Efficient Mapping onto Coarse-Grained Reconfigurable Architectures using Graph Drawing based Algorithm

Jonghee Yoon, Aviral Shrivastava[†], Sanghyun Park, Minwook Ahn, Reiley Jeyapaul[†], and Yunheung Paek

SO&R Research Group

Seoul National University, Korea

[†] CML Laboratory

Arizona State University, USA

Outline

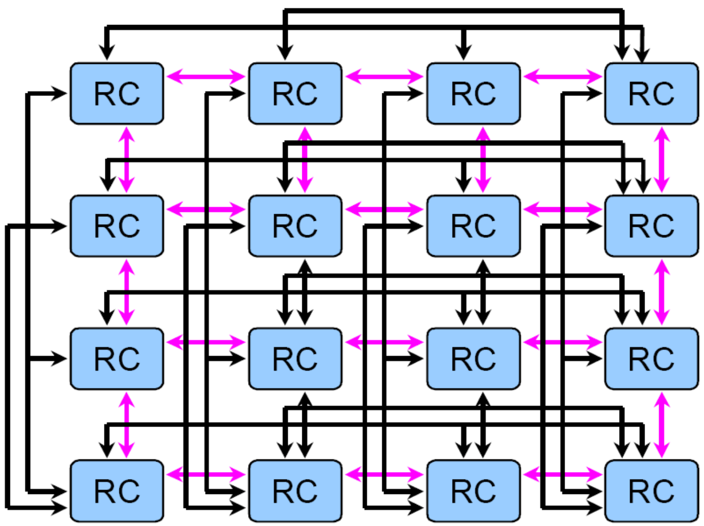
- Coarse-Grained Reconfigurable Architectures
- Issues on Application Mapping onto CGRAs
- ILP(Integer Linear Programming) Formulation
- Graph Drawing Algorithm
  - Split & Push
  - Matching-Cut
- Split & Push Kernel Mapping (SPKM)
- Experimental Results
- Conclusion

Reconfigurable Architecture

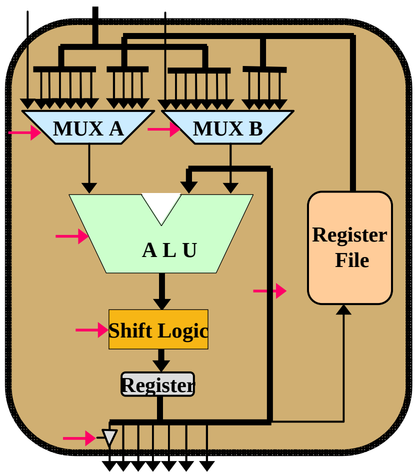
- Reconfiguration is emerging
  - increasing needs for flexible and high speed computing fabrics
  - ability to make substantial changes to the ***datapath itself*** in addition to the ***control flow***
- **CGRAs** (Coarse-Grained Reconfigurable Architectures)
  - High computation throughput
  - Low power consumption
  - Fast reconfiguration
  - Operation level granularity

CGRAs

- Array of processing elements (PEs)
  - PE (or reconfigurable cell, RC, in MorphoSys)
    - Light-weight processor
    - No control unit
    - Simple ALU operations
  - ex) Morphosys, RSPA, ADRES, .etc



MorphoSys RC Array



PE structure of RSPA

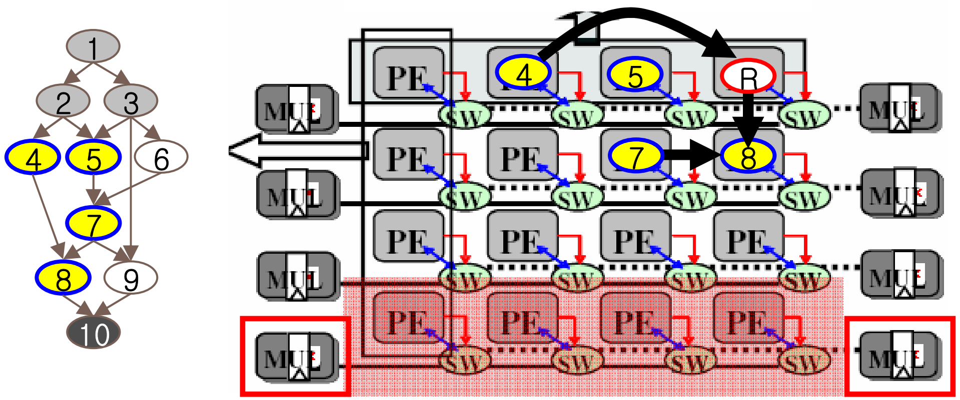
Application Mapping onto CGRAs

- Compiler's role for CGRAs
  - analyze the applications
  - transform the applications to be suitable for CGRA structure

- The main compiler issues in CGRAs are…
  - **Parallelism**
    - finding more parallelism in the application
      ➔ better use of CGRA features
    - e.g., s/w pipelining
  - **Resource Minimization**
    - to reduce power consumption
    - to increase throughput
    - to have more opportunities for further optimizations
      - e.g., power gating of PEs

CGRAs are becoming complex

- **Processing Element (PE) Interconnection**
  - 2-D mesh structure is not enough for high performance
- **Shared Resources**
  - cost, power, complexity,
  - multipliers and load/store units can be shared
- **Routing PE**
  - In some CGRAs, PE can be used for routing only
  - to map a node with degree greater than the # of connections of a PE



RSPA structure

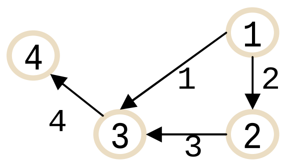
Problem Formulation

- Objective of Compiler is to generate mapping…
  - that utilizes ***less # of rows*** ⟶ more useful than *# of PEs* due to the shared resource constraints in practice
  - that uses ***less routing PEs***

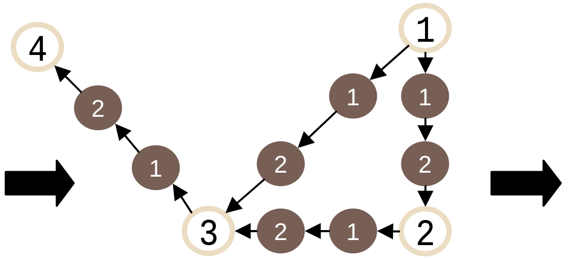
- Objective Function
  - *Given a kernel DAG K = (V, E), and a CGRA C = (P, L), find a mapping with the objectives described above*
- Constraints
  - Path existence
  - Simple path
  - Uniqueness of routing PE
  - No computation on routing PE
  - Shared resource constraints

ILP(Integer Linear Programming) Formulation

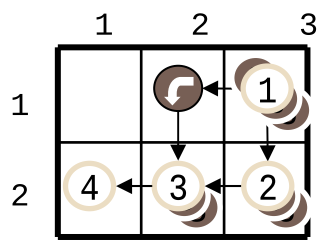
- Objectives
  - Minimize # of rows and # of routing operations(ROs)
- Constraints
  - Each operation can be mapped on one PE
  - Each PE can have only one operation
  - # of shared resources & Direct interconnections
- Suppose …
  - Each edge might include several RO nodes
  - Some ROs can be hidden under producer operation



(a) Kernel DAG K



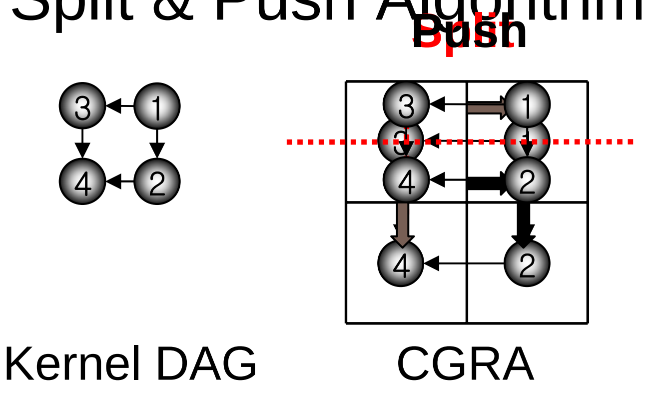
(b) K'



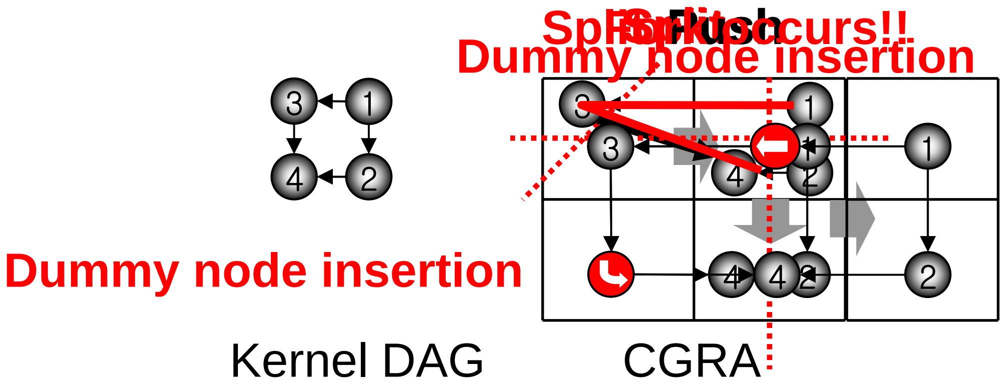
(c) Mapping Result

Graph Drawing Problem (I)

- Split & Push Algorithm[1]



Good Mapping



Bad Mapping

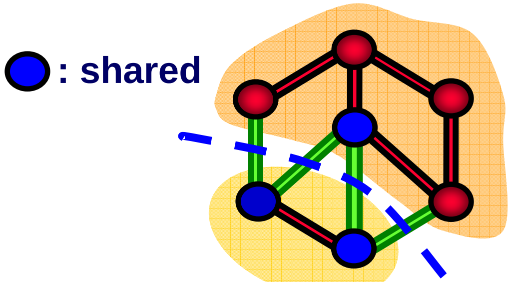
- Bad split decision incurs more uses of resources
  - 2 vs. 3 columns
  - Forks incurs dummy nodes, which are '*unnecessary routing PEs*'

- Now the question is, how can we reduce the forks?

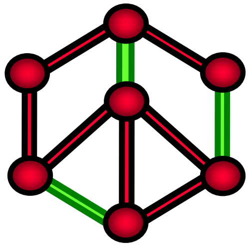
[1]*G. D. Battista et. al. A split & push approach to 3D orthogonal drawing. In Graph Drawing, 1998.*

Graph Drawing Problem (II)

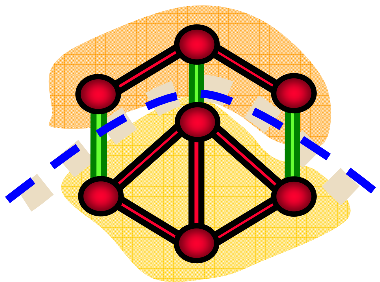
- Matching-Cut[2]
  - a set of edges which do not share nodes with any other edges and whose removal makes the graph disconnected



A cut, but not a matching

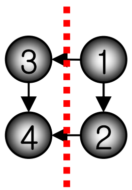


A matching, but not a cut

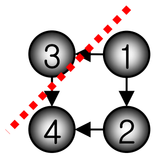


A matching-cut

- Forks can be avoided by finding matching-cut in DAG



A matching-cut, need 4 PEs, no routing PEs

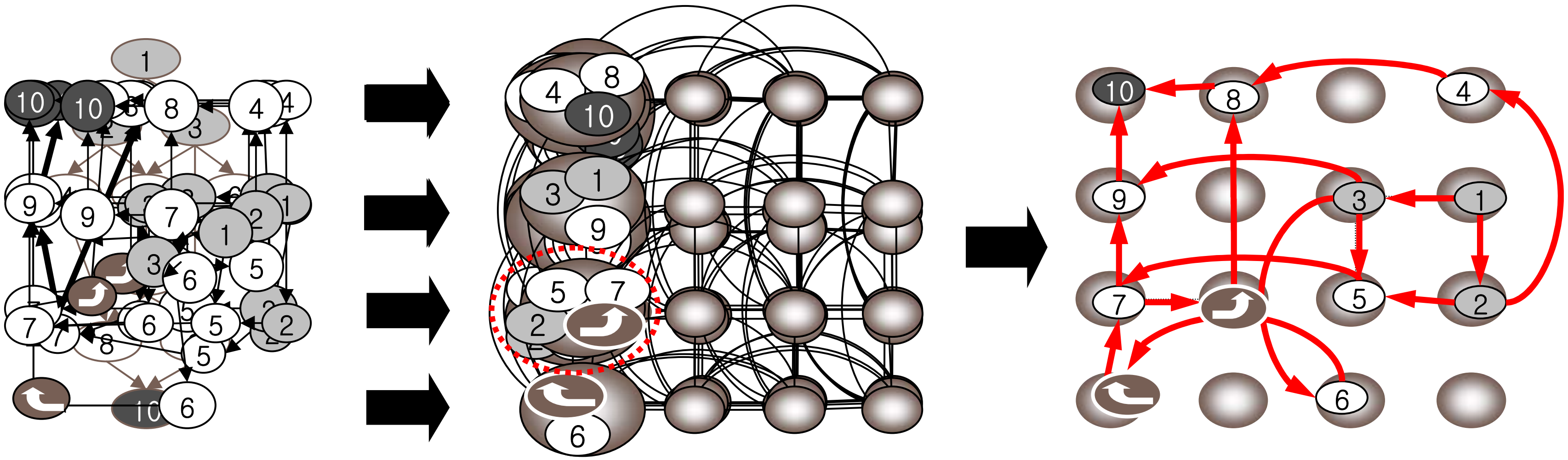


A cut, need 6 PEs, 2 routing PEs

[2]M. Patrignani and M. Pizzonia. The complexity of the matching-cut problem. In WG '01: Proceedings of the 27th International Workshop on Graph-Theoretic Concepts in Computer Science, 2001.

Split & Push Kernel Mapping

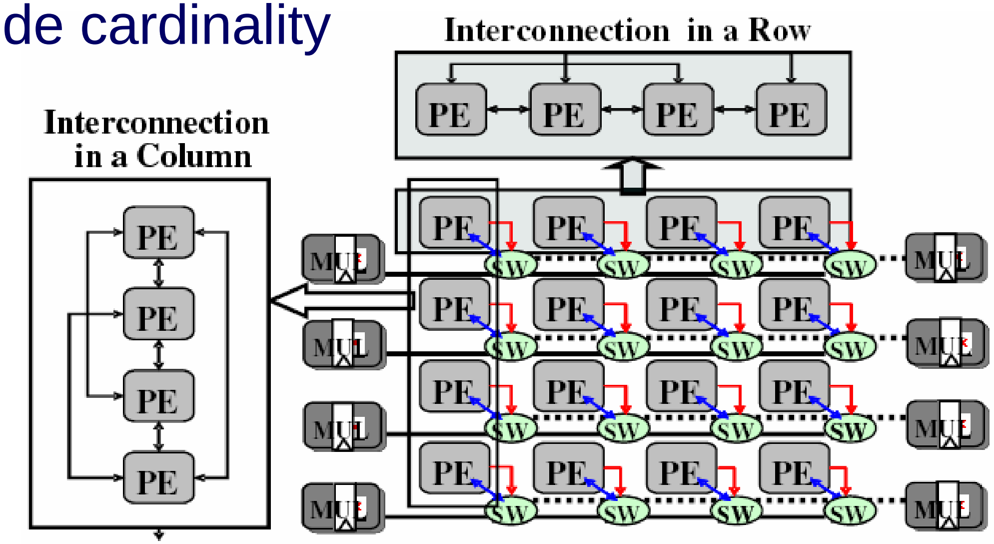
- PE is connected to at most 6 other PEs.
- At most 2 load operations and one store Operation can be scheduled.
  - Load : Store : ALU : RPE : Fork :
- # of node : 10
- # of load : 3
- # of store : 1
- Initial ROW$_{min}$ = $max(\lceil |V|/|N| \rceil, \lceil L/L_r \rceil, \lceil S/S_r \rceil)$ = max(3, 2, 1) = 3



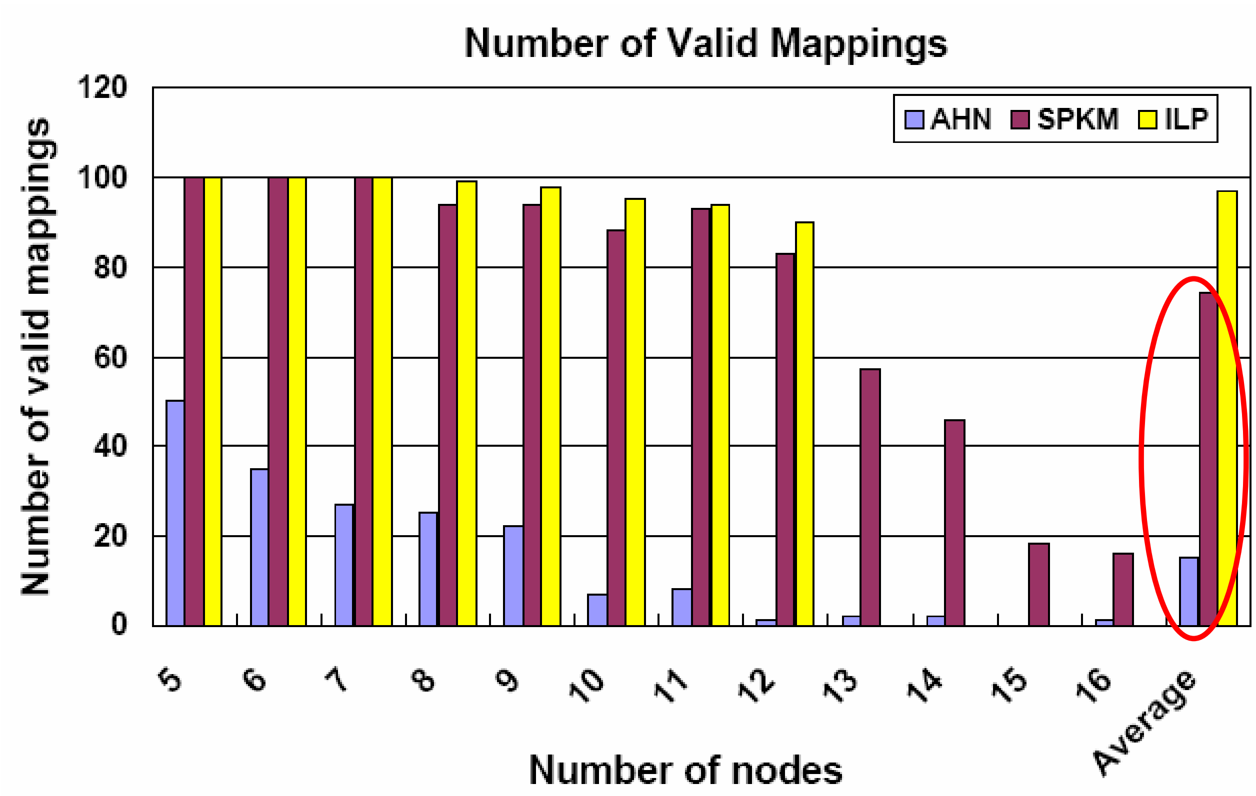
Row-wise Scattering

Experimental Setup

- We test SPKM on a CGRA called RSPA
  - RSPA has orthogonal interconnection (*irregular interconnection*)
  - Each row has 2 shared multipliers
    Each row can perform 2 loads and 1 store (*shared resource*)
  - PE can be used for routing only (*routing resource*)
- Random kernel DAG generator
  - 100 applications for each node cardinality
- Benchmarks from Livermore loops, MultiMedia and DSPStone



SPKM maps more applications



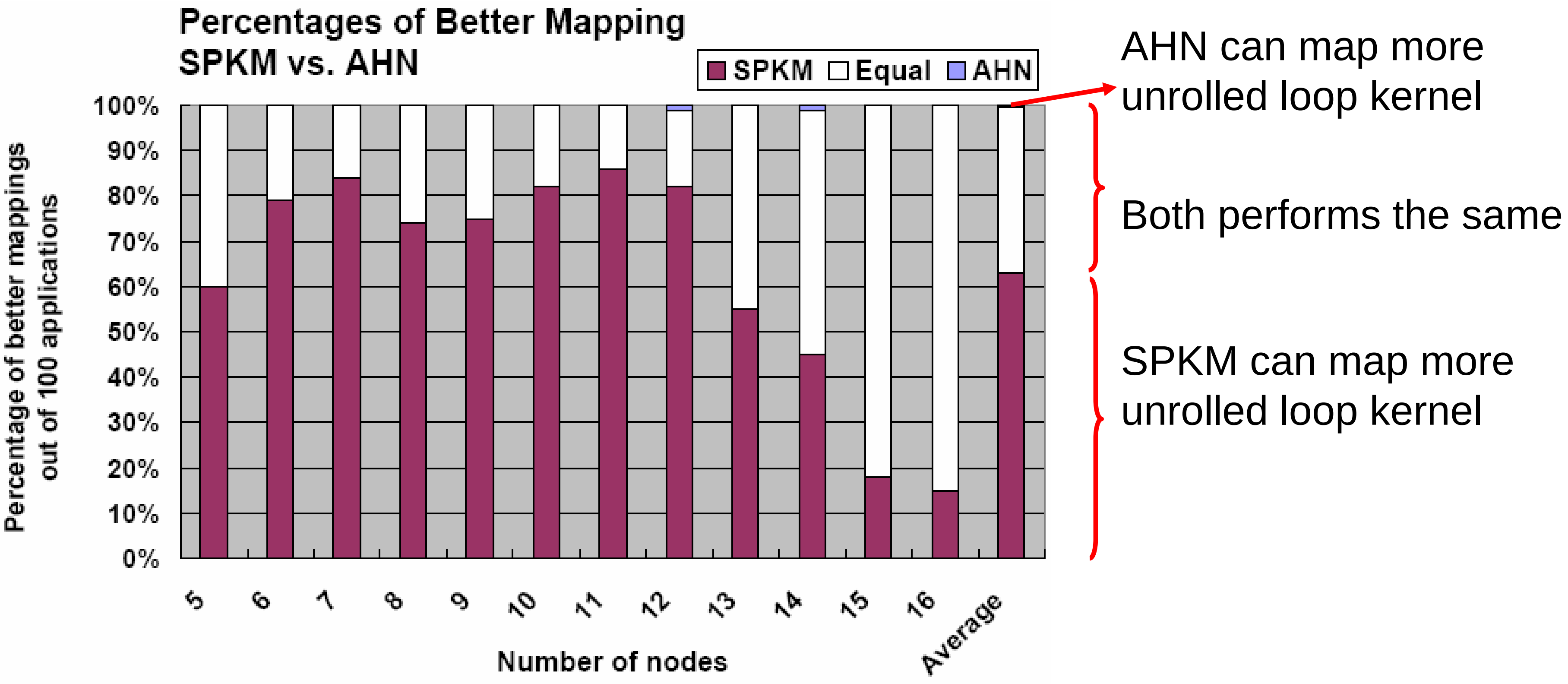
Y axis : # of applications that each technique can map

X axis : # of nodes that each application has

3.6X more applications

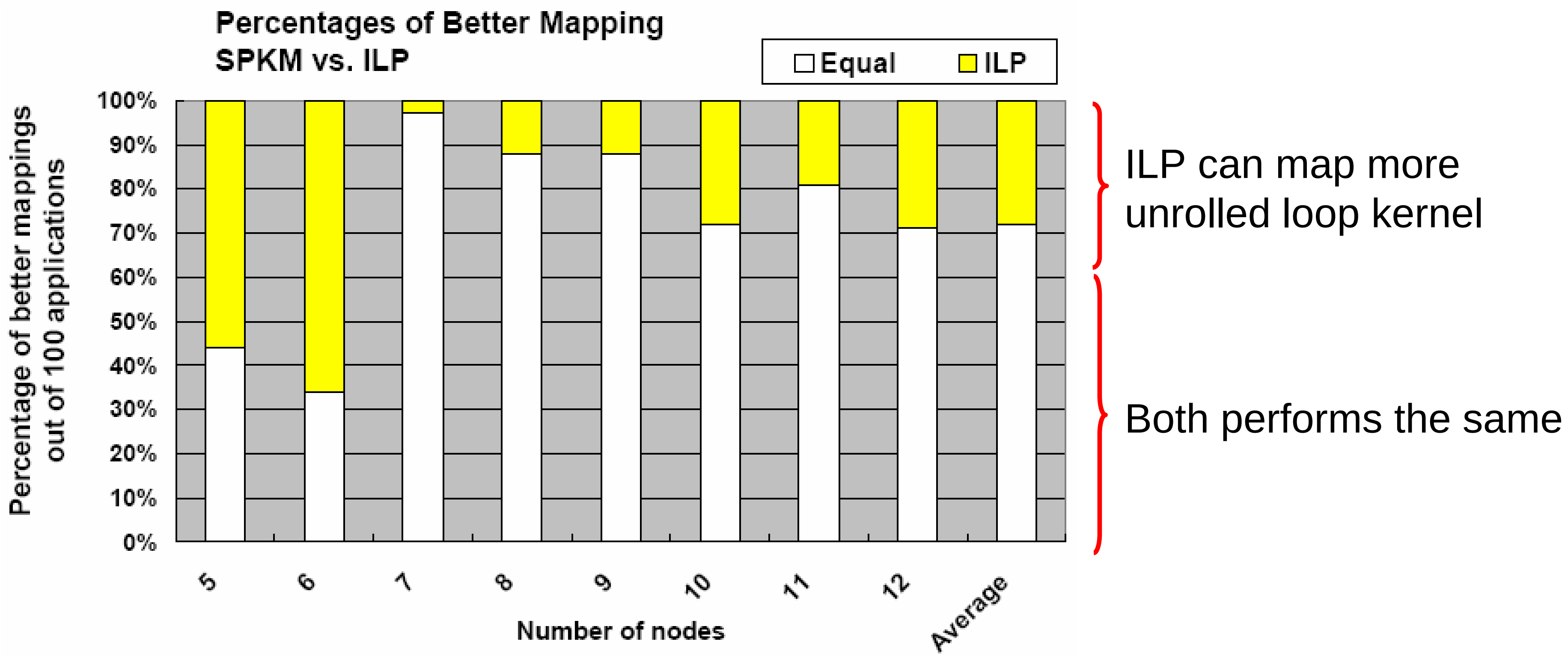
- SPKM can on average map 3.6X more applications than AHN
- For large application, SPKM shows high map-ability since it considers routing PEs well

SPKM generates better mapping



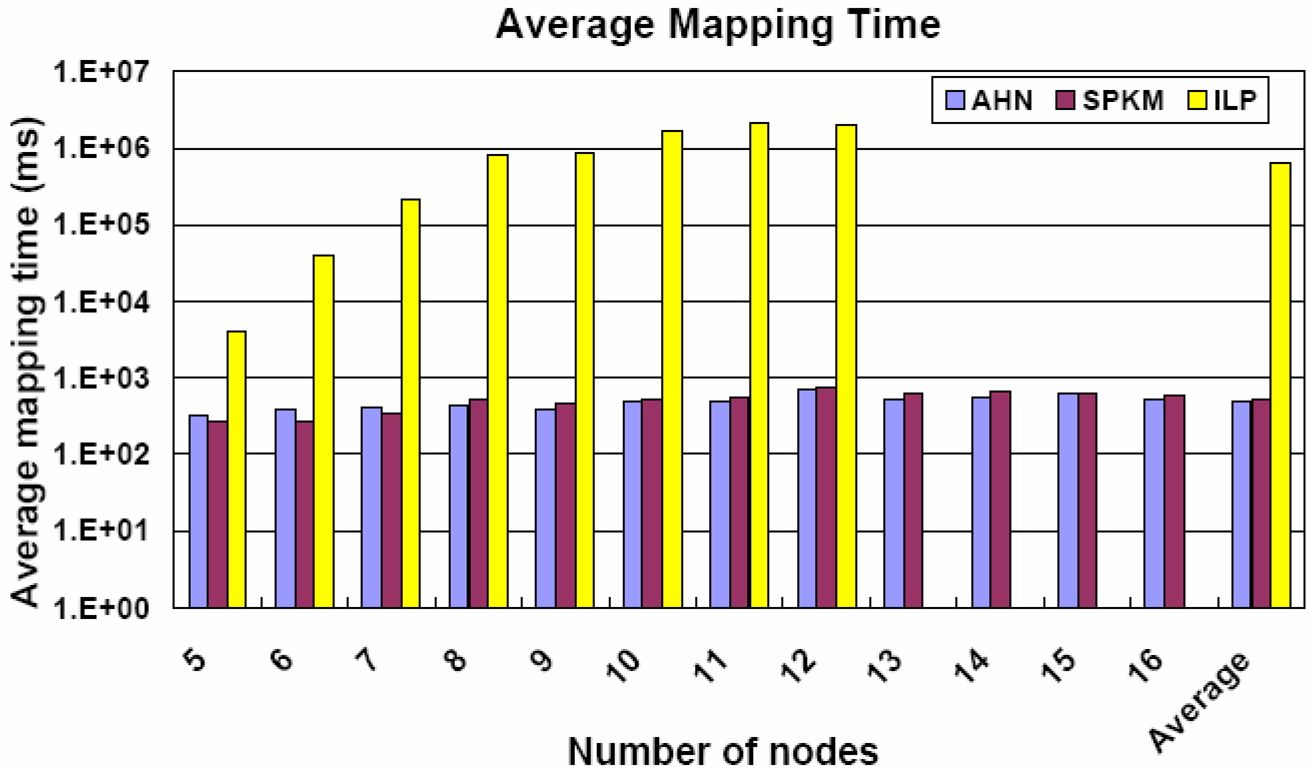
- The minimally utilized rows ➔ opportunities to map more operations ➔ ability to map more loop-unrolled kernel
- For 66% of the applications, SPKM generates better mappings

SPKM generates better mapping



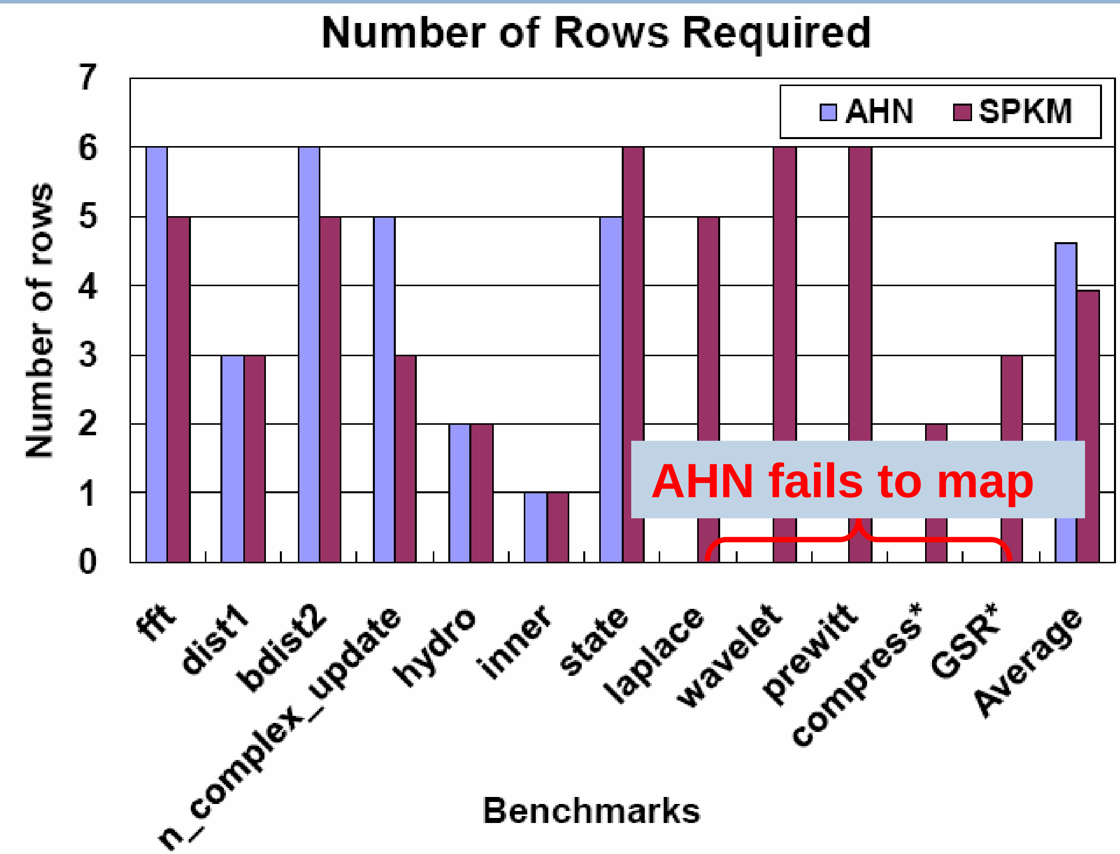
- The minimally utilized rows ➔ opportunities to map more operations ➔ ability to map more loop-unrolled kernel
- For 66% of the applications, SPKM generates better mappings

SPKM has less mapping time



- This execution time is measured with not-unrolled input kernels.
- SPKM has 8% less mapping time as compared to AHN.
- ILP takes very long time

SPKM for real benchmarks



- SPKM can map more benchmarks than AHN
- SPKM can map applications with larger unroll factor

Conclusion

- CGRA is a promising platform
- The success of CGRAs depends on the compiler
- CGRAs are becoming very complex
- We propose ILP approach and Graph-Drawing based heuristic, SPKM, that considers the details of CGRAs
- SPKM shows better ability to map application, better mapping quality (in power, performance), less mapping time than the existing heuristic

Thank you