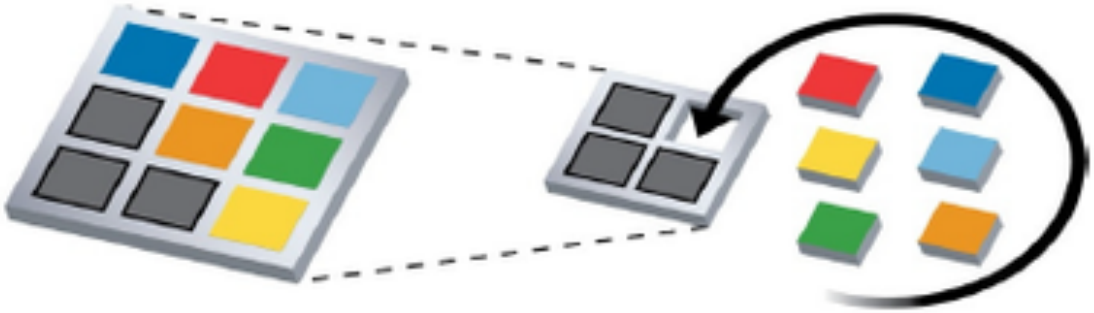
PRFloor: An Automatic Floorplanner for Partially Reconfigurable FPGA Systems

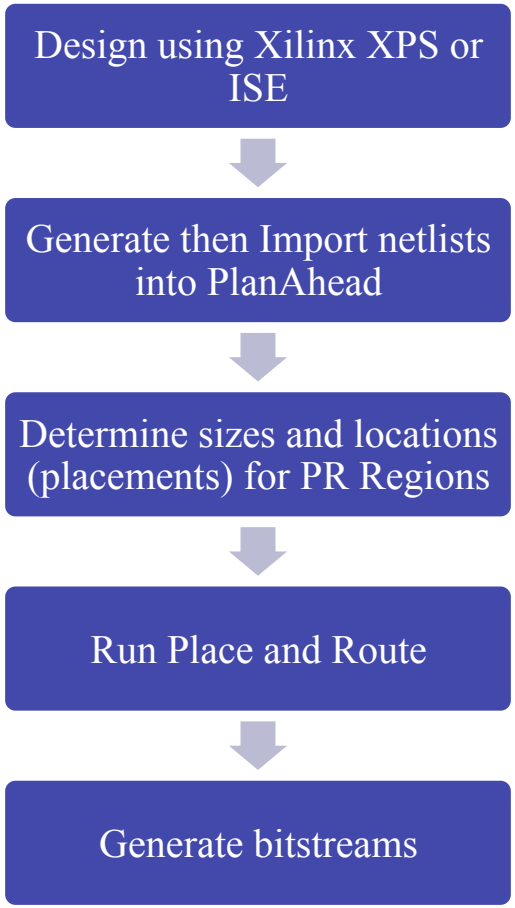
Tuan D. A. Nguyen [1] & Akash Kumar [2]

(1) ECE Department, National University of Singapore, Singapore
(2) Chair of Processor Design, Center for Advancing Electronics Dresden, TU Dresden, Germany

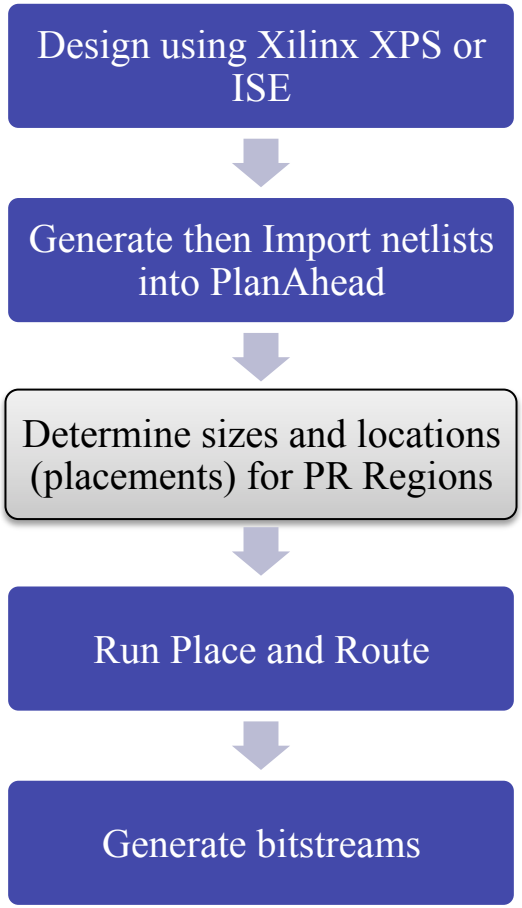
Partial Reconfiguration (PR)



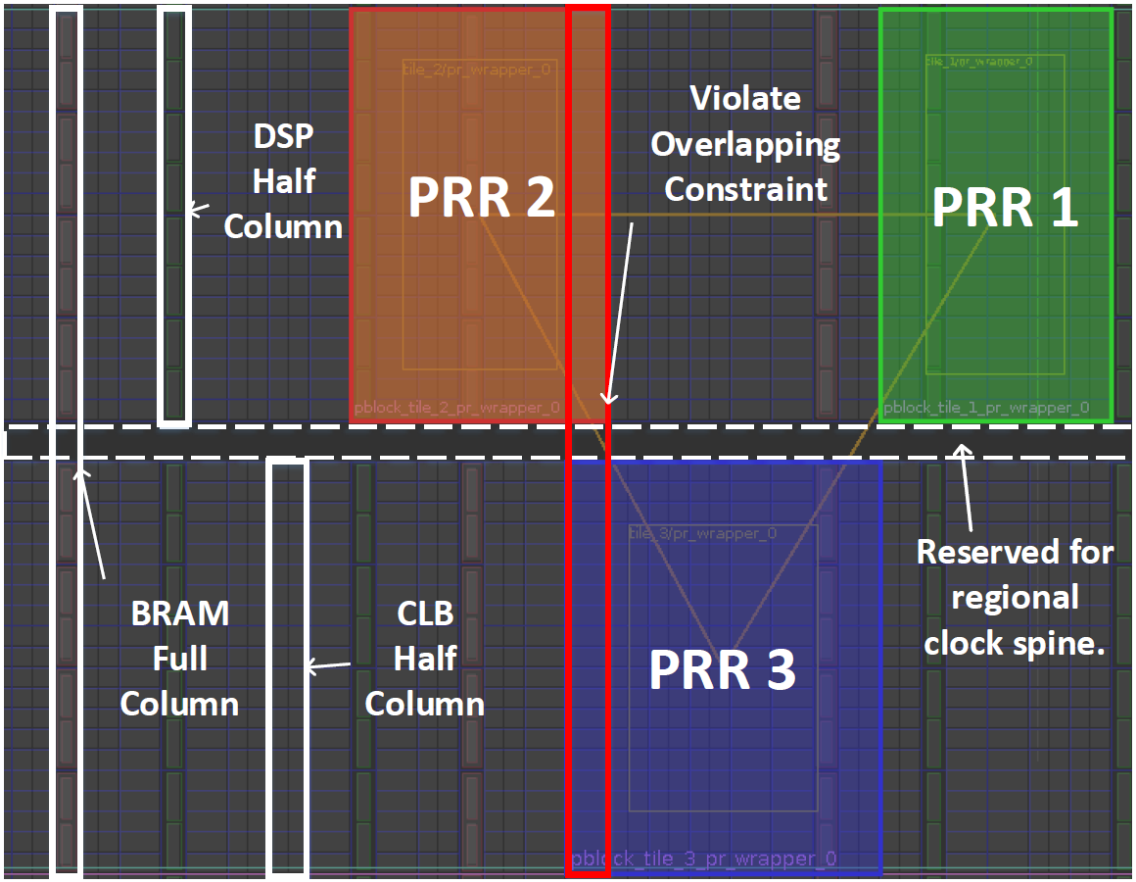
Xilinx ISE PR Design Flow



Xilinx ISE PR Design Flow

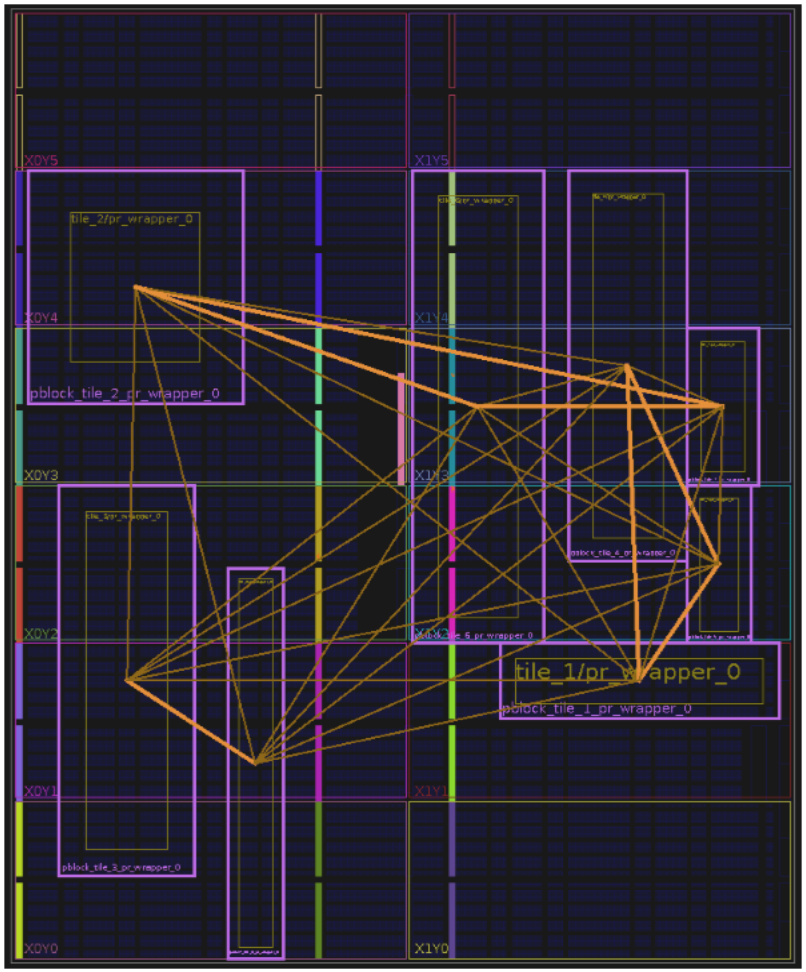


Floorplanning

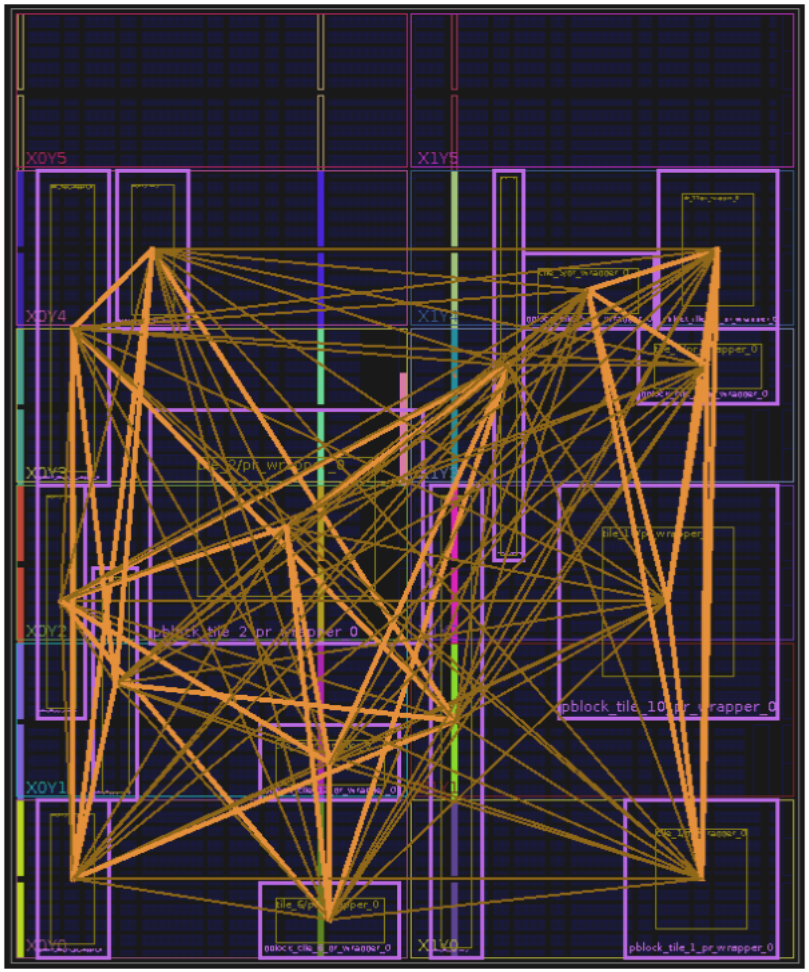


Problem?

Problem?

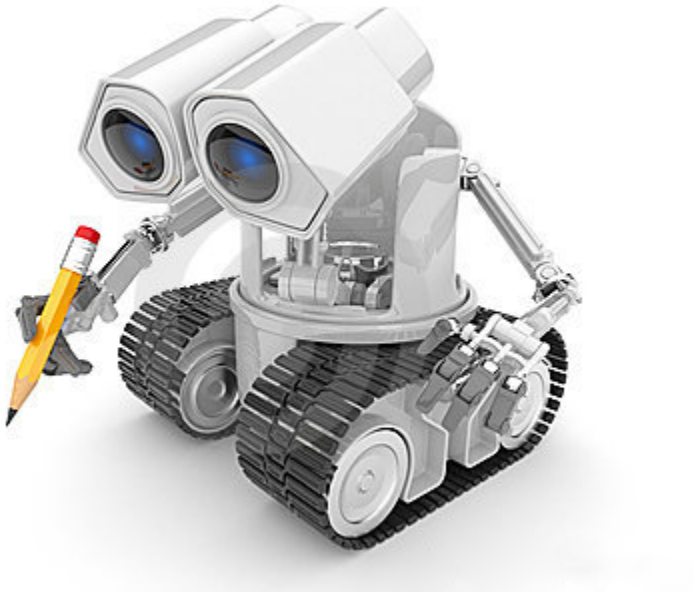


8 PRRs

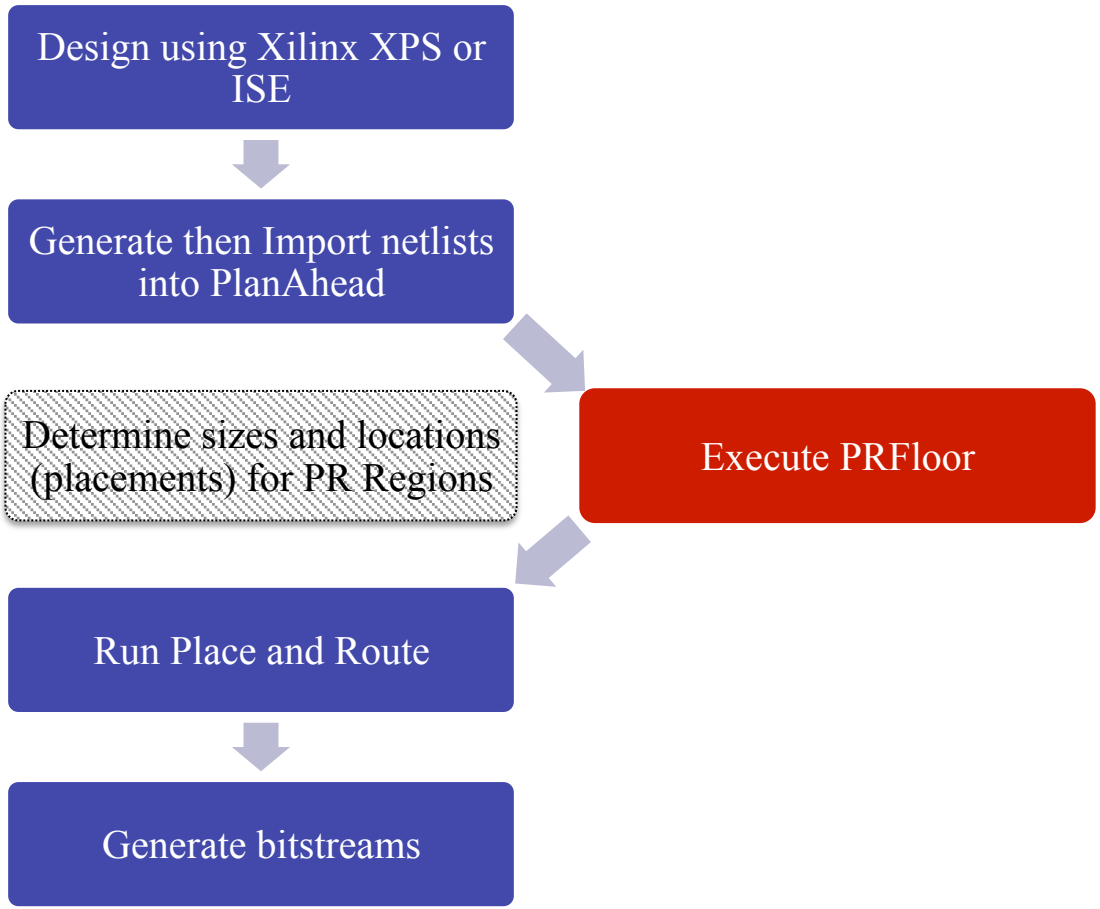


15 PRRs

So?



PRFloor


Design using Xilinx XPS or ISE
Generate then Import netlists into PlanAhead
Determine sizes and locations (placements) for PR Regions
Execute PRFloor
Run Place and Route
Generate bitstreams

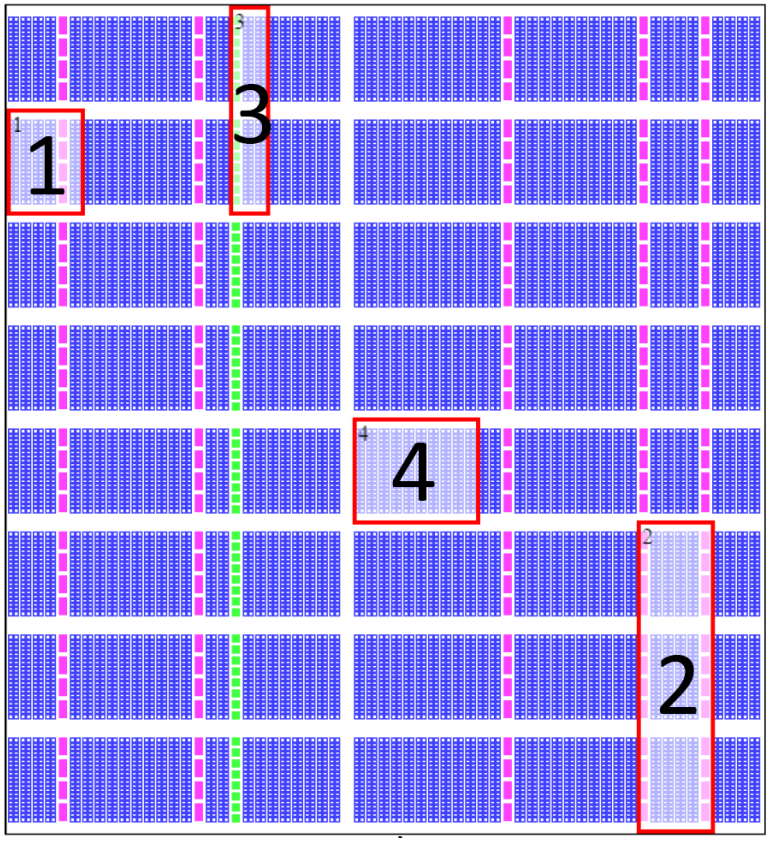
Common Issue of Previous Works

NUS National University of Singapore

Only consider PR regions (PRRs)

[Rabozzi14, Duhem13, Vipin12, Bolchini11, Montone11, Montone08]

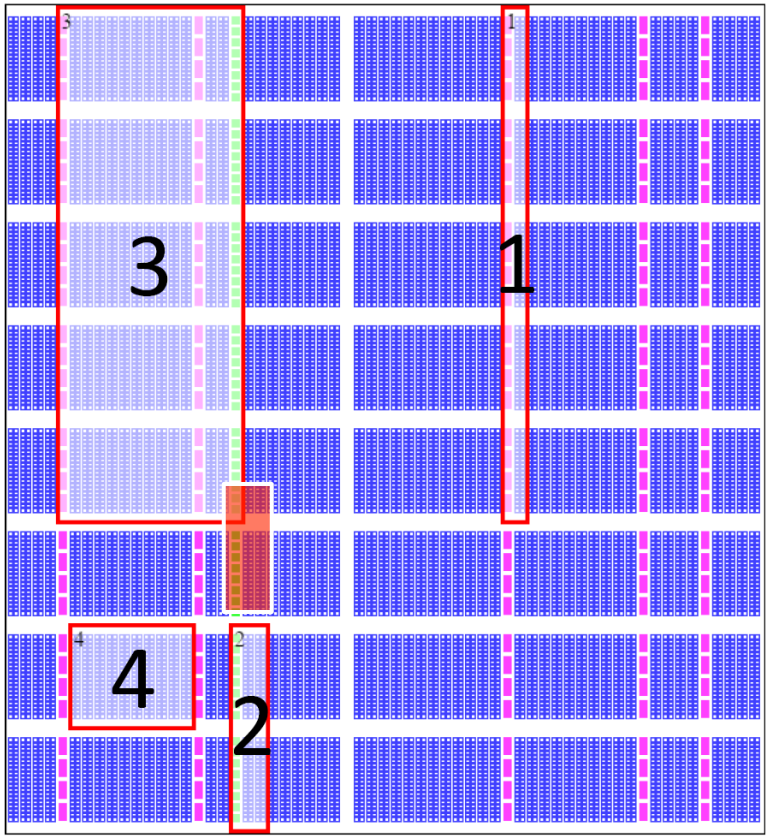
Common Issue



1. PRR 1 and 2 are too far away

[Rabozzi14] M. Rabozzi, J. Lillis, and M. D. Santambrogio. Floorplanning for Partially-Reconfigurable FPGA Systems via Mixed-Integer Linear Programming. In Field-Programmable Custom Computing Machines, Annual International Symposium on, pages 186–193. IEEE, 2014.

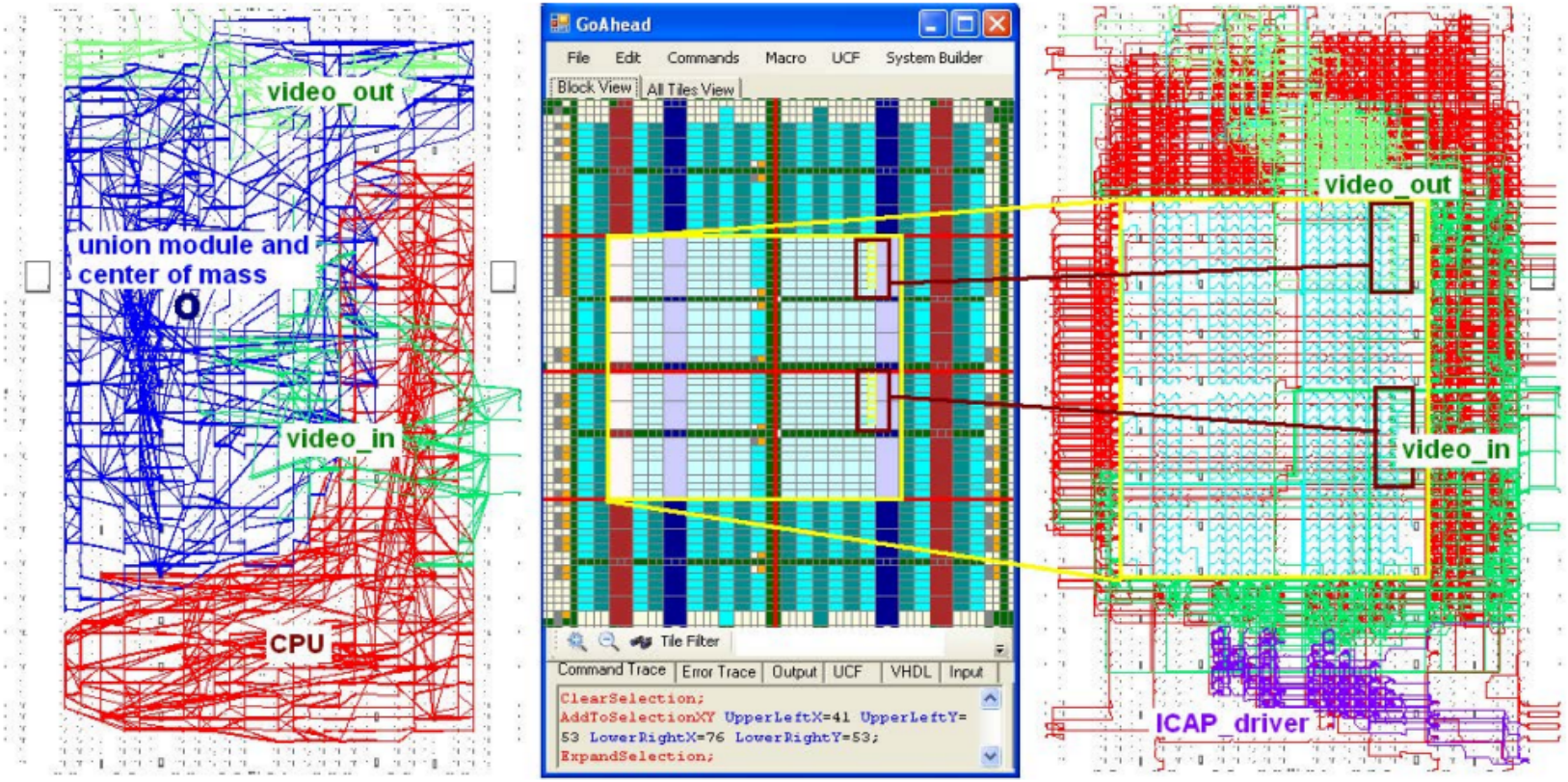
Common Issue



2. There is not enough DSP left for static module

[Rabozzi14] M. Rabozzi, J. Lillis, and M. D. Santambrogio. Floorplanning for Partially-Reconfigurable FPGA Systems via Mixed-Integer Linear Programming. In Field-Programmable Custom Computing Machines, Annual International Symposium on, pages 186–193. IEEE, 2014.

GOAHEAD [Beckhoff13]

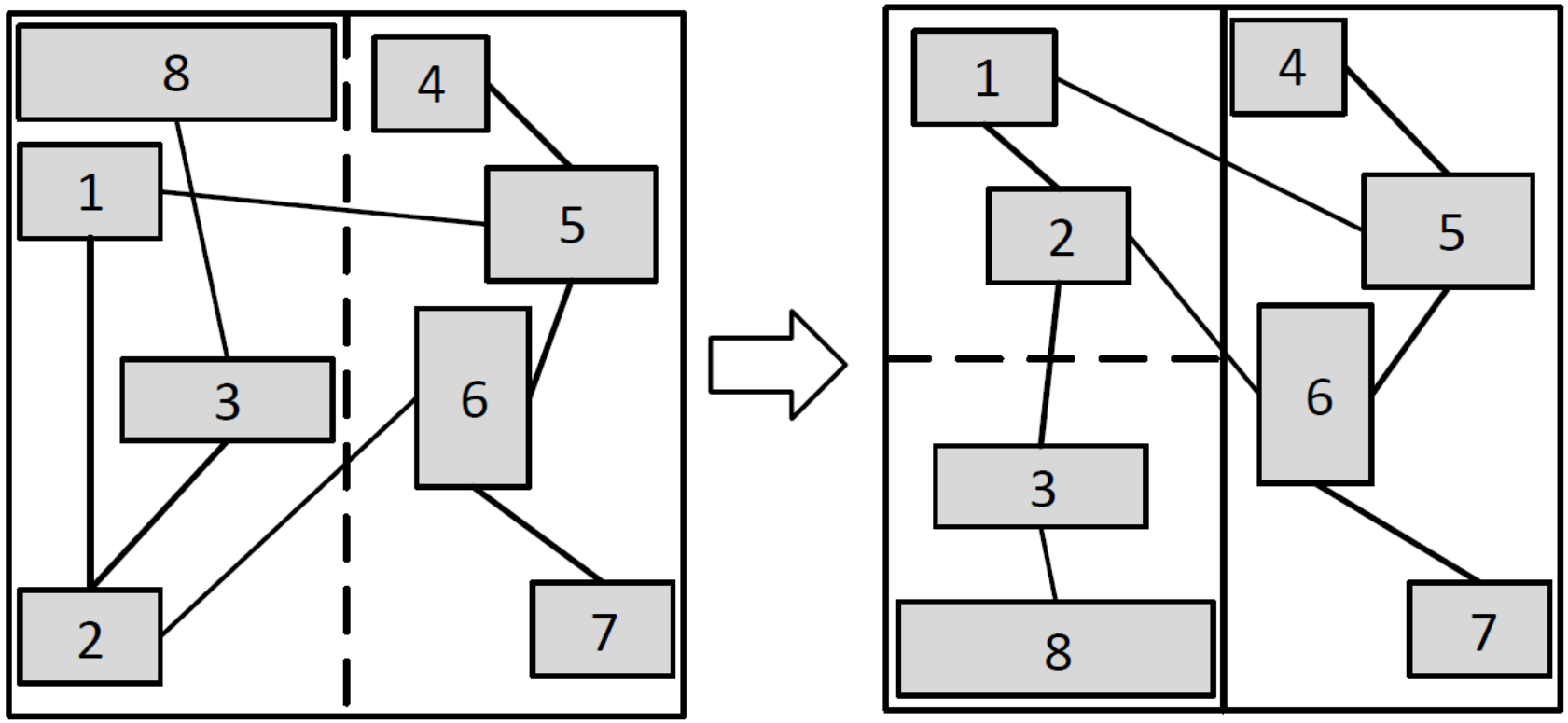


[Beckhoff13] C. Beckhoff, D. Koch, and J. Torreson. Automatic floorplanning and interface synthesis of island style reconfigurable systems with GOAHEAD. In Architecture of Computing Systems–ARCS 2013, pages 303–316. Springer, 2013.

Another issue

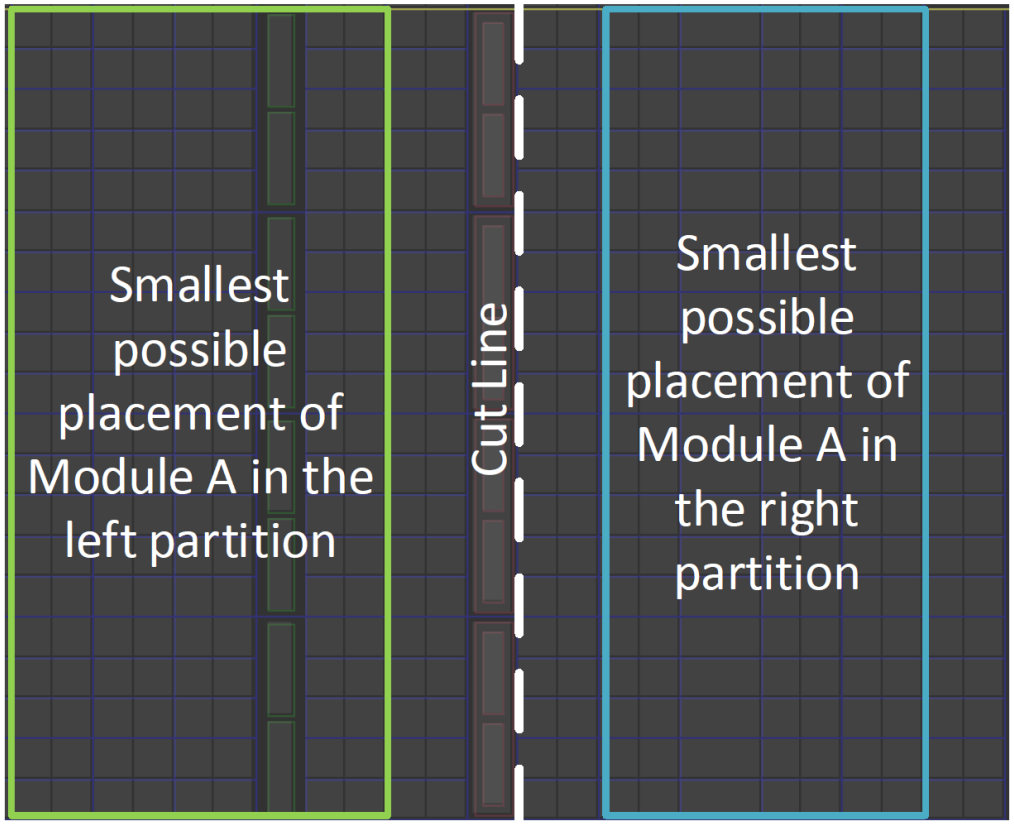
There are so many (static + PR) modules in MPSoC, up to hundreds in total

Recursive Cut-size Driven Netlist Bi-partitioning



[Yan10] J. Z. Yan and C. Chu. DeFer: deferred decision making enabled fixed-outline floorplanning algorithm. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 29(3):367–381, 2010.
[Lim08] S. K. Lim. Practical problems in VLSI physical design automation. Springer, 2008.
[Cong06] J. Cong, M. Romesis, and J. R. Shinnerl. Fast floorplanning by look-ahead enabled recursive bipartitioning. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on,25(9):1719–1732, 2006

Bipartitioning in FPGA



PRFloor - Overview

NUS National University of Singapore

Find all possible placements for modules on FPGA

↓

Use NLP-based bipartitioner to scatter the modules across the FPGA
Each module is assigned a preferred location called “anchor point”

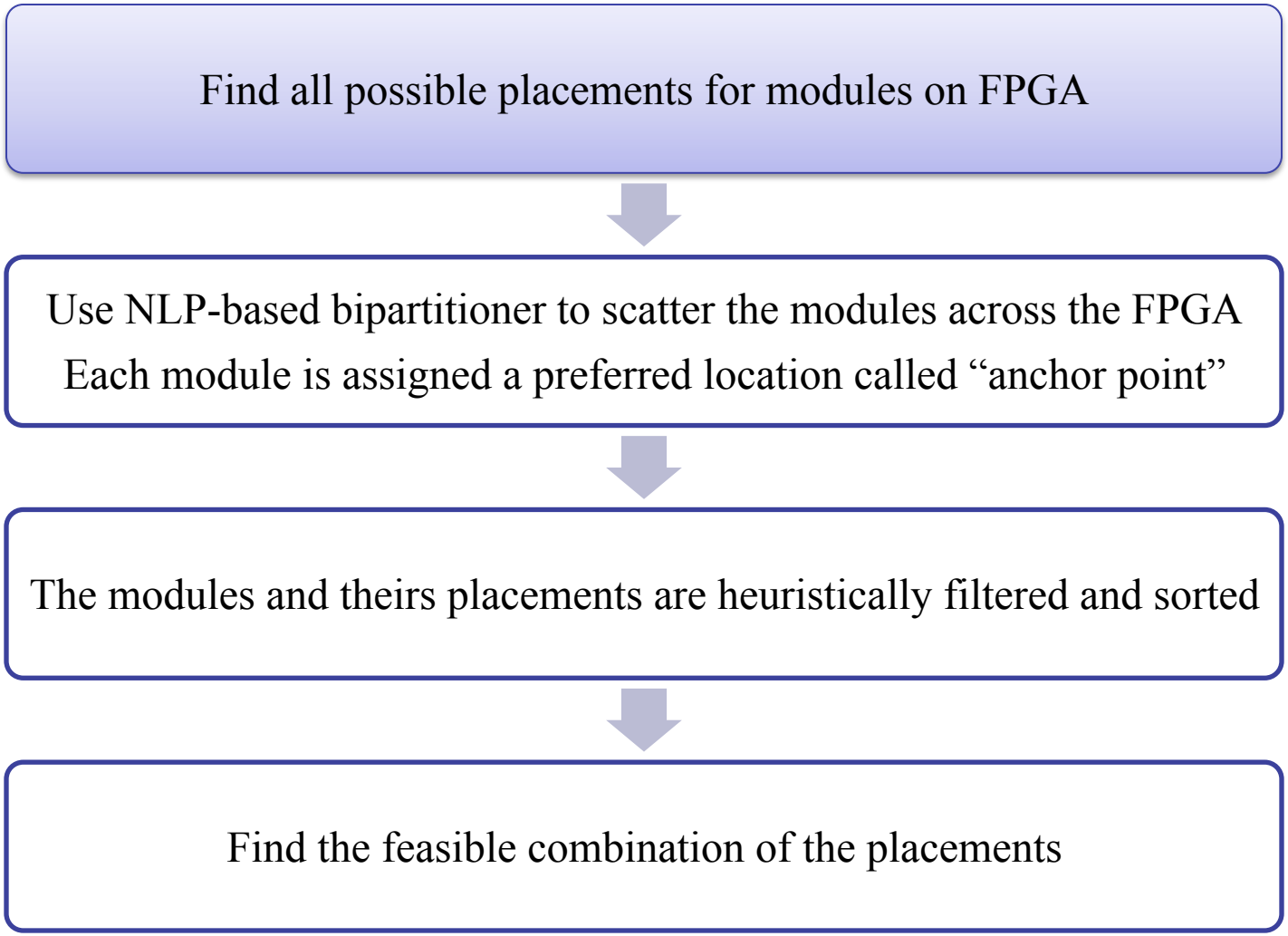
↓

The modules and theirs placements are heuristically filtered and sorted

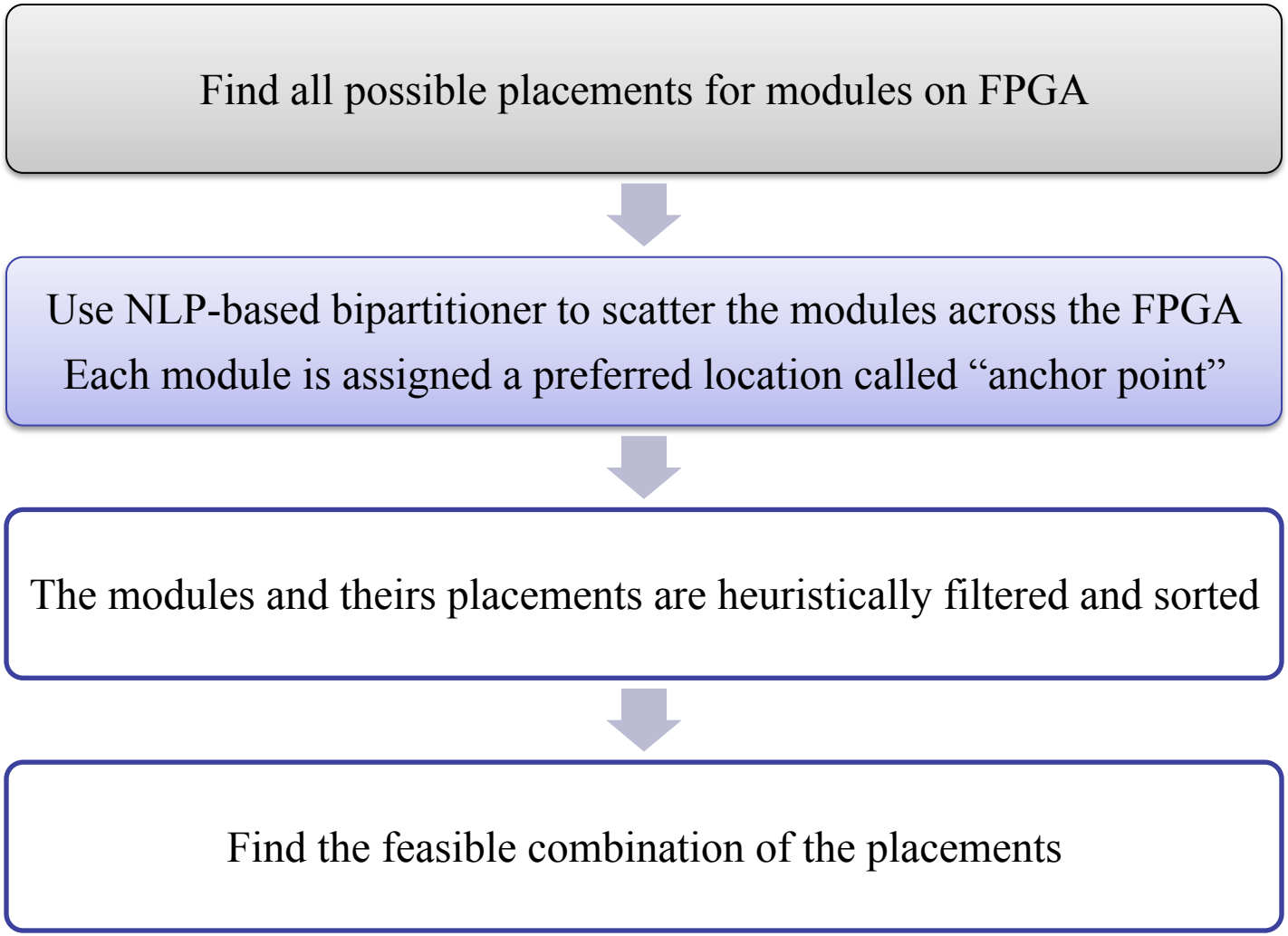
↓

Find the feasible combination of the placements

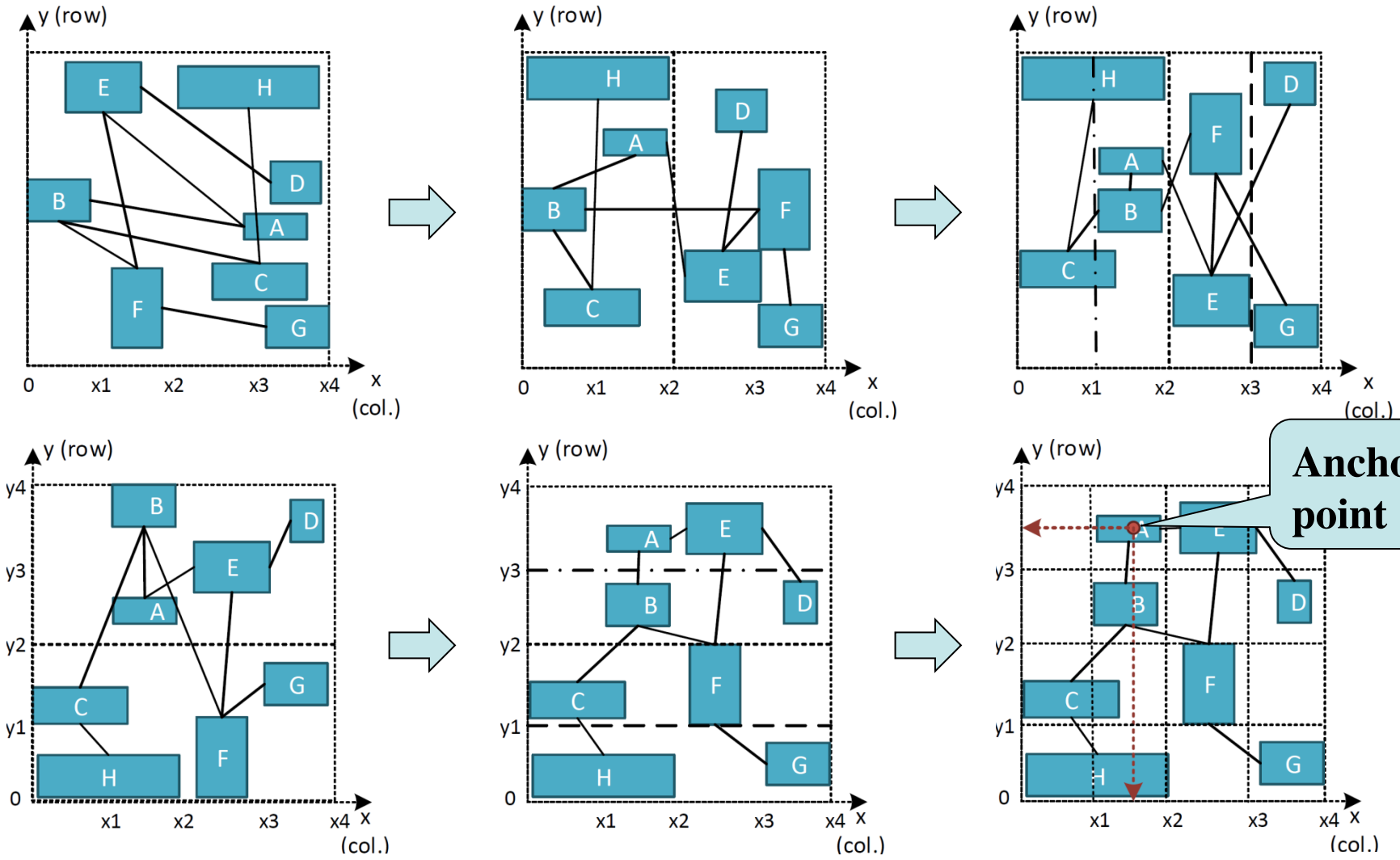
PRFloor - Overview



PRFloor - Overview



Recursive Pseudo-bipartitioning Heuristic



Non-linear Integer Program (NLP)

$$nets_{ij} = n_{ij} * (1 - m_i) * m_j + n_{ij} * m_i * (1 - m_j) \quad (1)$$
$$= n_{ij} * (m_i + m_j - 2 * m_i * m_j) \quad (2)$$

The number of nets between 2 modules that cross 2 partitions

Objective:

$$\sum_i m_i * \sum_{j \neq i} n_{ij} - 2 * \sum_i m_i * \sum_{j \neq i} (n_{ij} * m_j) \quad (3)$$

The total number of crossing-nets between all modules

Subject to:

$$Total0_{CLB} = \sum_i ((1 - m_i) * CLB0_i) <= MAX0_{CLB} \quad (4)$$

$$Total1_{CLB} = \sum_i (m_i * CLB1_i) <= MAX1_{CLB} \quad (5)$$

The total number of CLBs occupied by the modules in each partition should not exceed the available CLB in that partition

$$ub_{lower} * Total0_{CLB} <= ub_{upper} * Total1_{CLB} \quad (6)$$
$$ub_{lower} * Total1_{CLB} <= ub_{upper} * Total0_{CLB} \quad (7)$$

Balance the number of CLBs occupied in two partitions

[NLP] D. Li and X. Sun. Nonlinear integer programming, volume 84. Springer Science & Business Media, 2006.
[Gurobi] Gurobi Optimization version 6.0.2. http://www.gurobi.com, April, 2015.

PRFloor - Overview

Find all possible placements for modules on FPGA

Scatter the modules across the FPGA surface as uniformly as possible. Each module is assigned a preferred location called “anchor point”

The modules and theirs placements are heuristically filtered and sorted

Find the feasible combination of the placements

PRFloor - Overview

Find all possible placements for modules on FPGA

Scatter the modules across the FPGA surface as uniformly as possible. Each module is assigned a preferred location called “anchor point”

The modules and theirs placements are heuristically filtered and sorted

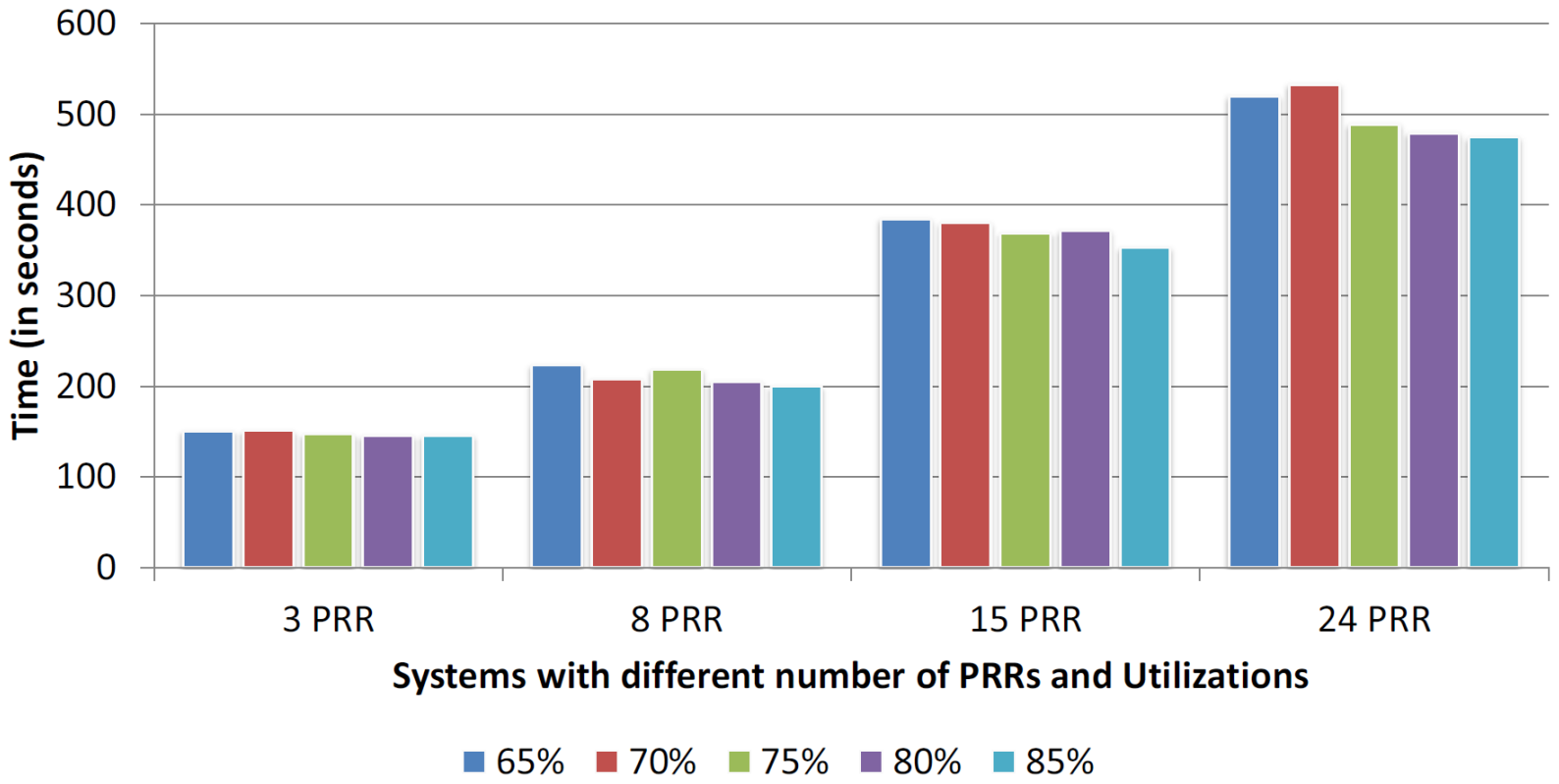
Find the feasible combination of the placements

Experiments: Synthetic Systems

NUS National University of Singapore

System	No. Mod	%CLB	%BRAM	%DSP
3 PRRs	99	65% → 85% **(41% → 60%)**	42% → 60% **(9% → 26%)**	6% → 13% **(4% → 11%)**
8 PRRs	116	65% → 85% **(36% → 56%)**	28% → 31% **(16% → 19%)**	14.5% → 15.1% **(11.1% → 11.7%)**
15 PRRs	130	65% → 87.8% **(34% → 57%)**	45% → 53% **(27% → 34%)**	25% → 28% **(22% → 25%)**
24 PRRs	126	65% → 85% **(33% → 52%)**	45% → 60% **(21% → 36%)**	23% → 32% **(22% → 31%)**

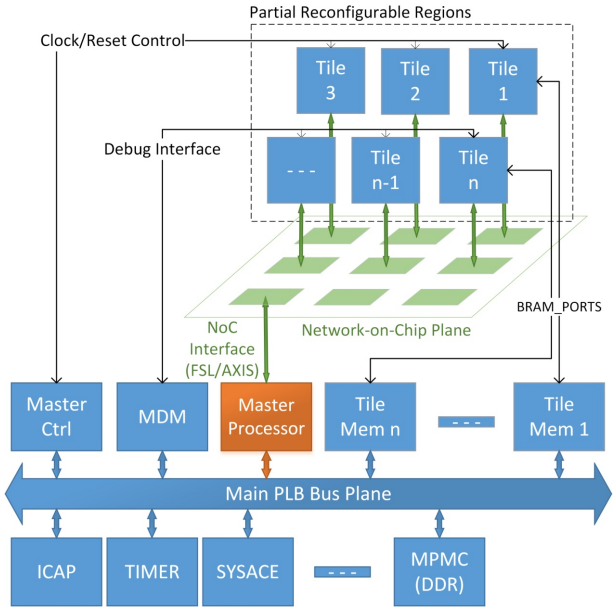
Execution Time



Increases almost linearly with the number of modules

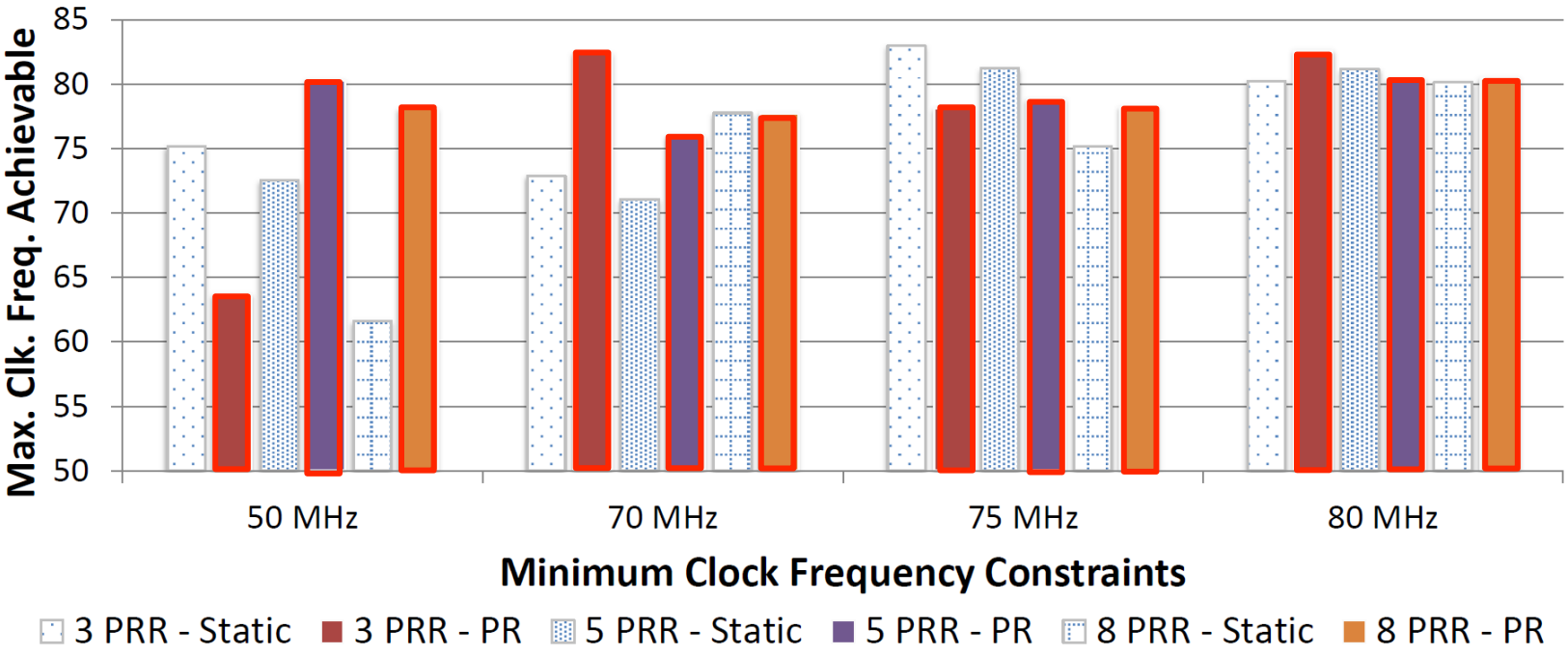
Experiments: Real systems

- **Instantiate PR-HMPSoC [Nguyen14] with varying number of PRRs (3 to 8)**
- **Compare the maximum achievable clock frequency with the comparable static system**



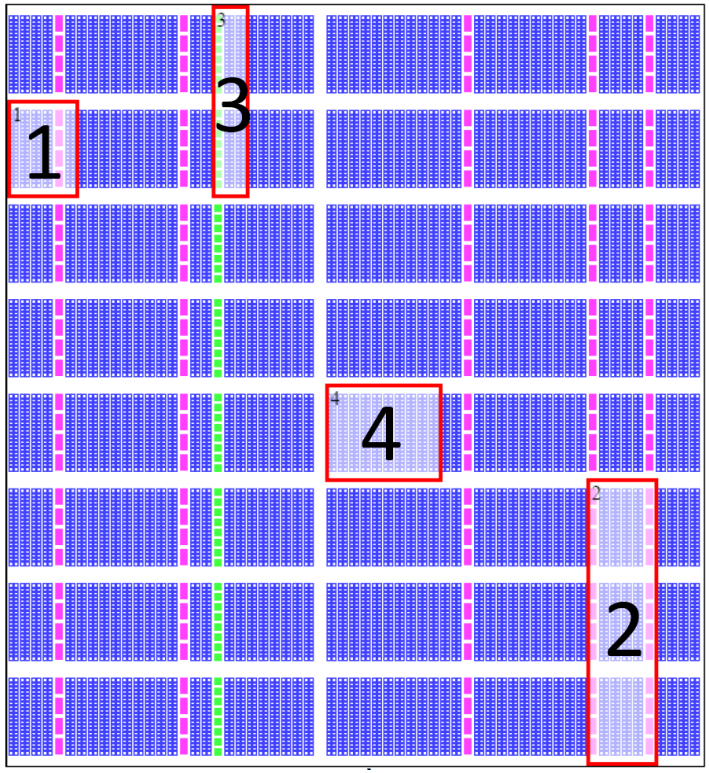
[Nguyen14] Nguyen, T.D.A.; Kumar, A., "PR-HMPSoC: A versatile partially reconfigurable heterogeneous Multiprocessor System-on-Chip for dynamic FPGA-based embedded systems," in Field Programmable Logic and Applications (FPL), 2014 24th International Conference on , vol., no., pp.1-6, 2-4 Sept. 2014

PR Systems vs. Static Systems

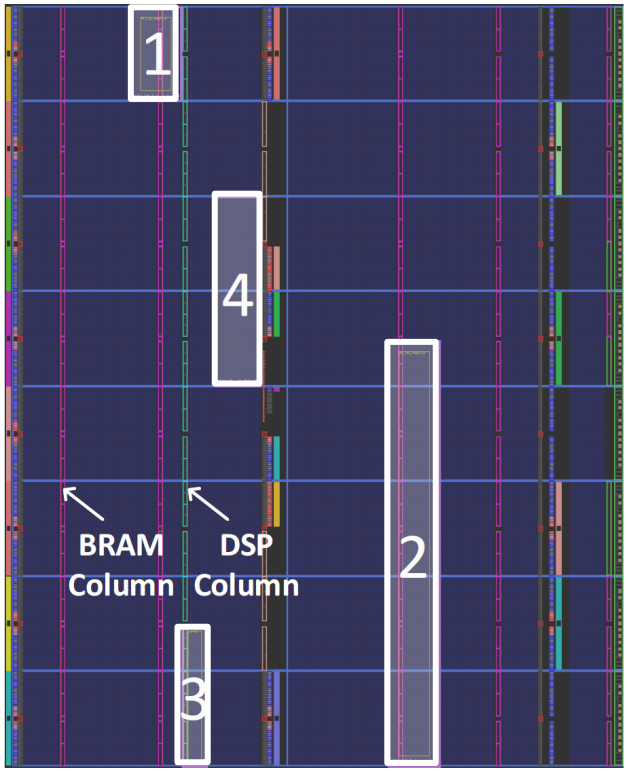


The maximum clock frequency results obtained from PR systems are not worse than the static ones.

Compare with [Rabozzi14]



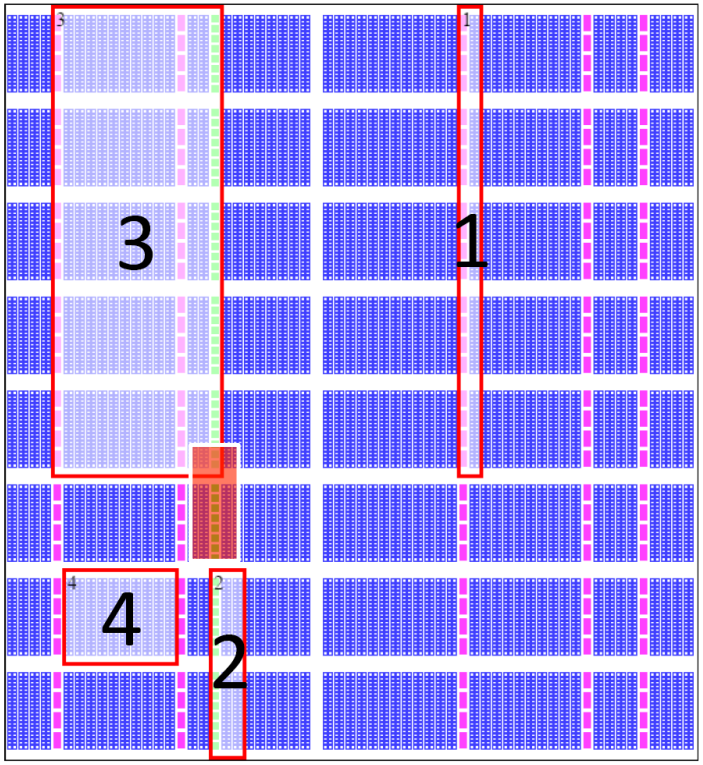
PRR 1 and 2 are too far away



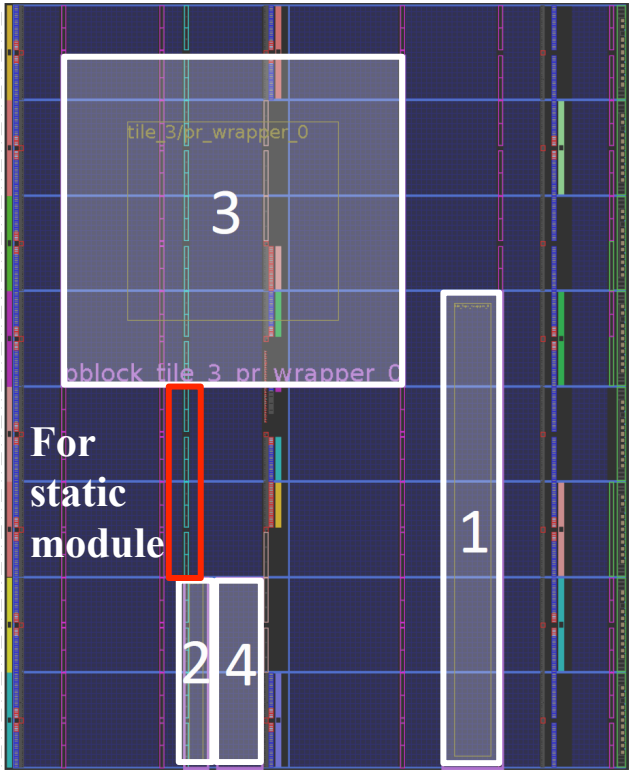
Wastage is 19% lower
Total Manhattan distances is 35% smaller

[Rabozzi14] M. Rabozzi, J. Lillis, and M. D. Santambrogio. Floorplanning for Partially-Reconfigurable FPGA Systems via Mixed-Integer Linear Programming. In Field-Programmable Custom Computing Machines, Annual International Symposium on, pages 186–193. IEEE, 2014.

Compare with [Rabozzi14]



There is not enough DSP left for static module



There is sufficient DSP resources for static module

[Rabozzi14] M. Rabozzi, J. Lillis, and M. D. Santambrogio. Floorplanning for Partially-Reconfigurable FPGA Systems via Mixed-Integer Linear Programming. In Field-Programmable Custom Computing Machines, Annual International Symposium on, pages 186–193. IEEE, 2014.

Conclusion

- The automatic floorplanner, PRFloor, is presented with the NLP-based bipartitioner
- PRFloor can provide high quality result in couple of minutes

Future Work

- **Improve the quality and performance**
  - Control the designer choices over wire-length or wastage better
  - Accelerate the first step of finding placements for modules
- **Support bitstream relocation [Oomen15]**

[Oomen15] Oomen, R.; Tuan Nguyen; Kumar, A.; Corporaal, H., "An automated technique to generate relocatable partial bitstreams for Xilinx FPGAs," in Field Programmable Logic and Applications (FPL), 2015 25th International Conference on , vol., no., pp.1-4, 2-4 Sept. 2015

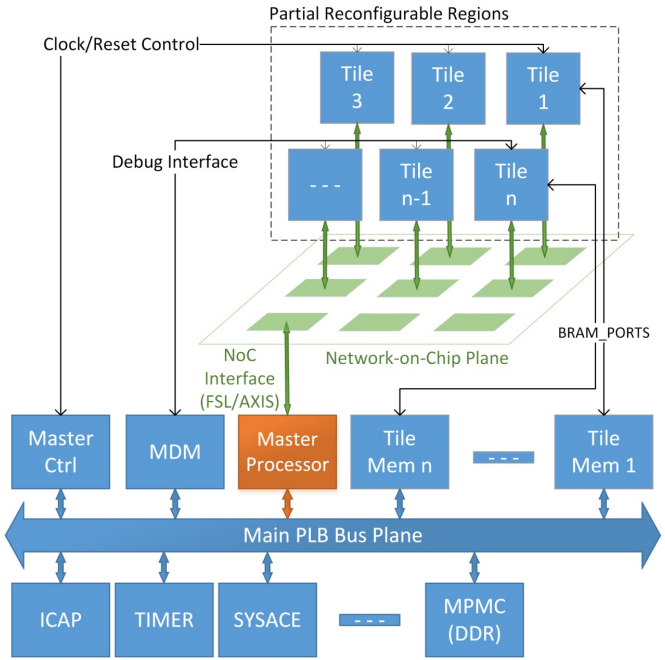
Demo

```
36          if "pr_module" is 0, this tile will be considerred as normal module
37          however, setting "pr_module" to 0 has not been tested yet
38        -->
39        <processor name="tile_mb_hw" type="tile_mb_hw" pr_module="1" >
40          <!--properites of PR module -->
41          <pr_properties>
42
43            <!--
44              The "module_name" and "inst_name" must be identical to the name of PR module and its instance
   name in Tile
45            -->
46            <pr_instance module_name="pr_wrapper" inst_name="pr_wrapper_0" />
47
48            <!--
49              "pr_variant" means different implementation of one PR module
50              PRGen will look for variant's netlist in:  "./pr_netlist/<type>/<module_name>.ngc
51              the "netlist" field is ignored in the current version of PRGen but it must be there
52              You can add as many variants as you want for a Tile
53            -->
54            <pr_variant name="microblaze" type="microblaze" netlist="pr_netlist" />
55            <pr_variant name="adder_xor" type="hw_adder_xor" netlist="pr_netlist"  />
56          </pr_properties>
57        </processor>
58      </tile>
59      <tile name="tile_2">
archgraph.xml [xml] [54,1] &ff
```

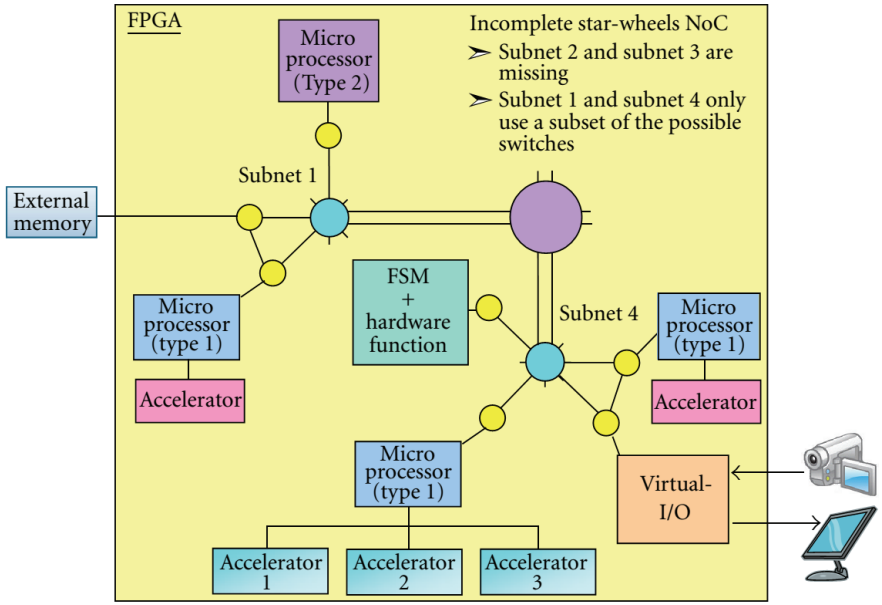
Thank you!

Appendix

Large PR MPSoC



[Nguyen14]

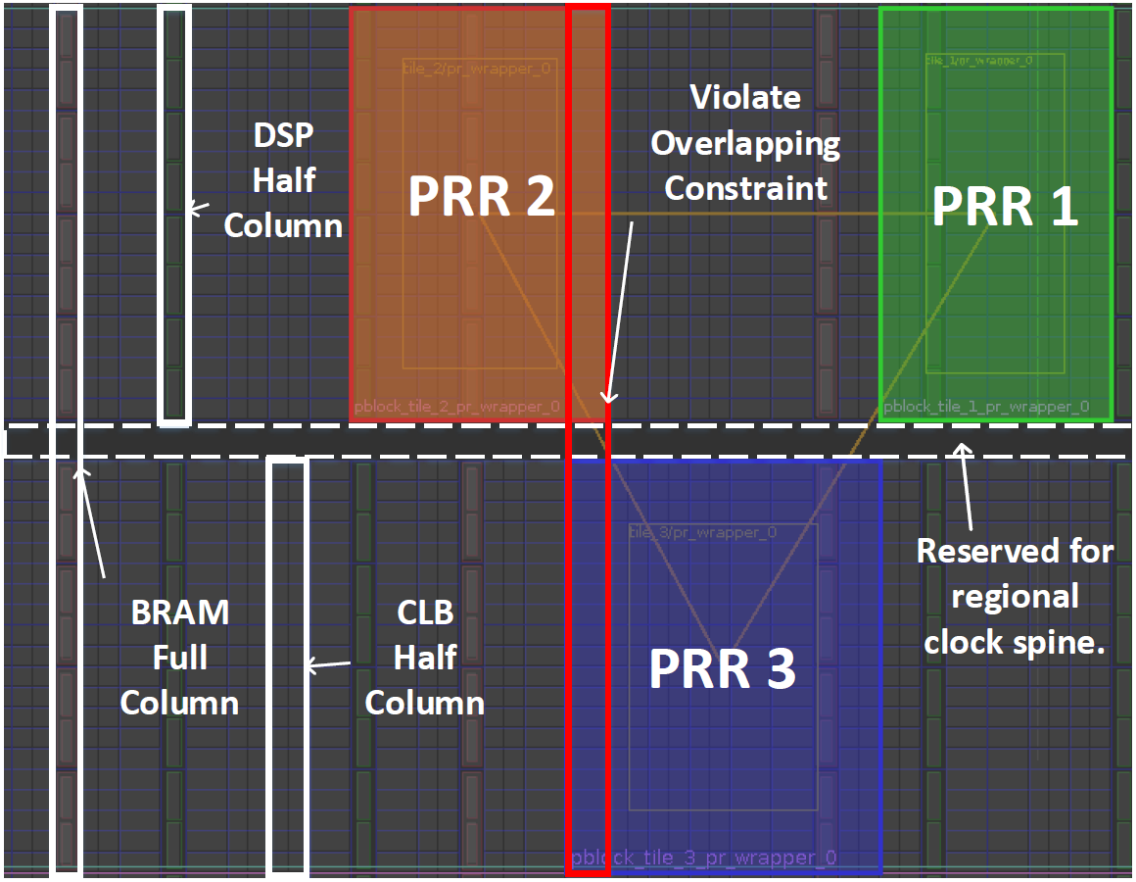


[Gohringer11]

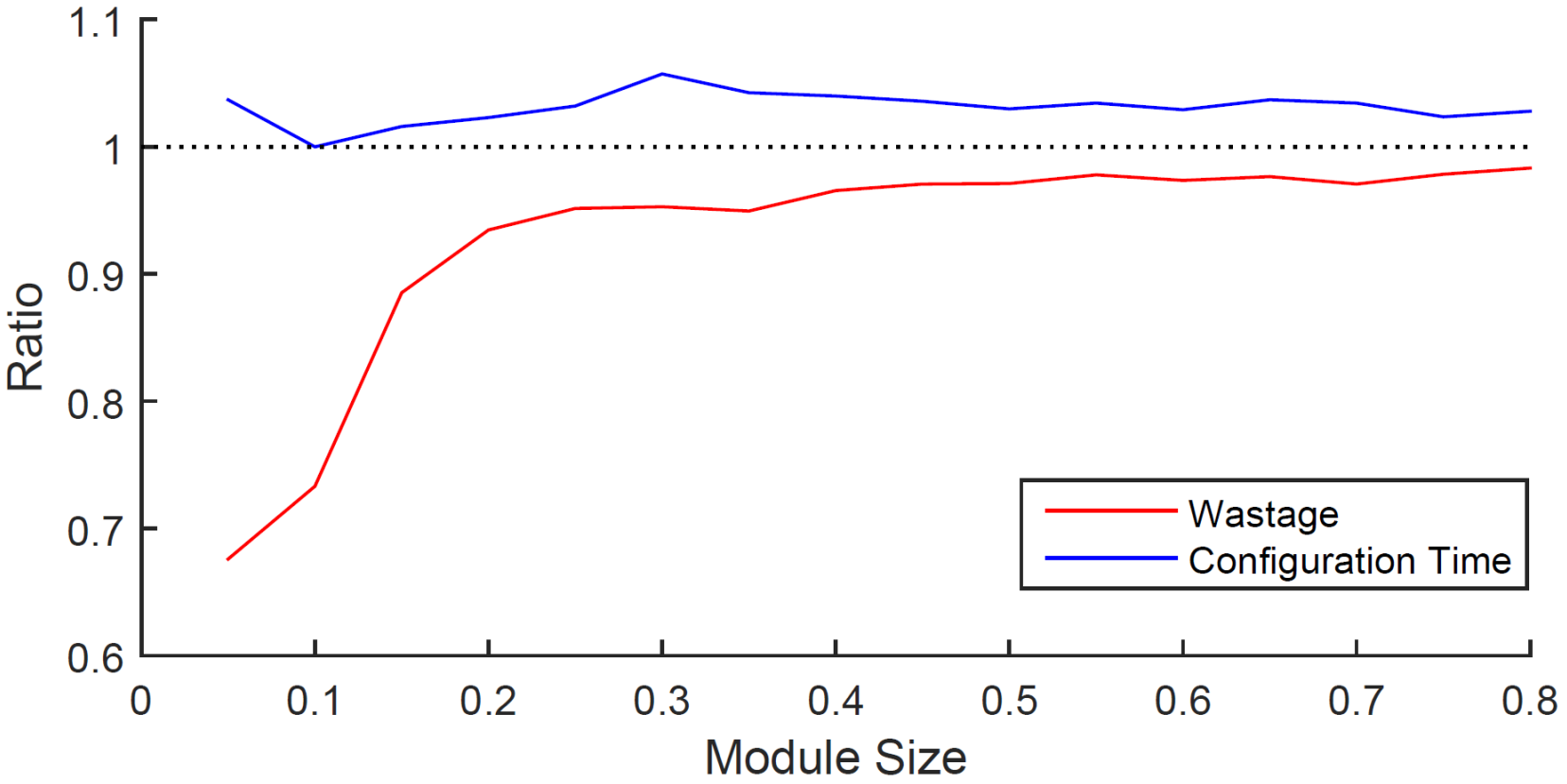
[Nguyen14] Nguyen, T.D.A.; Kumar, A., "PR-HMPSoC: A versatile partially reconfigurable heterogeneous Multiprocessor System-on-Chip for dynamic FPGA-based embedded systems," in Field Programmable Logic and Applications (FPL), 2014 24th International Conference on , vol., no., pp.1-6, 2-4 Sept. 2014

[Gohringer11] D. Gohringer, M. Hübner, E. N. Zeutebouo, and J. Becker, "Operating system for runtime reconfigurable multiprocessor systems," *International Journal of Reconfigurable Computing*, vol. 2011, p. 3, 2011

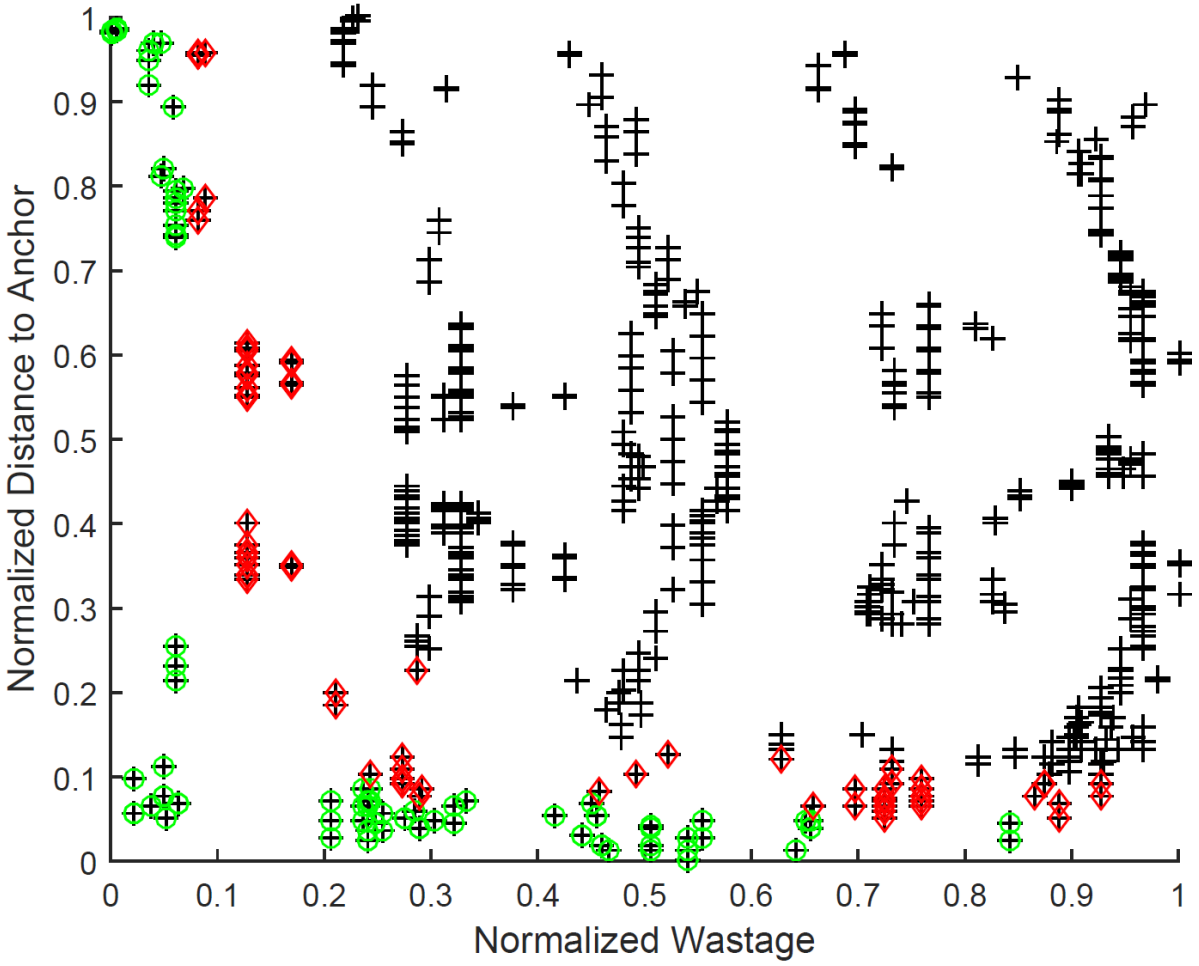
FPGA Model


DSP
Half
Column
PRR 2
Violate
Overlapping
Constraint
PRR 1
pblock_tile_2_pr_wrapper_0
pblock_tile_1_pr_wrapper_0
BRAM
Full
Column
CLB
Half
Column
PRR 3
Reserved for
regional
clock spine.

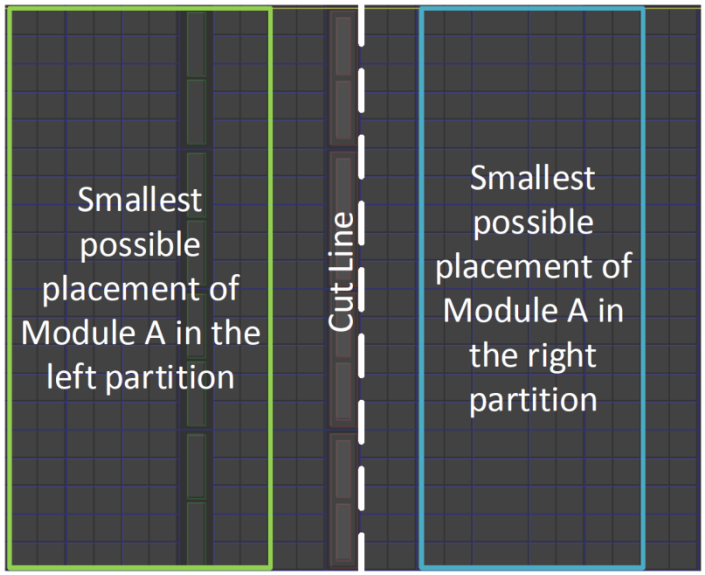
Why half-column granularity?

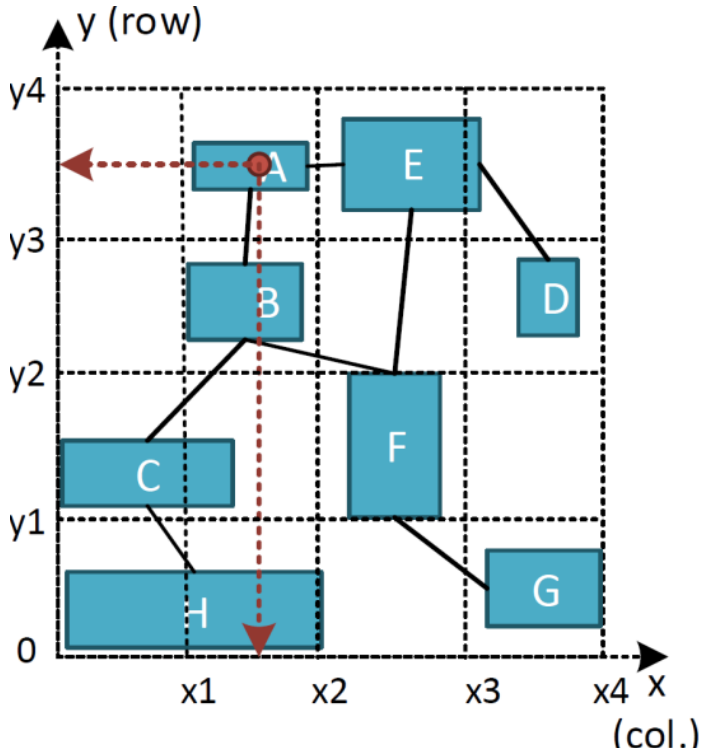


Pareto Ranking



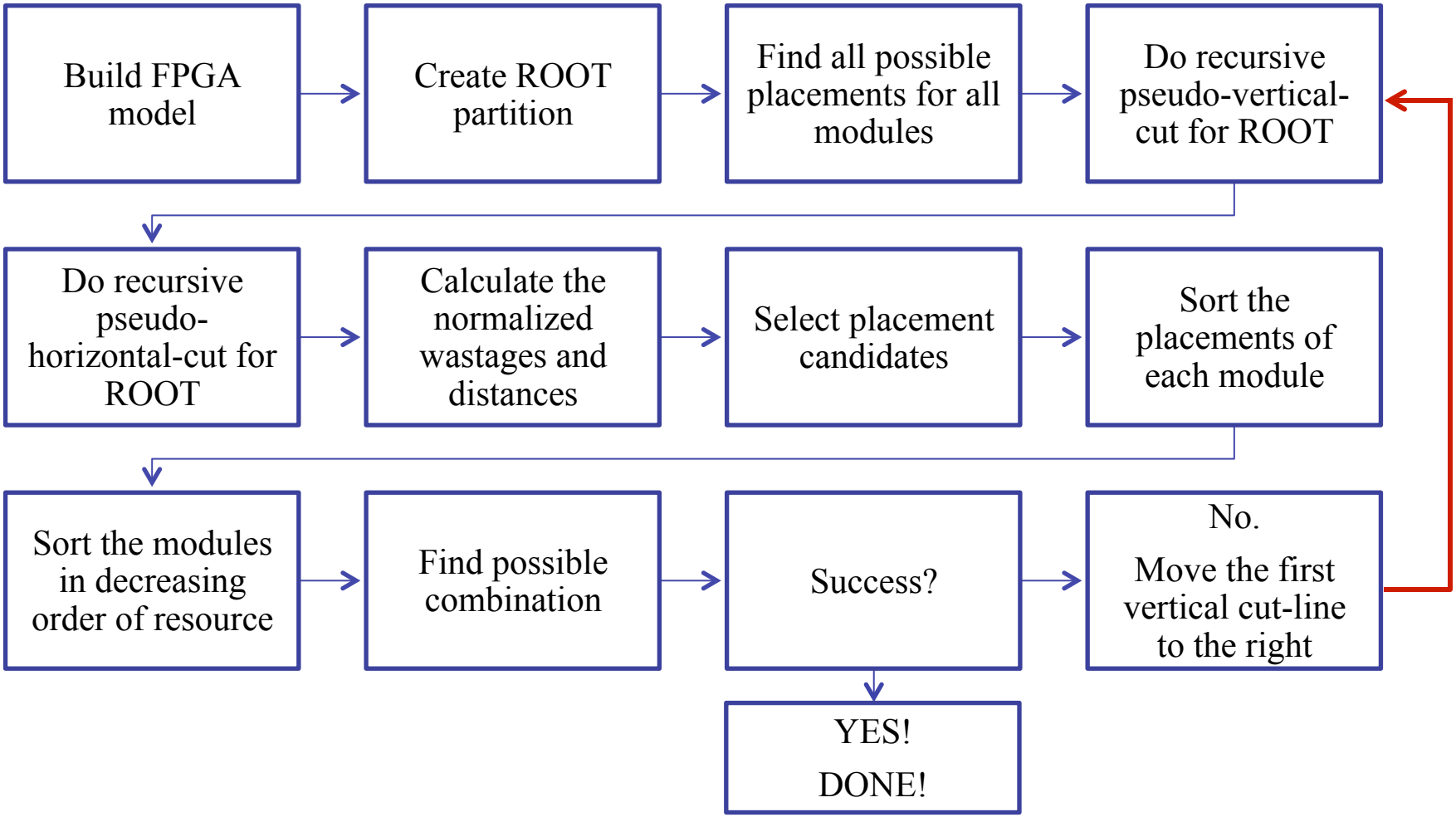
Sort the placements





$$OBJ{\downarrow}placement = \alpha * wastage + \beta * dist\_to\_anchor$$

PRFloor - Overview



Recursive Pseudo-bipartitioning Heuristic

NUS National University of Singapore

Algorithm 2 Recursive Pseudo-bipartitioning Heuristic

Require: *parent_partition* $\neq \emptyset$ **and** valid *cut_line* **and** cut_type

create *partition*0 $\leftarrow \emptyset$ from *cut_line* and *parent_partition*
create *partition*1 $\leftarrow \emptyset$ from *cut_line* and *parent_partition*
list_mod $\leftarrow \emptyset$ {list of modules used for bipartition process}
area_constraint $\leftarrow 90\%$
if cut_type = vertical cut **and** first cut **then**
  area_constraint $\leftarrow 100\%$
end if
for all module $\in$ parent_partition **do**
  *list_placement*0 $\leftarrow \emptyset$; *list_placement*1 $\leftarrow \emptyset$
  for all placement of module $\in$ parent_partition **do**
    *area_ratio*0 $\leftarrow$ *area_of_placement* $\cap$ *partition*0
    *area_ratio*1 $\leftarrow$ *area_of_placement* $\cap$ *partition*1
    if *area_ratio*0 $\geq$ *area_constraint* **then**
      add placement to *list_placement*0
    else if *area_ratio*1 $\geq$ *area_constraint* **then**
      add placement to *list_placement*1
    end if
  end for
  if *list_placement*0 $= \emptyset$ **and** *list_placement*1 $= \emptyset$ **then**
    deduct the resource of *module* from the total resource of *partition*0 and *partition*1
  else
    *resource*0 $\leftarrow$ *estimate_resources*(*list_placement*0)
    *resource*1 $\leftarrow$ *estimate_resources*(*list_placement*1)
    add *module* $\rightarrow$ *list_mod*
  end if
end for
run *NLP_Bipartitioner*(*list_mod*, *partition*0, *partition*1)
if SUCCESS **then**
  update anchor points of modules
  execute Recursive Pseudo-bipartitioning for partition0
  execute Recursive Pseudo-bipartitioning for partition1
end if

Estimate occupied resources

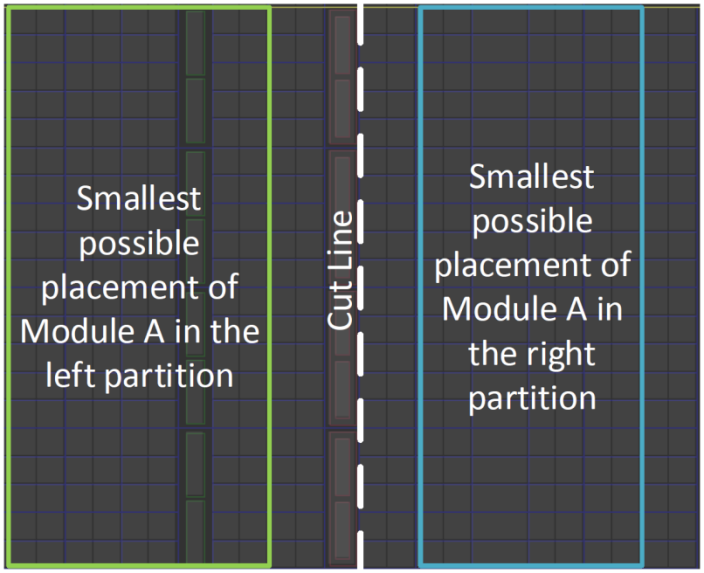
Algorithm 3 Estimate Occupied Resources

1: $a = \bar{x}$
2: **if** $a > \tilde{x}$ **then**
3: $\quad a = \tilde{x}$
4: **end if**
5: $result = a - 1.5 * \sigma_x$
6: **if** $result < minimum\_occupied$ **then**
7: $\quad result = minimum\_occupied$
8: **end if**
9: **return** $result$

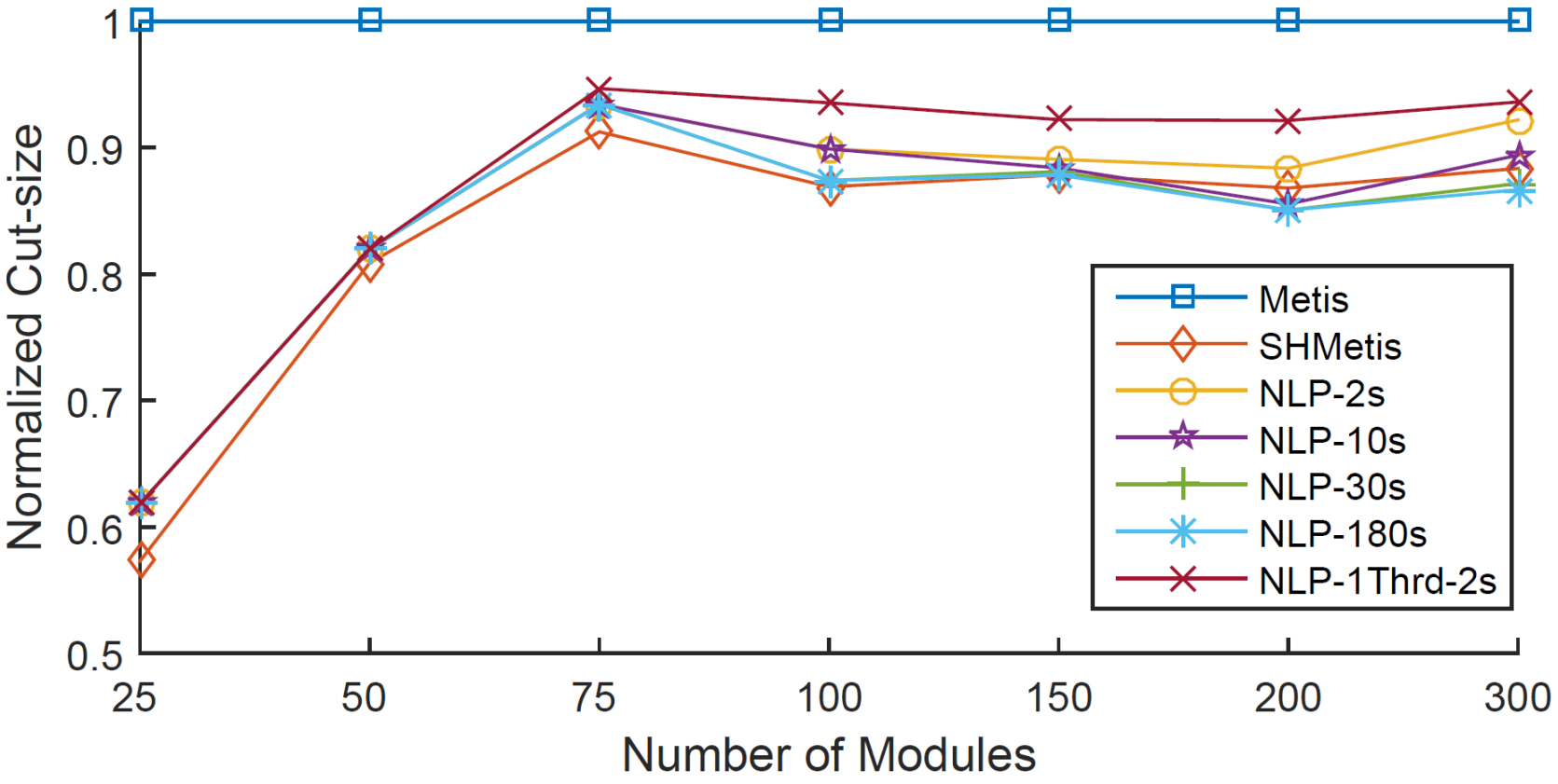
$\bar{x}$: arithmetic mean $\tilde{x}$: median $\sigma \downarrow x$: standard deviation

Bipartitioner

- The available resources in two partitions can be different
- The resources occupied by the possible placements of one module in two partitions can be different
- Each type of resource occupied by modules in two partitions can be balanced individually

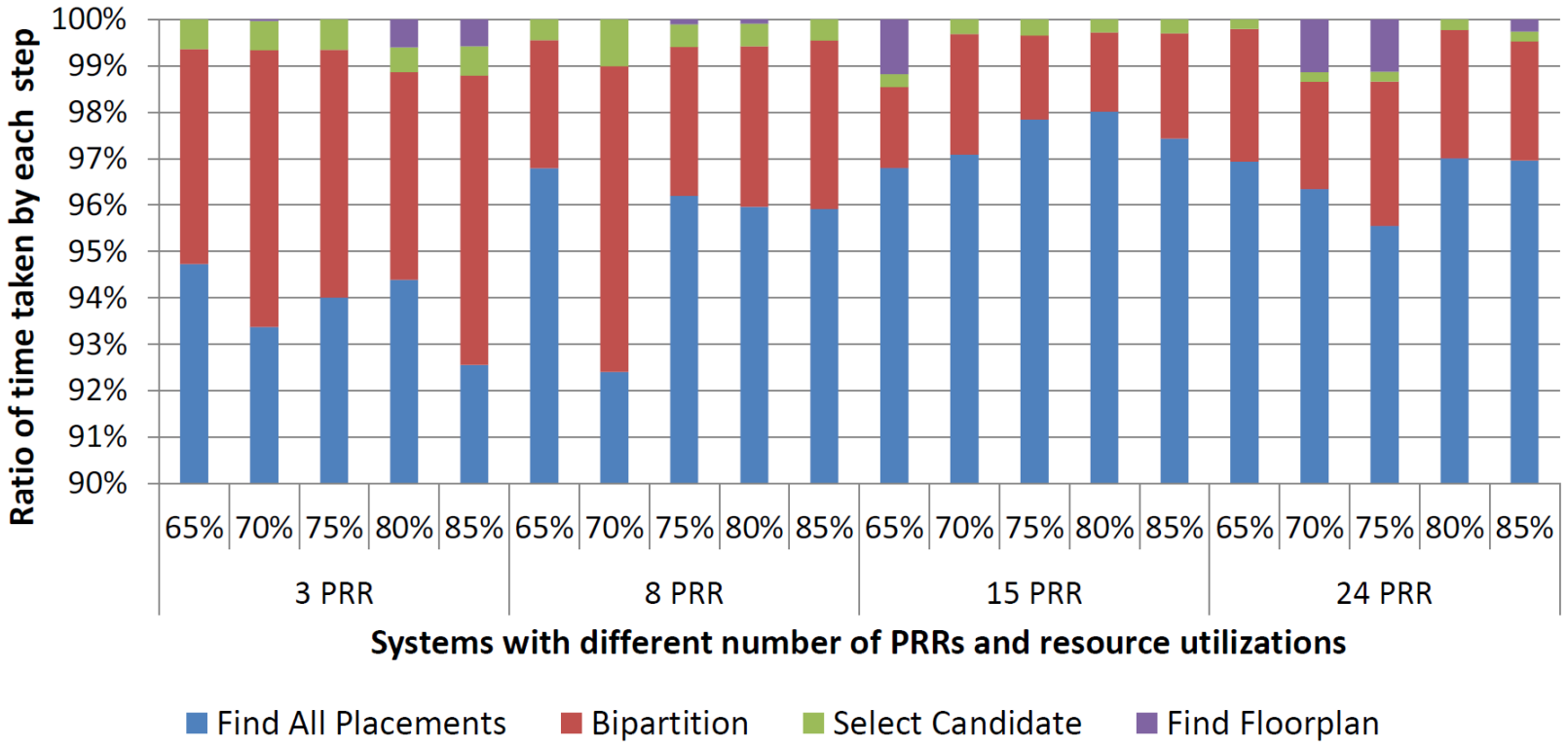


Quality of the NLP Bipartitioner



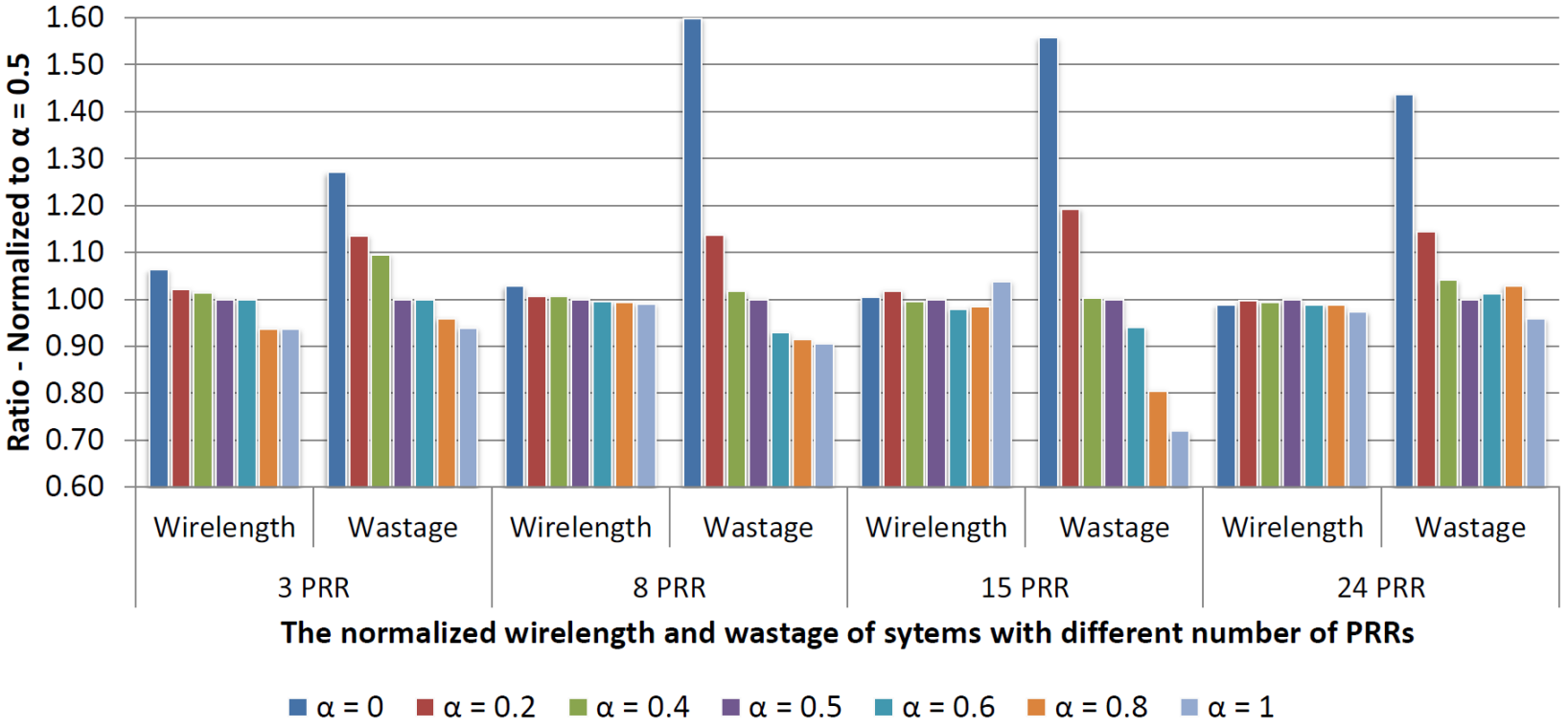
[Hmetis98] G. Karypis and V. Kumar. hmetis: A hypergraph partitioning package, version 1.5. 3. 1998.
[Metis13] G. Karypis and V. Kumar. Metis - a software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices version 5.1.0. 4, March 2013.

Execution Time - Breakdown



- **The recursive process used to find the floorplan is very fast.**
- **It takes only at most 1.2% of the total runtime. In most cases, almost 0.**

Effect of α and β to Wire-length and Wastage



$$OBJ\downarrow placement = \alpha * wastage + \beta * dist\_to\_anchor$$

Resource requirement PRRs compared with [Rabozzi14]

PR Regions	Fig. 14a CLB	Fig. 14a BRAM	Fig. 14a DSP	Fig. 14b CLB	Fig. 14b BRAM	Fig. 14b DSP
PRR 1	100	4	0	100	20	0
PRR 2	50	18	0	50	0	14
PRR 3	50	0	10	50	40	20
PRR 4	200	0	0	200	0	0

[Rabozzi14] M. Rabozzi, J. Lillis, and M. D. Santambrogio. Floorplanning for Partially-Reconfigurable FPGA Systems via Mixed-Integer Linear Programming. In Field-Programmable Custom Computing Machines, Annual International Symposium on, pages 186–193. IEEE, 2014.