

HETRIS: Adaptive Floorplanning for Heterogeneous FPGAs

Kevin E. Murray and Vaughn Betz



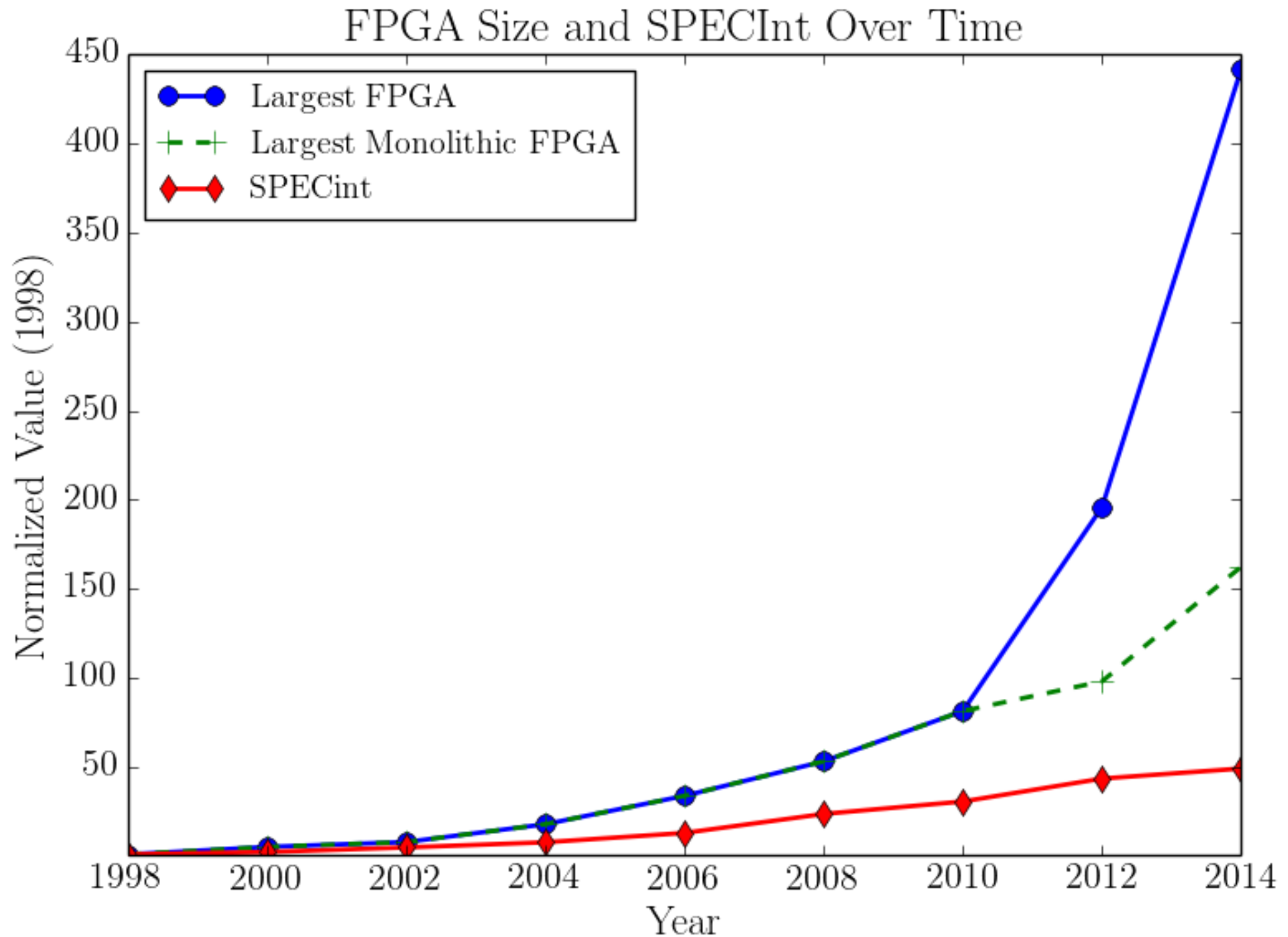
UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE & ENGINEERING

Overview

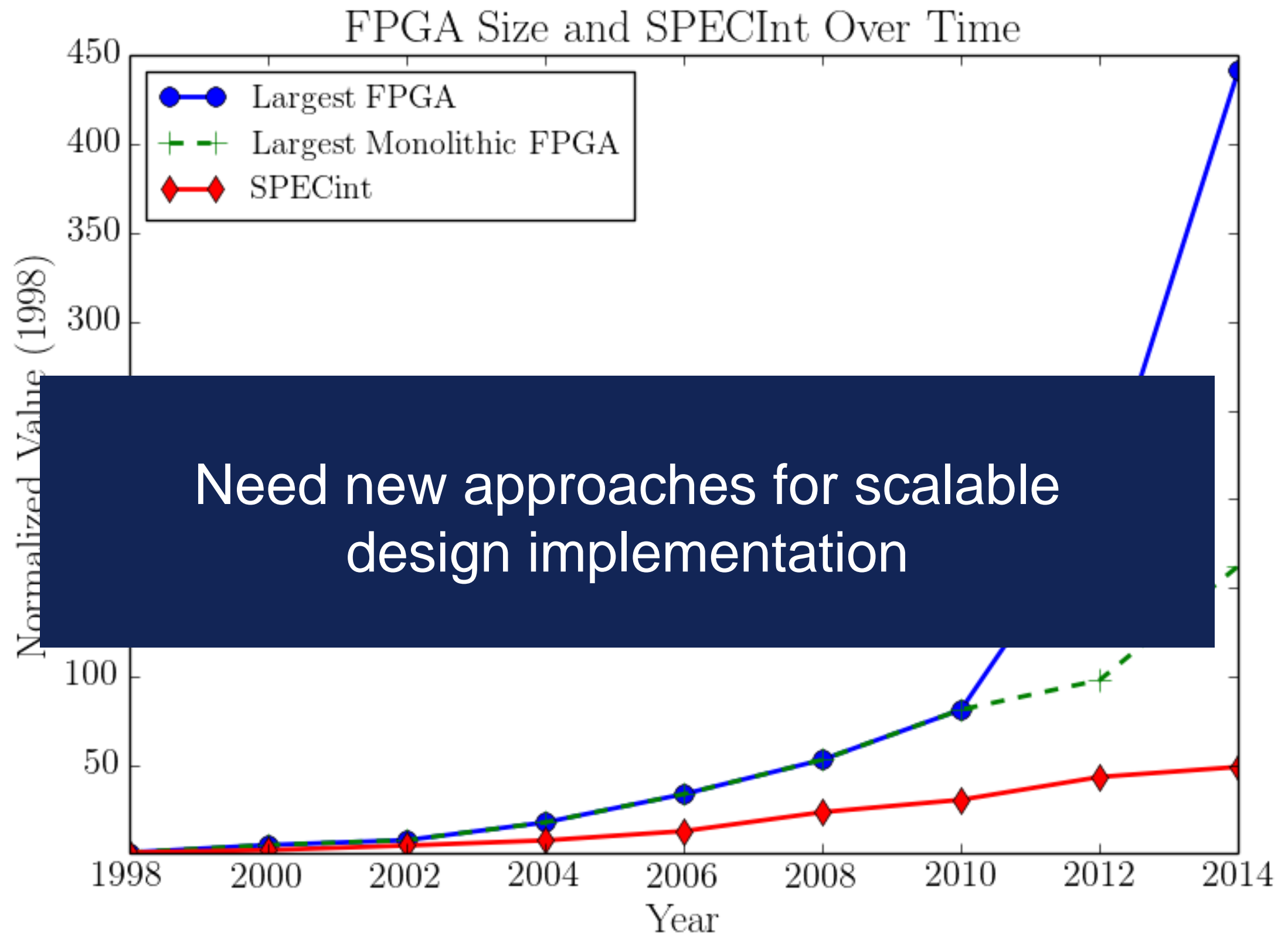
- Heterogeneous FPGA Floorplanner
 - Dynamically adapts to targeted FPGA Architecture
 - 15.6x faster than prior work
 - Open Source
- Investigate nature of heterogeneous FPGA floorplanning
- First evaluation of a heterogeneous FPGA floorplanner on realistic benchmarks and architectures
 - Comparison to a commercial tool



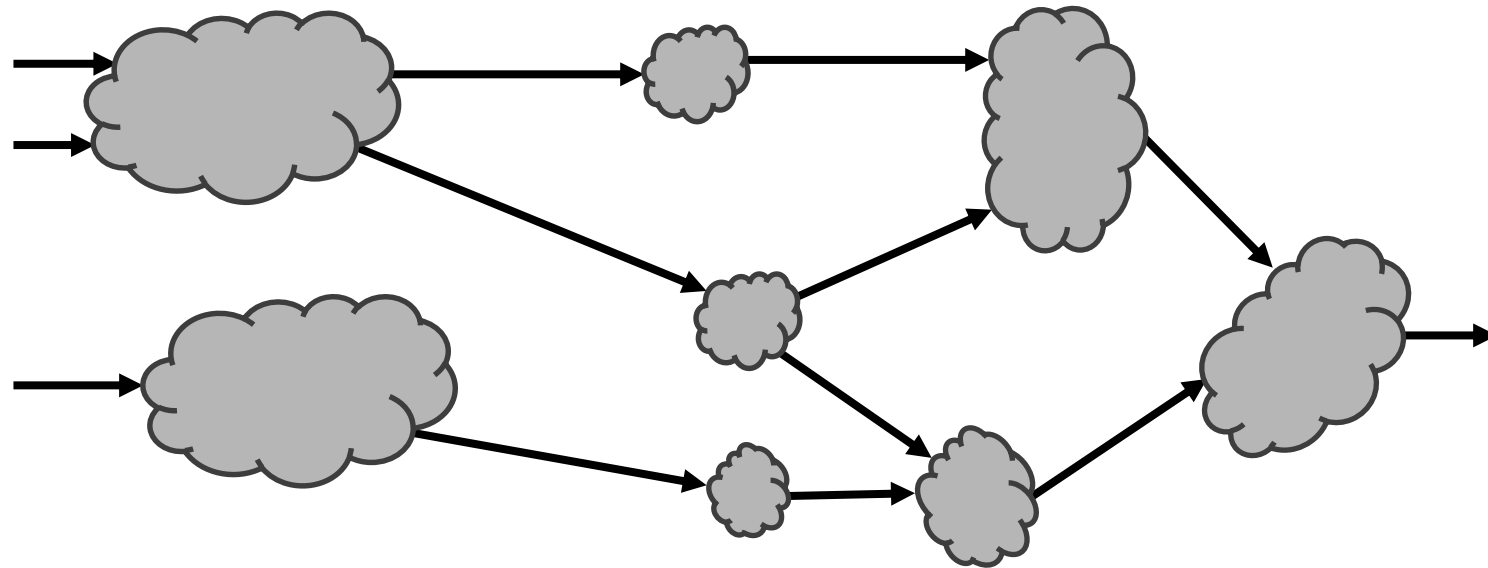
Increasing FPGA Design Size



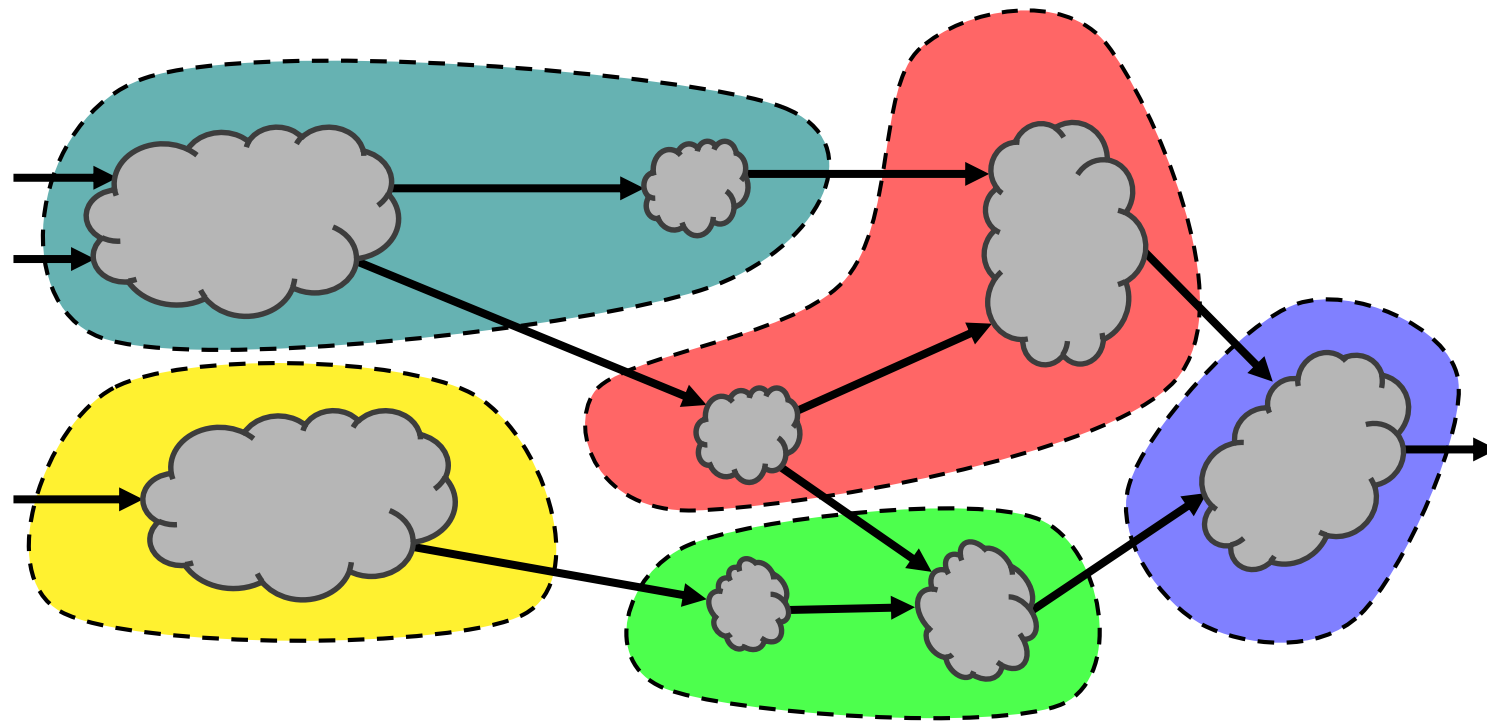
Increasing FPGA Design Size



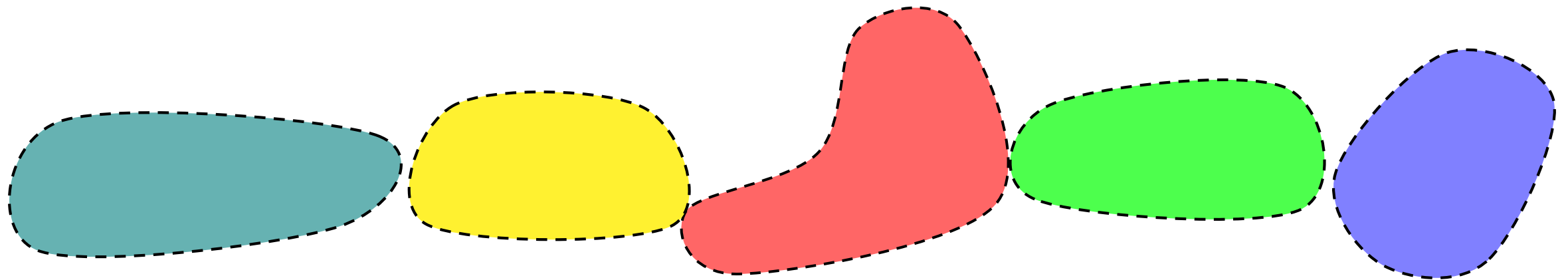
Floorplanning



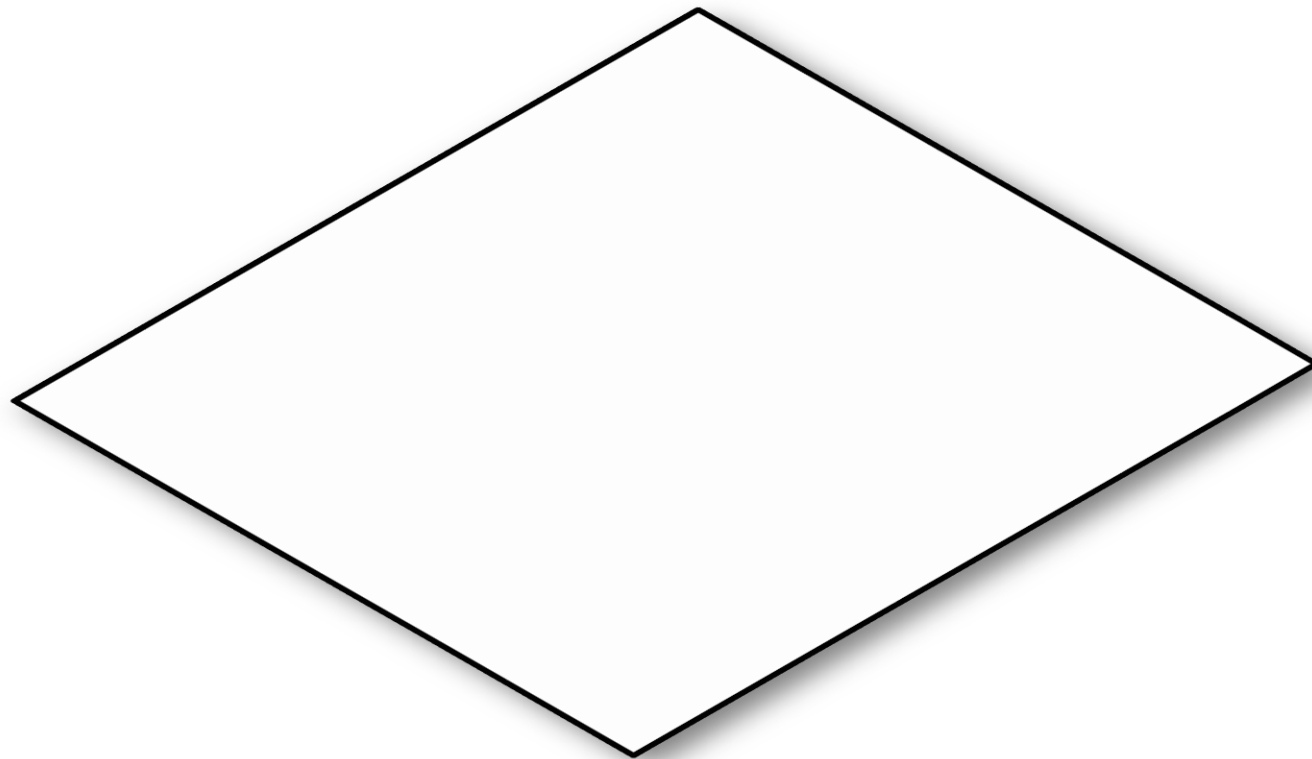
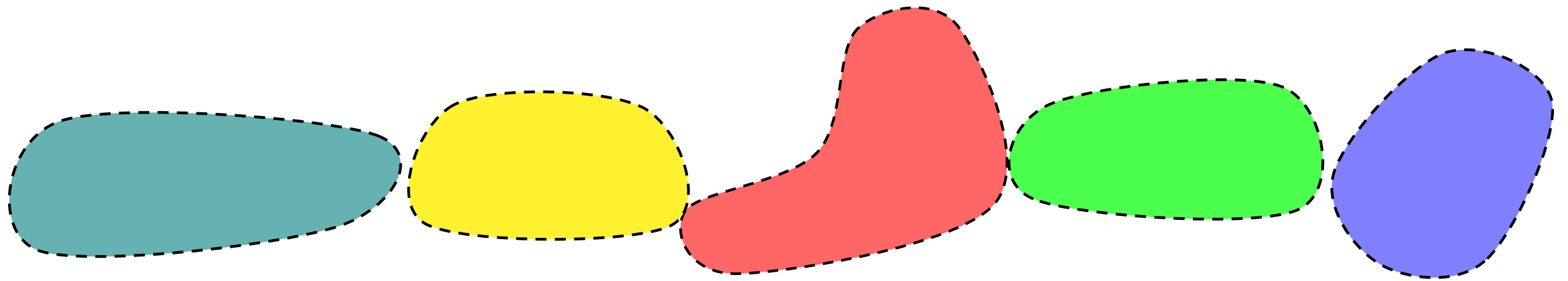
Floorplanning



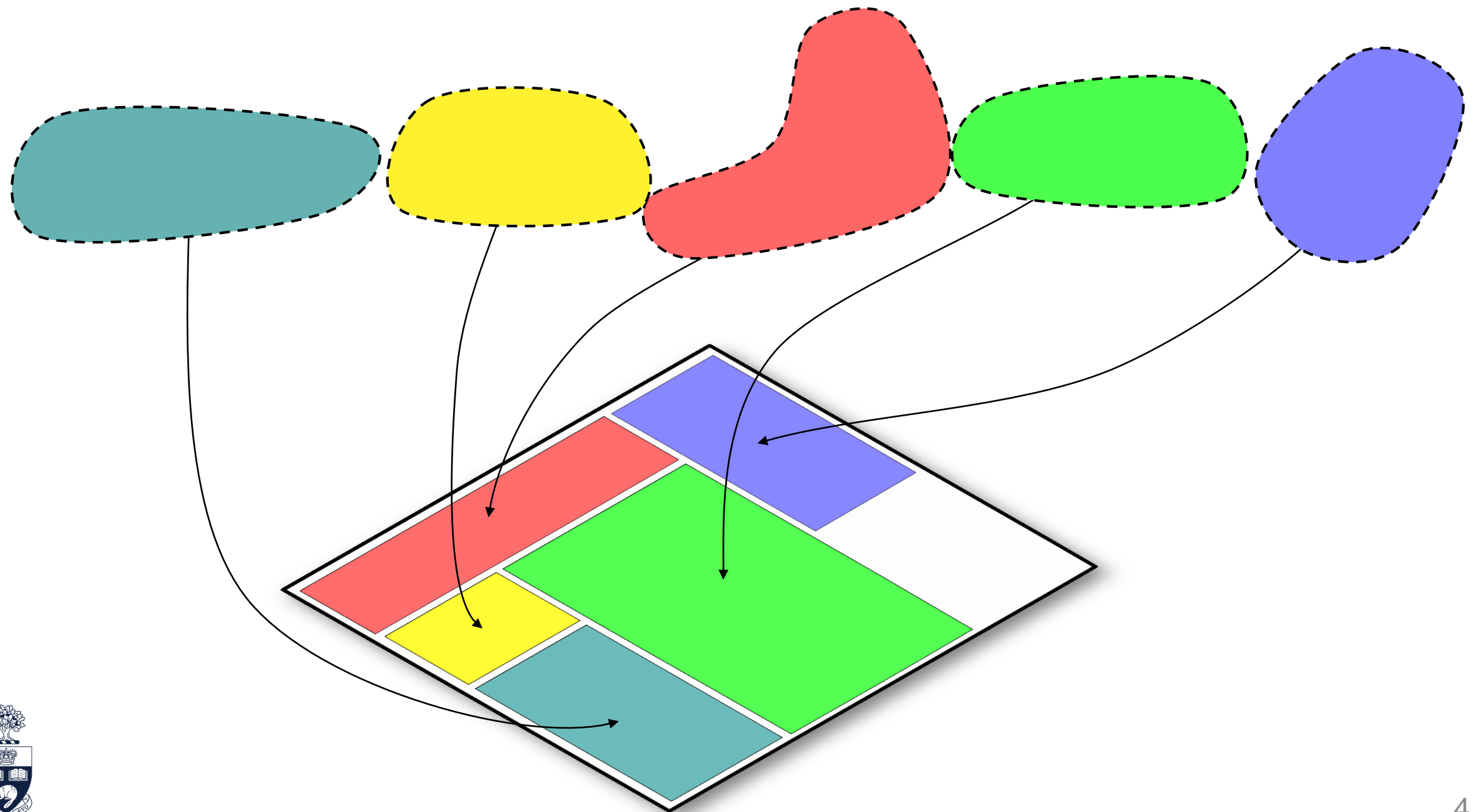
Floorplanning



Floorplanning

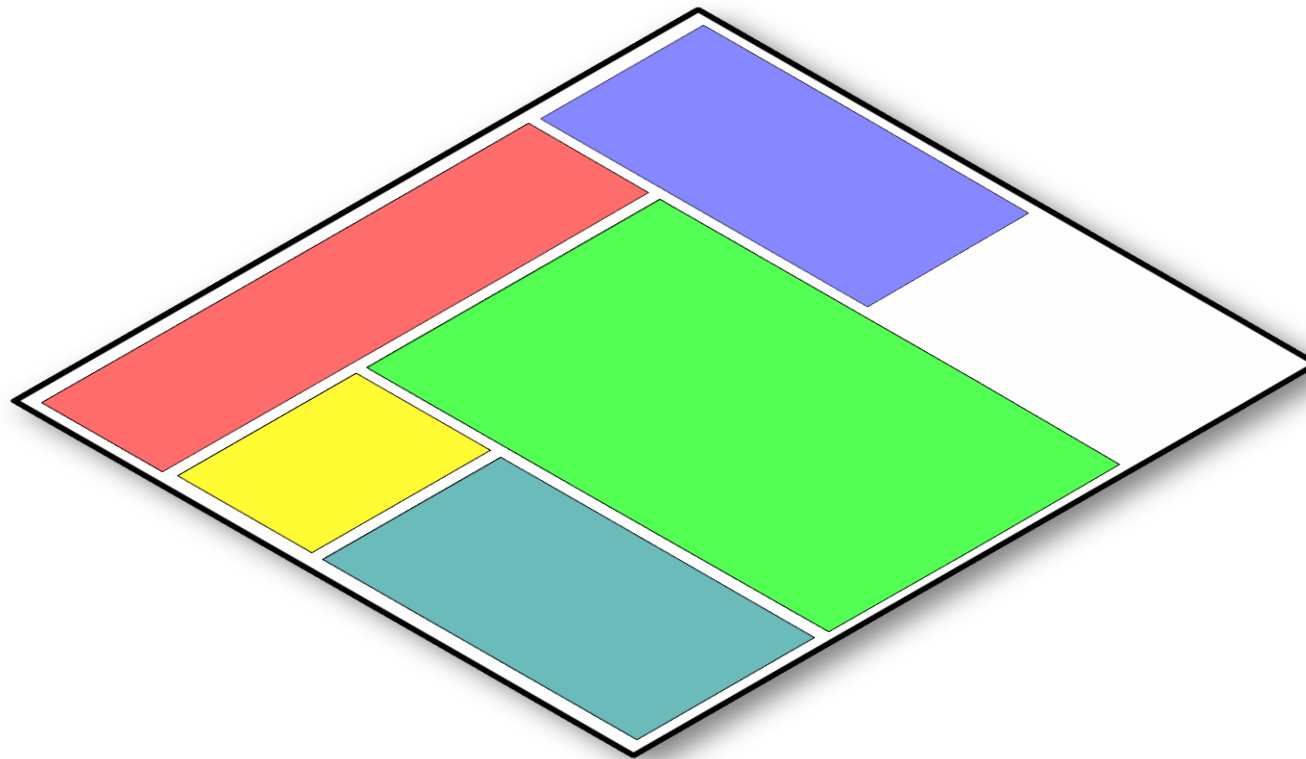


Floorplanning



Floorplanning

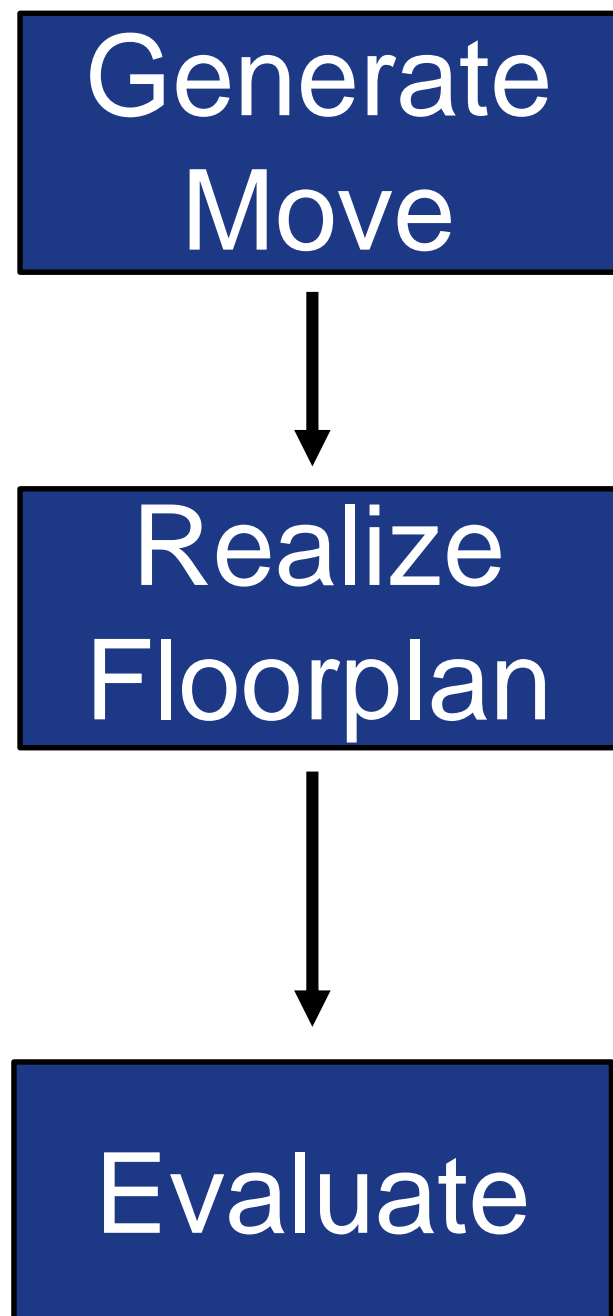
- Divide-and-conquer design implementation
 - Solve smaller sub-problems (potentially in parallel)
 - Re-use existing CAD tools and algorithms
 - Improved team-based design
- Required for Partial Reconfiguration



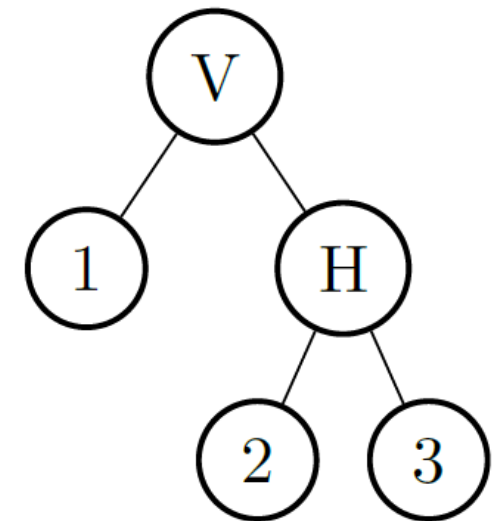
HETRIS: Heterogeneous Region Implementation System



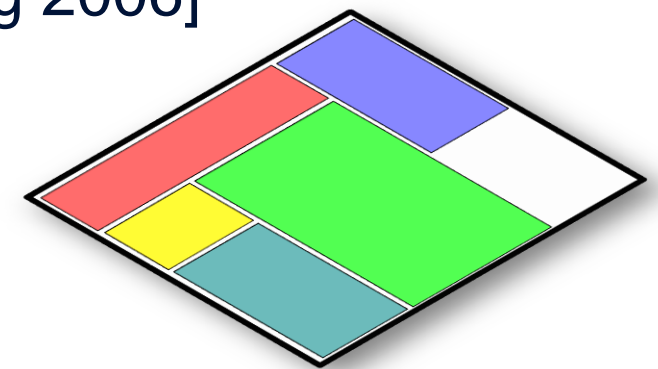
Hetris: Overview



- Slicing Tree



- Irreducible Realization Lists
[Cheng & Wong 2006]



- Area & Wirelength Costs

Hetris: Overview

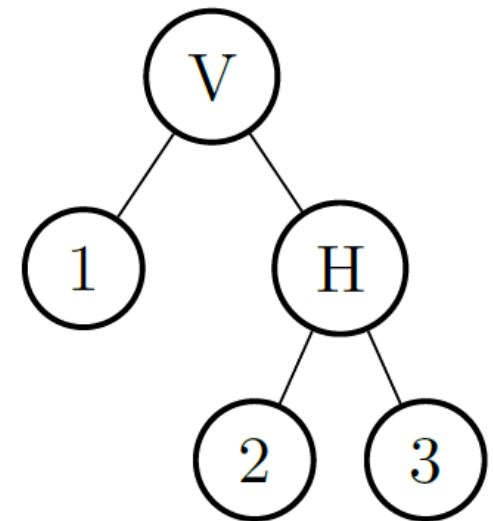
Simulated Annealing

Generate
Move

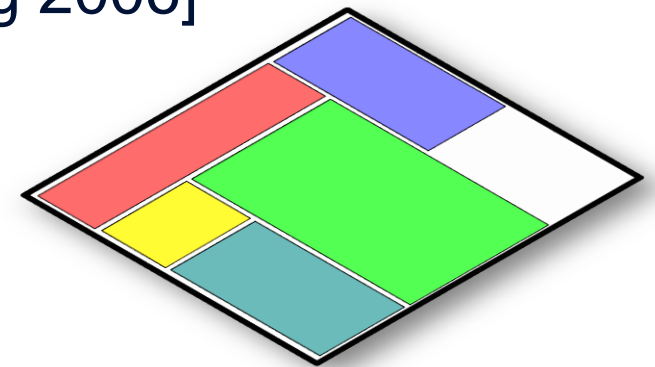
Realize
Floorplan

Evaluate

- Slicing Tree

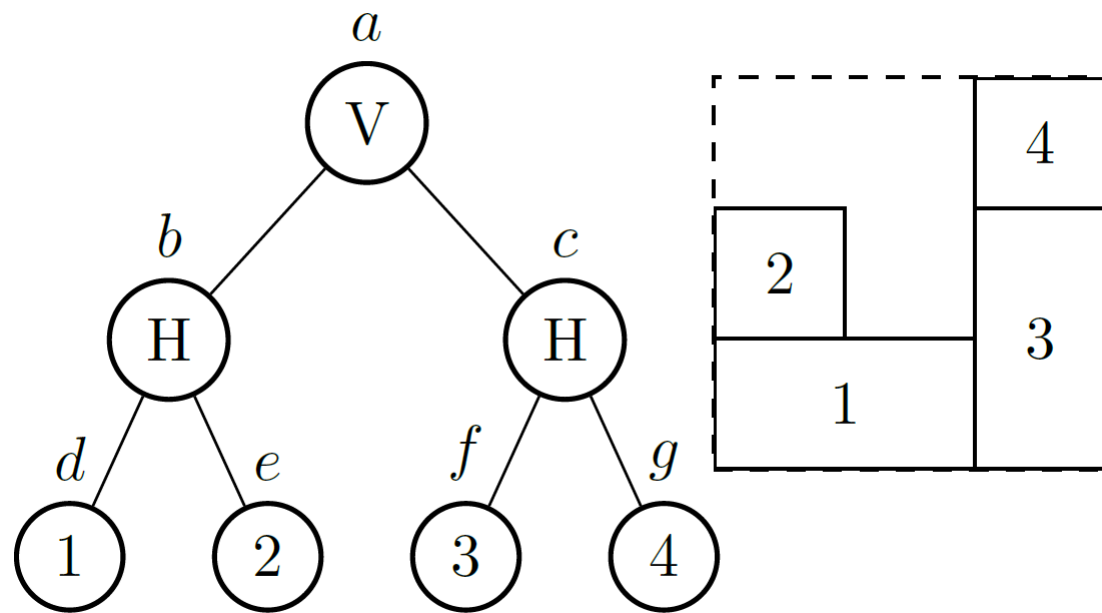


- Irreducible Realization Lists
[Cheng & Wong 2006]

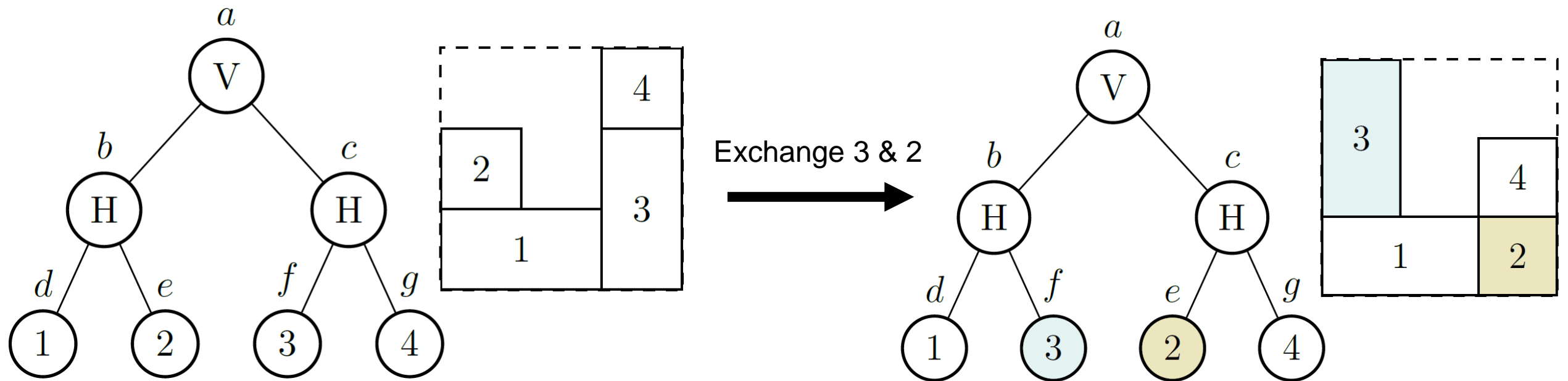


- Area & Wirelength Costs

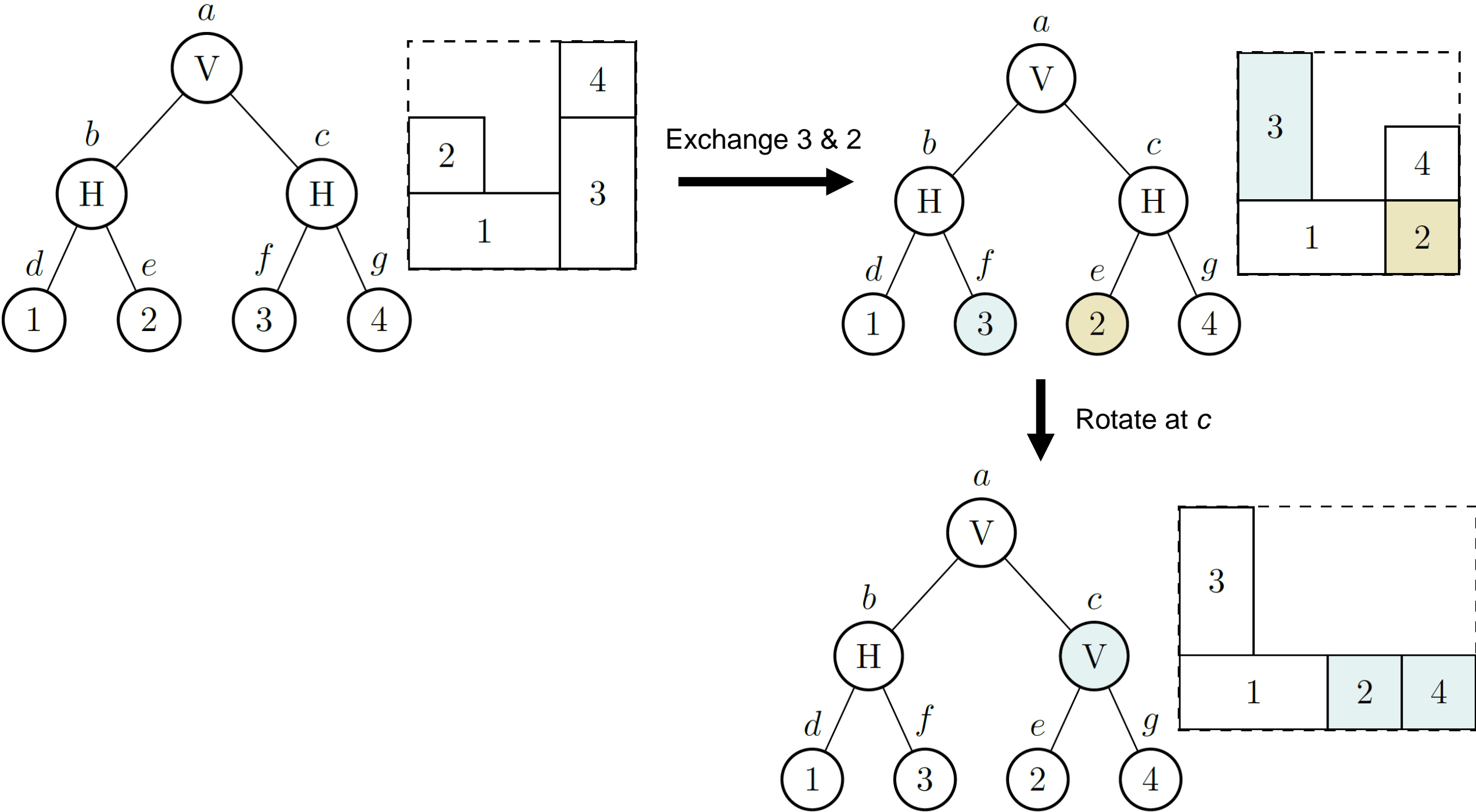
Slicing Tree Moves



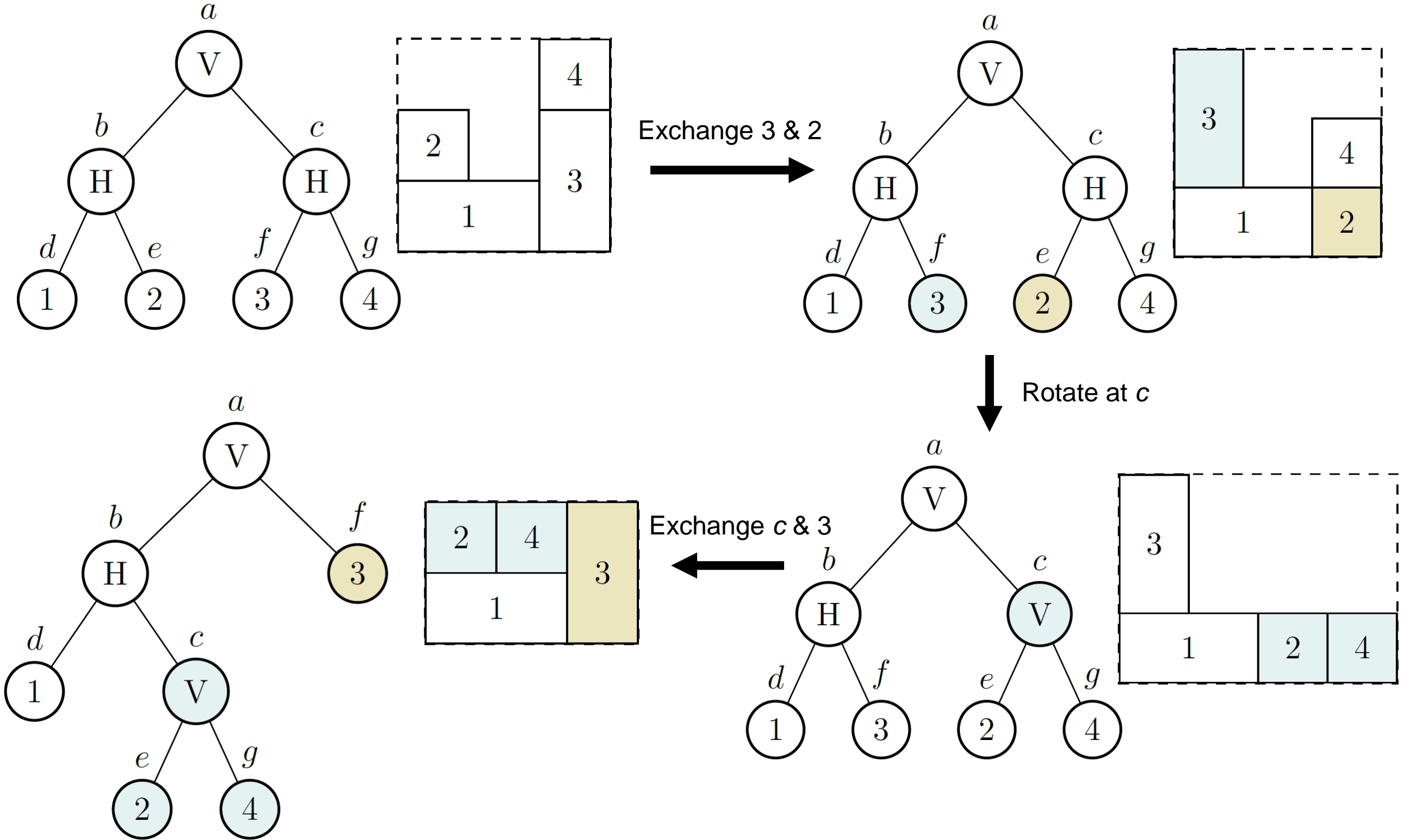
Slicing Tree Moves



Slicing Tree Moves

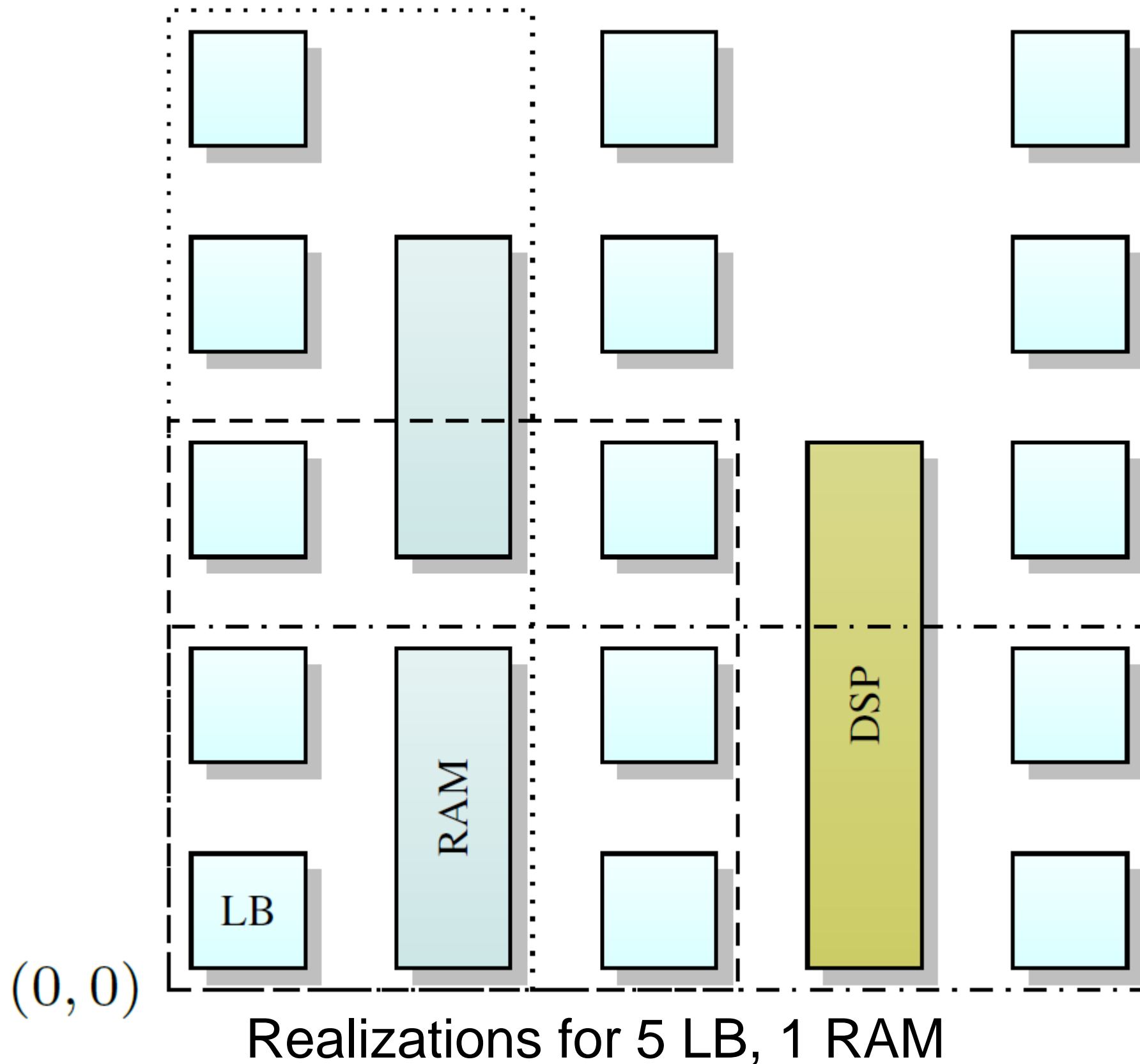


Slicing Tree Moves



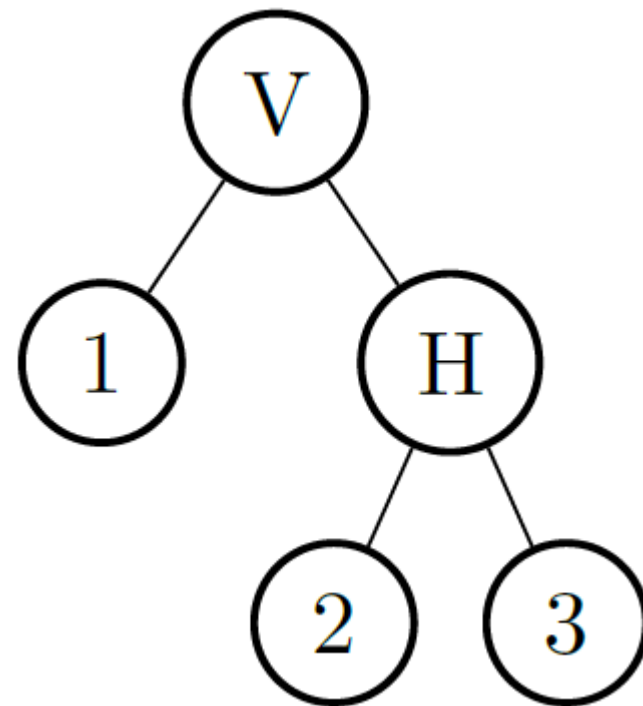
Handling Heterogeneity: Irreducible Realization Lists

- Unique to every location on the FPGA



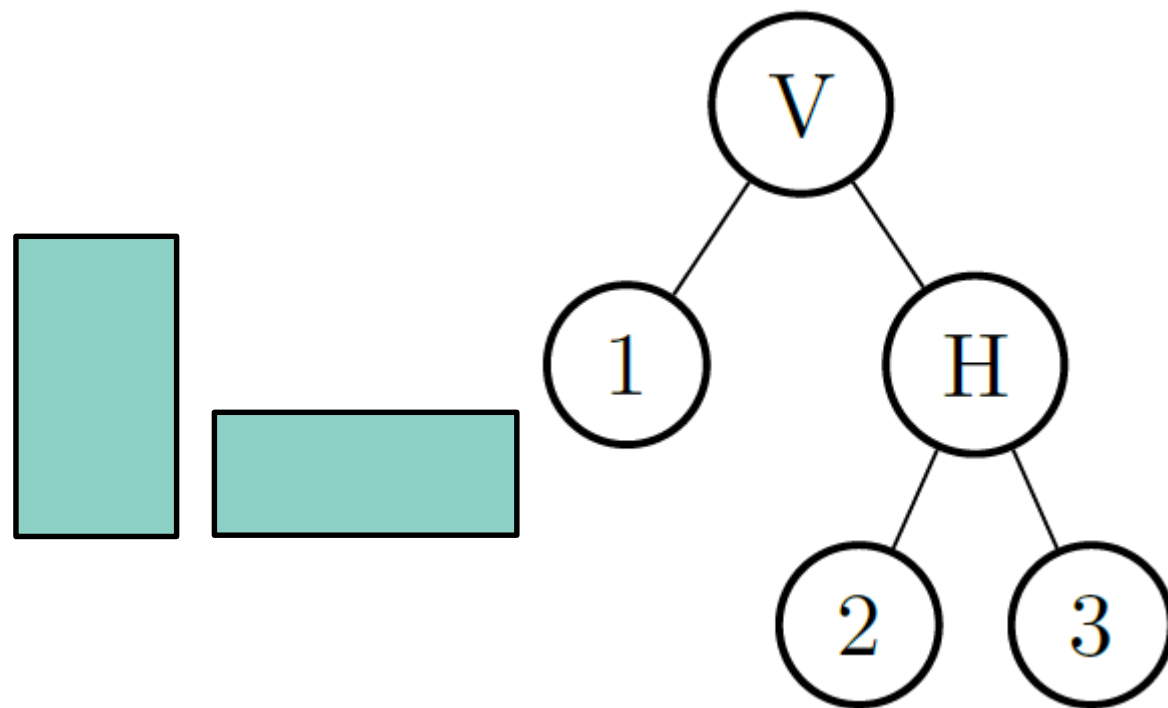
Realizing Slicing Trees

- Recursively calculate shapes at each node in the tree [Cheng & Wong 2006]
- Realizations at root encode full floorplans



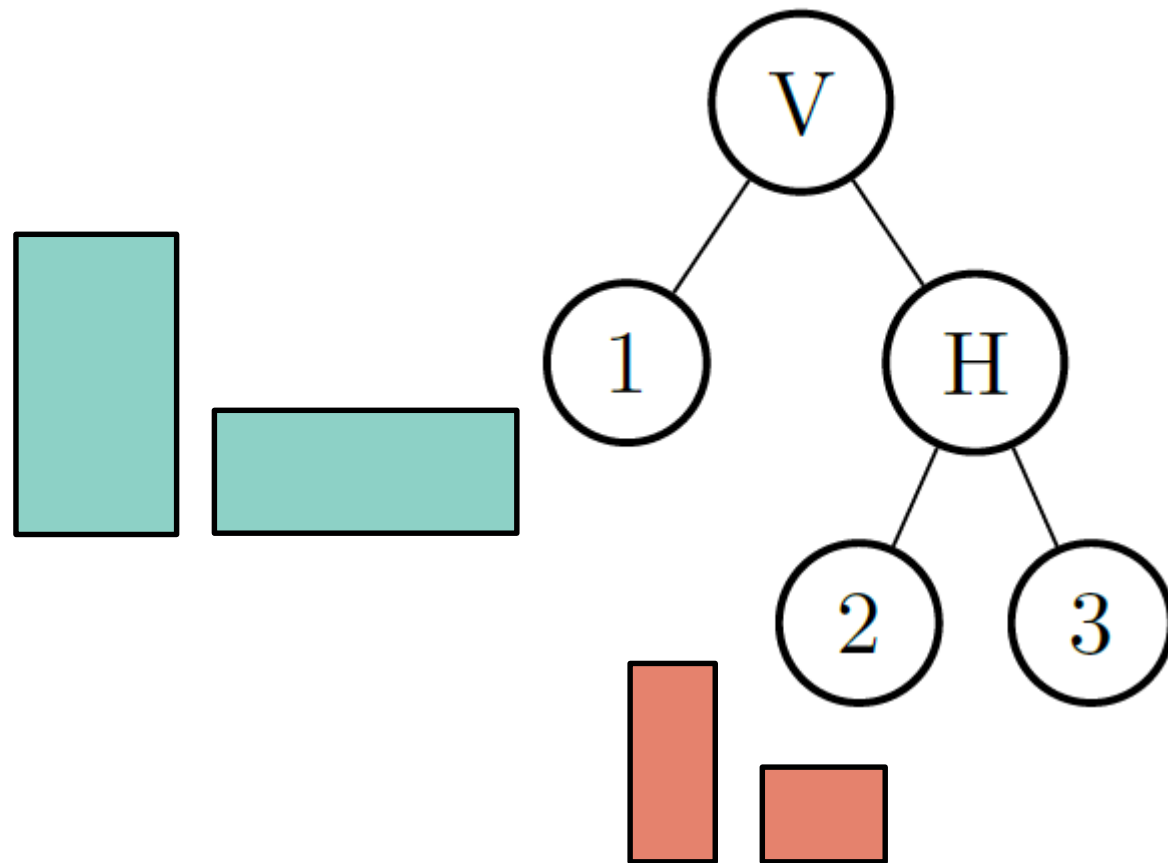
Realizing Slicing Trees

- Recursively calculate shapes at each node in the tree [Cheng & Wong 2006]
- Realizations at root encode full floorplans



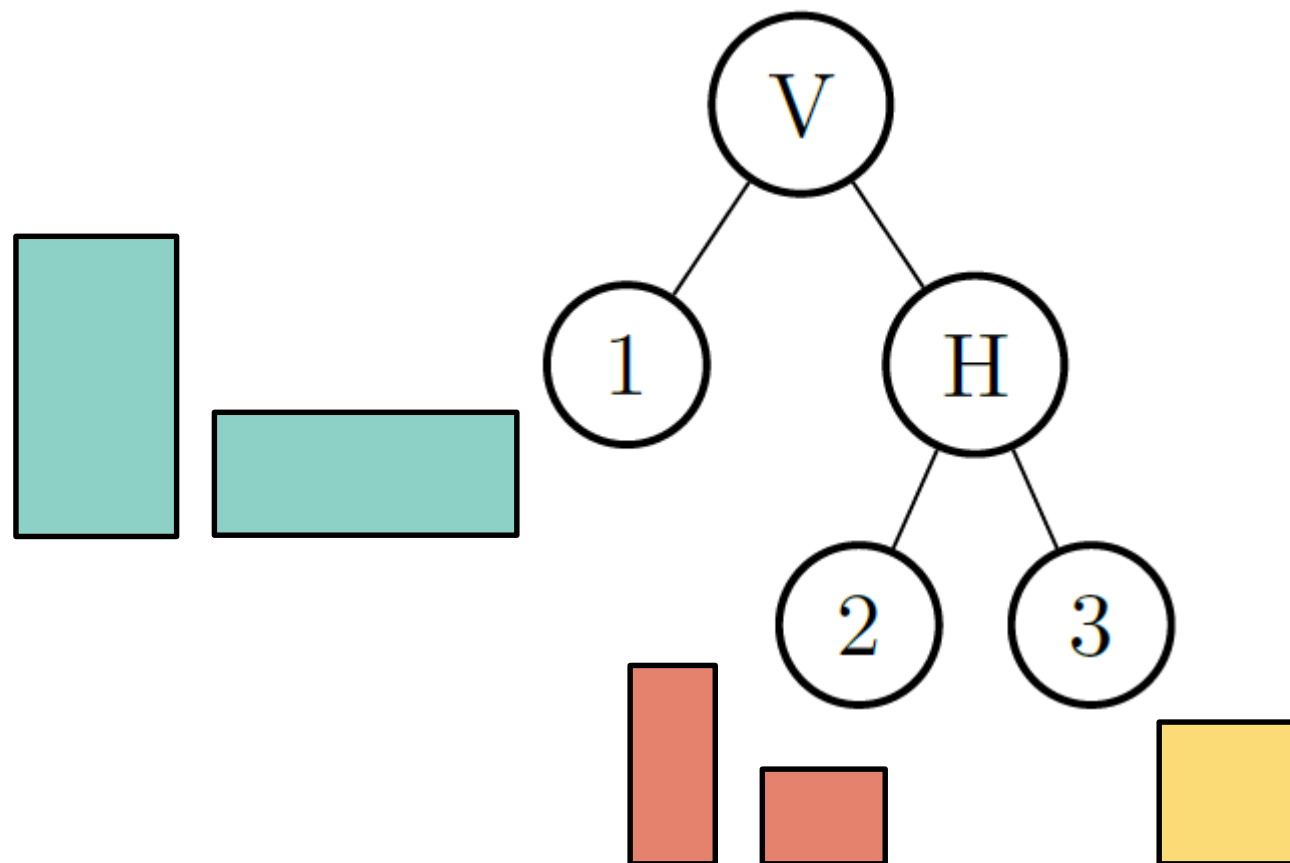
Realizing Slicing Trees

- Recursively calculate shapes at each node in the tree [Cheng & Wong 2006]
- Realizations at root encode full floorplans



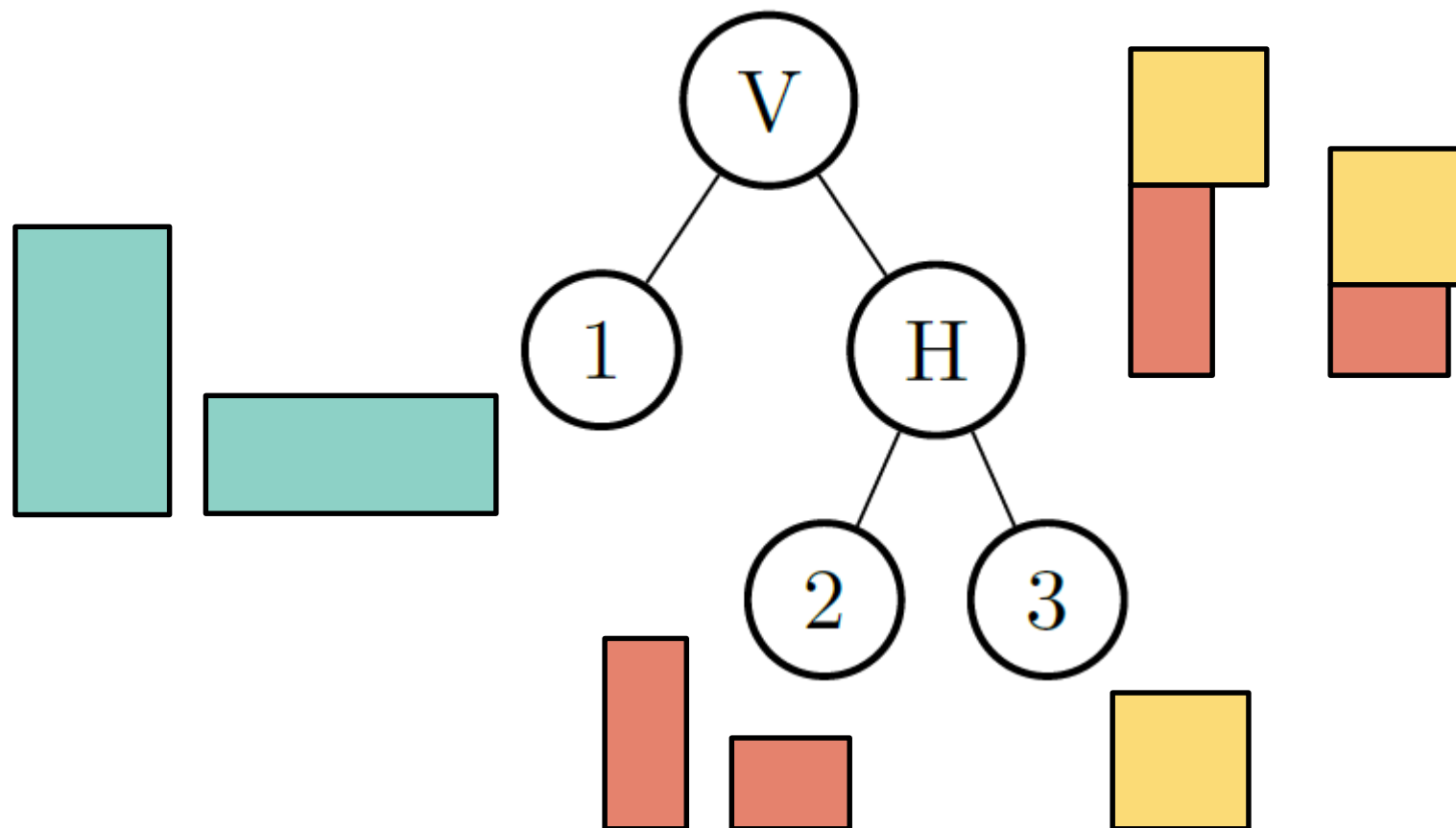
Realizing Slicing Trees

- Recursively calculate shapes at each node in the tree [Cheng & Wong 2006]
- Realizations at root encode full floorplans



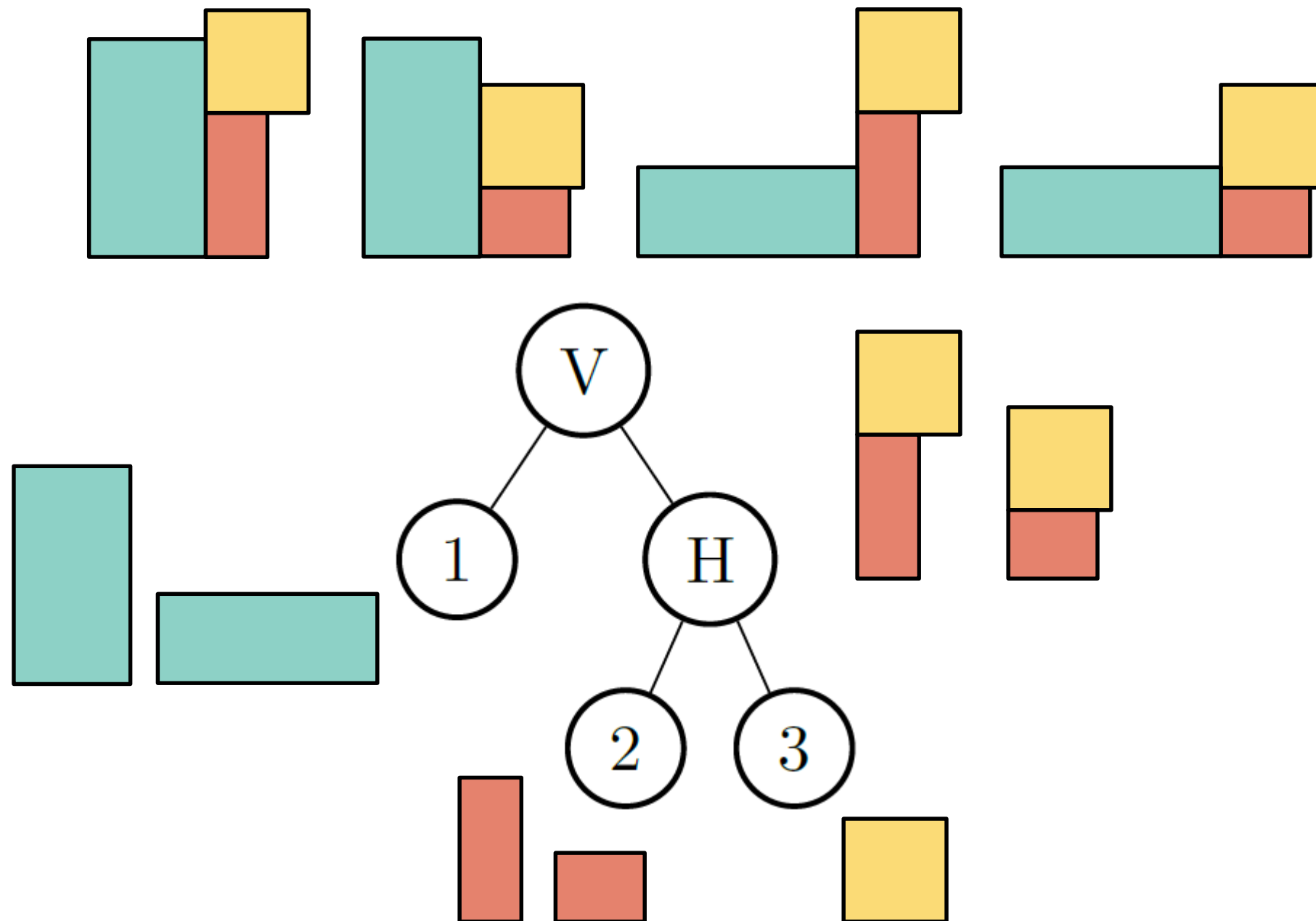
Realizing Slicing Trees

- Recursively calculate shapes at each node in the tree [Cheng & Wong 2006]
- Realizations at root encode full floorplans



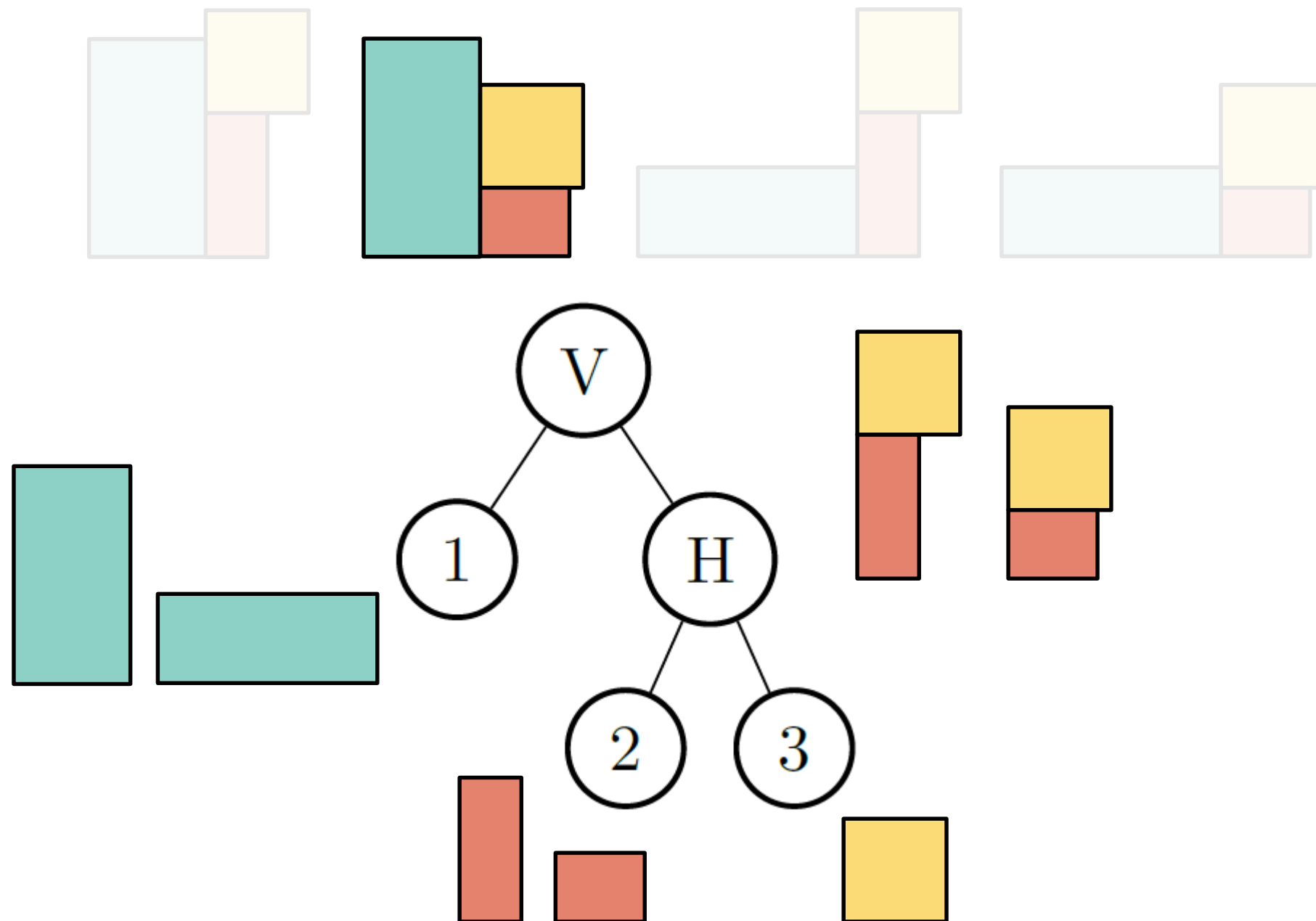
Realizing Slicing Trees

- Recursively calculate shapes at each node in the tree [Cheng & Wong 2006]
- Realizations at root encode full floorplans

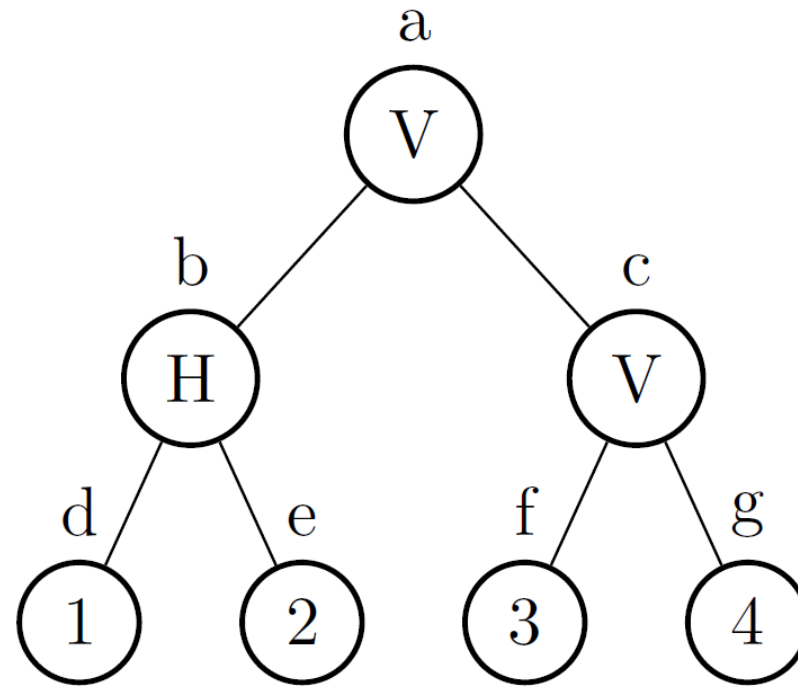


Realizing Slicing Trees

- Recursively calculate shapes at each node in the tree [Cheng & Wong 2006]
- Realizations at root encode full floorplans



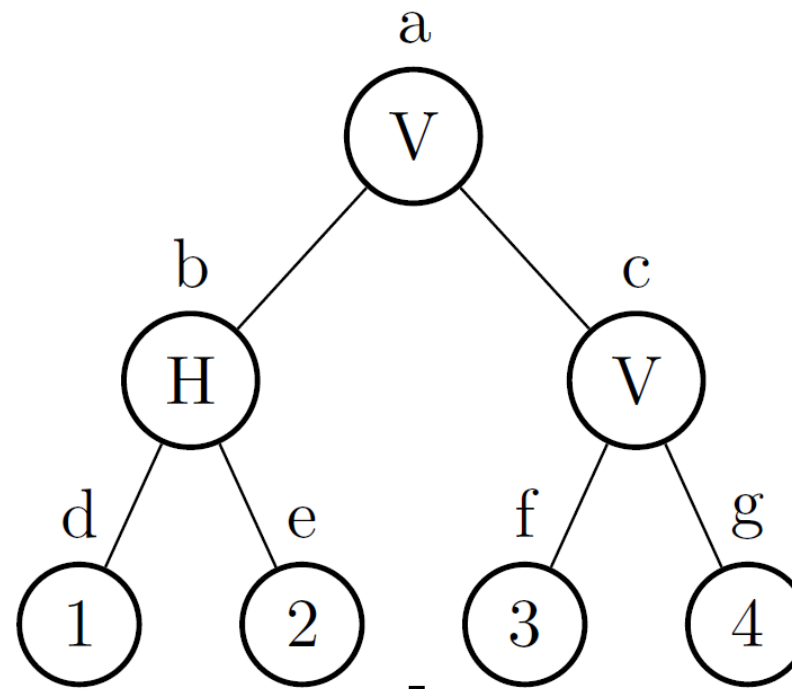
Algorithmic Enhancements



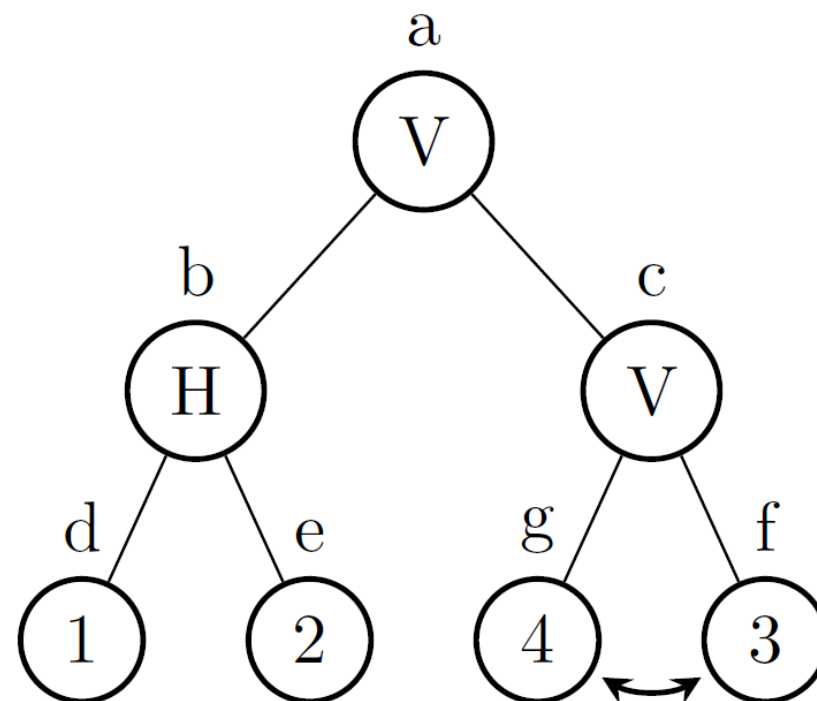
2	3	4
1		



Algorithmic Enhancements



↓ Exchange 3 & 4

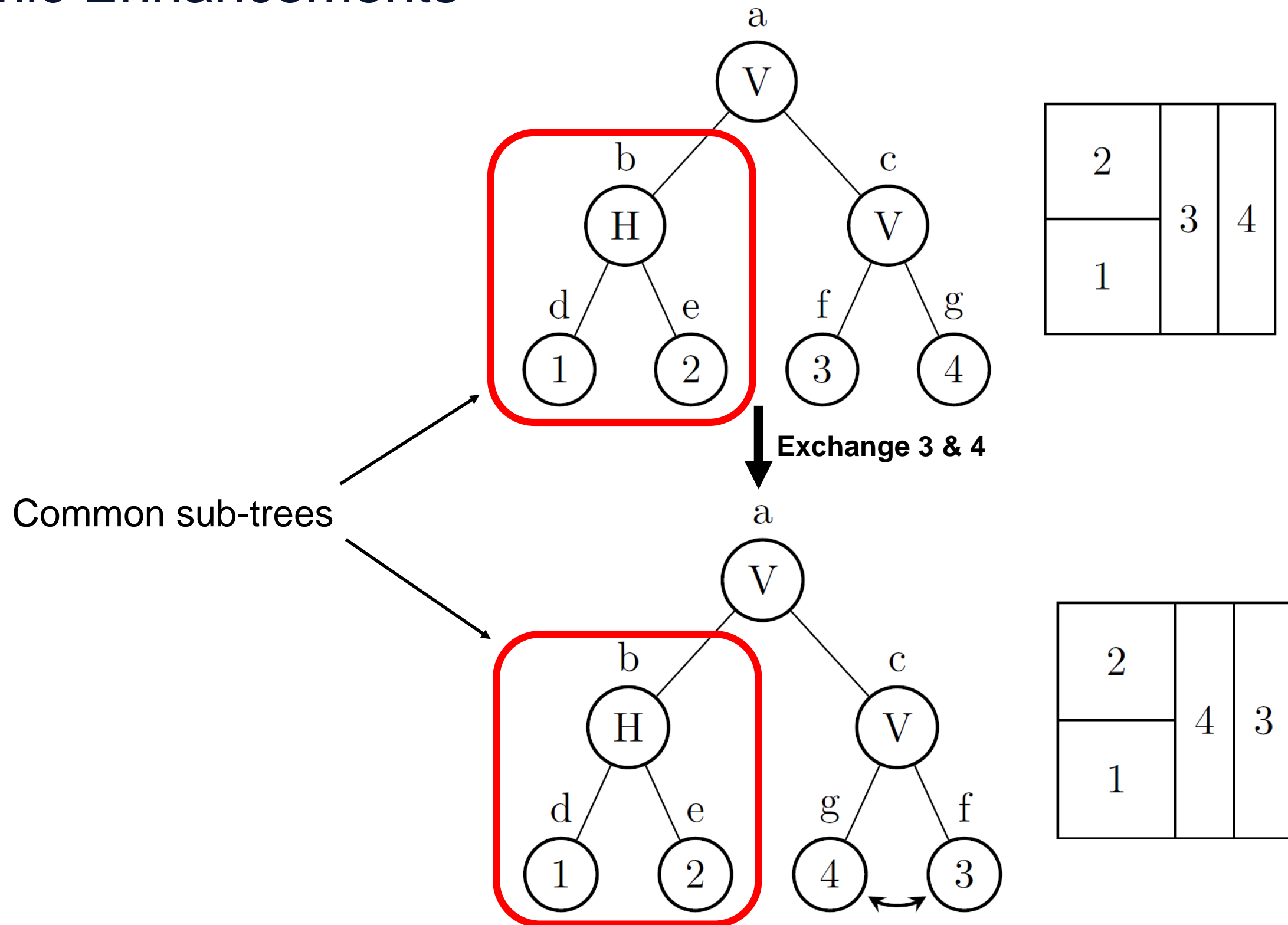


2	3	4
1		

2	4	3
1		



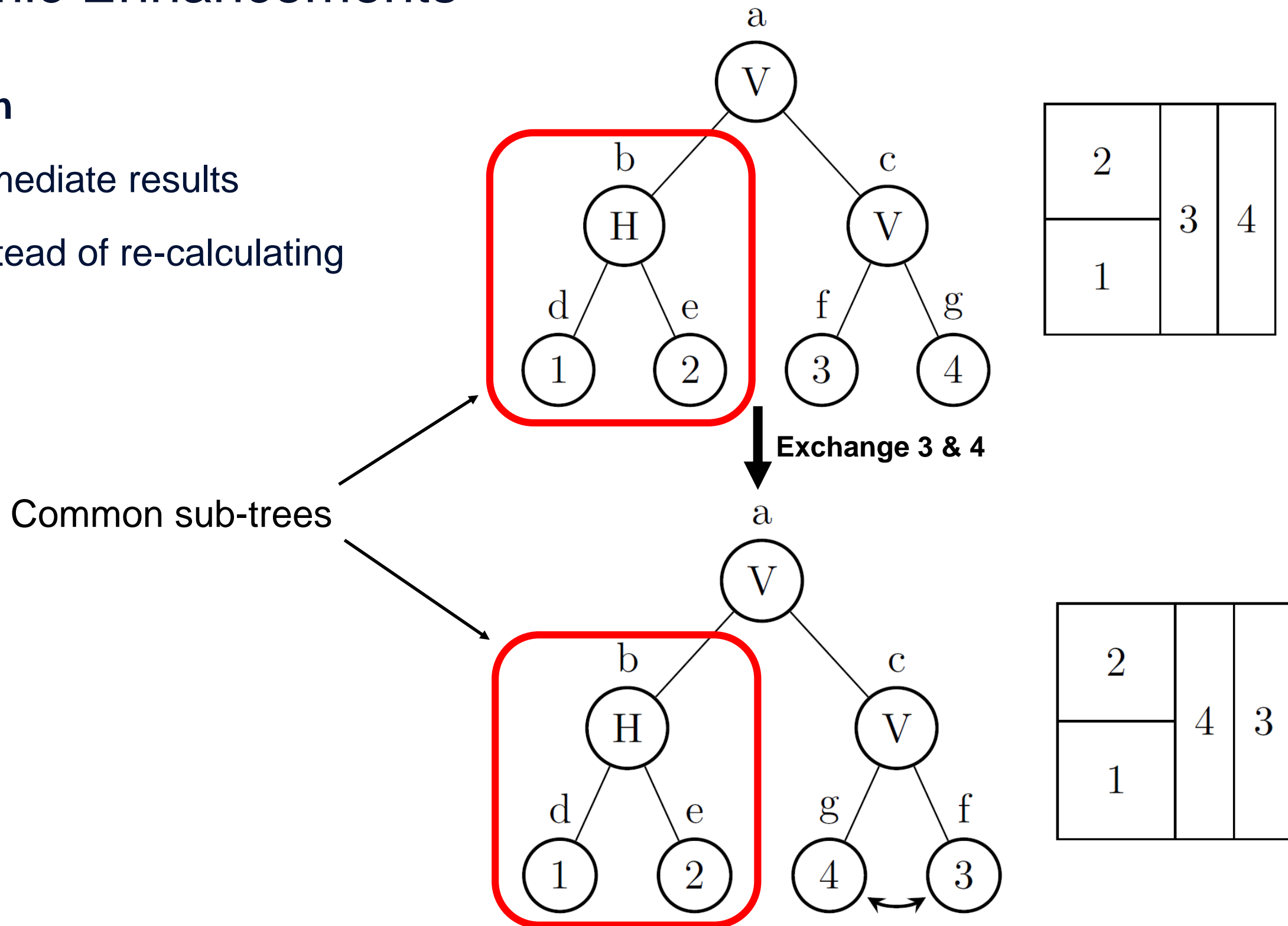
Algorithmic Enhancements



Algorithmic Enhancements

Memoization

- Save intermediate results
- Re-use instead of re-calculating



Algorithmic Enhancements

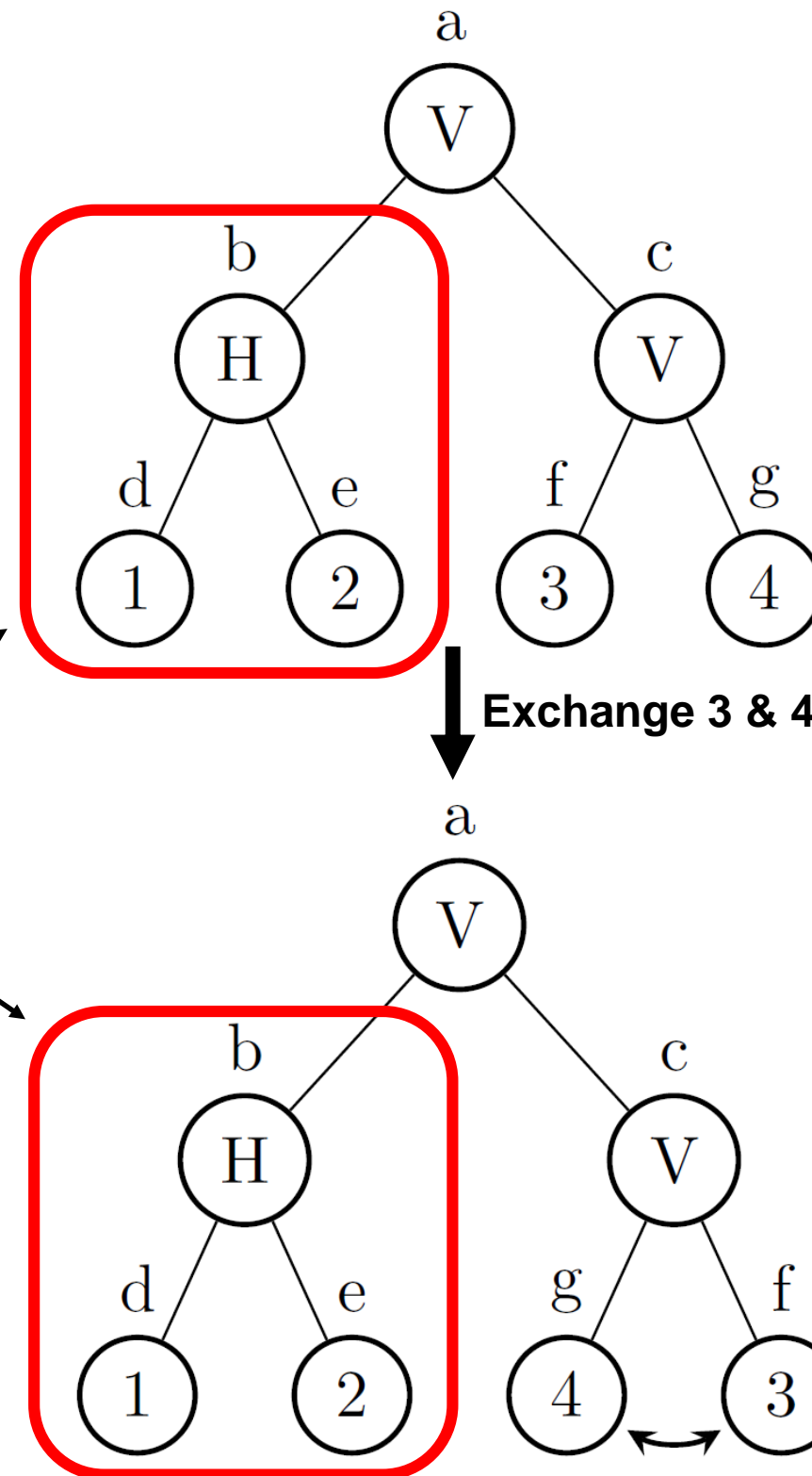
Memoization

- Save intermediate results
- Re-use instead of re-calculating

Common sub-trees

Lazy Evaluation

- Calculate leaf shapes as needed to avoid wasted work
- Important for non-tileable FPGAs



2	3	4
1		

2	4	3
1		



Impact of Algorithmic Enhancements

Configuration	Speed-Up
Baseline	1.0X
Memoization	2.3X
Lazy Evaluation	5.4X
Memoization & Lazy Evaluation	15.6X

- Titan Benchmarks: 90K – 550K primitives
- **Average run-time:** 9 minutes @ 32 partitions

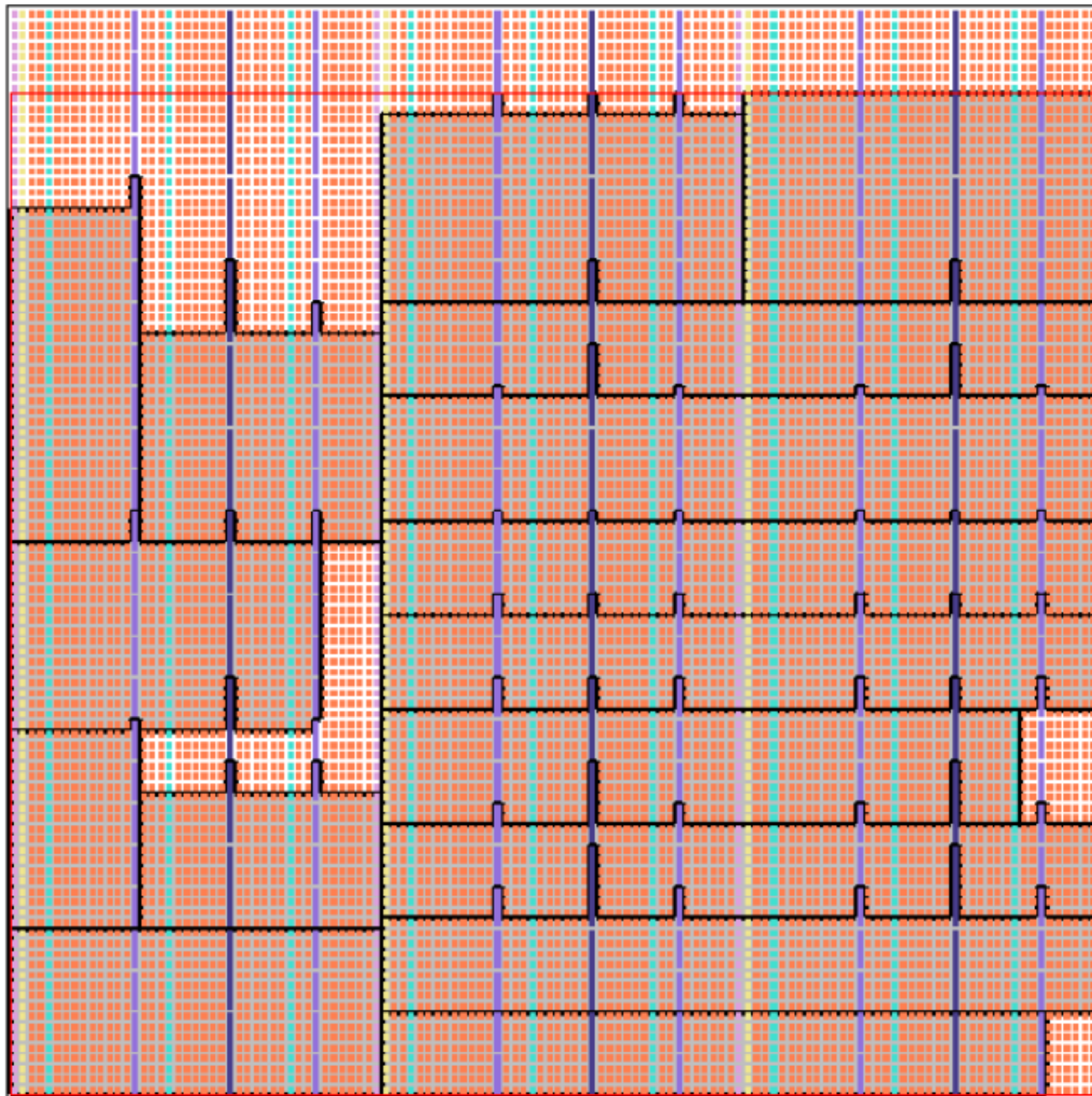


Floorplan Legality



How to ensure legal solution?

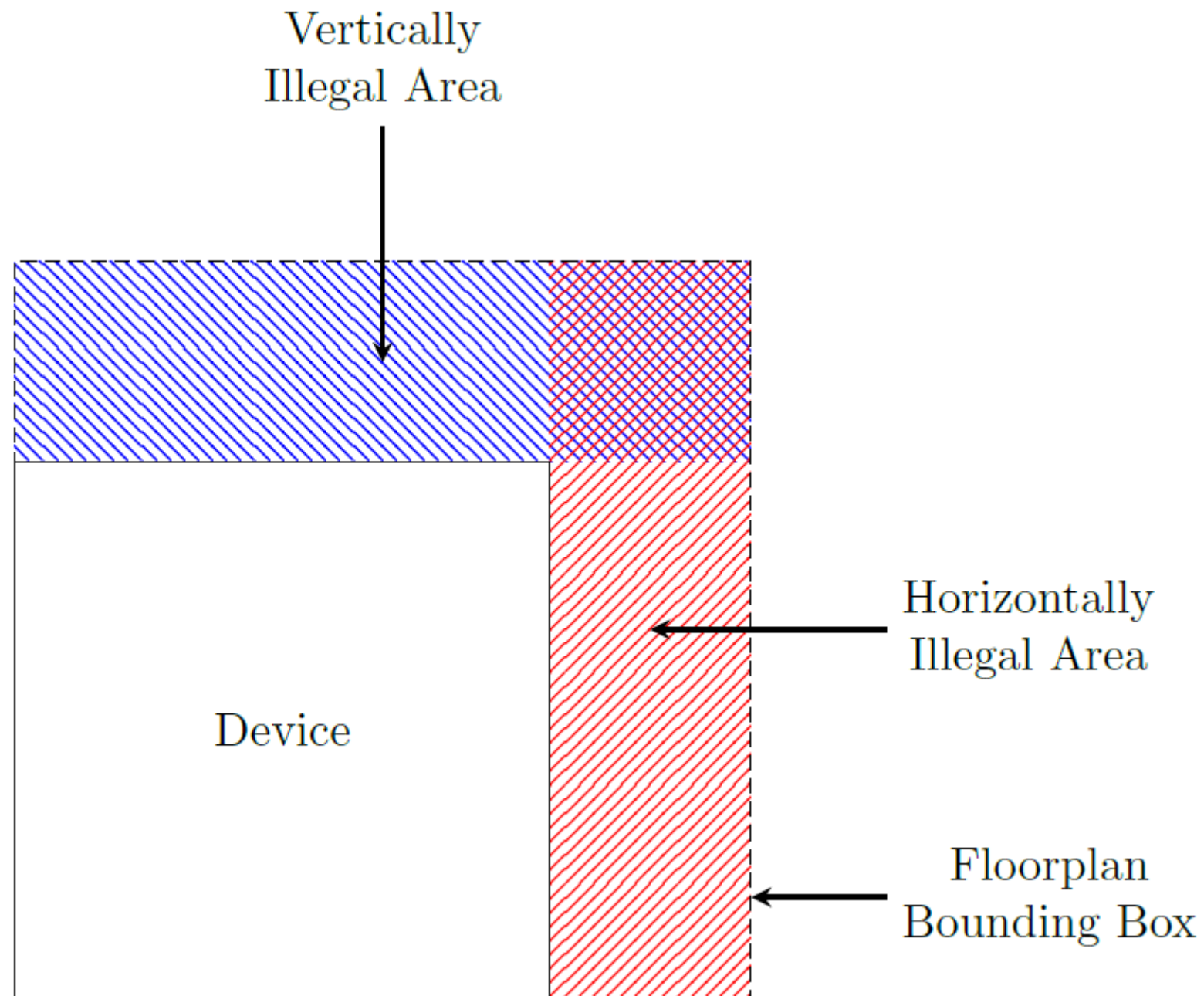
- Impractical to forbid illegal solutions
- Cost penalty: Floorplan area outside the device



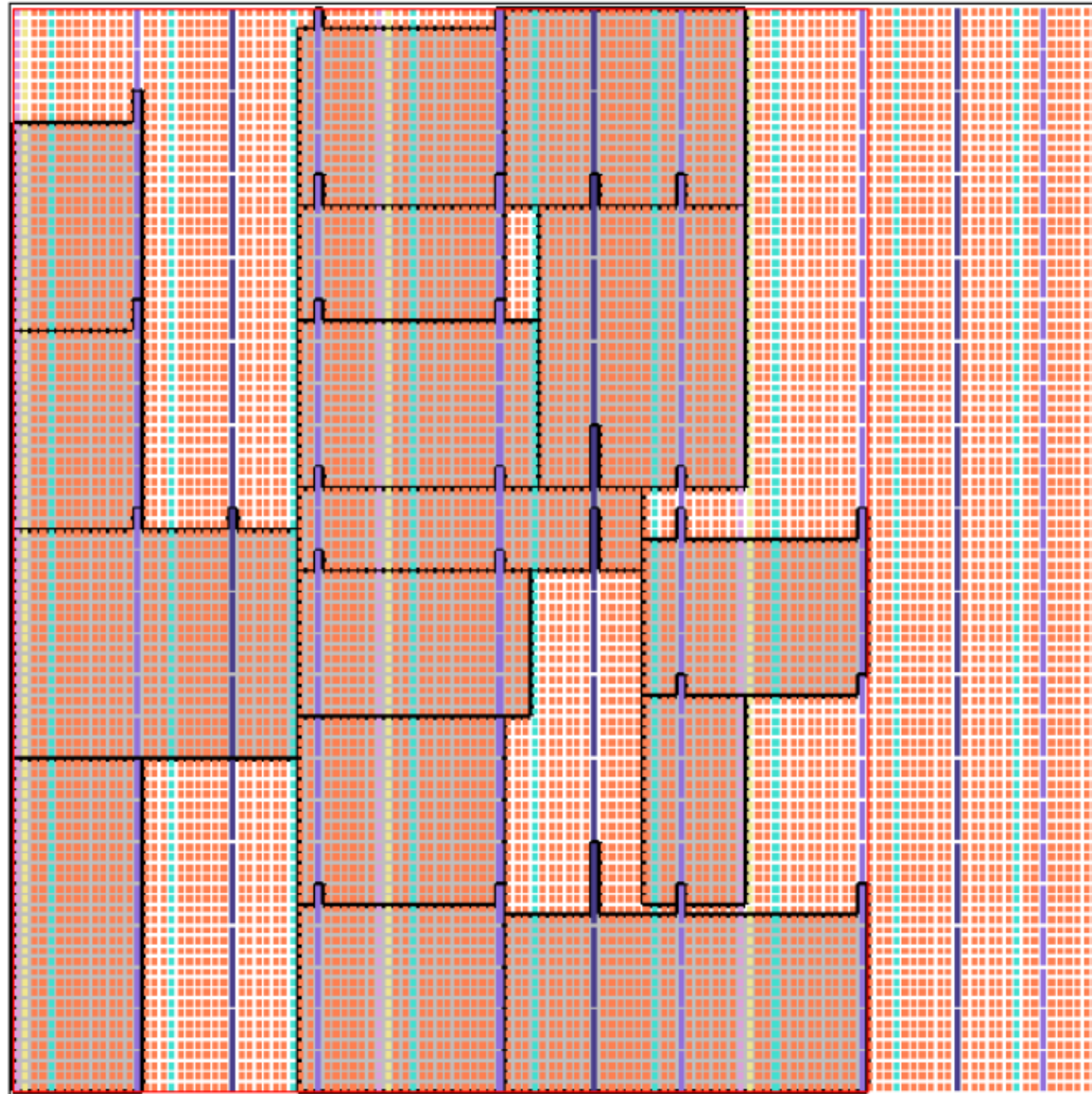
One column
too wide!

Split Cost Penalty

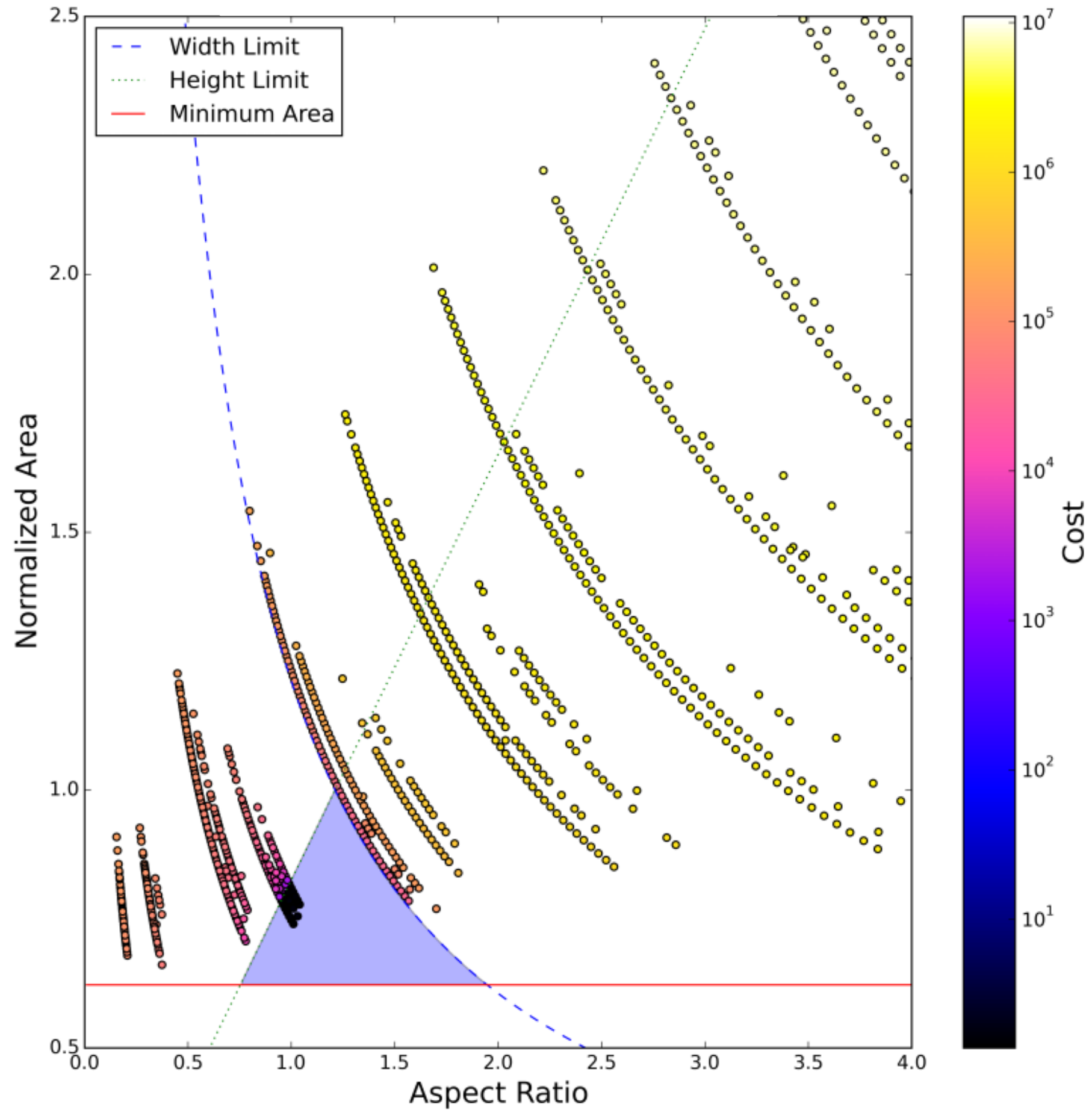
- Use separate cost penalties for horizontal and vertical legality



Legal Solution

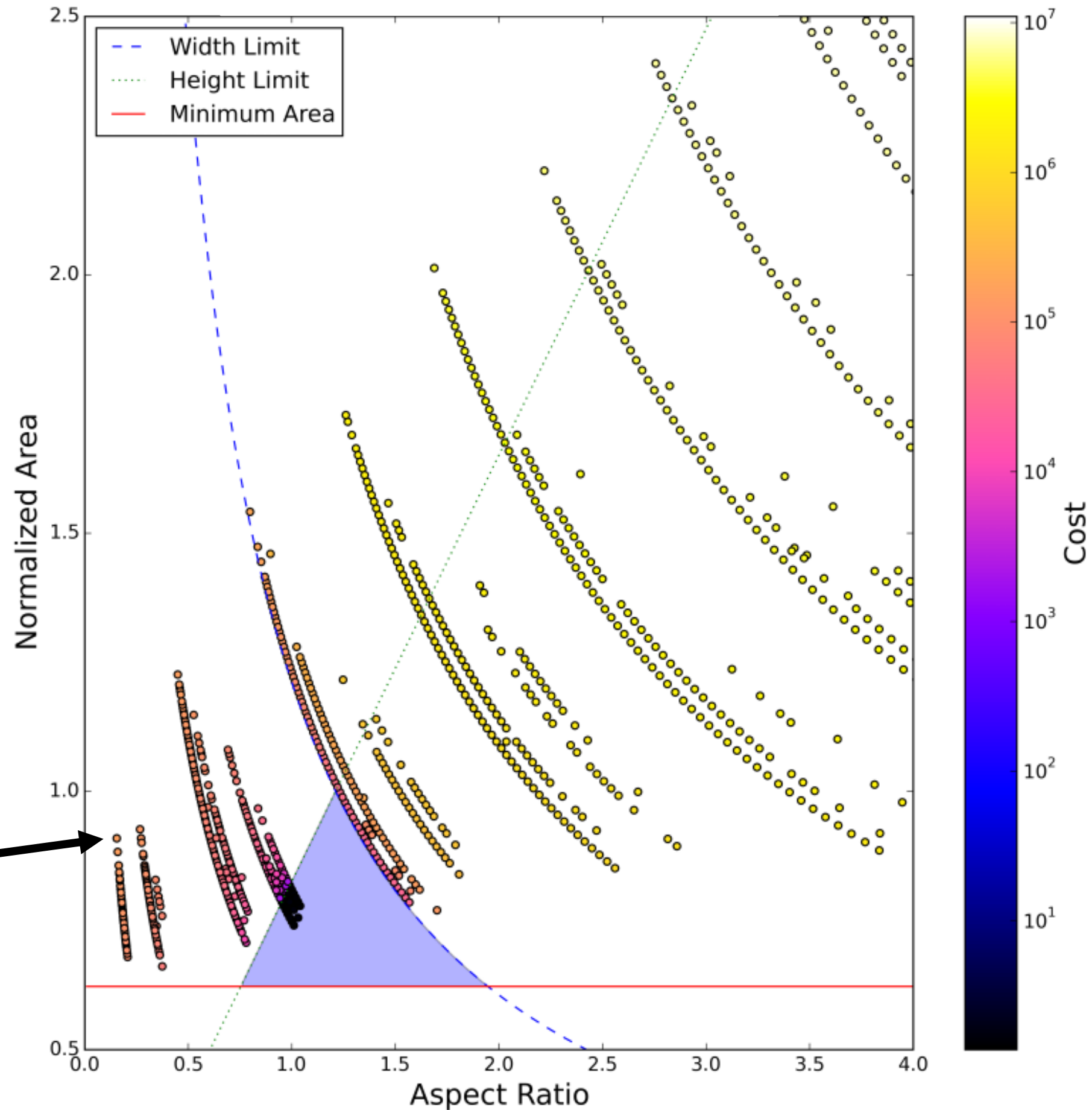
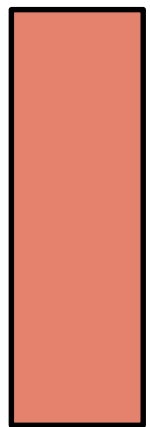


Search Space

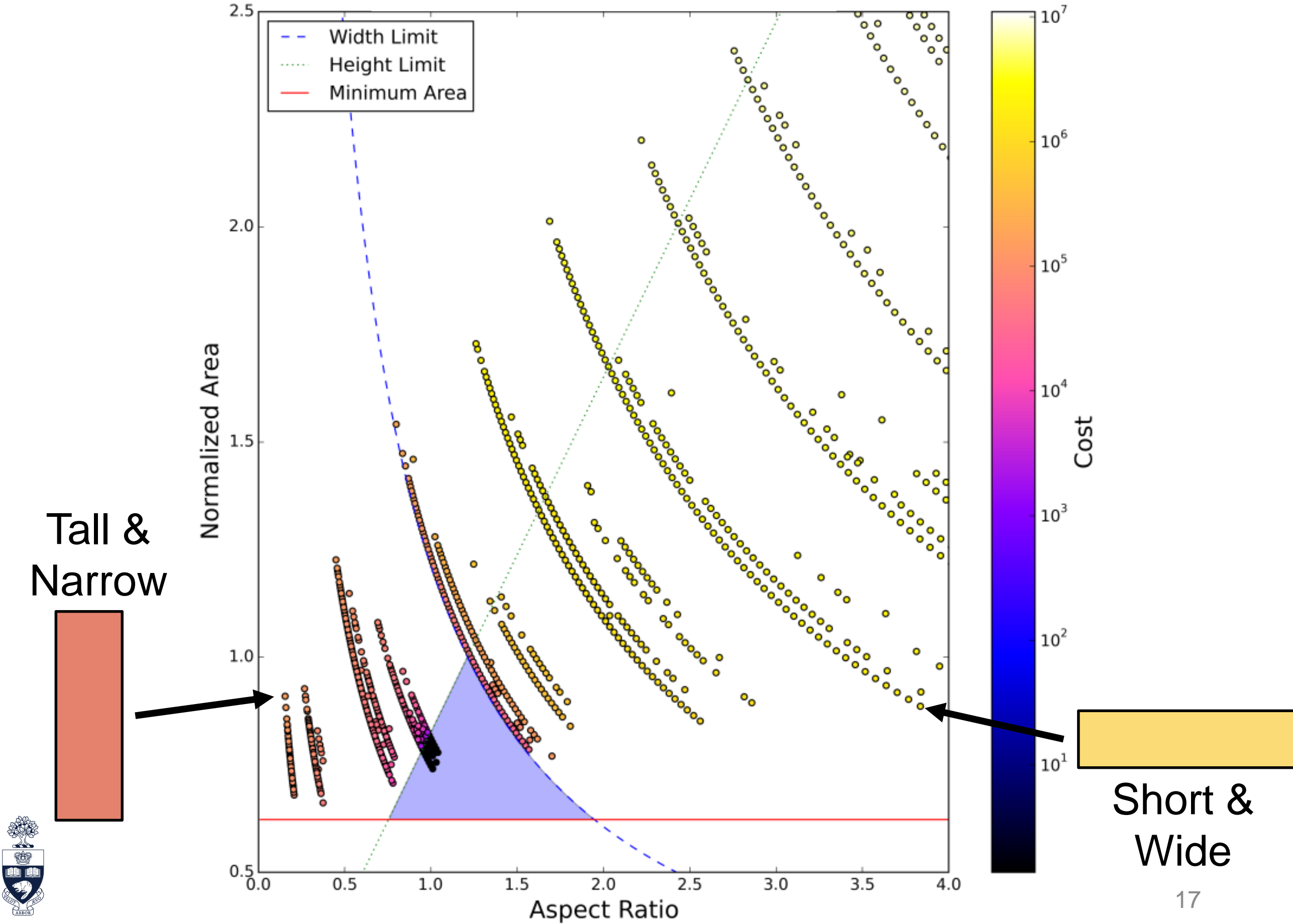


Search Space

Tall &
Narrow

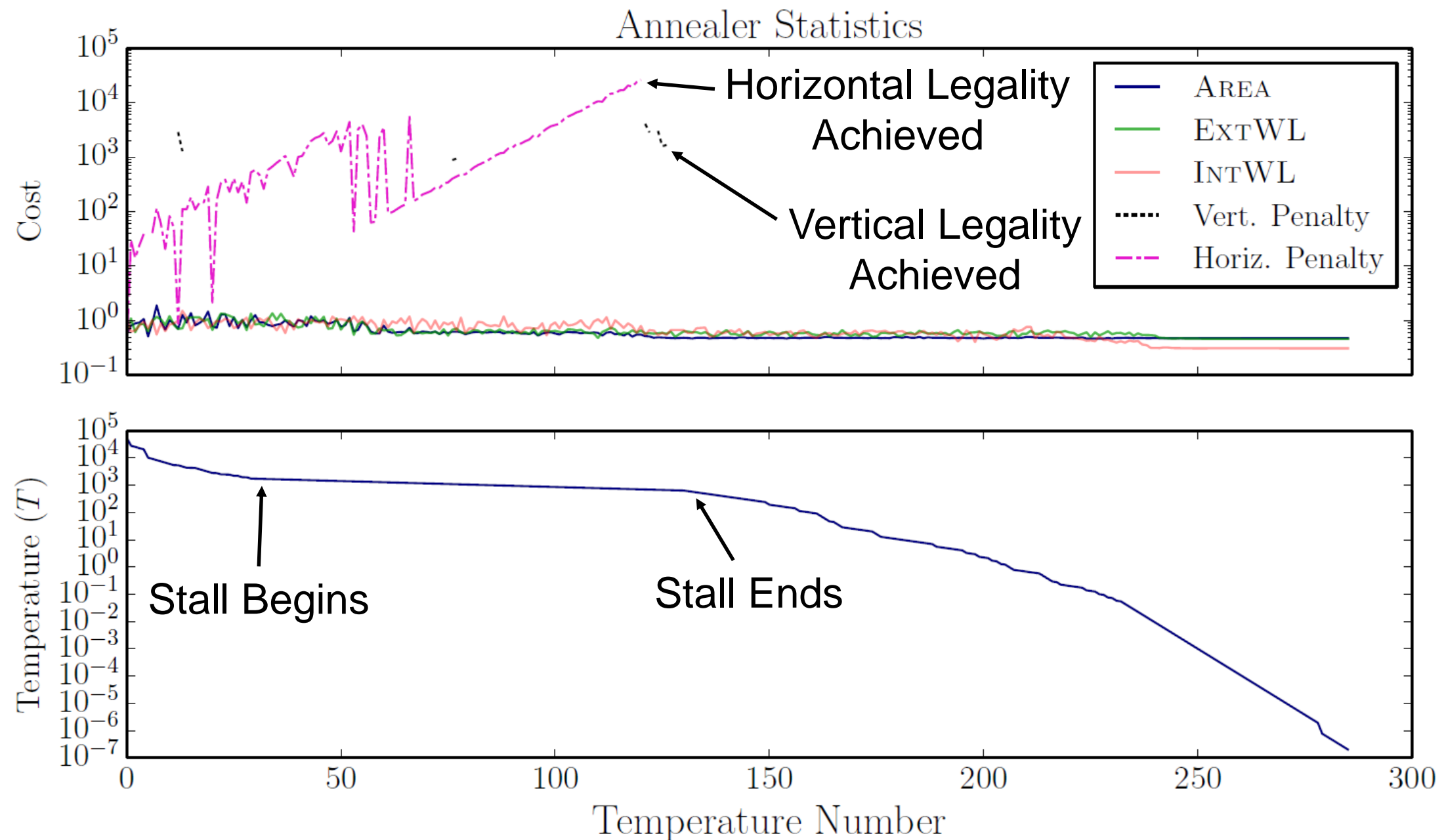


Search Space



Adaptive Legality

- Need robust cost penalty
- Dynamically adapt penalty based on legal acceptance rate
- Stall the anneal until legality achieved

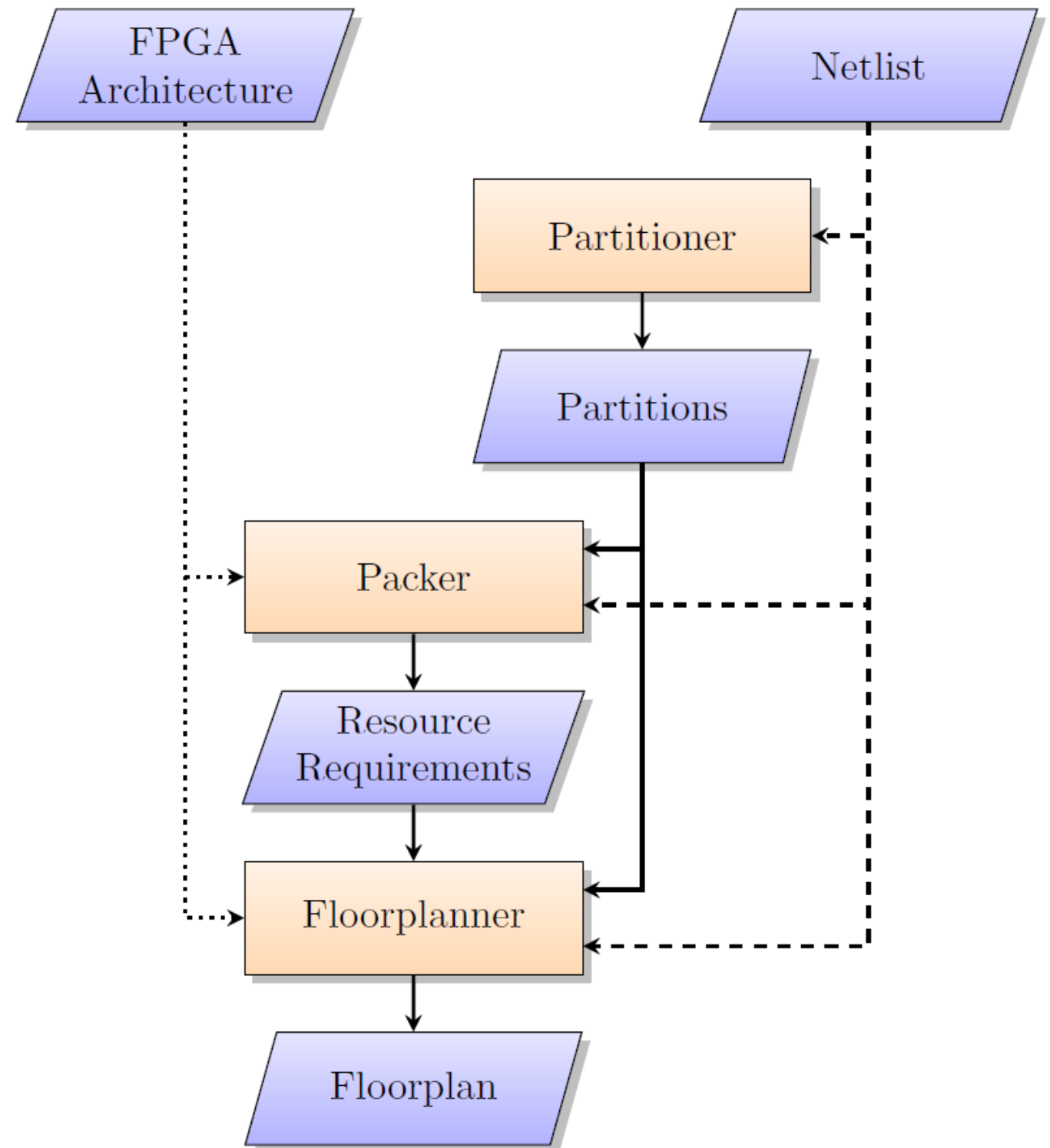


Experimental Results

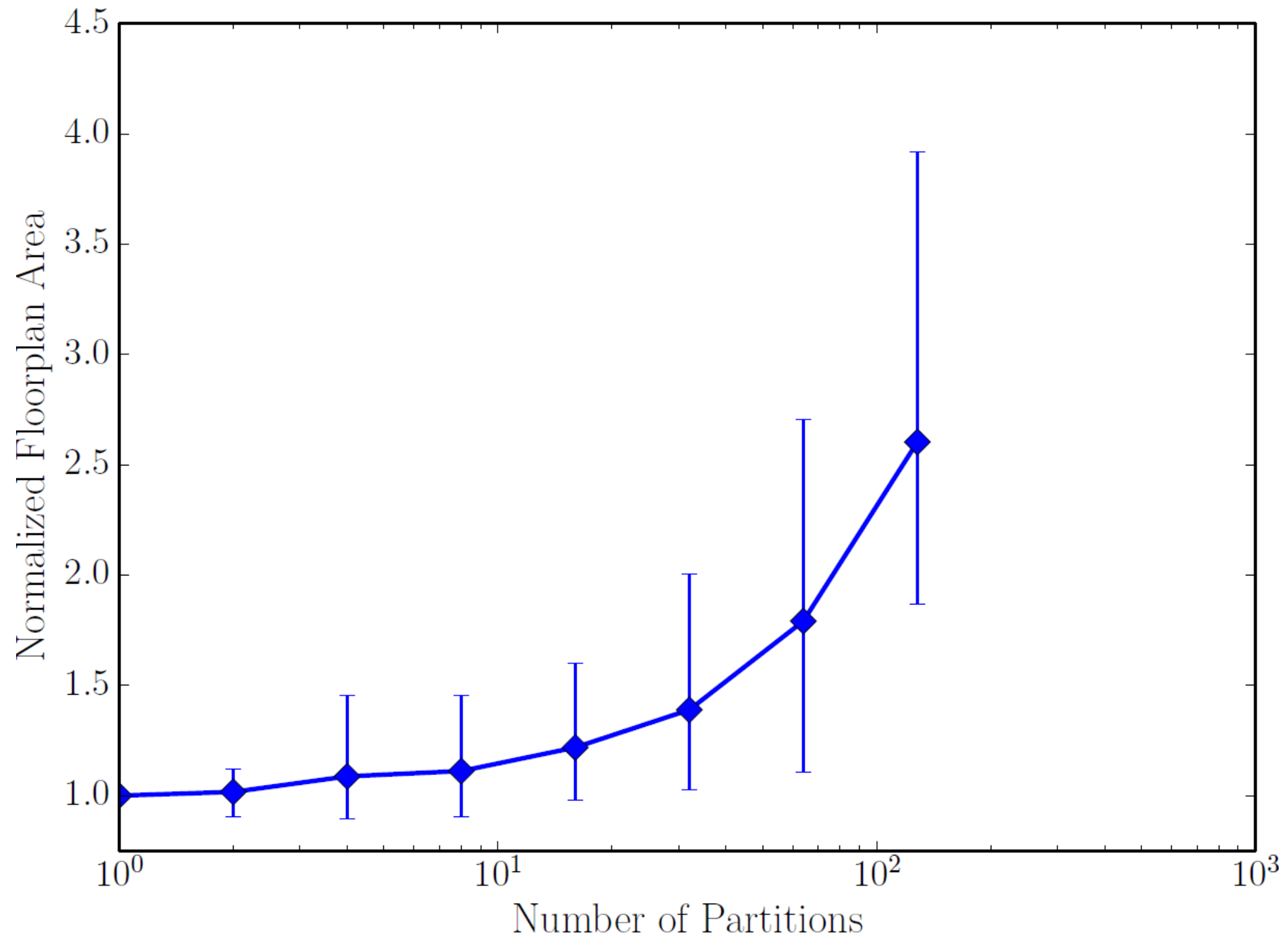


Experimental Setup

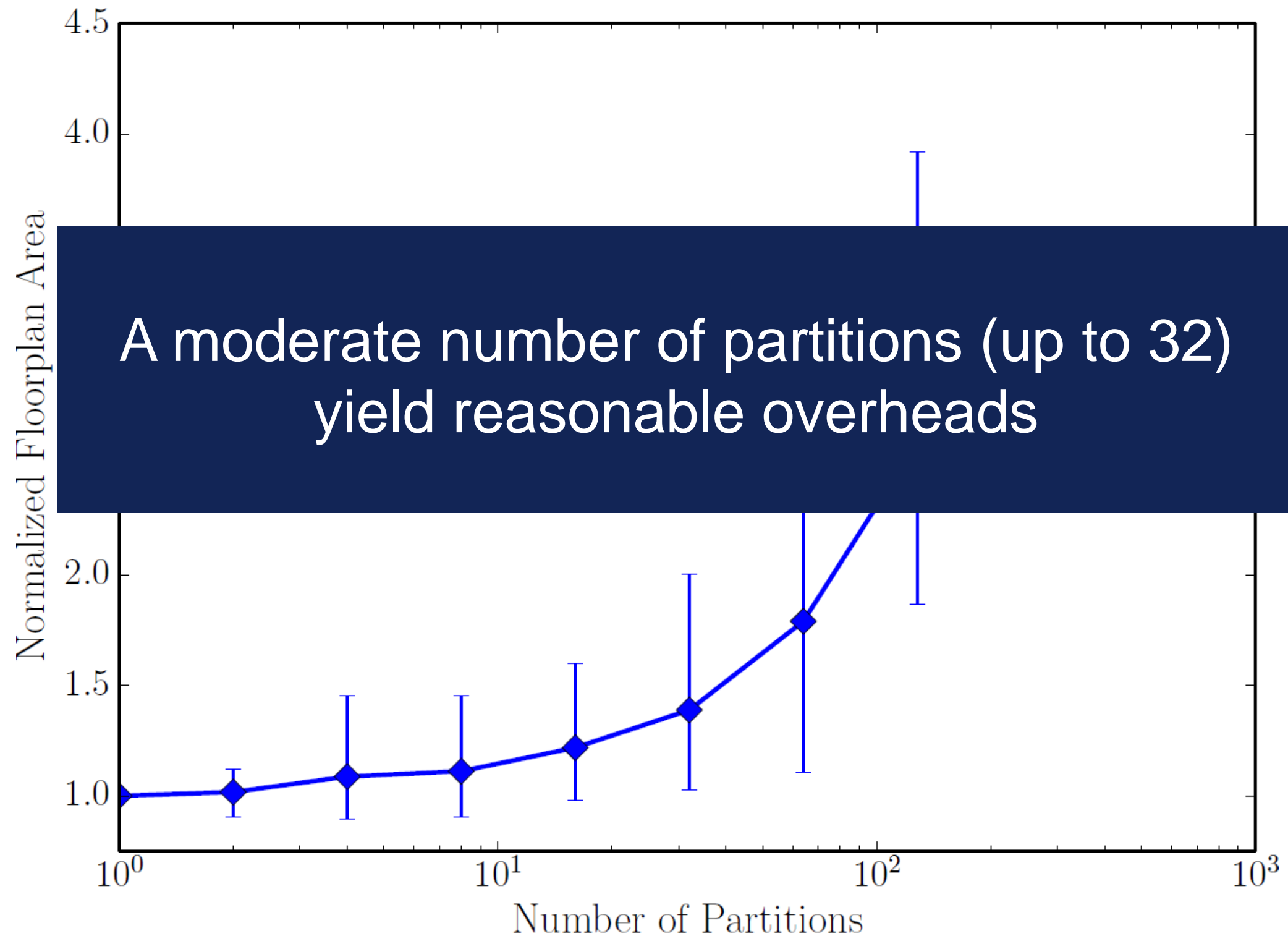
- **Benchmarks:** Titan (90K - 550K primitives)
- **Architecture:** Stratix IV-like
- **Partitioner:** Metis
- **Packer:** VPR
- **Floorplanner:** Hetris
 - Area and Wirelength Optimization



Floorplan Area and Number of Partitions



Floorplan Area and Number of Partitions



Comparison with Quartus II

- Scalable benchmark (Cascaded FIR filters)
 - Limited by DSP blocks on EP4SGX230 device
- Consider both 1-FIR and 2-FIR instances per partition

Automated Design Flow	Max. FIR Inst. 1-FIR	Max. FIR Inst. 2-FIR
Quartus II	37	40
Hetris Default	38	44
Hetris High-Effort	39	44



Conclusion and Future Work



Conclusion

- Hetris open source FPGA floorplanning tool
- Algorithmic enhancements yielding 15.6x speed-up
- Adaptive optimization techniques to robustly handle legality
- First evaluation of FPGA floorplanning using realistic benchmarks and architectures



Future Work

Hetris

- Further algorithmic enhancements
- Timing-driven optimization
- Support for non-rectangular shapes

Design Flow

- Improved automated design partitioning
- Full post-place & route evaluation of floorplanning



Thanks!

Questions?

Email: kmurray@eecg.utoronto.ca

HETRIS Release:

uoft.me/hetris



UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE & ENGINEERING