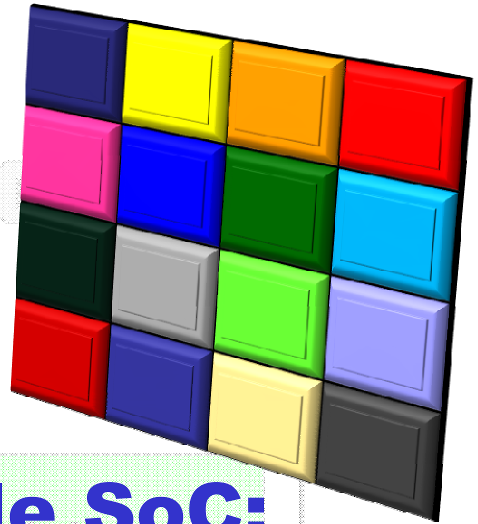


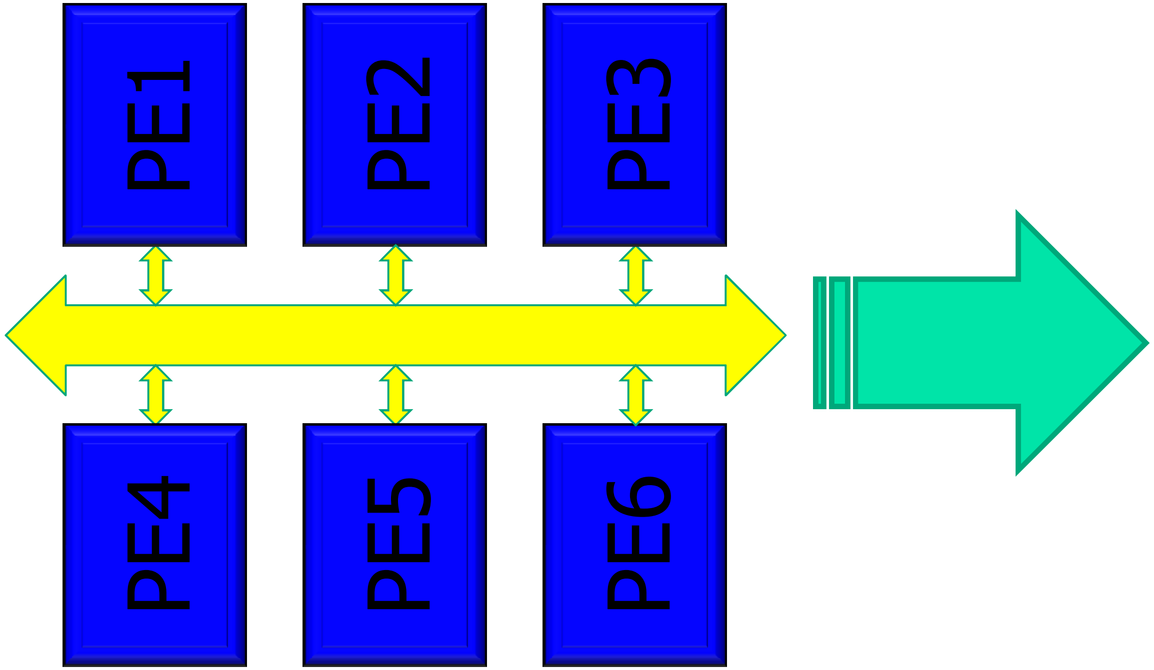
The Era of Many-Module SoC: Revisiting the NoC Mapping Problem

Isask'har (Zigi) Walter, Israel Cidon, Avinoam Kolodny, Daniel Sigalov

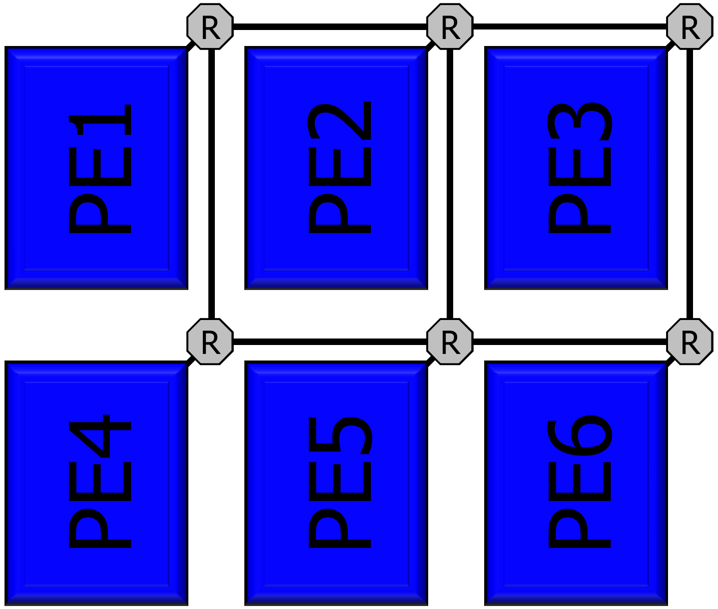
Technion – Israel Institute of Technology

December, 2009

SoC Revolution

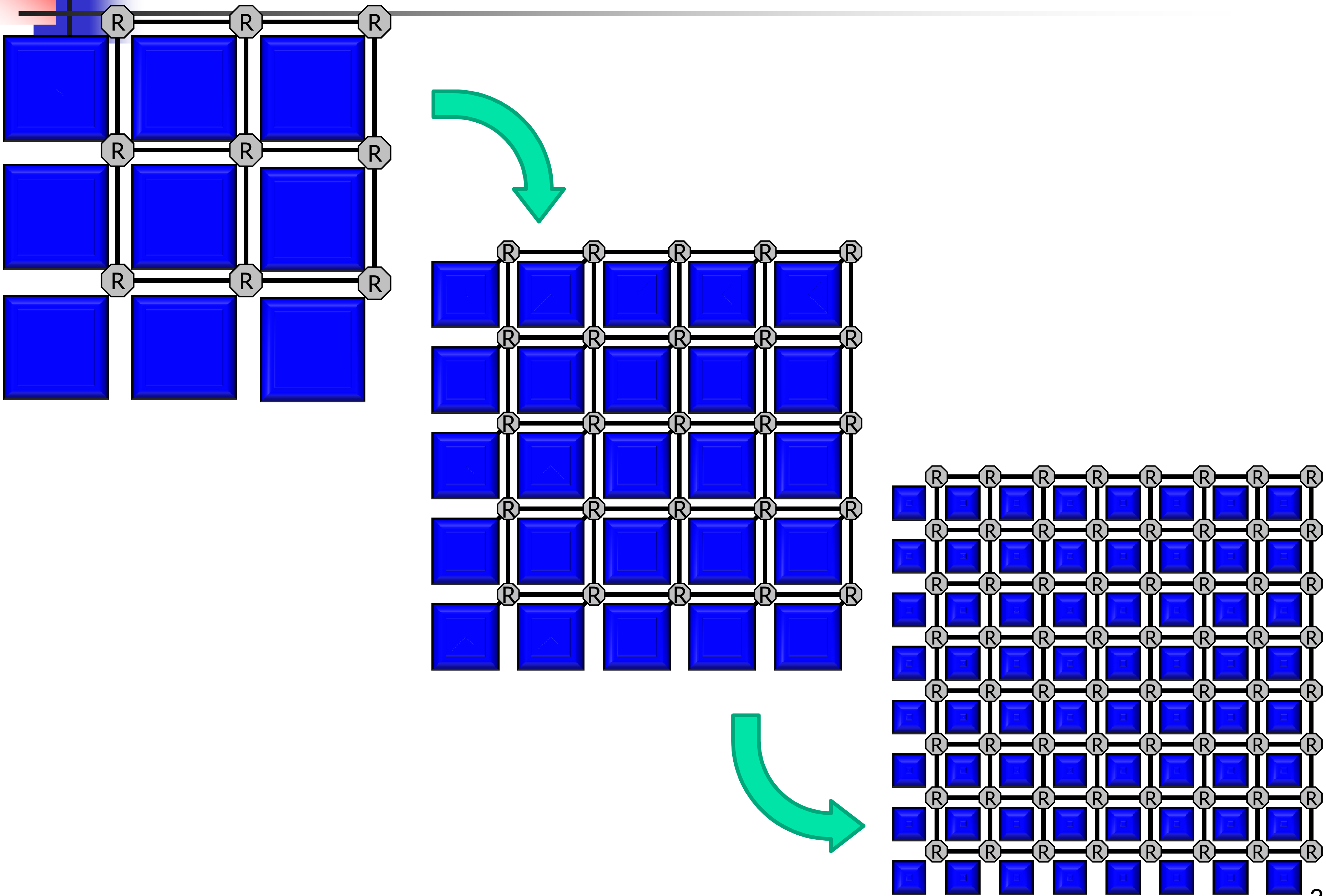


Bus-based system

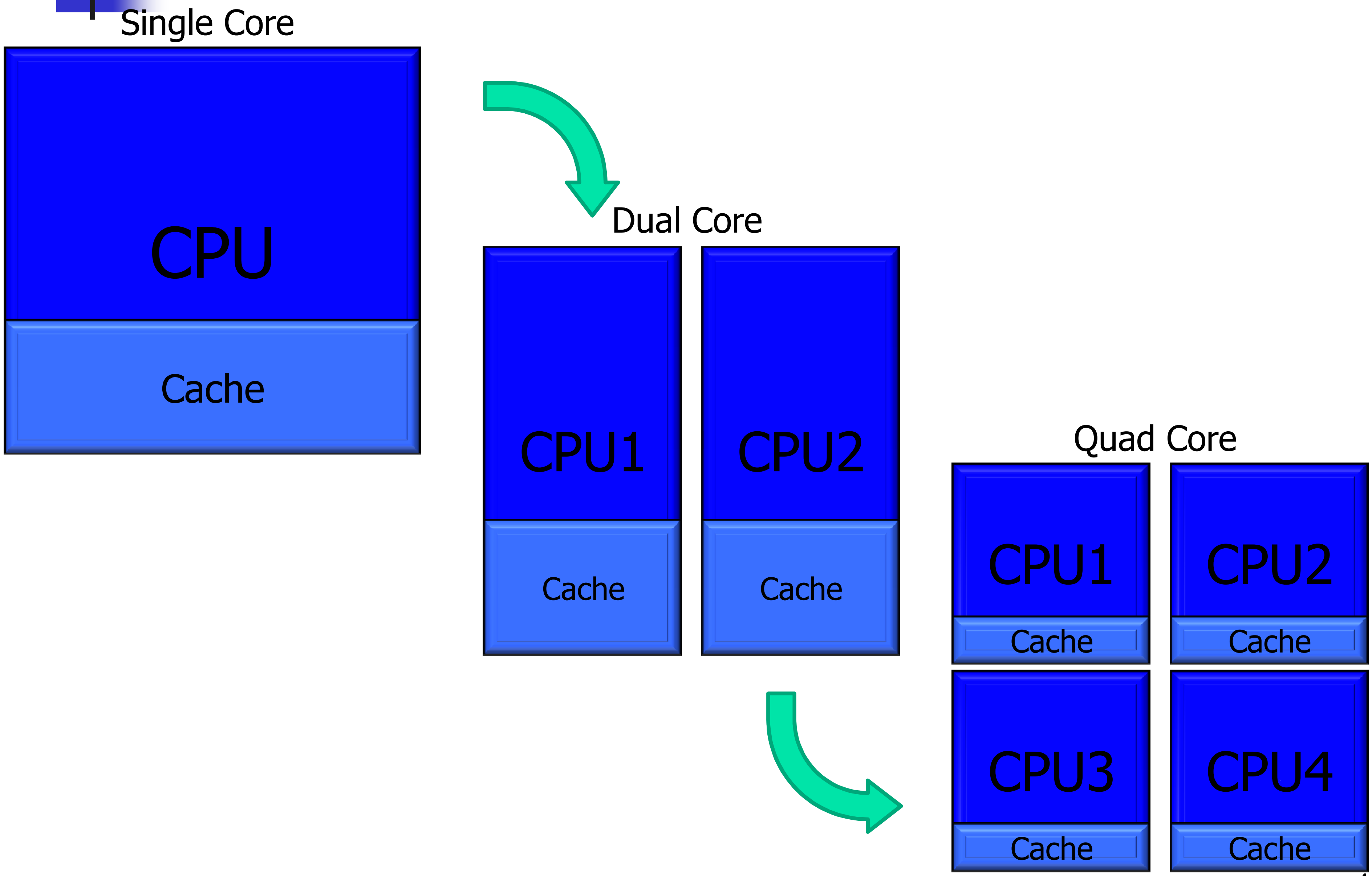


NoC-based system

SoC Evolution

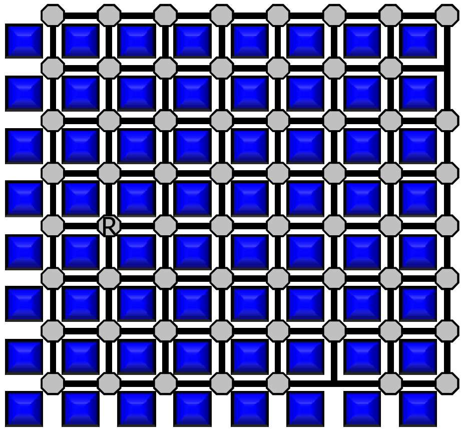


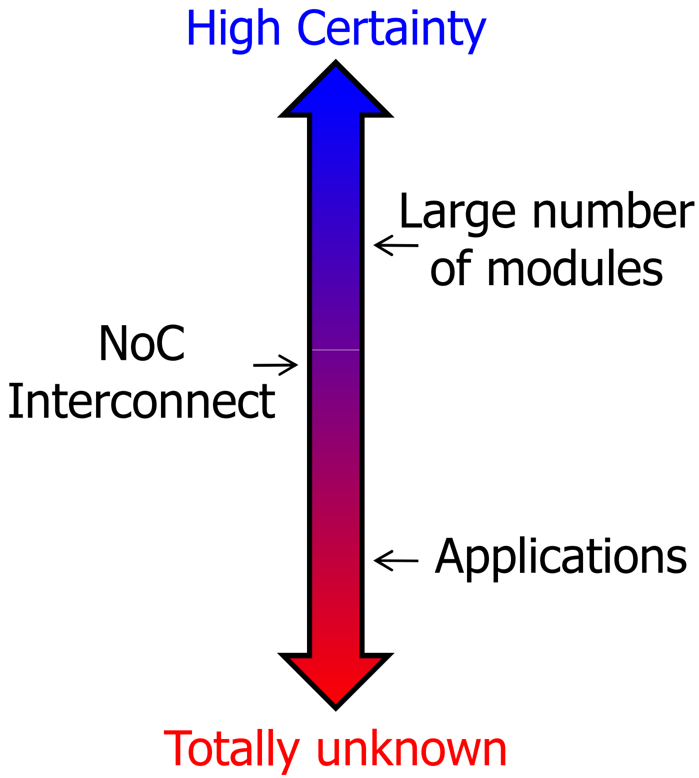
Processor Evolution



The Era of Many-Module SoC

- How would such chips be like?

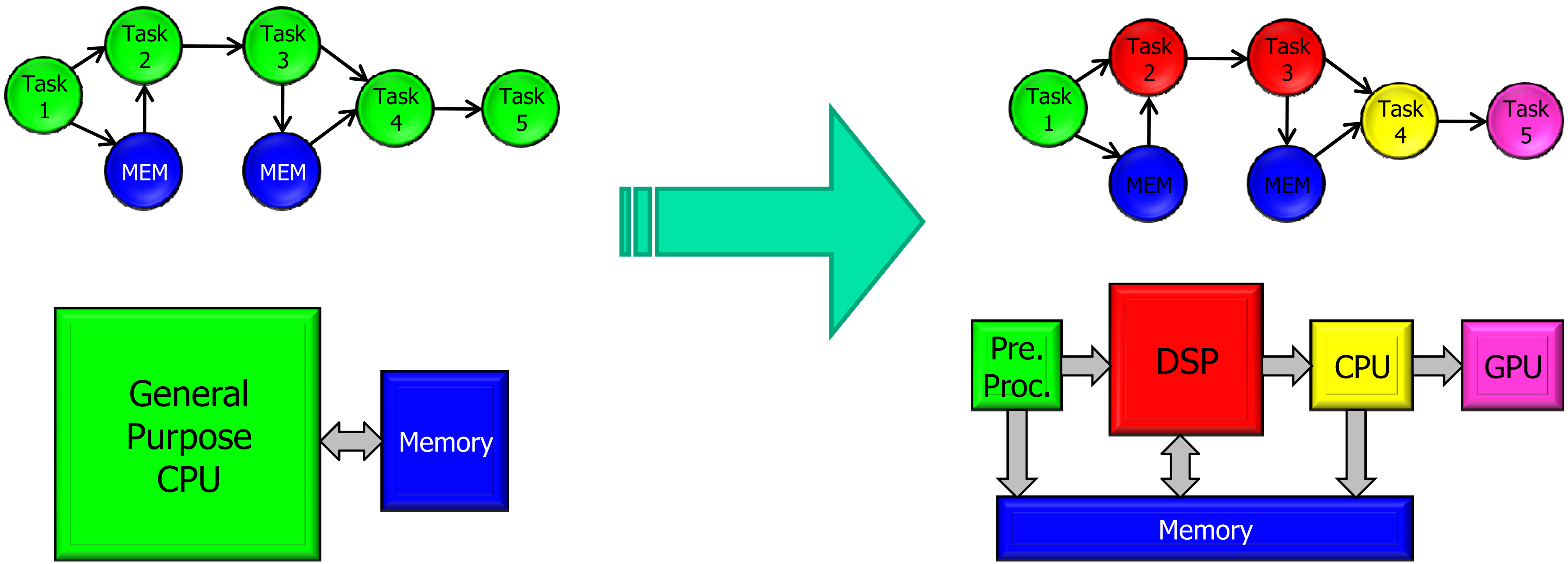




- Most likely
  - Power still important
  - Highly parallel
  - IP reuse
    - Ease of design and verification

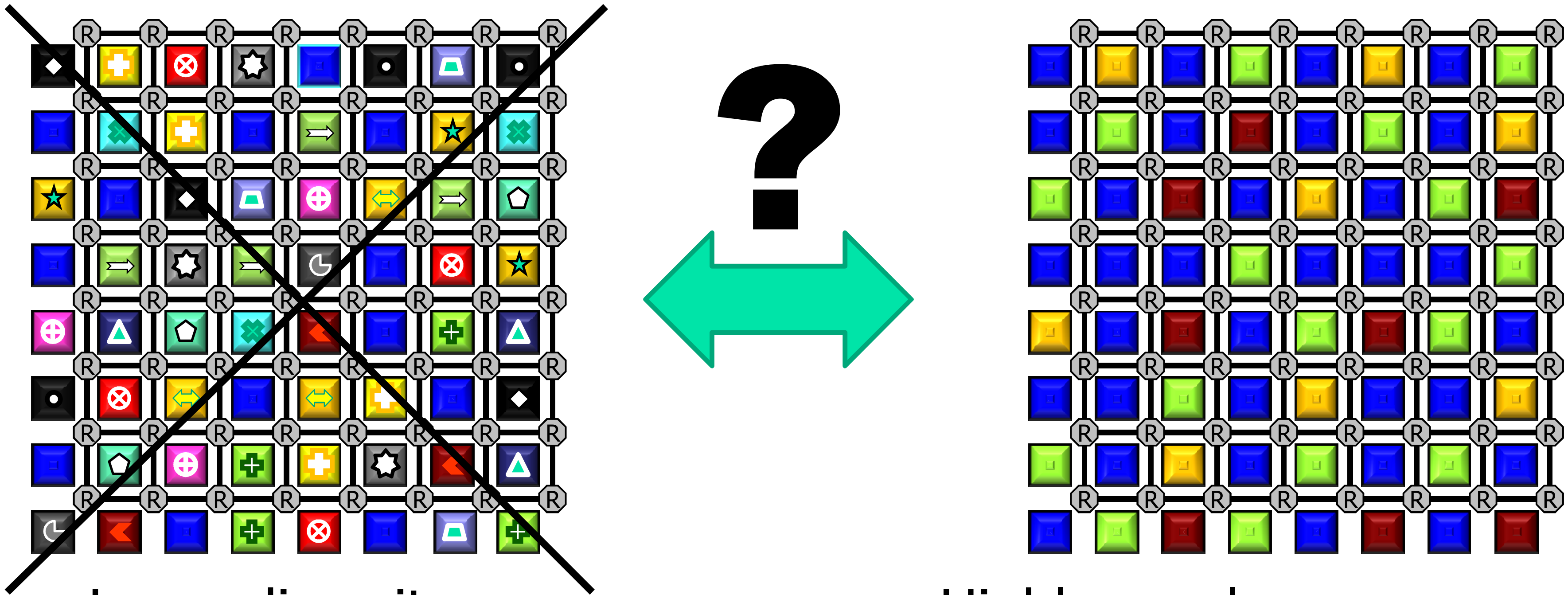
Future SoCs - Observation#1

- Special purpose cores replace general purpose processors
  - Power considerations



- Processing pipes are getting longer

Future SoCs - Observation#2



- Large diversity
- All modules are unique

- Highly regular
- Classes of Replicated cores
  - standard modules (DSP, HW accelerators, Cache banks, etc.)

The Era of Many-Module SoC

- Increased use of specialized cores
  - Pipes are getting longer

Observation#1

- Replication of processing elements

Observation#2

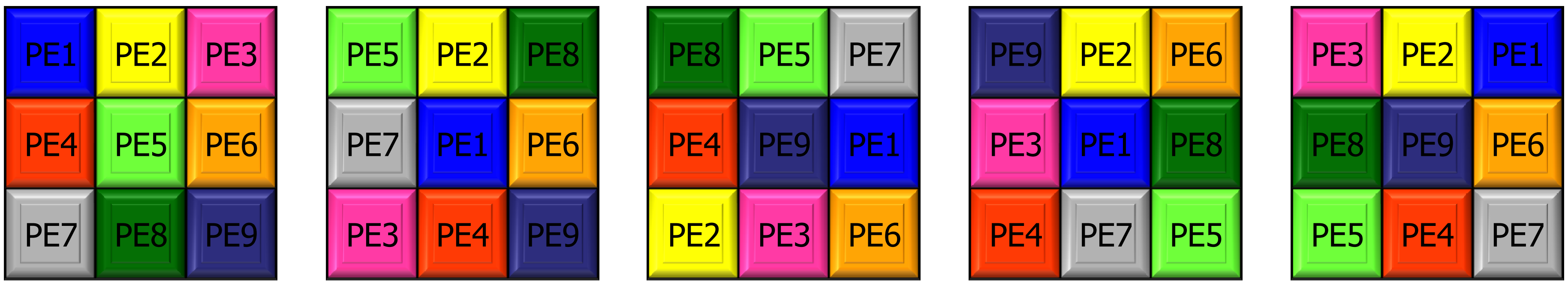
- How is the design flow affected?
  - This work – mapping of the NoC

Outline

- The Era of Many Module SoC
- Revisiting the Mapping Problem
- Cross-Entropy Optimization
- Evaluation

NoC Mapping

- Given
  - Traffic pattern(s)
    - a set (or sets) of pair-wise bandwidth requirements and timing constraints
  - Routing
  - Topology
- Goal
  - Find efficient mapping of cores to tiles



Mapping Optimization

- An important design step
  - Mapping affects power and performance!
- A difficult problem!
  - Often heuristic algorithms are used

- Common optimization goals
  - Minimize (dynamic) power
  - Minimize power + maximize performance
  - Minimize power subject to *performance constraints*

Modeling

- Typical modeling
  - Power and latency proportional to distance
  - Cost function:

$$Cost(\pi \in P) = \sum_{l \in L} BW_l = \sum_{1 \le i, j \le N} \left[b_{i \to j} \cdot Dist(i, j) \right]$$

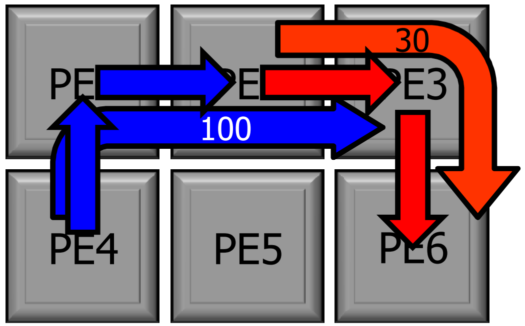
Calculating Mapping Cost

$$Cost(\pi \in P) = \sum_{l \in L} BW_l = \sum_{1 \le i, j \le N} \left[b_{i \to j} \cdot Dist(i, j) \right]$$

$$Cost(\pi_1) = \left[bw(PE_2 \to PE_6) \cdot Dist(PE_2 \to PE_6) \right] + \left[bw(PE_4 \to PE_3) \cdot bw(PE_4 \to PE_3) \right]$$

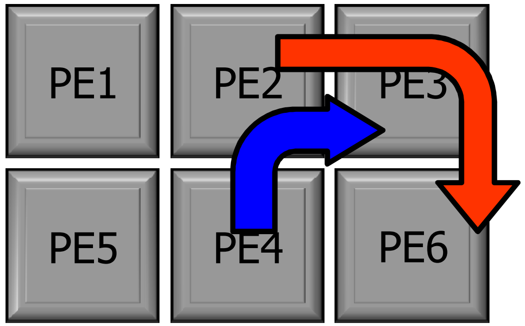
$$Cost(\pi_1) = 30 \cdot Dist(PE_2 \to PE_6) + 100 \cdot Dist(PE_4 \to PE_3)$$

$$Cost(\pi_1) = 30 \cdot 2 + 100 \cdot 3 = 360$$



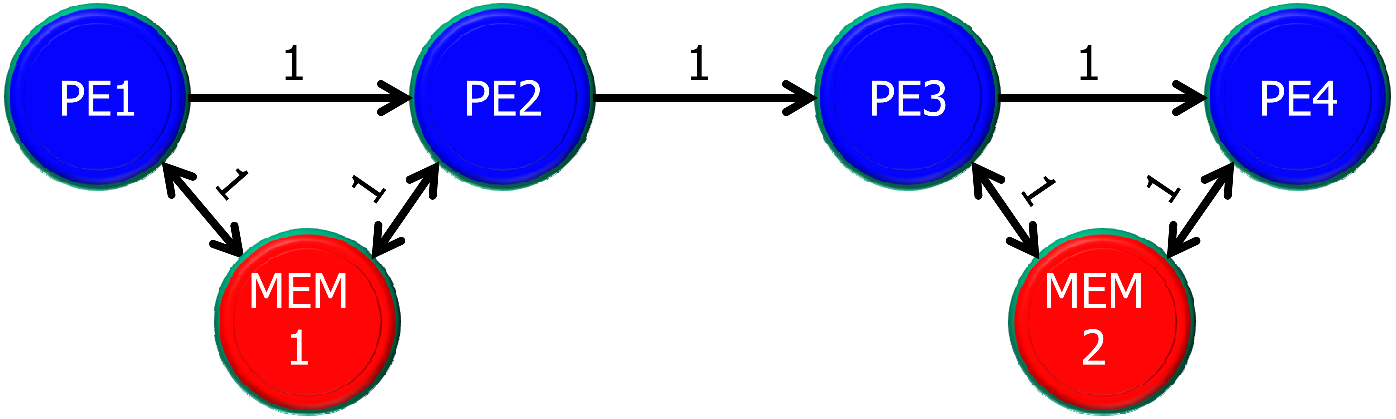
Mapping π_1

$$Cost(\pi_2) = 30 \cdot 2 + 100 \cdot 2 = 260$$

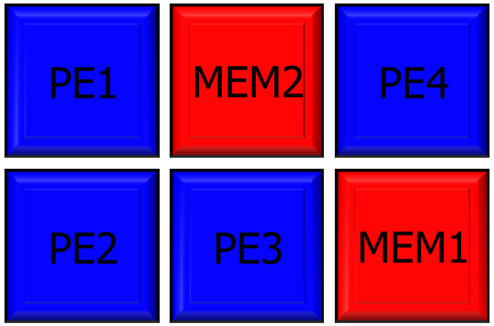


Mapping π_2

Motivation - Example #1



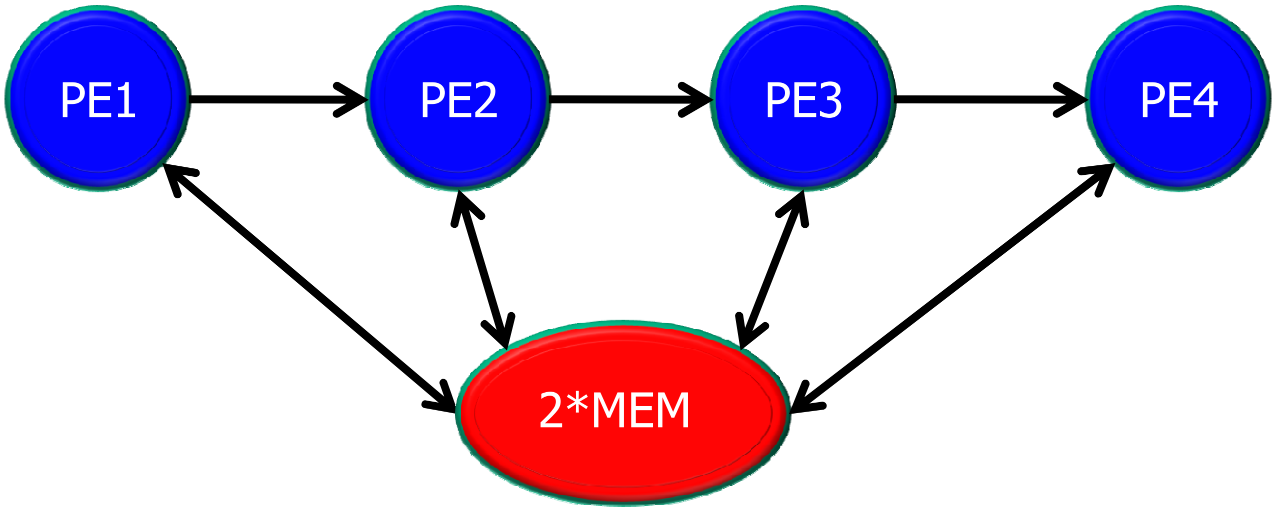
- Optimal mapping (π_1):



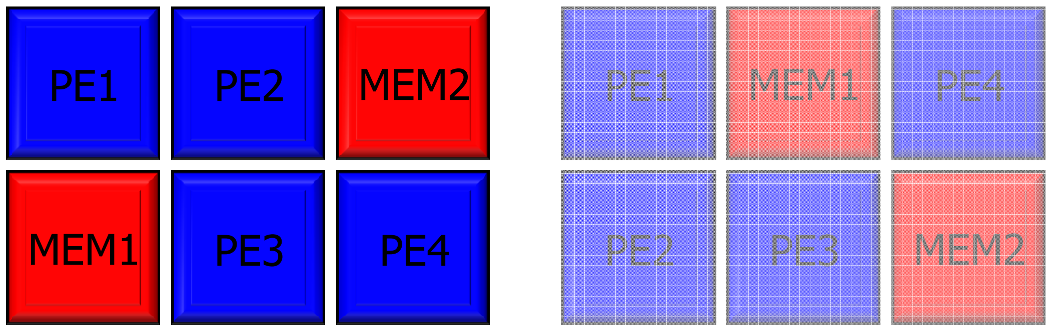
$$Cost(\pi_1) = \sum_{1 \le i, j \le N} \left[b_{i \to j} \cdot Dist(i, j) \right] = 9$$

Motivation - Example #1 (cont.)

- Let the mapping algorithm assign the flows!

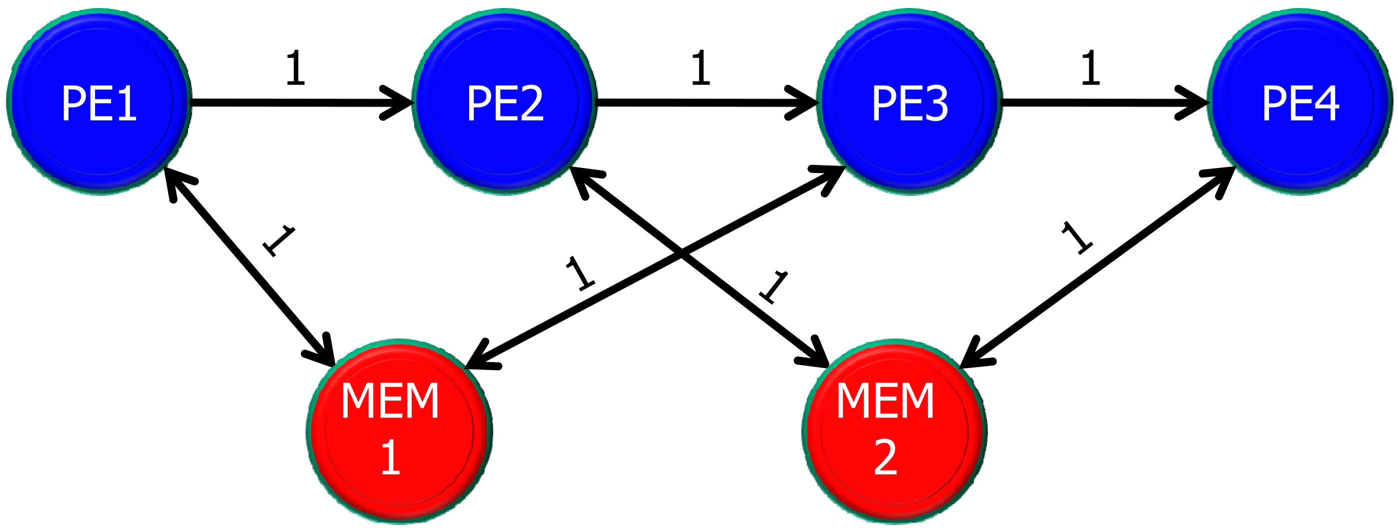


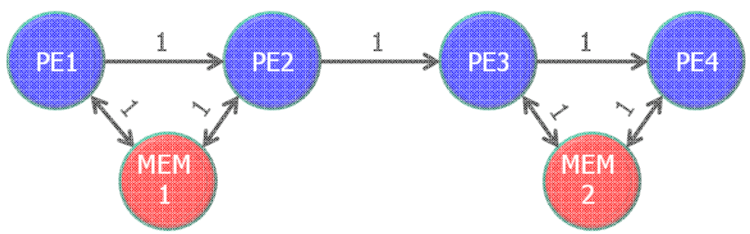
- Optimal mapping (π_2):

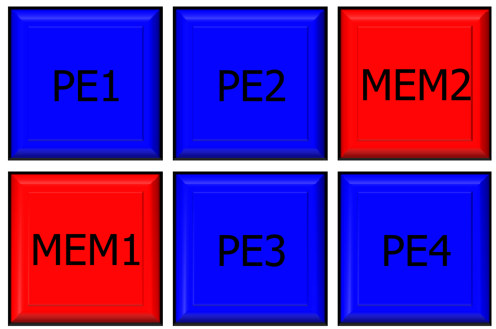


Cost(π_2)=7 Cost(π_1)=9

Motivation - Example #1 (cont.)







$Cost(\pi_2)=7$

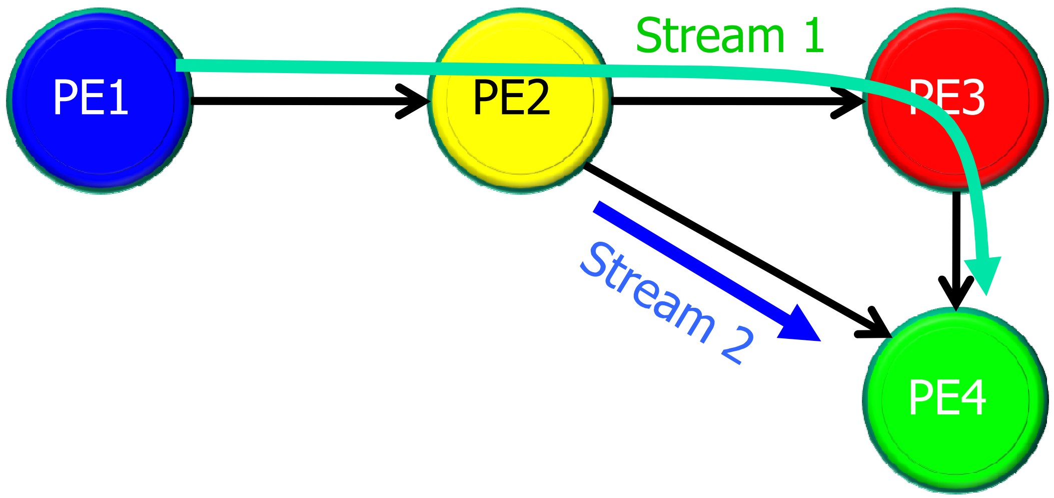
- The mapping algorithm should be aware of replicated modules!

Classic Performance Constraints

- Pair-wise point-to-point requirements
- For example, in a 4-module system:

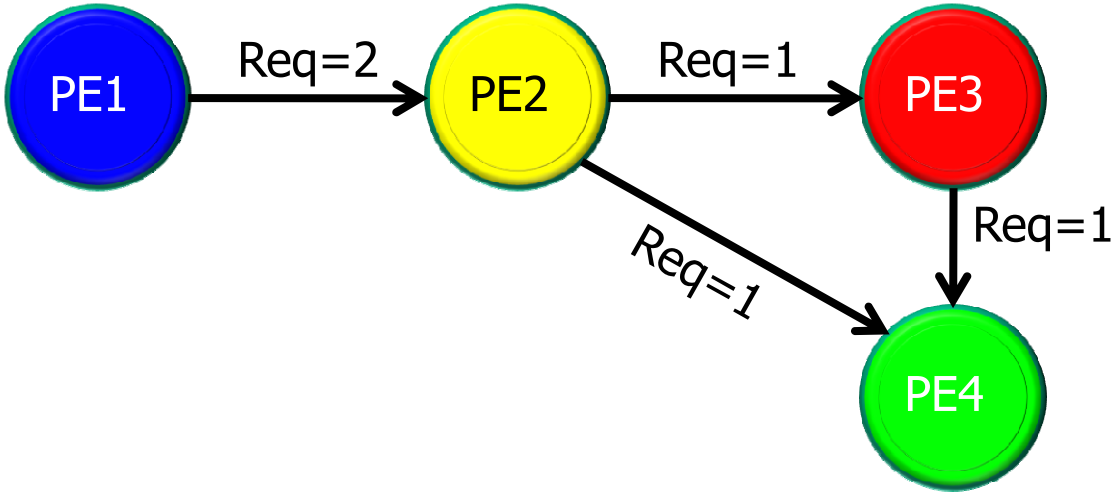
PE1				
PE2	2			
PE3		1		
PE4		1	1	
	PE1	PE2	PE3	PE4

Motivation - Example #2

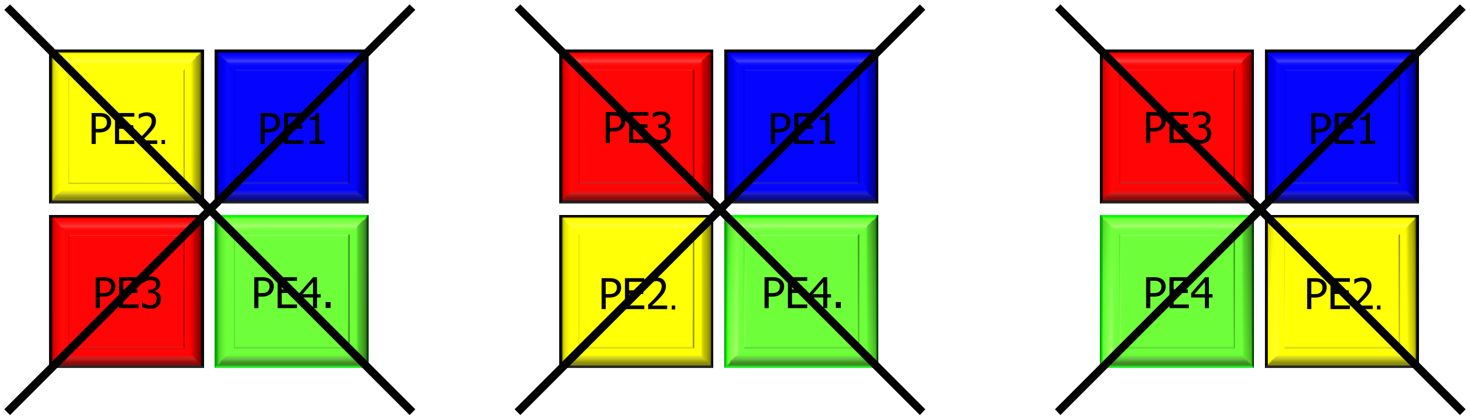


Stream ID	PEs	Timing Requirement
Stream 1	PE1→PE2→PE3→PE4	4
Stream 2	PE2→PE4	1

Example #2 – Pair-wise req.

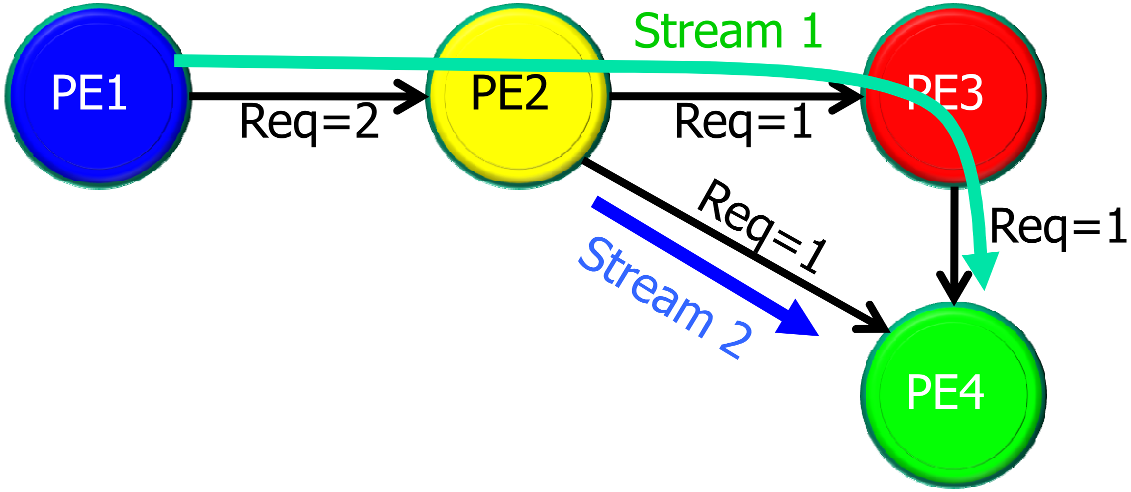


PE2	2		
PE3.		1	
PE4		1	1
	PE1	PE2	PE3



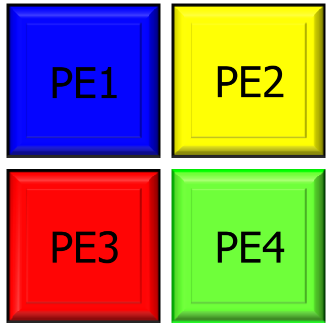
- No feasible mapping!

Application-Level Requirements



Stream ID	PEs	Requirement
Stream 1	PE1→PE2→PE3→PE4	4
Stream 2	PE2→PE4	1

- A feasible mapping does exist!



- It's better to work with the application level requirements

This Work

- Find efficient mappings by extending the formulation of the mapping problem
  - Adding degrees of freedom

- Degree of freedom #1
  - Leverage existence of replicated modules
- Degree of freedom #2
  - Replace p2p constraints with end-to-end, application-level requirements

Modifying the Formulation (1)

- Leverage existence of replicated modules
  - Allow the mapping algorithm to allocate flows to the best replicated module

Flow	BW	Time Req.
$PE_1 \rightarrow DSP_3$	100	3
$PE_2 \rightarrow DSP_4$	200	12
$PE_2 \rightarrow SRAM_1$	100	15
$PE_3 \rightarrow SRAM_2$	100	5
…	…	…

Flow	BW	Time Req.
$PE_1 \rightarrow$ <ANY DSP>	100	3
$PE_2 \rightarrow$ <ANY DSP>	200	12
$PE_2 \rightarrow$ <ANY SRAM>	100	15
$PE_3 \rightarrow$ <ANY SRAM>	100	5
…	…	…

Modifying the Formulation (2)

- Replace p2p constraints with end-to-end, application-level requirements

∞									
2	∞								
3	4	4							
3	1	3	∞						
3	7	7	4	∞					
∞	4	2	∞	3	∞				
5	∞	3	2	7	∞	4			
∞	6	5	1	∞	2	∞	∞		
1	∞	3	5	∞	∞	3	7	7	
1	3	∞	2	1	2	3	∞	3	∞

P2P timing req.

Stream ID	Stream's PEs	E2E Req.
1	$PE_1 \rightarrow PE_3 \rightarrow PE_9 \rightarrow PE_4 \rightarrow PE_{10}$	23
2	$PE_5 \rightarrow PE_2 \rightarrow PE_3 \rightarrow PE_8 \rightarrow PE_7 \rightarrow PE_6 \rightarrow PE_{10}$	12
3	$PE_5 \rightarrow PE_3 \rightarrow PE_9$	15
4	$PE_7 \rightarrow PE_8 \rightarrow PE_2 \rightarrow PE_3$	20
5	$PE_1 \rightarrow PE_2$	2
…	…	…

E2E timing req.

- In this paper, for synthetic task graphs
  - Did so for a real application too

Outline

- The Era of Many Module SoC
- Revisiting the Mapping Problem
- Cross-Entropy Optimization
- Evaluation

Cross Entropy Optimization

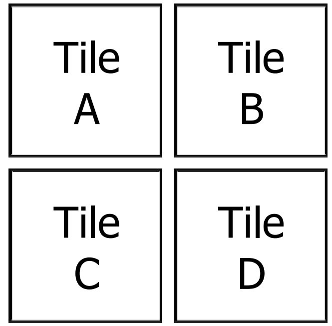
- Modern optimization heuristic
  - Good at combinatorial optimization problems
- Akin to evolutionary algorithms
  - Generation of new solutions is based on sampling and estimation
- Inherently a global search method
  - Reduced risk of getting trapped in a local minimum

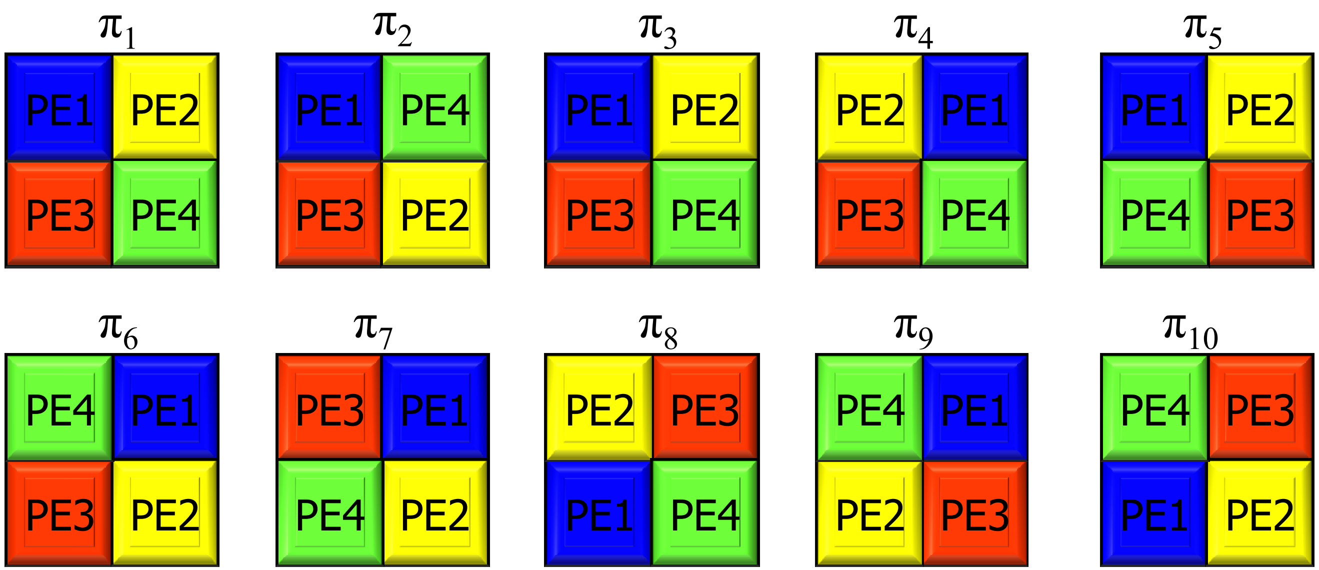
Cross Entropy Optimization

- Given an initial parameter vector $v=v_0$, sample a random population of K solutions $x_1,x_2,\ldots,x_k$ from the distribution given by $f(x;v)$.
- Evaluate the costs $S(xi), i=1,\ldots,K$.
- Using the ρK ($0<\rho<1$) elite (lowest cost) samples, obtain a new density function $f(x;v)$ by calculating a new vector v via Maximum Likelihood (ML) estimation.
- Repeat steps 1-3 with the new vector v unless maximum number of iterations is reached or no improvement is obtained for a predefined number of iterations.

- For example:
  1. Generate 10 random mappings: $\pi_1, \pi_2, \ldots, \pi_{10}$
  2. Find 3 lowest cost mappings: π_2, π_5, π_7
  3. Examine those 3 best mappings:
     A. For each tile, calculate the probability core PE_i is mapped to that tile
     B. Update probabilities accordingly

CE Example





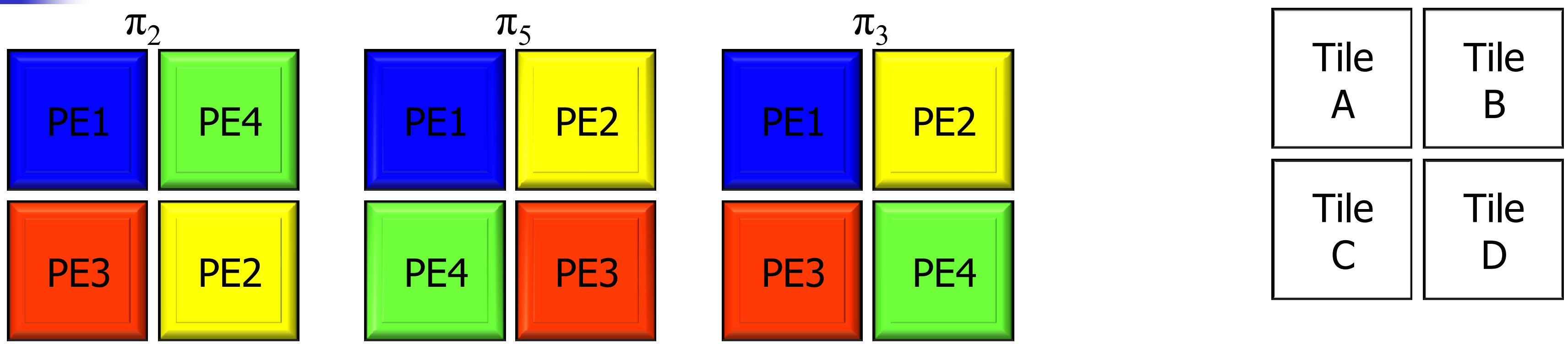
$\text{Prob}(\text{TileA} \leftarrow PE_1) = \text{Prob}(\text{TileA} \leftarrow PE_2) = \text{Prob}(\text{TileA} \leftarrow PE_3) = \text{Prob}(\text{TileA} \leftarrow PE_4) = 0.25$

$\text{Prob}(\text{TileB} \leftarrow PE_1) = \text{Prob}(\text{TileB} \leftarrow PE_2) = \text{Prob}(\text{TileB} \leftarrow PE_3) = \text{Prob}(\text{TileB} \leftarrow PE_4) = 0.25$

$\text{Prob}(\text{TileC} \leftarrow PE_1) = \text{Prob}(\text{TileC} \leftarrow PE_2) = \text{Prob}(\text{TileC} \leftarrow PE_3) = \text{Prob}(\text{TileC} \leftarrow PE_4) = 0.25$

$\text{Prob}(\text{TileD} \leftarrow PE_1) = \text{Prob}(\text{TileD} \leftarrow PE_2) = \text{Prob}(\text{TileD} \leftarrow PE_3) = \text{Prob}(\text{TileD} \leftarrow PE_4) = 0.25$

Updating Probabilities



- Prob(TileA←PE_1)=1
- Prob(TileB←PE2)=2/3
- Prob(TileB←PE4)=1/3
- Prob(TileC←PE3)=2/3
- Prob(TileC←PE4)=1/3
- Prob(TileD←PE2)=1/3
- Prob(TileD←PE3)=1/3
- Prob(TileD←PE4)=1/3

- Following iteration uses these updates probabilities
- Gradually, probabilities converge to 0/1

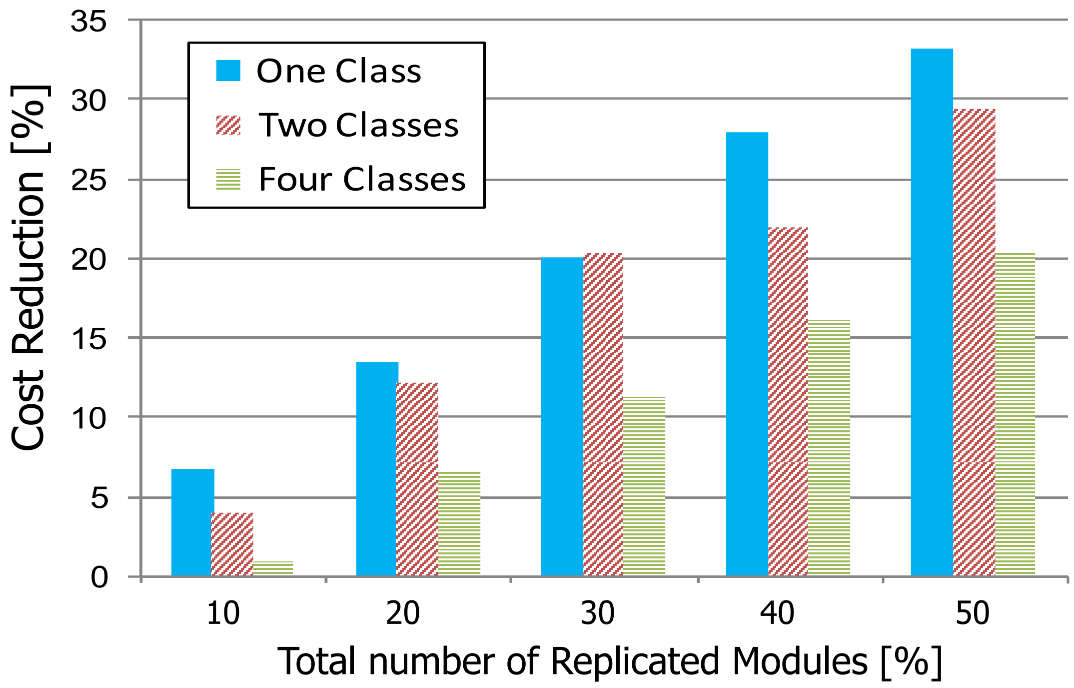
Outline

- The Era of Many Module SoC
- Revisiting the Mapping Problem
- Cross-Entropy Optimization
- Evaluation

Evaluation

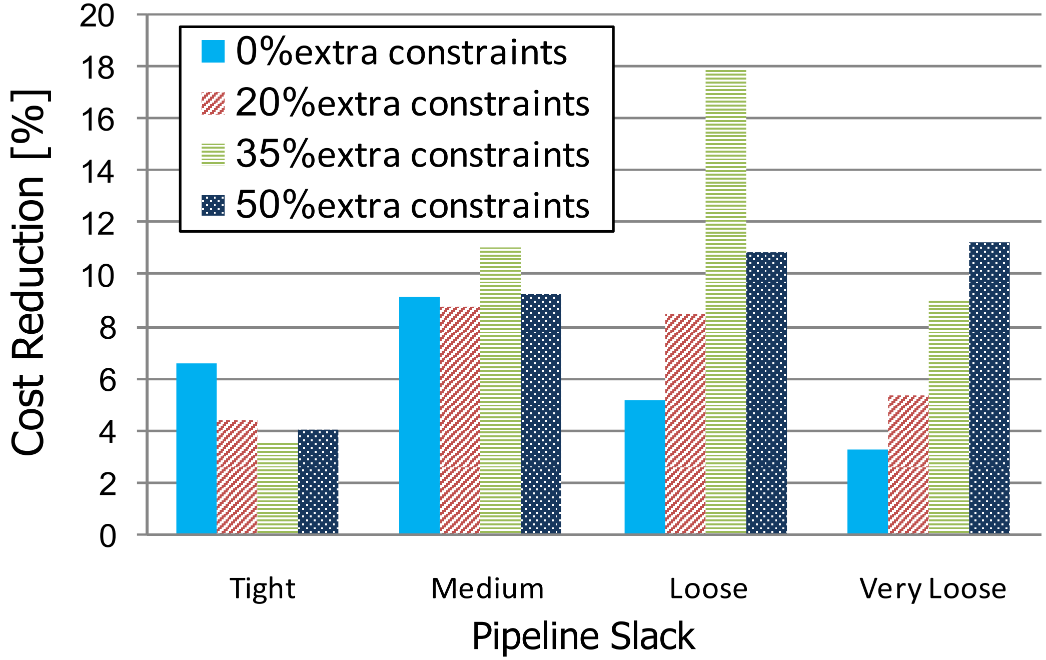
- Scenario
  - 6x6 mesh NoC
  - Synthetic, randomized SoC
    - Task graphs (and task-to-core mapping)
    - Varying number of replicated modules
    - Varying timing constraints
    - (Real application in DATE10 paper)
- Compare with best cost of classic mapping
  - Averaging multiple runs

Accounting for Replication



- "Class": a group of identical PEs
  - Total number of replicated cores= {Number of classes}*{class size}

Application-Level Requirements



- SoCs with a pipeline data path and background P2P traffic
  - Varying pipeline slack
  - Different amounts of background constraints

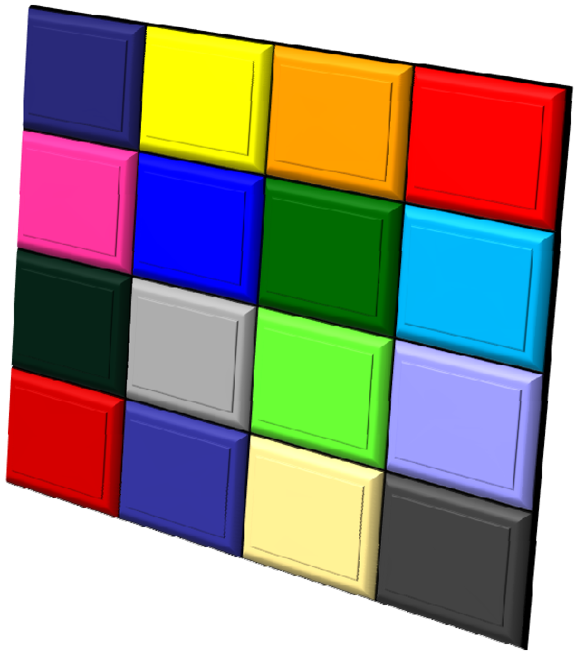
Conclusions and Future Work

- We are going into the era of "Many module SoC"
- Extend the mapping to account for
  - Classes of replicated modules
  - Application-level requirements
- Meaningful power savings

- But mapping is an example
  - Routing? Task assignment? Link design? Topology selection?

The Era of Many-Module SoC

Thank you!



Questions?

zigi@tx.technion.ac.il

QNoC Research Group

QNoC Research Group