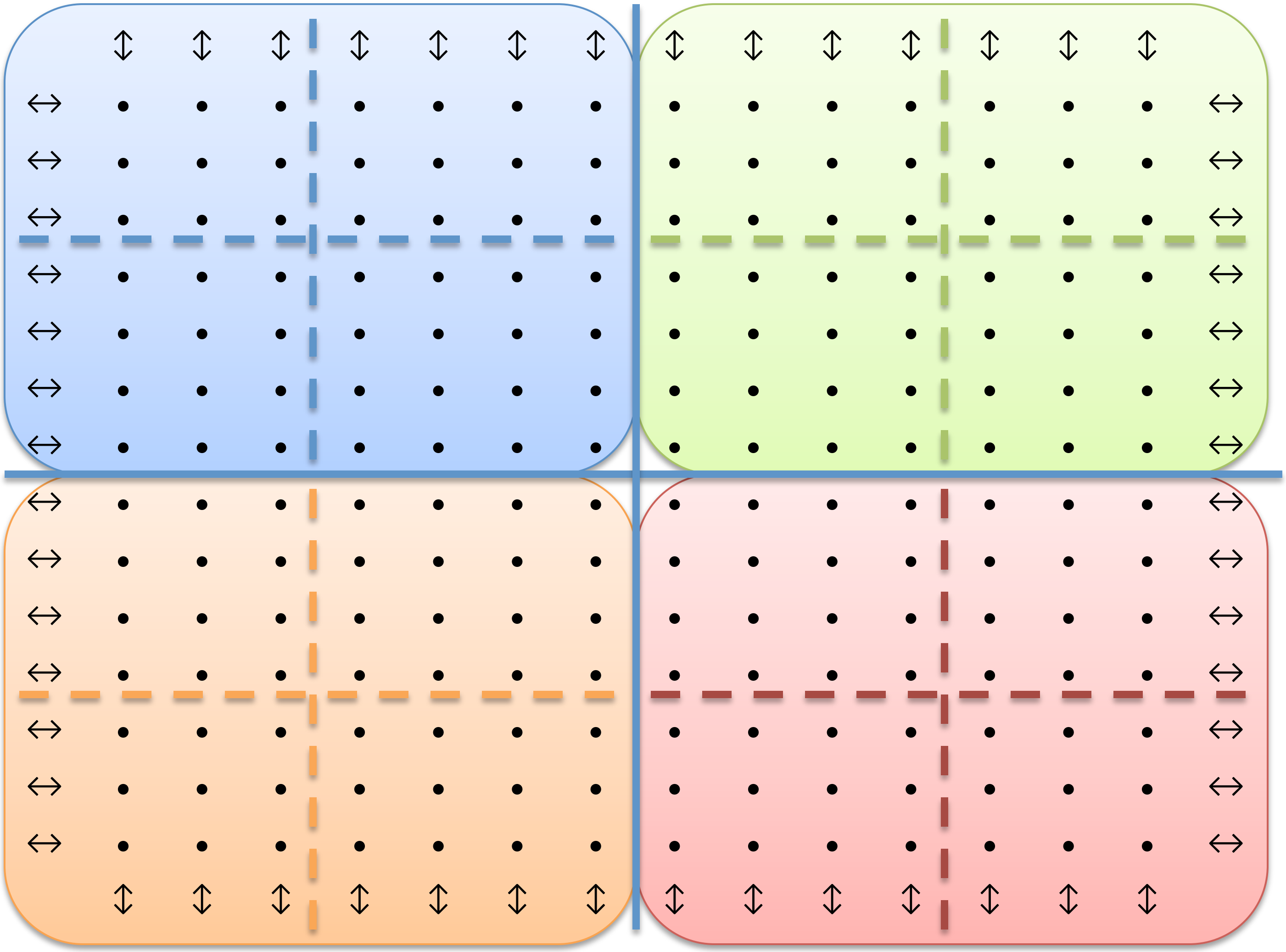


Putting it together



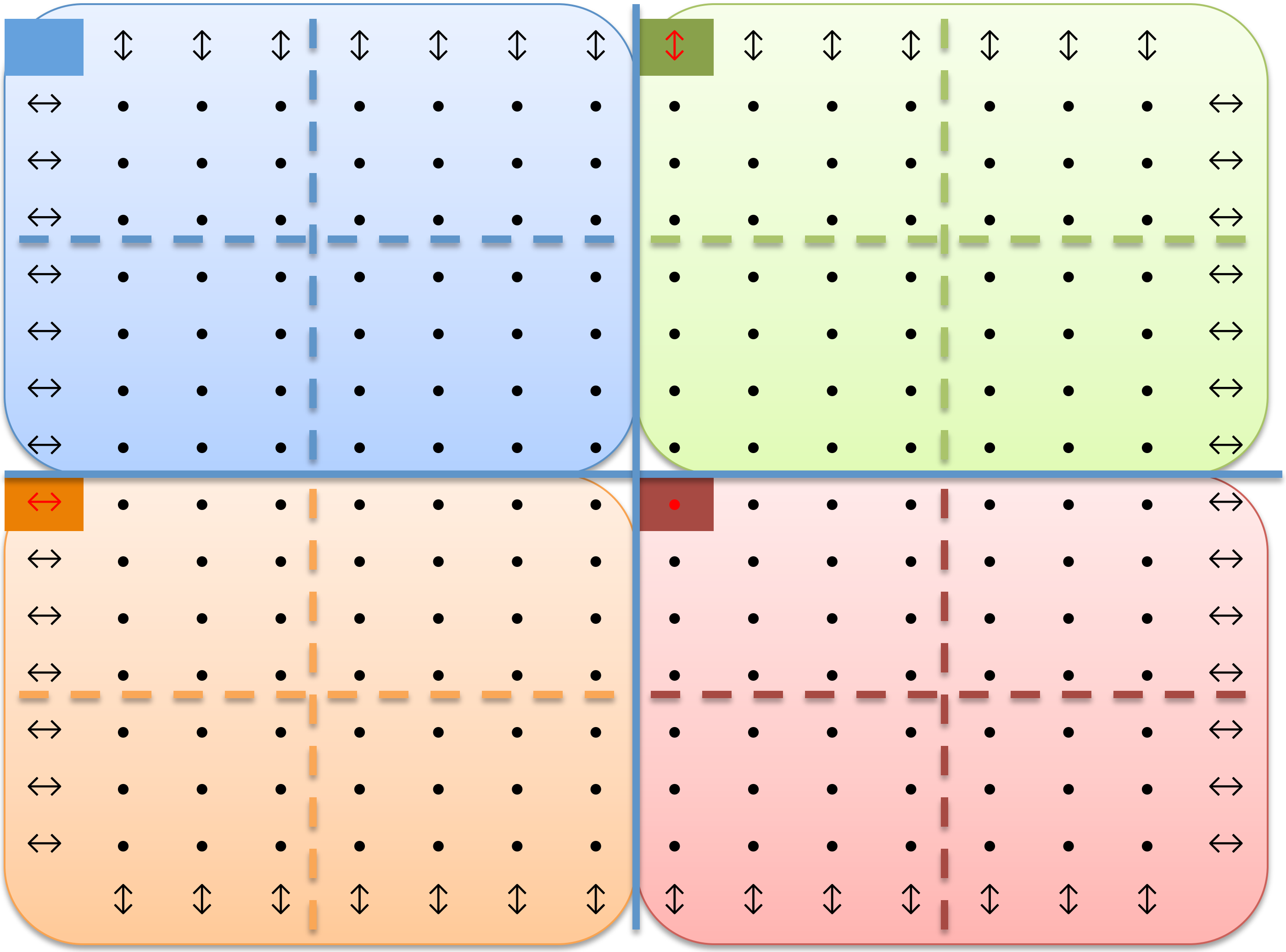
● CLB
↔ I/O

Putting it together



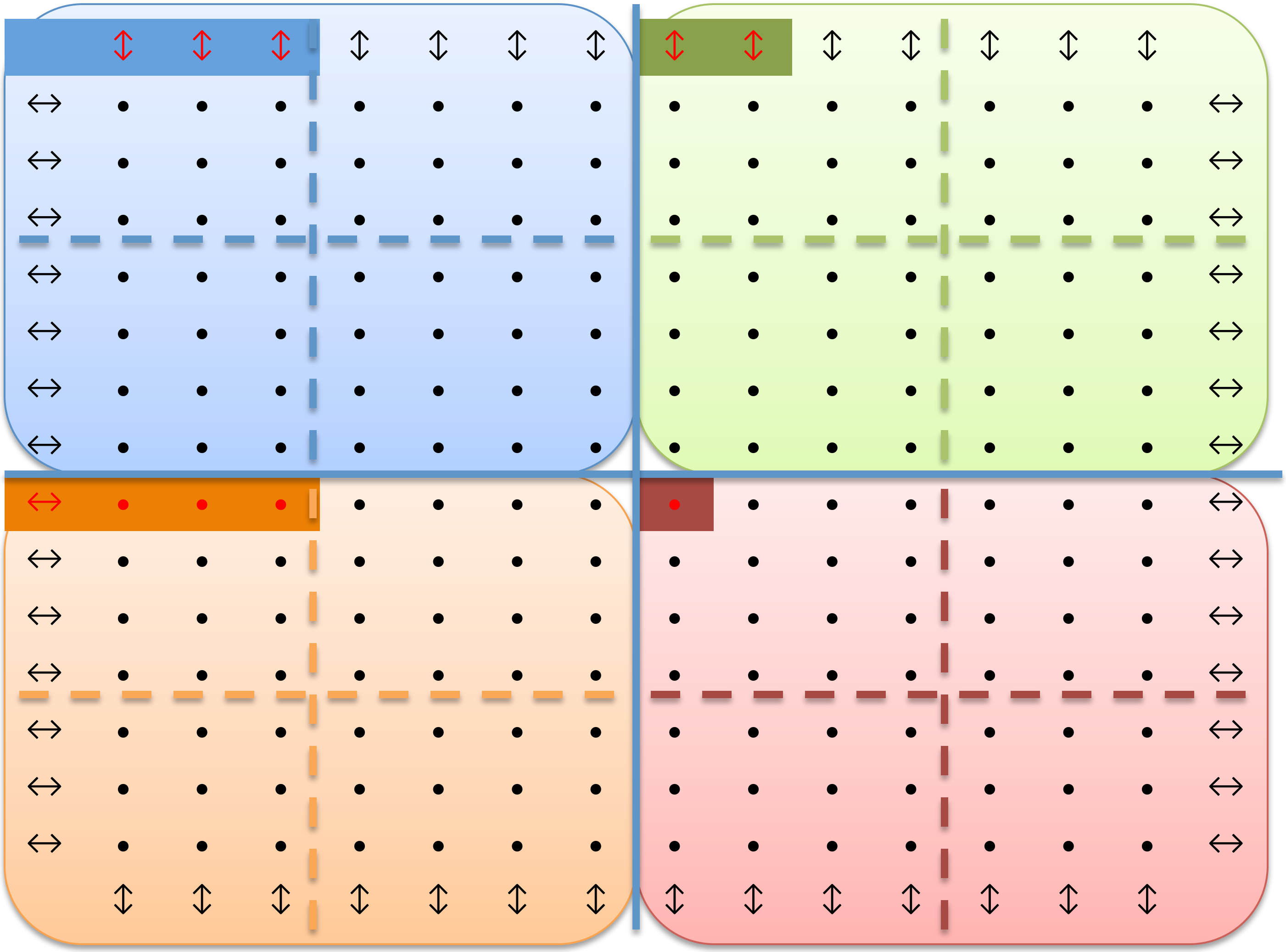
- ● CLB
- ↔ I/O

Putting it together

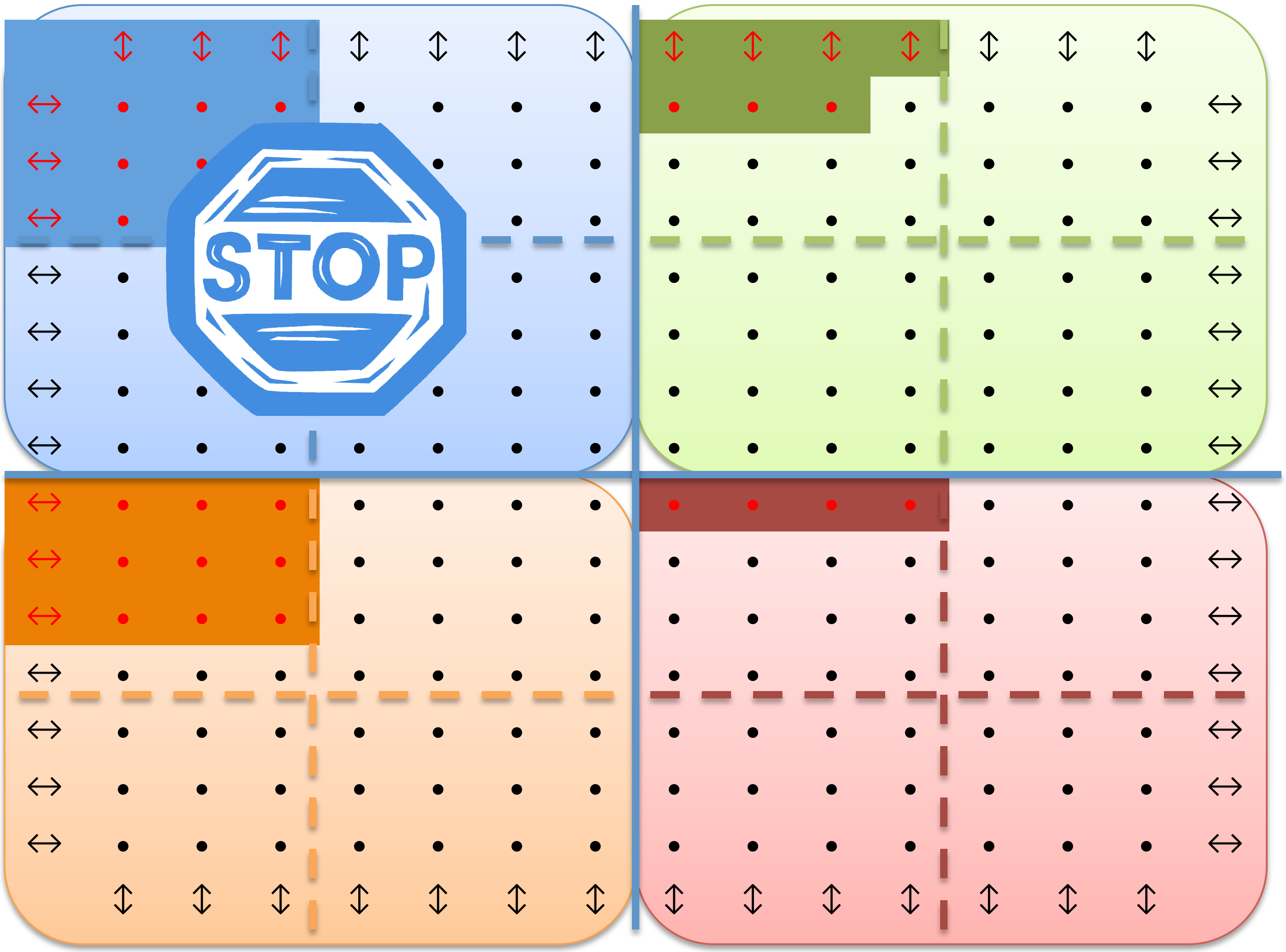


- ● CLB
- ↔ I/O

Putting it together



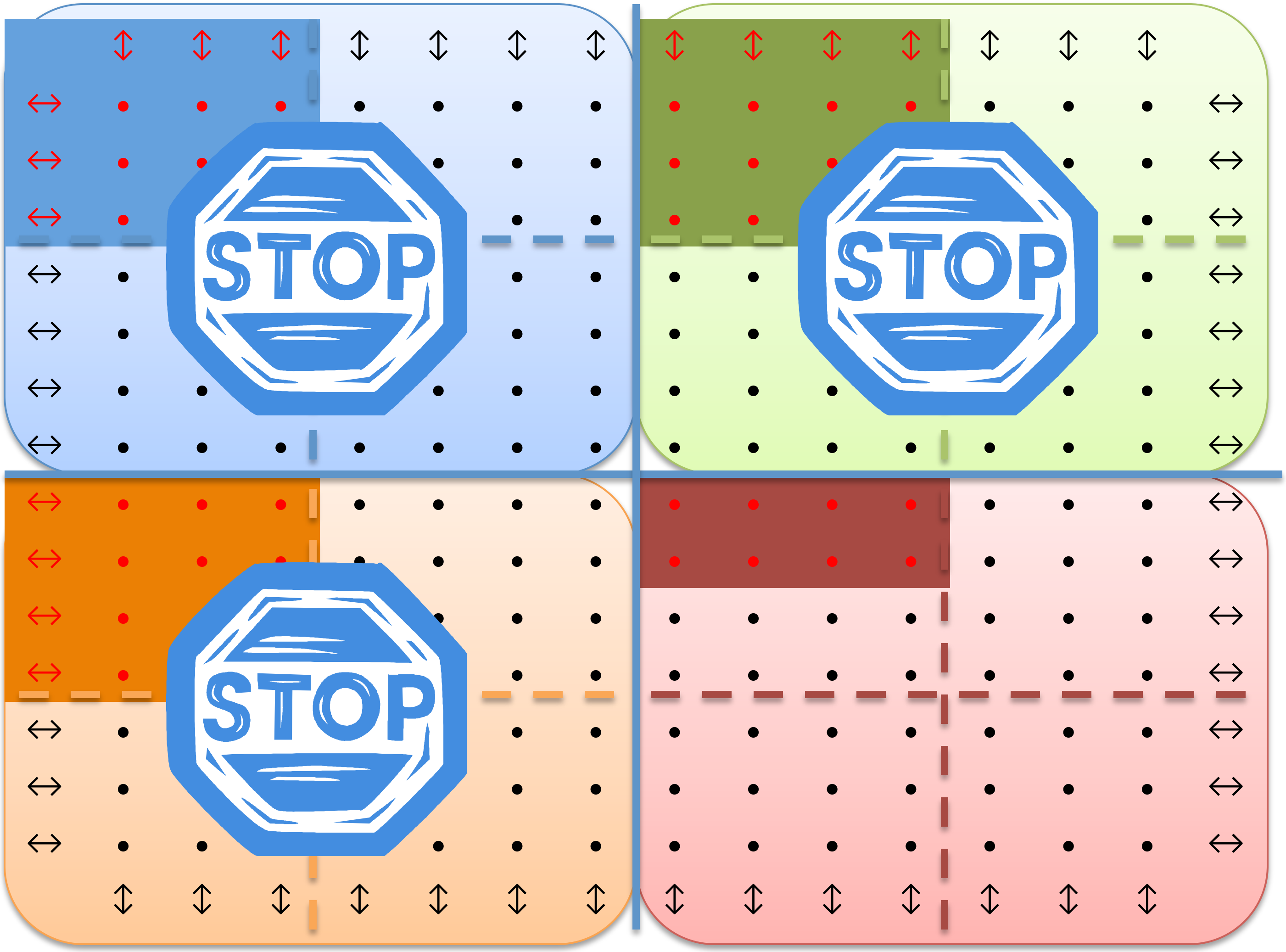
Putting it together



● CLB

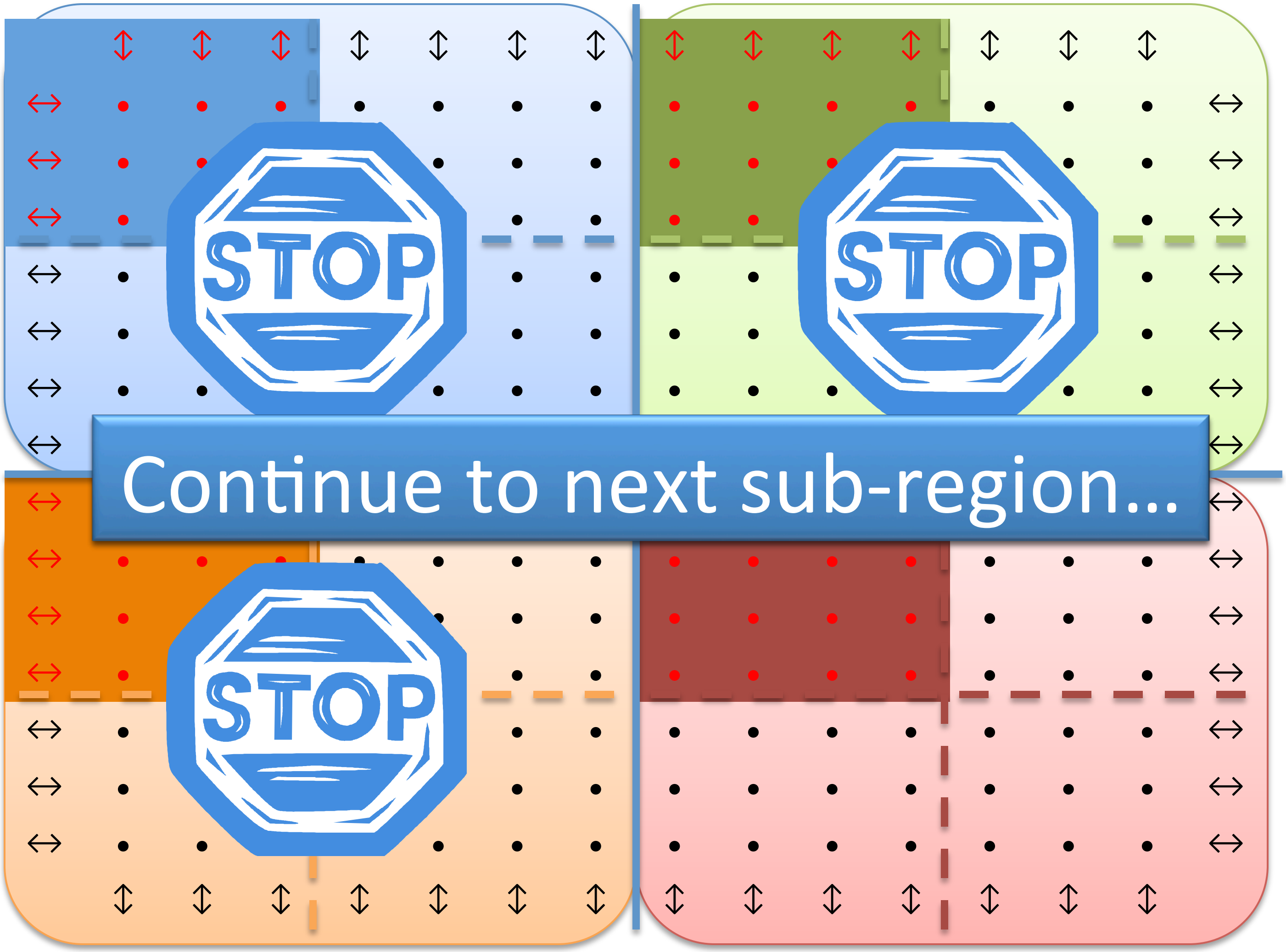
↔ I/O

Putting it together



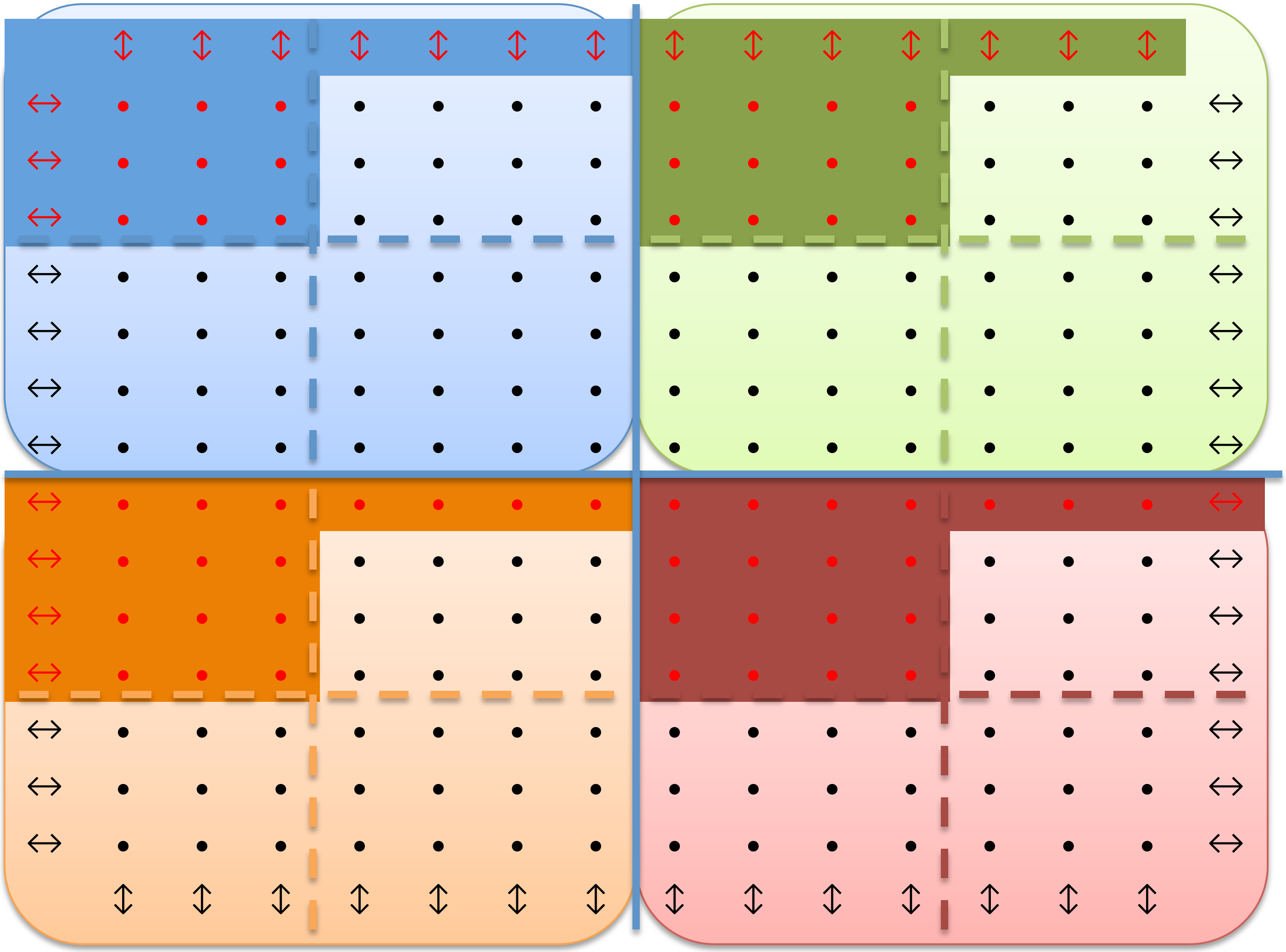
● CLB
↔ I/O

Putting it together

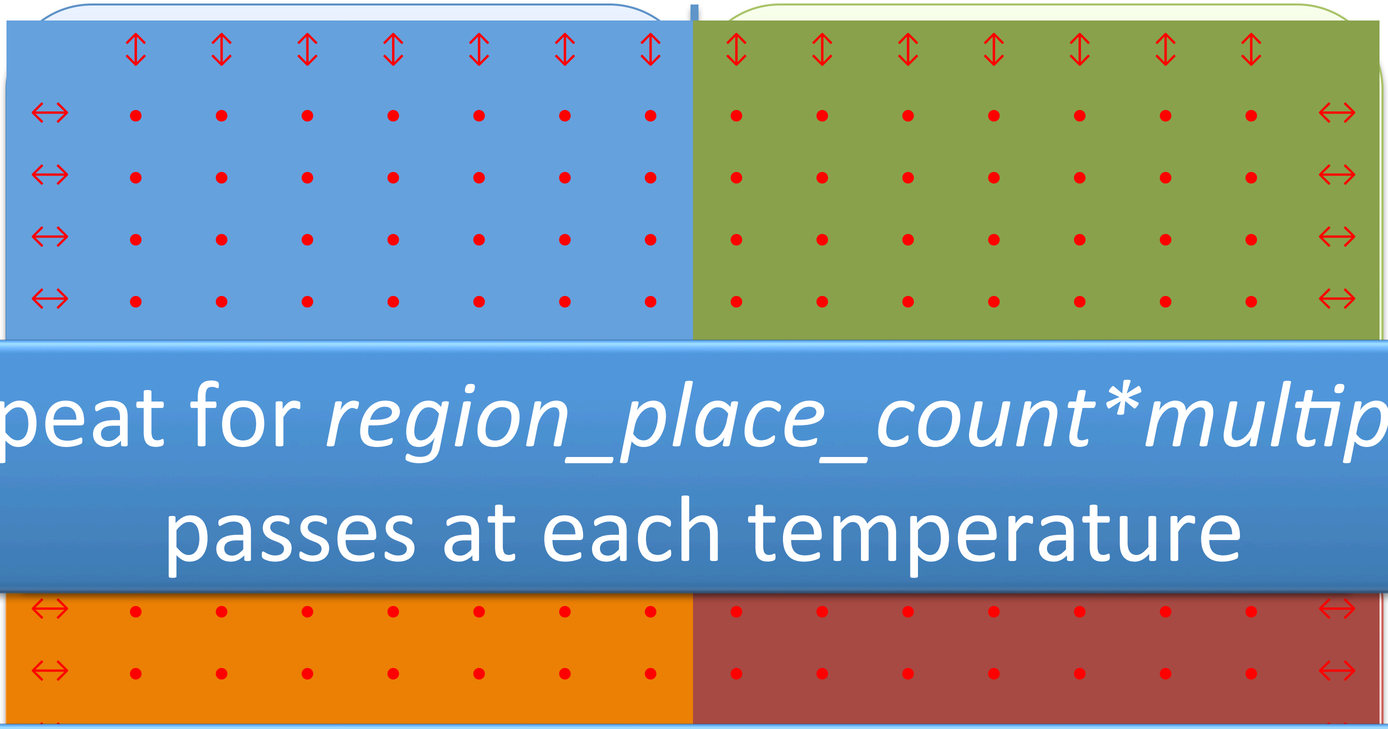


- ● CLB
- ↔ I/O

Putting it together



Putting it together



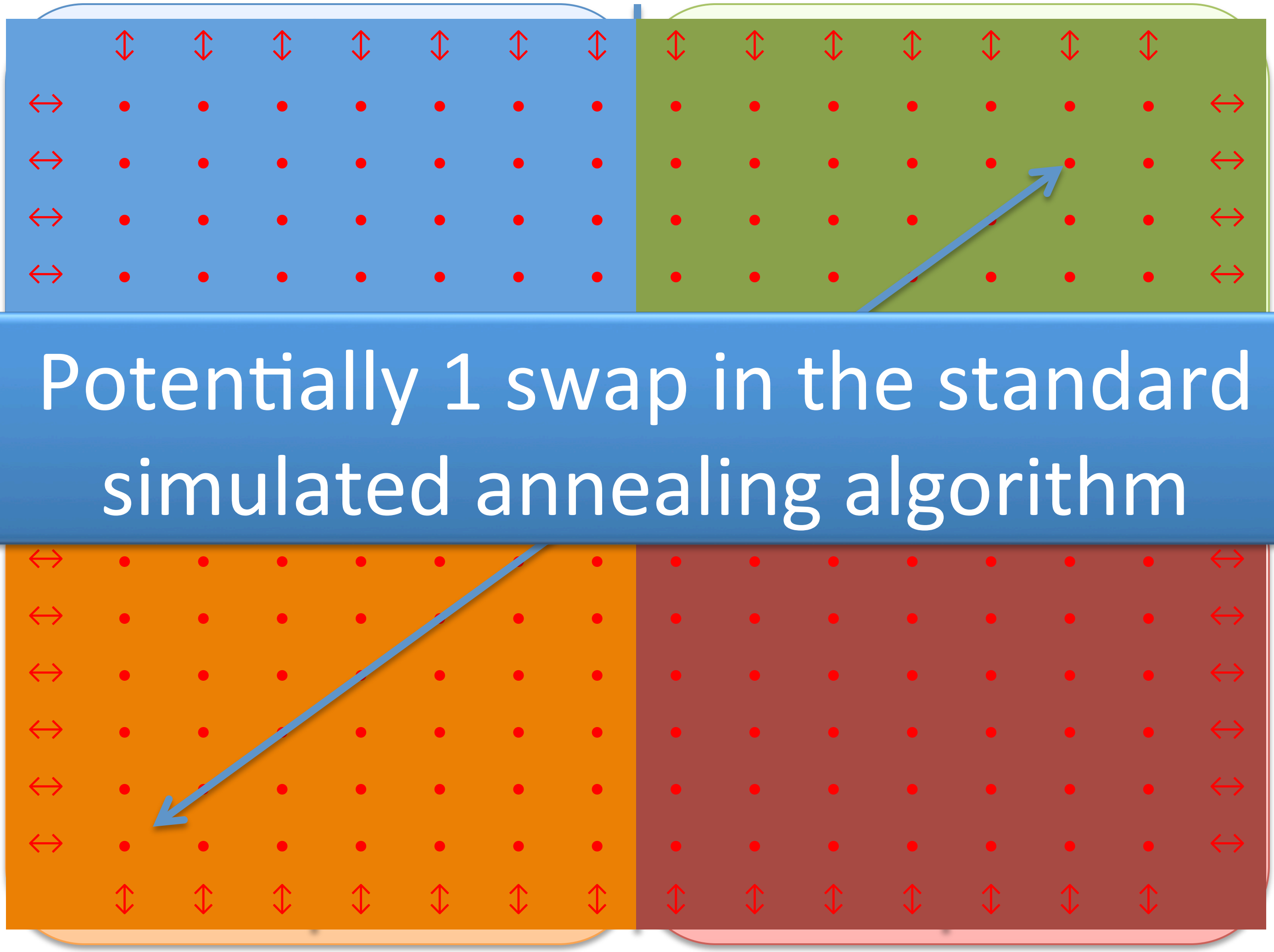
Repeat for *region_place_count*multiplier* passes at each temperature

sample region_place_count: 90

multiplier at high T.: 1

multiplier at low T.: 1/20

Putting it together

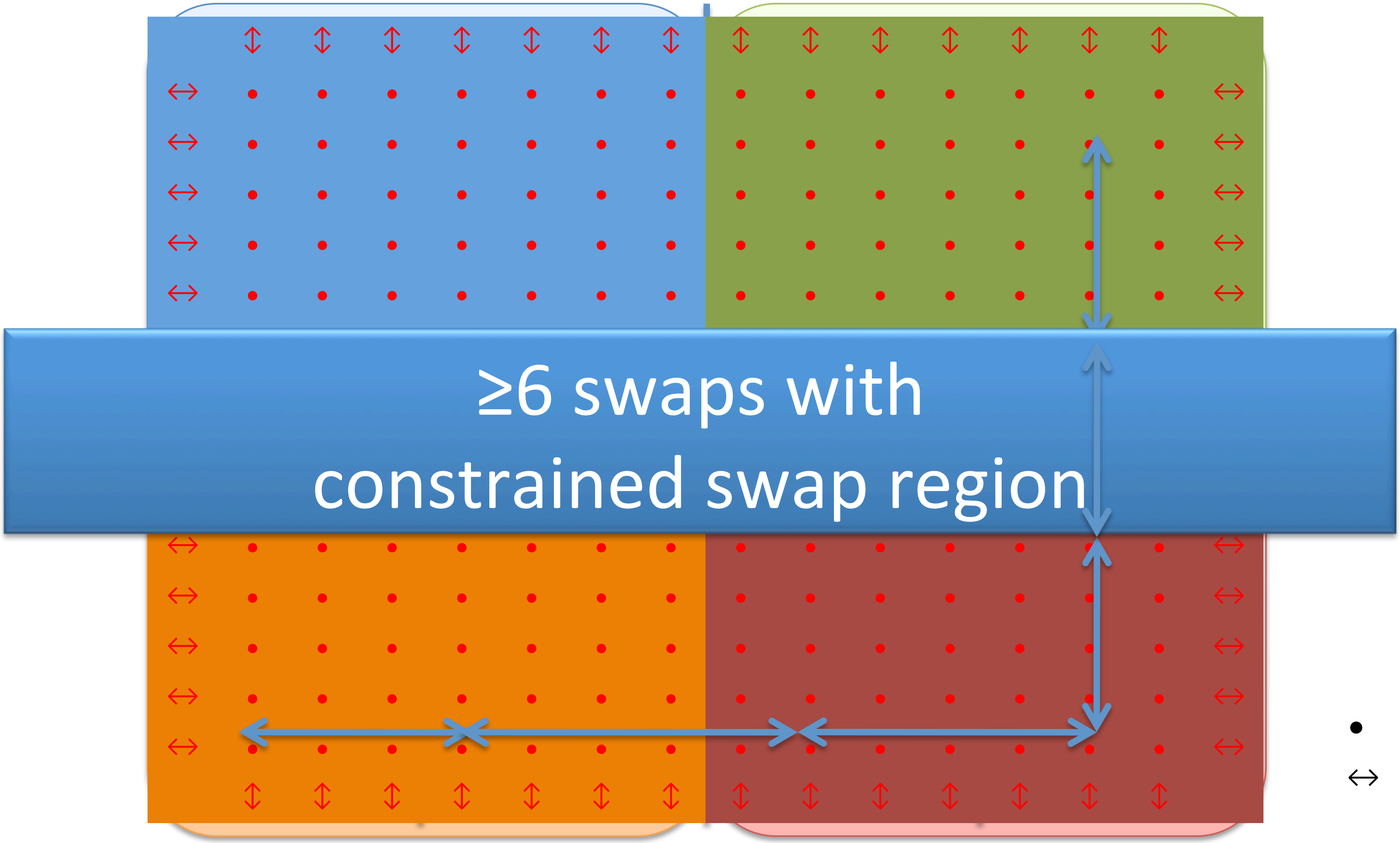


Potentially 1 swap in the standard simulated annealing algorithm

- CLB

↔ I/O

Putting it together



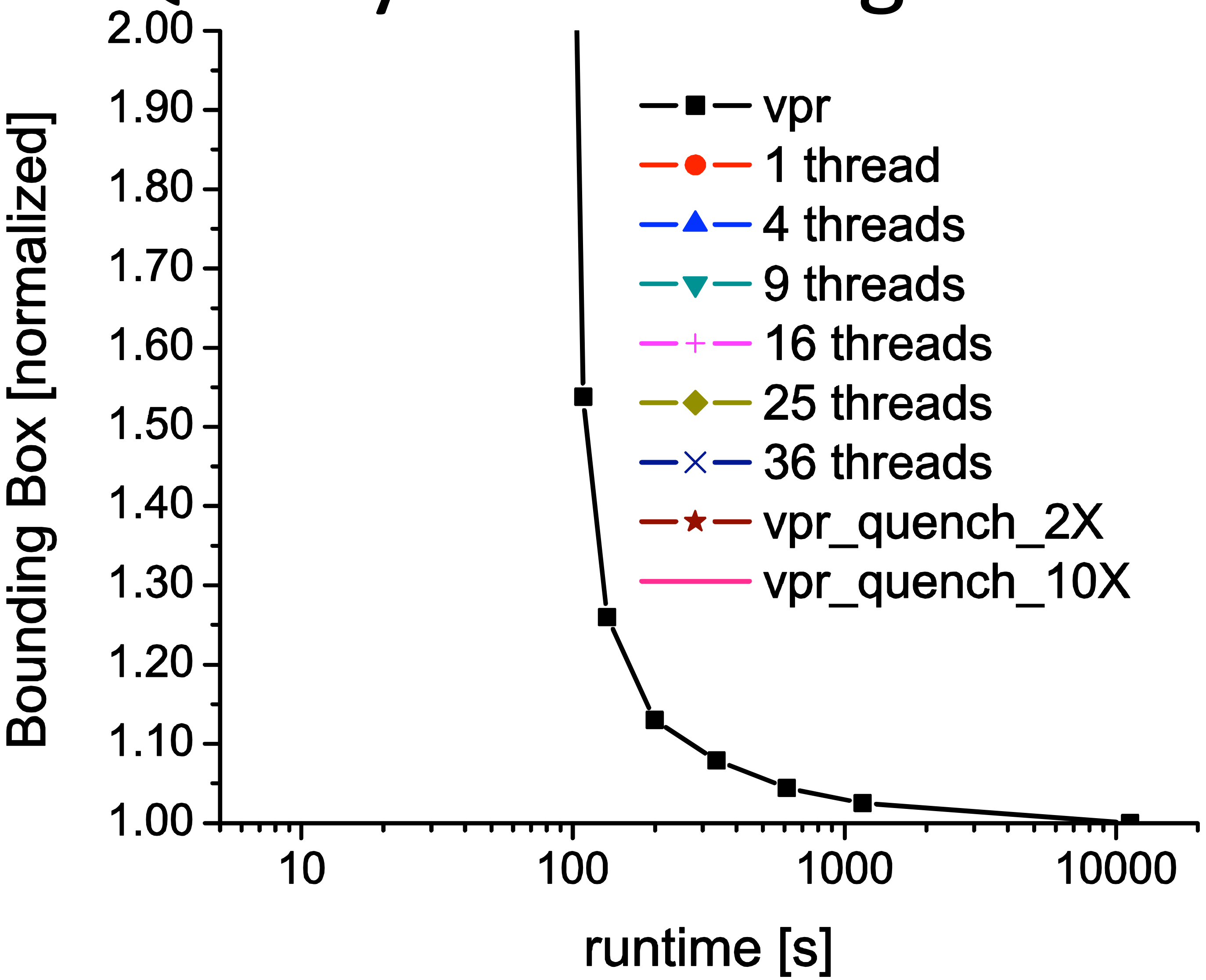
≥6 swaps with
constrained swap region

- ● CLB
- ↔ I/O

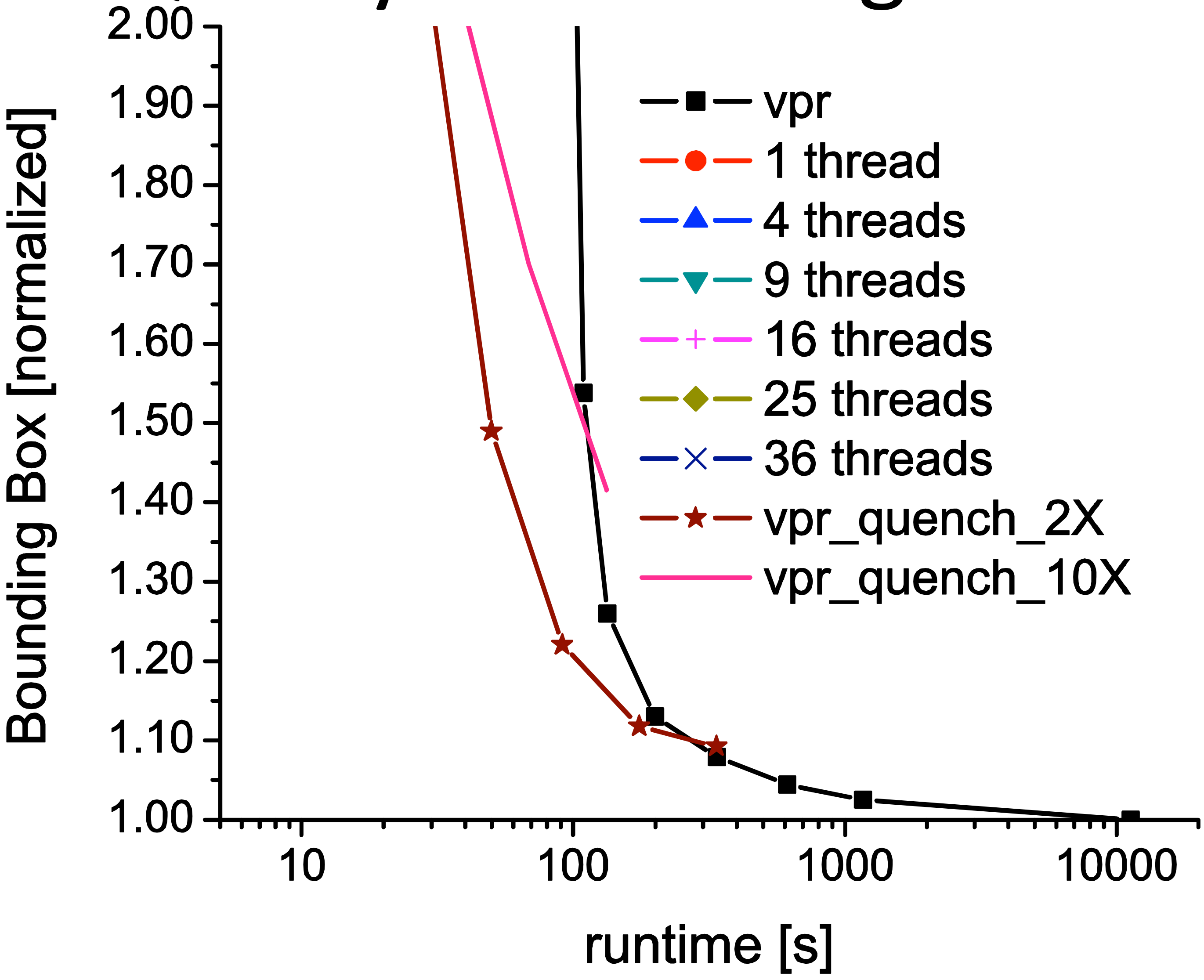
Result

- 7 synthetic circuits from Un/DoPack flow
- Clustered with T-Vpack 5.0.2
- Niagara 2, 64 threads, 1165 MHz, 32GB
- Baseline: `VPR 5.0.2 –place_only`
- Only placement time
  - – Exclude netlist reading…etc

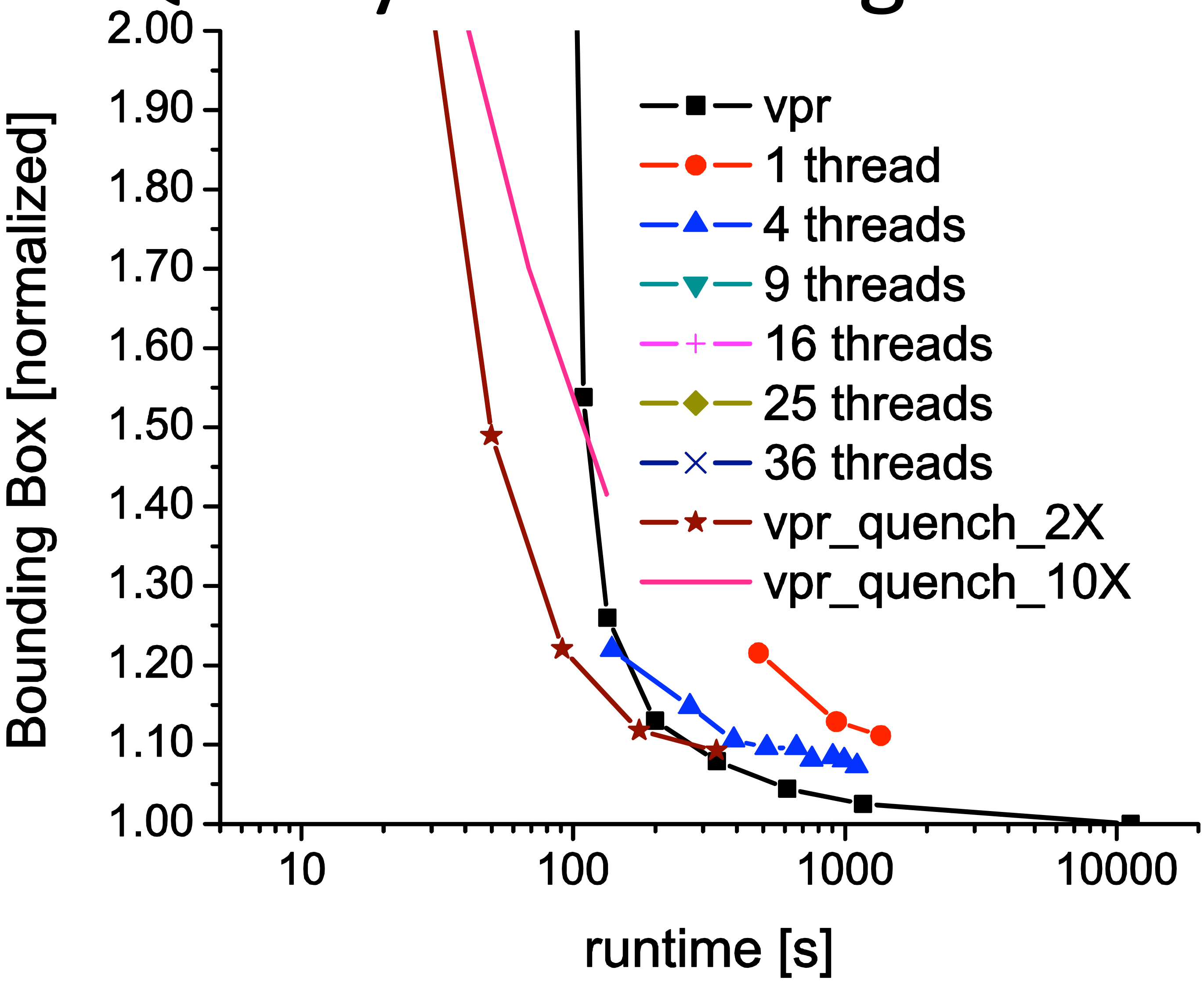
Quality – Bounding Box



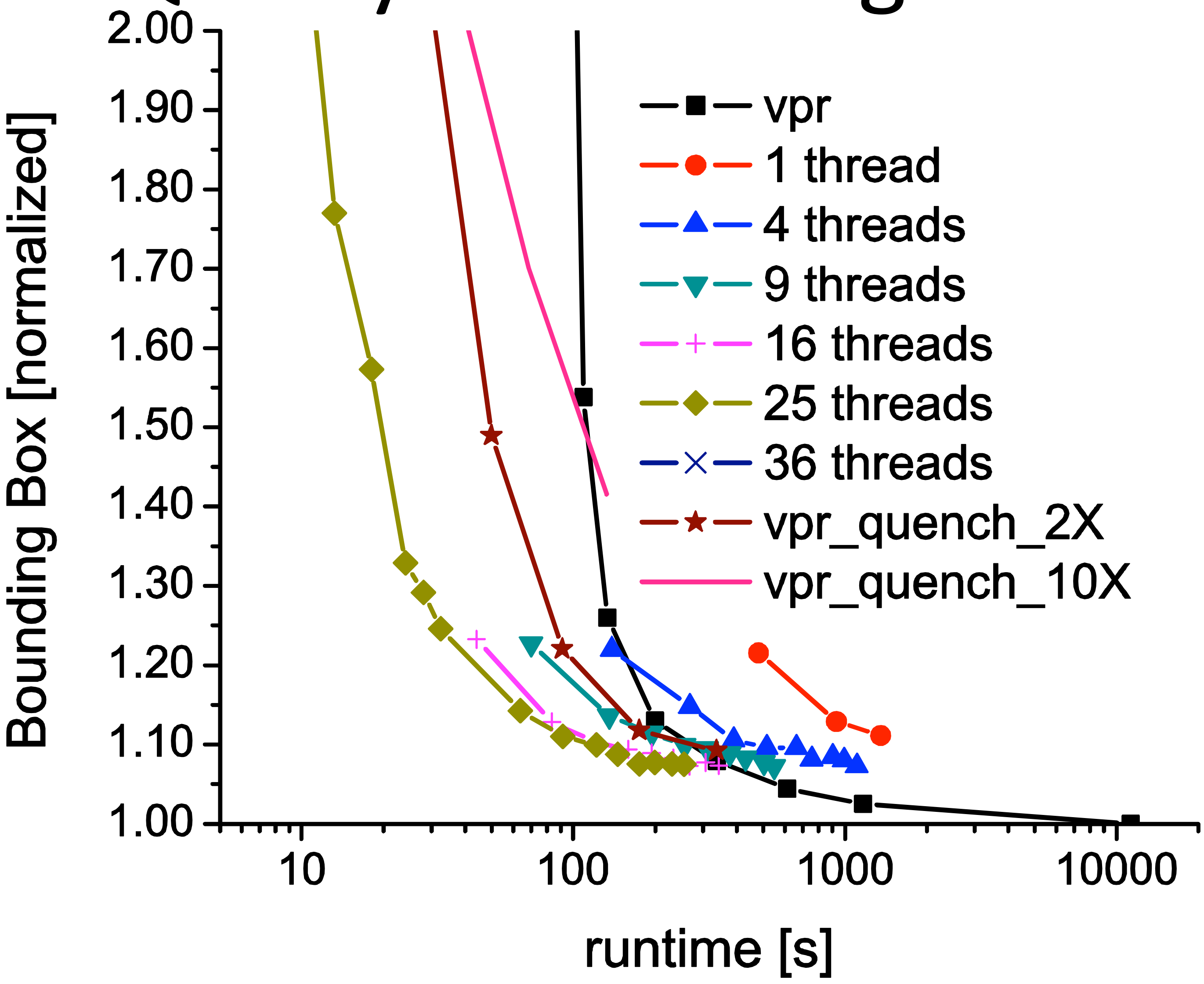
Quality – Bounding Box



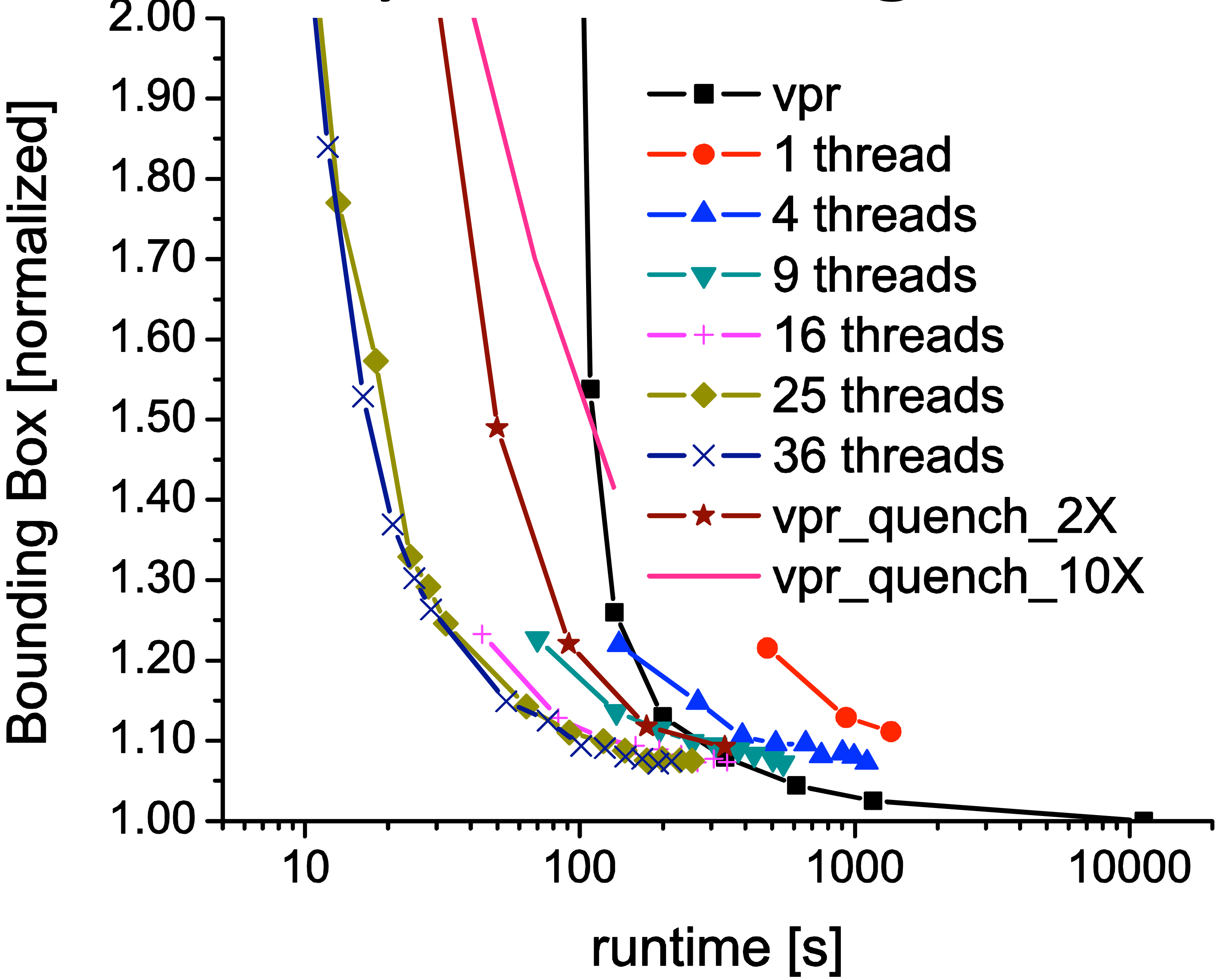
Quality – Bounding Box



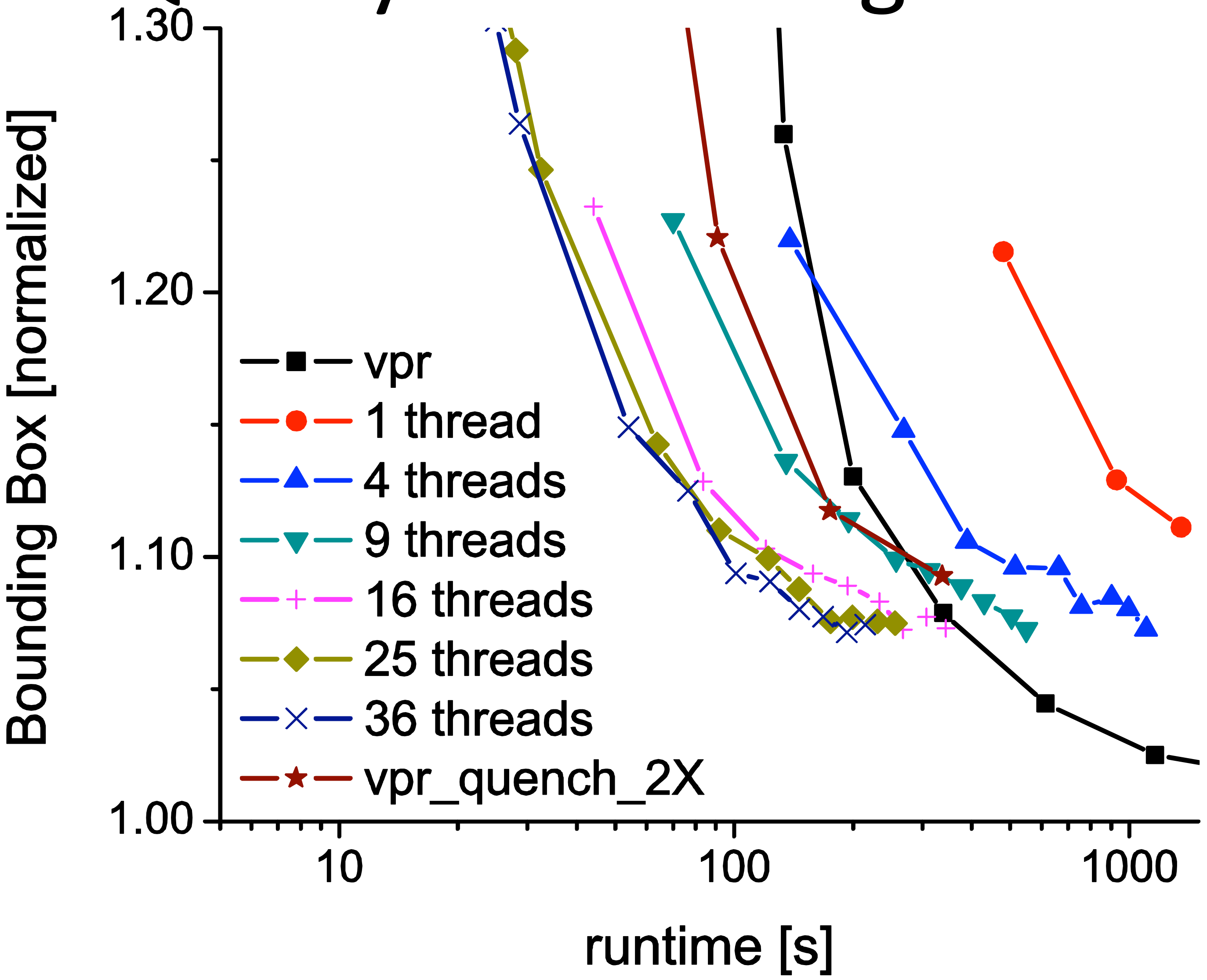
Quality – Bounding Box



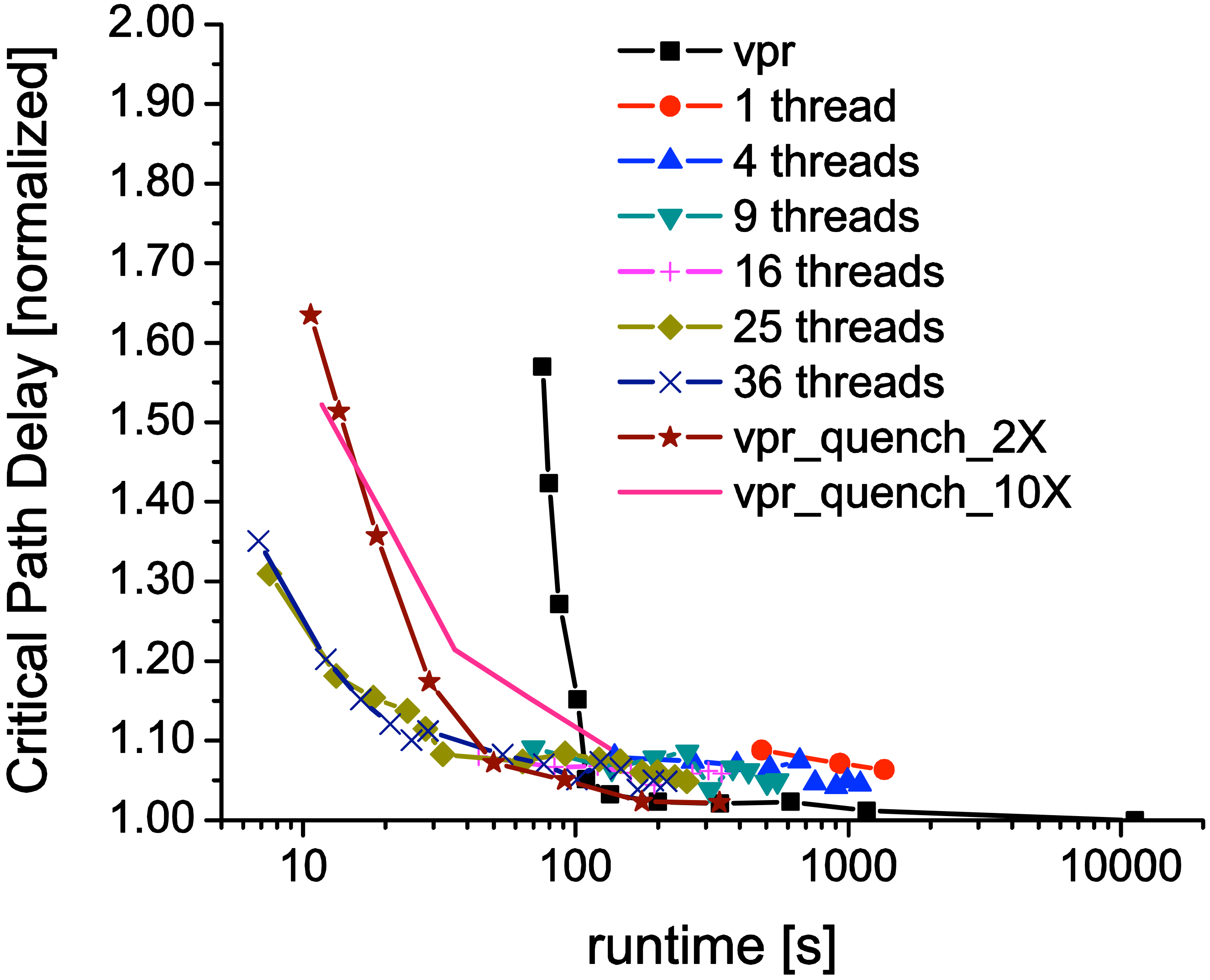
Quality – Bounding Box



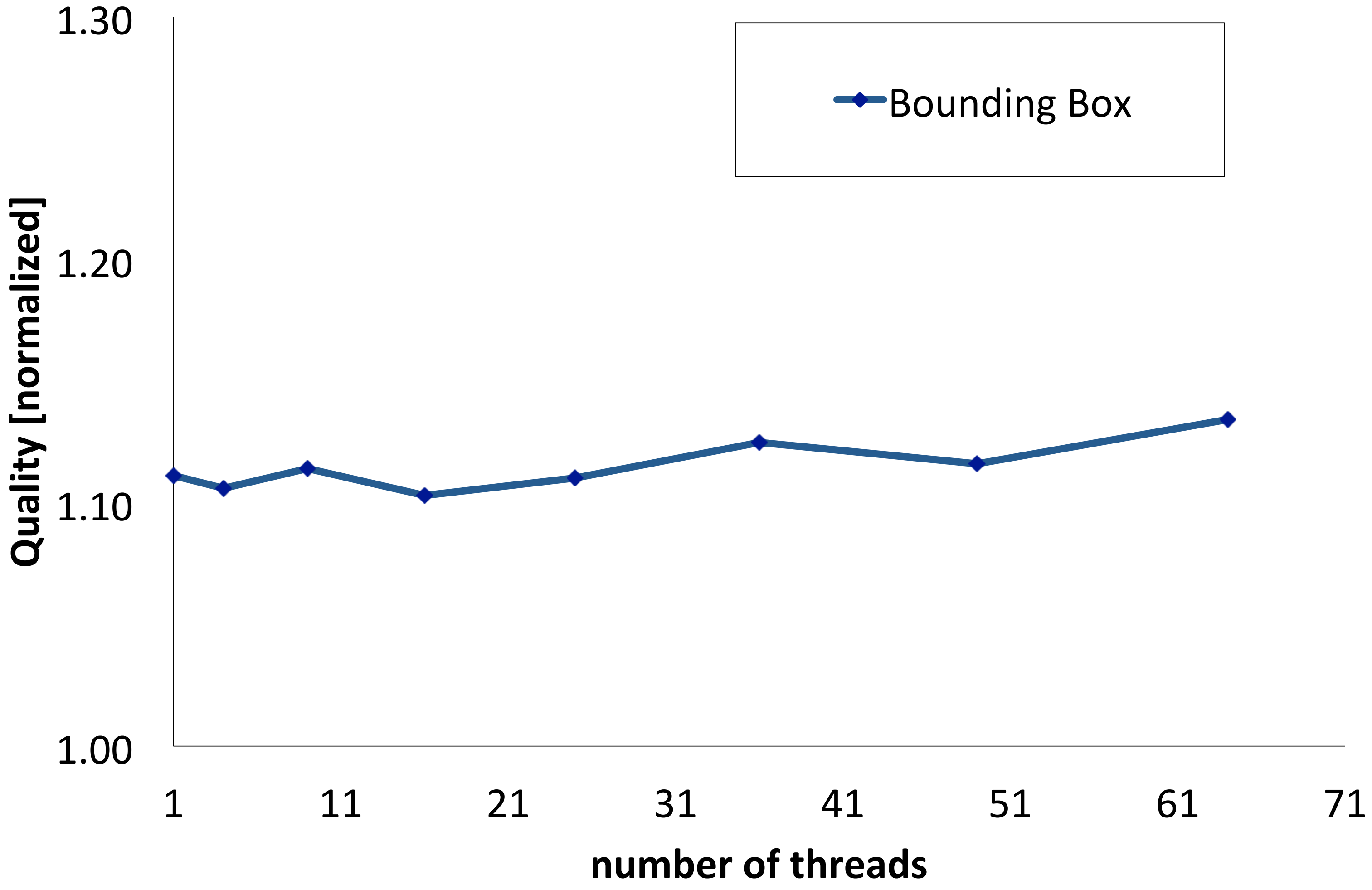
Quality – Bounding Box



Quality – Critical Path Delay

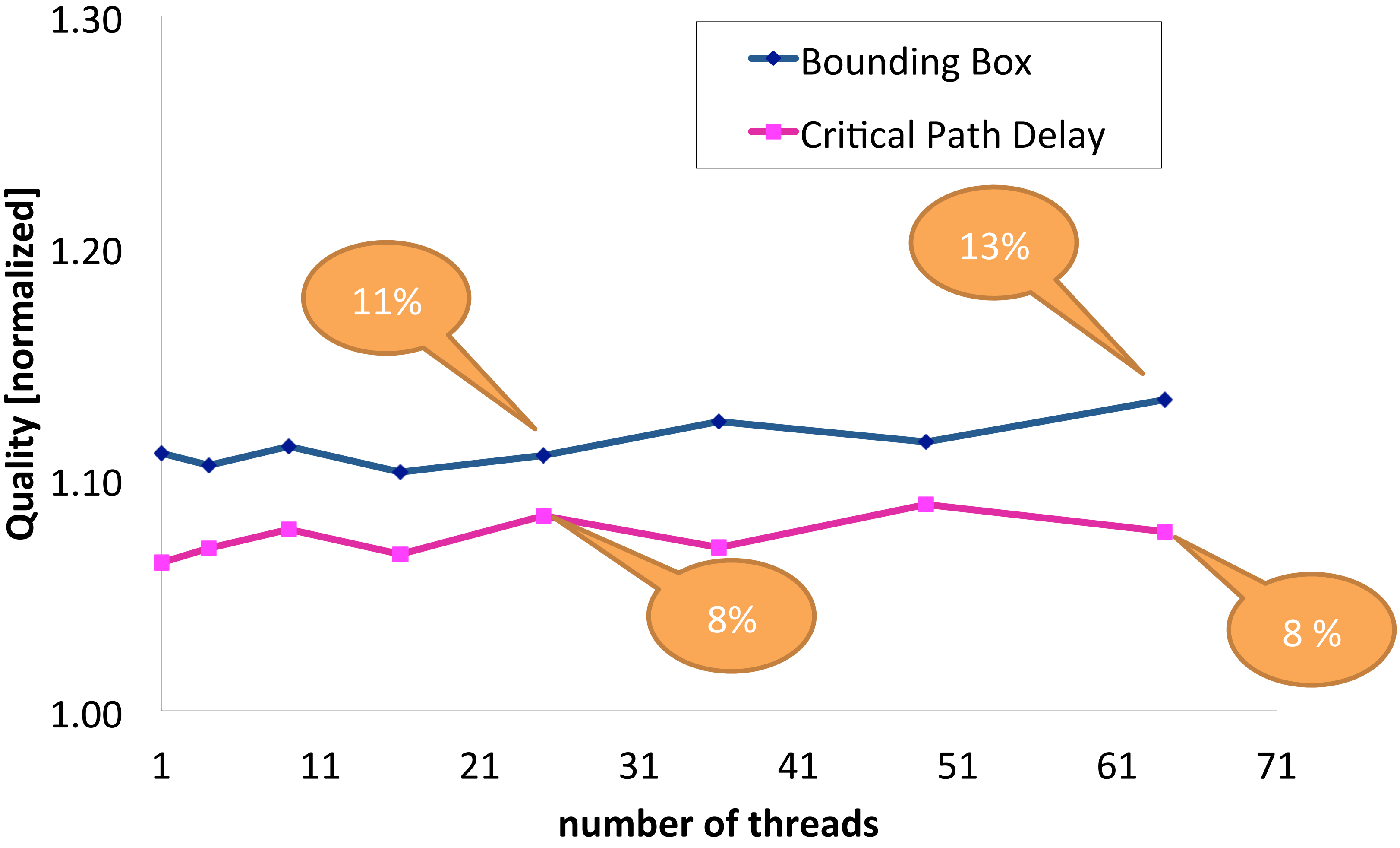


Effect of scaling on QoR



region_place_count = 90

Effect of scaling on QoR



Conclusion

- QoR
  - Constrained swap region → more swaps at higher T
  - Extended sub-region to allow blocks to migrate
  - Mildly affected by the number of threads
- Determinism without fine-grain synchronization
  - Split work into sub-regions
  - Local (stale) copy of global data
- Runtime scalable, timing-driven
- Speedup: 12X over VPR –fast, 123X over VPR

Future Work

Question: Can we scale beyond 25 threads?

YES!

- Better load balance techniques
  - Improved region partitioning
- New data structures
  - Support fully parallelizable timing updates
  - Reduce inter-processor communication
- Incremental timing analysis update
  - May benefit QoR as well!

Questions?

Chris Wang, Guy Lemieux
{chrisw, lemieux} @ ece.ubc.ca

University of British Columbia

backup

Bibliography

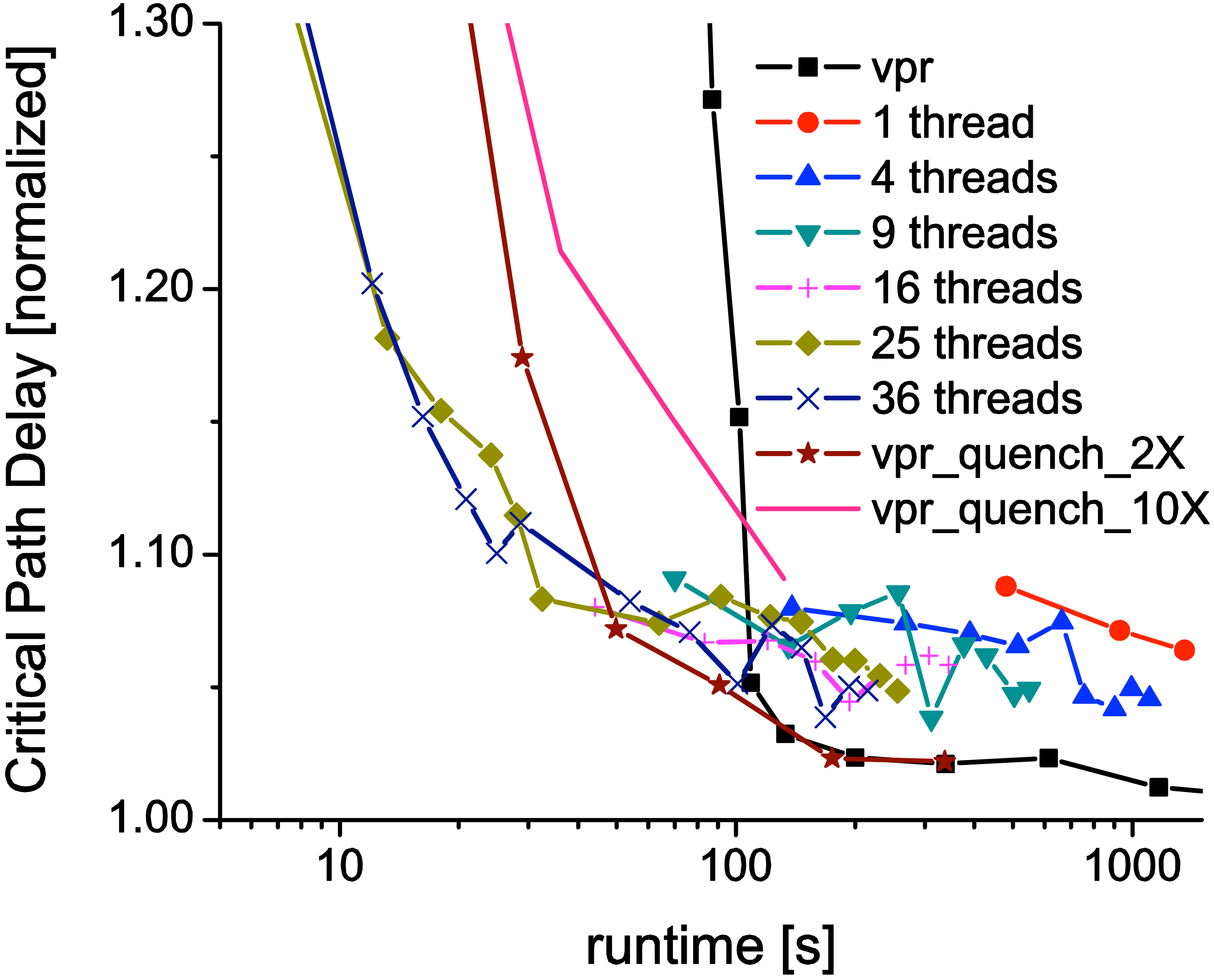
- [1] Moores Law - http://www.ieee.org/portal/site/sscs/menuitem.f07ee9e3b2a01d06bb9305765bac26c8/index.jsp?&pName=sscs_level1_article&TheCat=6010&path=sscs/06Sept&file=Halfhill.xml

- [2] A. Ludwin, V. Betz, and K. Padalia. High-quality, deterministic parallel placement for FPGAs on commodity hardware. In FPGA ’08: Proc. International Symposium on Field Programmable Gate Arrays, pages 14–23, 2008.

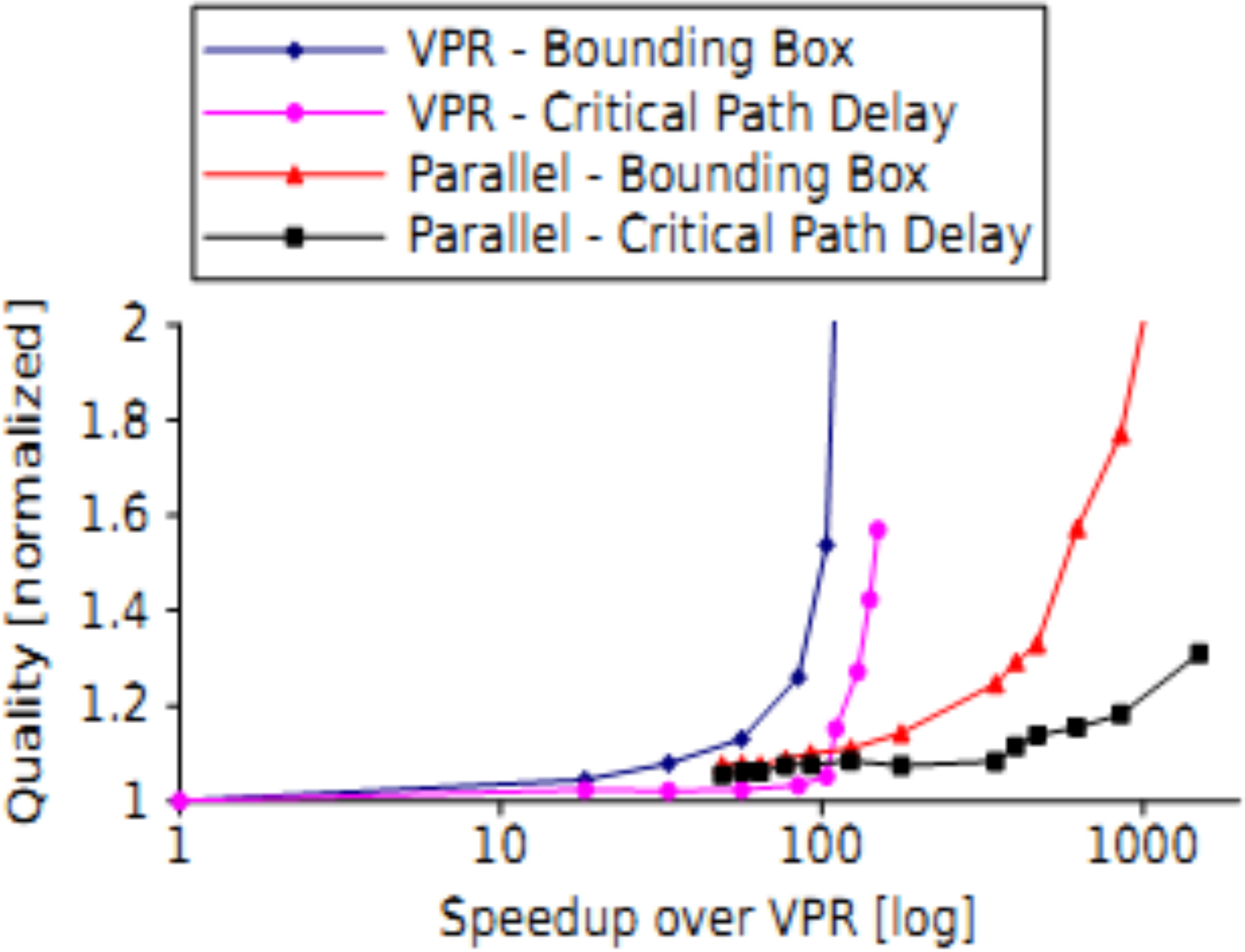
- [3] Michael G. Wrighton , André M. DeHon, Hardware-assisted simulated annealing with application for fast FPGA placement, Proceedings of the 2003 ACM/SIGDA eleventh international symposium on Field programmable gate arrays, February 23-25, 2003, Monterey, California, USA

- [4] G. Smecher, S. Wilton, G. Lemieux, "Self-hosted placement for massively parallel processor arrays," *Field-Programmable Technology, 2009. FPT 2009. International Conference on* , vol., no., pp.159-166, 9-11 Dec. 2009

Quality – Critical Path Delay

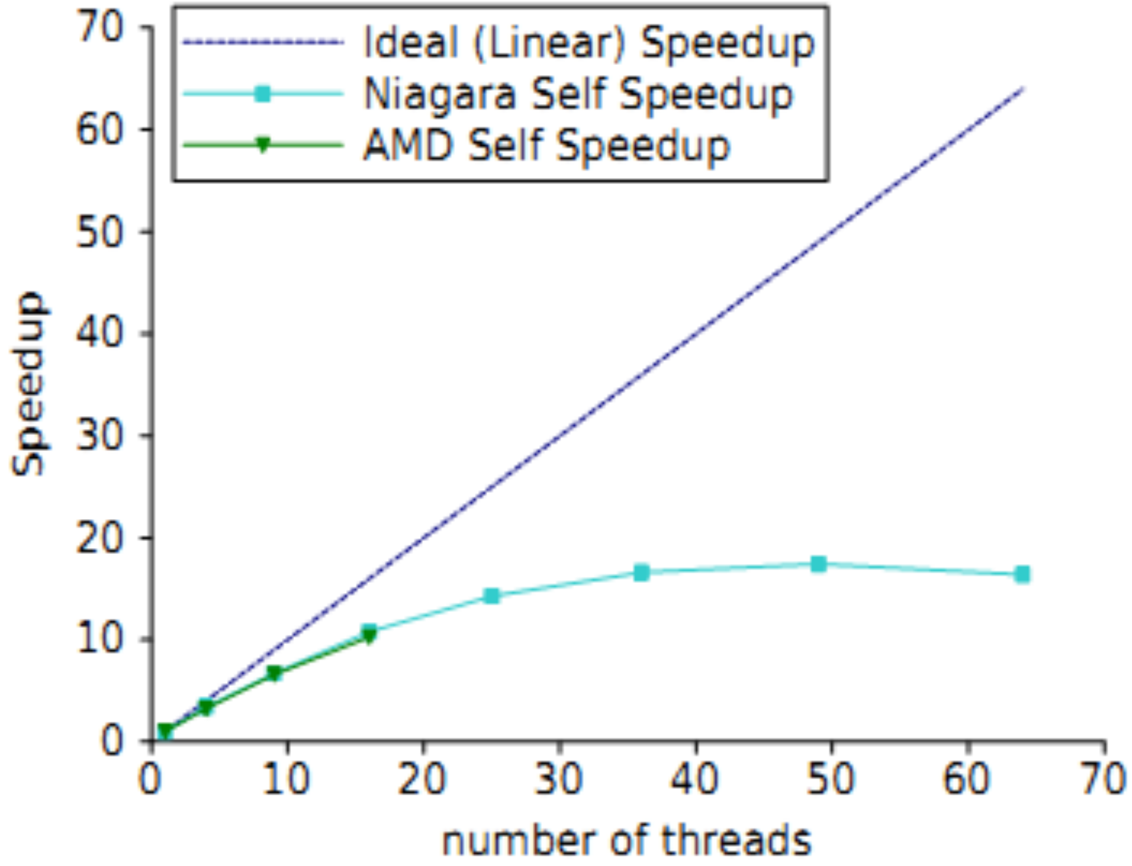


Quality from speeding up VPR and Parallel



Self Speedup

(Fig. 8)


Ideal (Linear) Speedup
Niagara Self Speedup
AMD Self Speedup
Speedup
number of threads