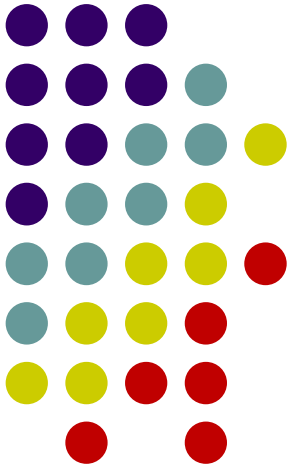
Memory System Synthesis for MPSoCs with 3D-stacked Memories



Yi-Jung Chen (陳依蓉)

Department of Computer Science & Information Engineering

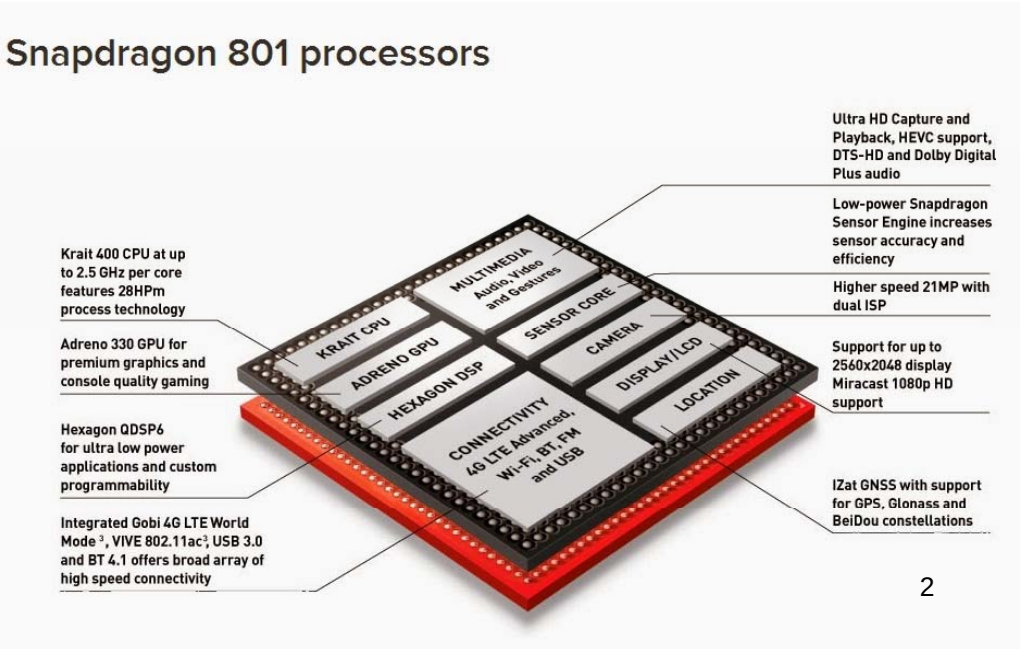
National Chi Nan University

國立暨南國際大學
National Chi Nan University

Design Challenges of Modern Embedded Systems

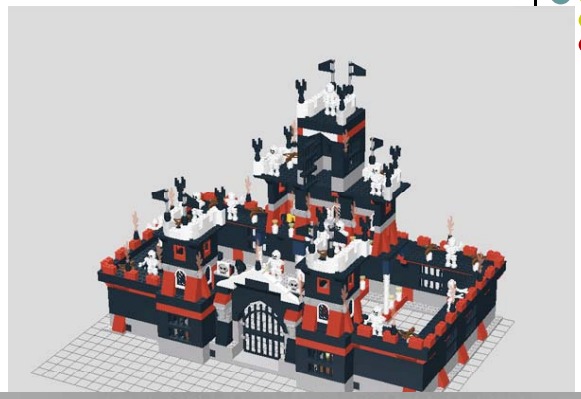
- Highly complex
  - An MPSoC architecture
    - MPSoC: Multi-Processor System-on-Chip
  - Several to tends of cores or IPs (e.g., CPU, mp3 encoder/decoder, JPEG encoder/decoder etc.)
    - Billions to trillions of transistors in a chip
  - It is very possible to have hundred of cores in the future

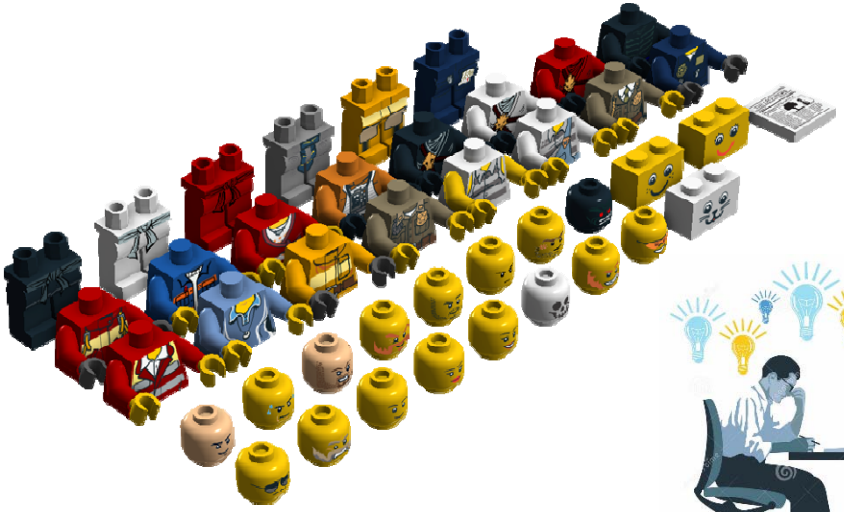
- Design challenges
  - Impossible to start the design from transistor-level
  - High memory bandwidth demands



System-level Synthesis

- Taking CPUs, IP cores, and memory modules as the basic building blocks
- Building the target system by these blocks based on the requirements & constraints of the target system.



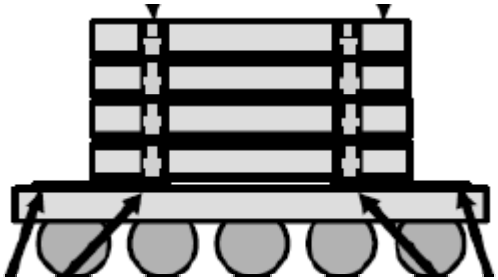


Outline

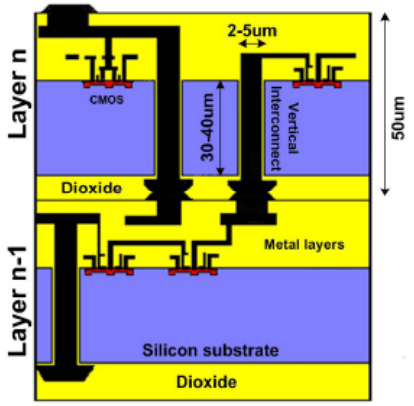
- Overview of system-level synthesis
- 3D-enabled MPSoC and Memory Integration
- Synthesis of Distribute Memory Interface for MPSoCs with 3D-stacked DRAMs
- Synthesis of Memory & Data Allocation for MPSoCs with Reconfigurable 3D-stacked SRAMs
- Conclusion

The 3D Integration Technology

- Utilizing Through Silicon Vias (TSVs) to stack multiple active device in the third dimension
- Properties of TSVs
  - Very low latency compared to traditional wires of 2D ICs
  - High density
    - More than 300 1K-bit TSV buses can be integrated on a 1cm^2 chip
- Heterogeneous integration
  - E.g., CPU+DRAM, CPU+Flash



3D die-stacking [JSAP '01]

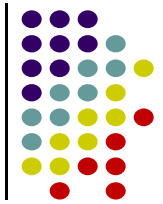


Cross-section view of an n-layer stacked layers connected by TSVs [DAC '06]

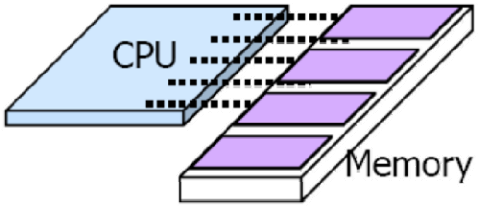
• Takahashi et al., "Current status of research and development for three-dimensional chip stack technology", JSAP '01

•Loi et al., "A thermally-aware performance analysis of vertically integrated (3-D) processor-memory hierarchy", DAC '06

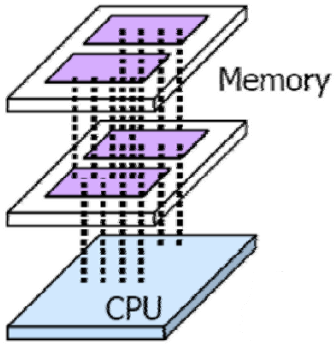
3D MPSoC-Memory Connection to Solve the Memory Wall Problem



- **How can 3D technology help address the *Memory Wall* problem?**
  - **High degree of integration**
    - **integrating more memories with CPUs in a small area**
  - **High TSV density**
    - **Providing abundant bandwidth**
  - **Heterogeneous integration**



2D CPU-memory connection [ISCA '08]



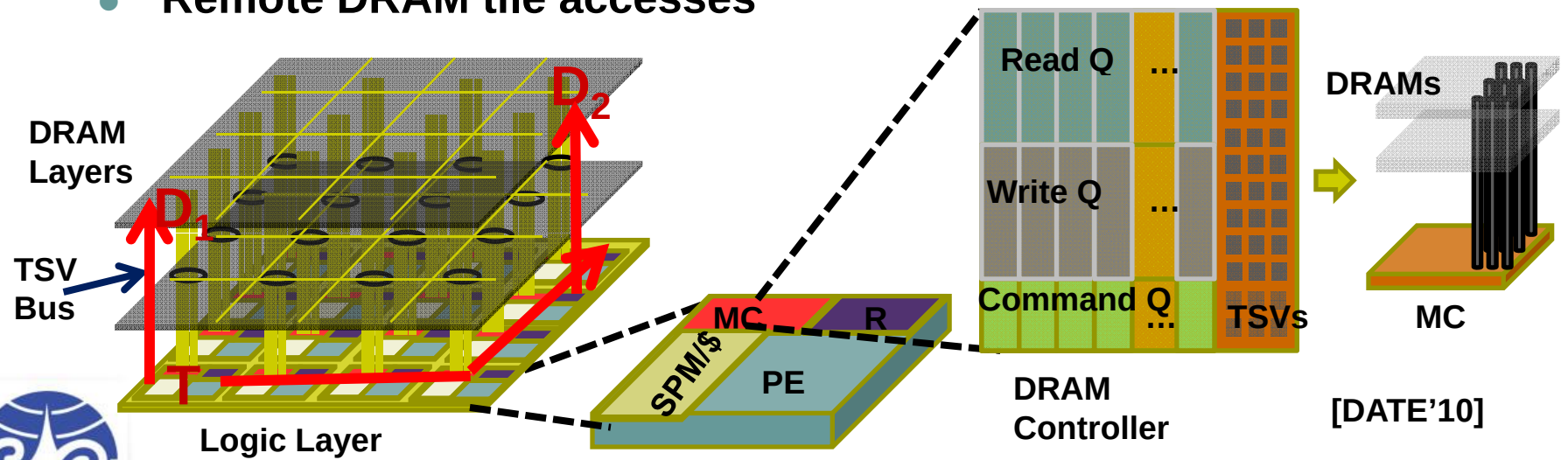
3D CPU-memory connection [ISCA '08]

Outline

- Overview of system-level synthesis
- 3D-enabled MPSoC and Memory Integration
- Synthesis of Distribute Memory Interface for MPSoCs with 3D-stacked DRAMs
- Synthesis of Memory & Data Allocation for MPSoCs with Reconfigurable 3D-stacked SRAMs
- Conclusion

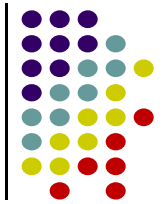
MPSoC with 3D-stacked DRAMs

- A generic design for MPSoC with 3D-stacke DRAMs
  - One layer of MPSoC
    - Composed of regular tiles that are connected by a 2D mesh network
    - Each tile has a PE (Processing Element), a local memory controller (MC), a SPM (scratch-pad memory), and a router
    - All MCs has the same # of TSVs
- Two kinds of DRAM accesses in the 3D NoC
  - Local DRAM tile accesses
  - Remote DRAM tile accesses



Loi et al, "An efficient distributed memory interface for many-core platform with 3D stacked DRAM", DATE 2010.

A New Design Dimension in the Memory Subsystem

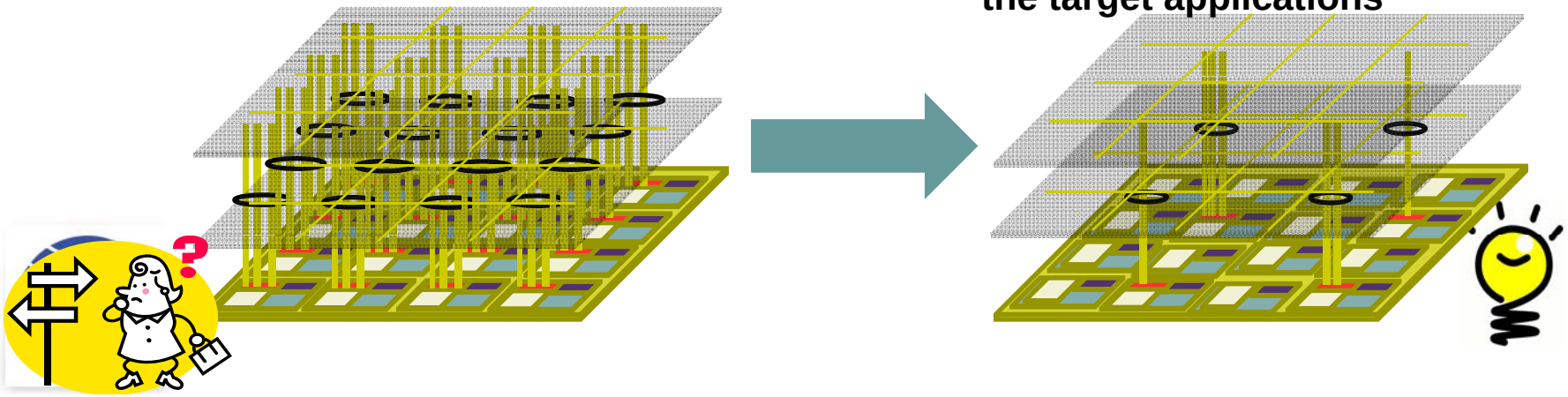


- **Is it necessary to have a generic design?**
  - **More TSVs**
    - **Degradation of chip yield [ASPDAC 2009]**
    - **Higher costs (e.g., manufacturing, testing, etc.)**
  - **More MCs**
    - **Taking more tile resources**
      - **If a tile does not need an MC, the resource can be traded for other modules (e.g., more scratch-pad memory (SPM) capacity)**

An optimization opportunity of tailoring the distributed memory interface design to the need of the target applications

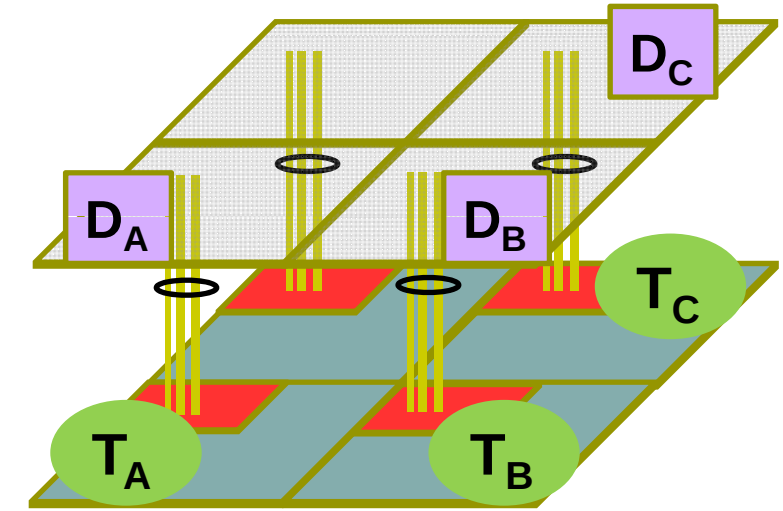
Generic design

A customized design tailored for the target applications



Design Challenges

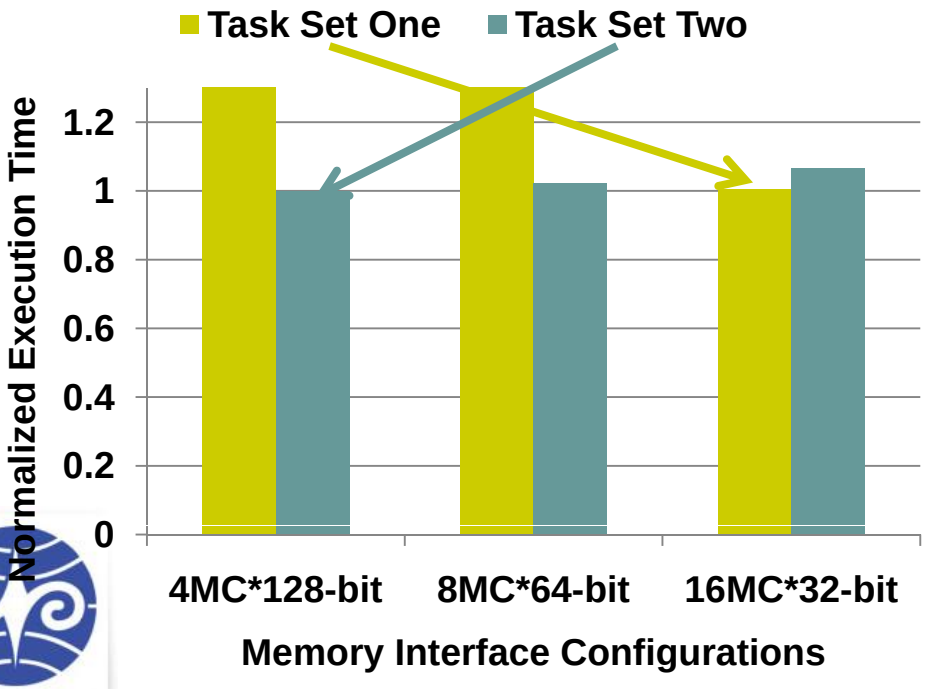
- **HW/SW co-design is necessary**
  - ***Task mapping*** **and** ***data assignment*** **should be redesigned for a design of the distributed memory interface to achieve good performance**

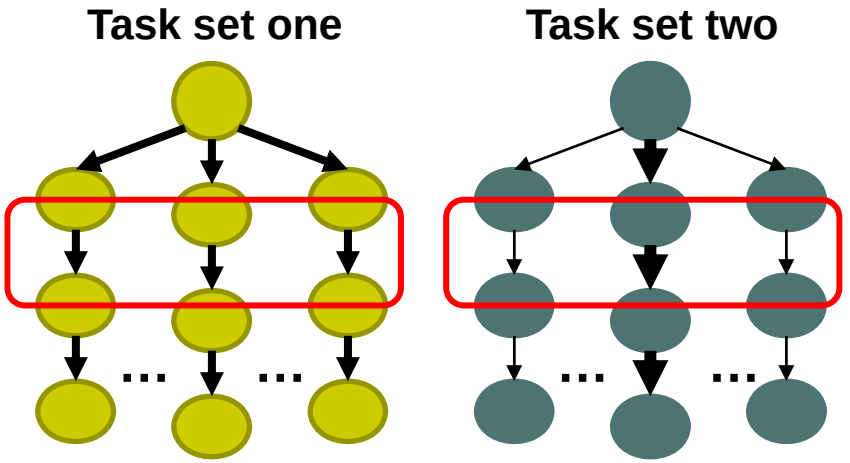


$T_C \leftrightarrow D_C > T_B \leftrightarrow D_B$

Design Challenges (cont.)

- **Program behaviors that affect the allocation of MCs and TSVs**
  - **# of critical data accesses issued in parallel → # of MCs needed**
  - **# of bits accessed per data access → TSV data bus width needed by each MC**

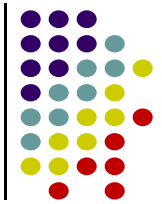




Same total bandwidth
Different access distribution

Same total # of TSV data bits, different # of MCs

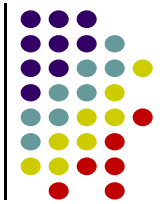
Contribution of this research



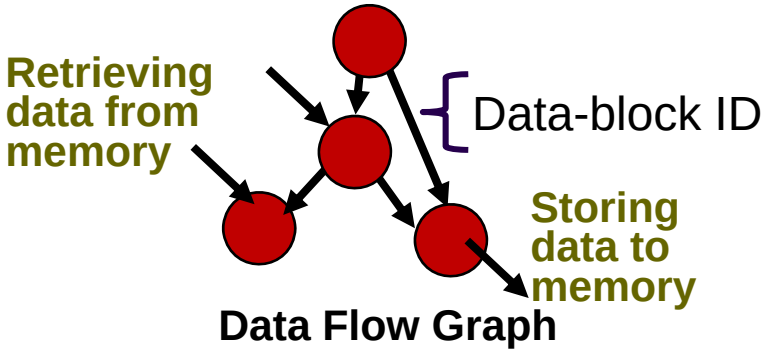
- **The first synthesis framework for the distributed memory interface of MPSoCs with 3D-stacked DRAMs**
  - **Designed for application-specific systems**
    - **In application-specific systems, the behavior of the system (e.g., applications executed in the system) is known as a priori**
  - **The synthesis algorithm**
    - **Input:**
      - **the hardware architecture and application behavior**
    - **Output:**
      - **HW architecture: the configuration of the memory interface**
      - **SW architecture: the configuration of task allocation and data mapping**
  - **Goal:**
    - **Find a design of the distributed memory interface that utilizes the least resources while the performance constraint is met**
- **This part of research is published in [ICCAD 2012]**

Y.-J. Chen, C.-L. Yang, and J.-J. Chen, "Distributed memory interface synthesis for network-on-chips with 3D-stacked DRAMs", ICCAD '12.

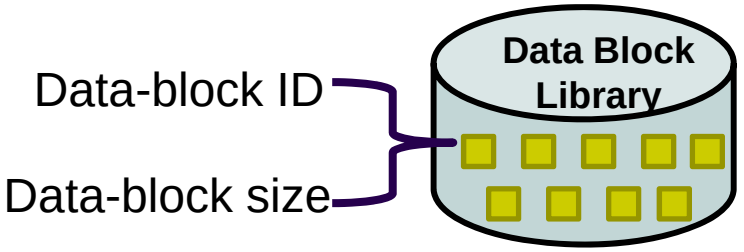
System Specifications – Application Model



- **Data flow graphs (DFG)**
  - **Vertex**
    - **a task or a kernel function**
  - **Edge**
    - **Transferring of a data block (a collection of scalars or arrays)**
    - **Each edge is associated with a data block id**

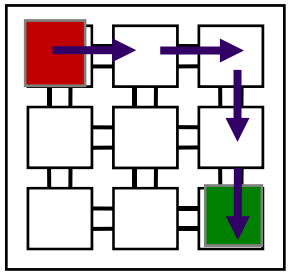


- **Data block library**
  - **Storing the information of the data blocks that are accessed in the task set**

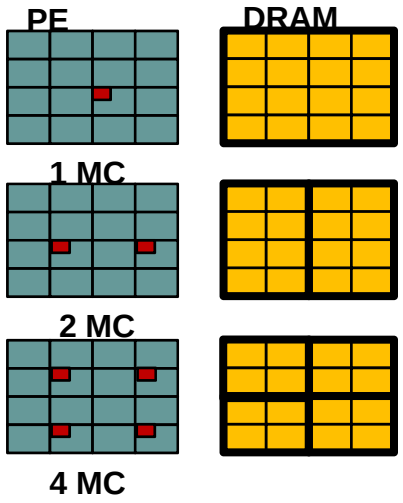


System Specification – Architectural Model

- **NoC model**
  - 2D Mesh topology
    - M x N nodes
  - X-Y routing

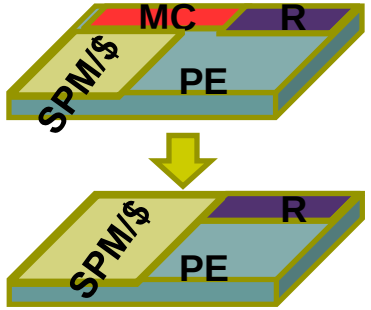


NoC



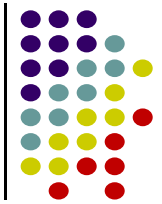
- **DRAM model**
  - DRAMs are partitioned into M x N tiles

- **Model of the distributed memory interface**
  - 3 parameters are modeled
    - # of MCs, position of each MCs, and the TSV data bus width of each MC
  - We assume a set of predefined eligible configurations of MC # and positions

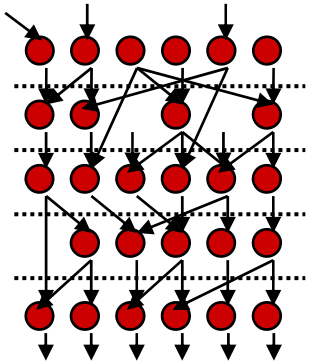


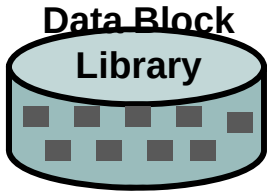
Due to the area constraint, an MC can only have the maximum of B_{max} bits for the TSV data bus

Problem Formulation

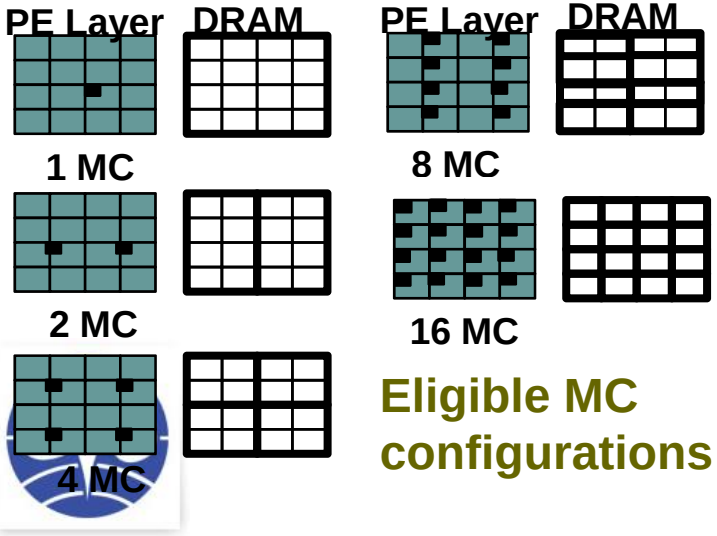


Given:

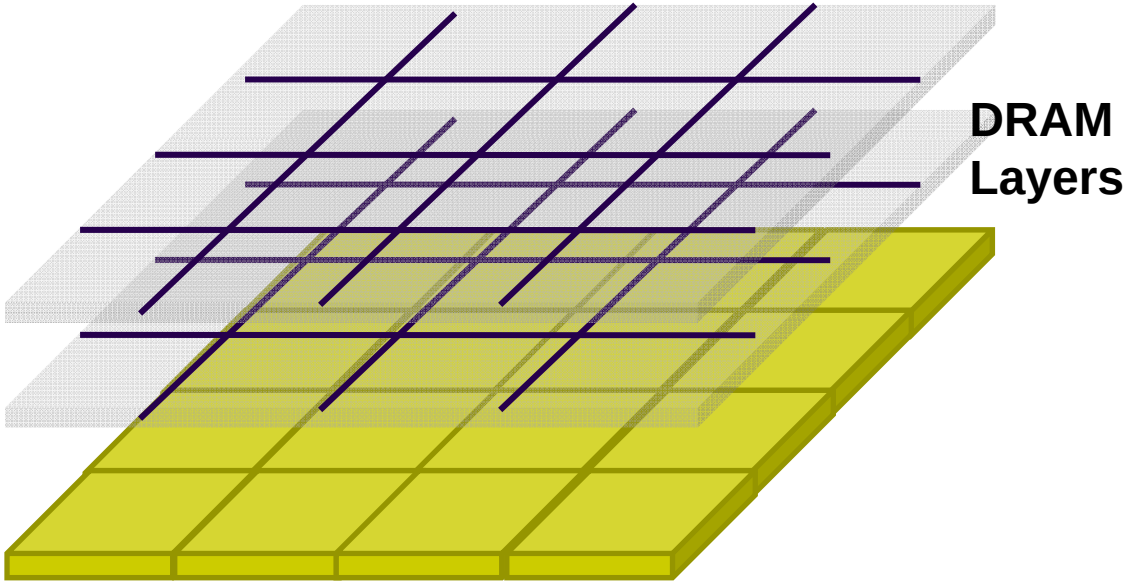




Data Flow Graph



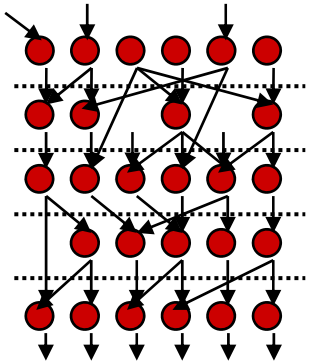
Eligible MC configurations



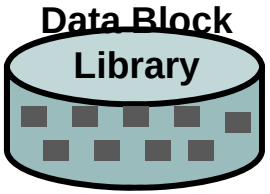
PE Layer

Problem Formulation

Given:



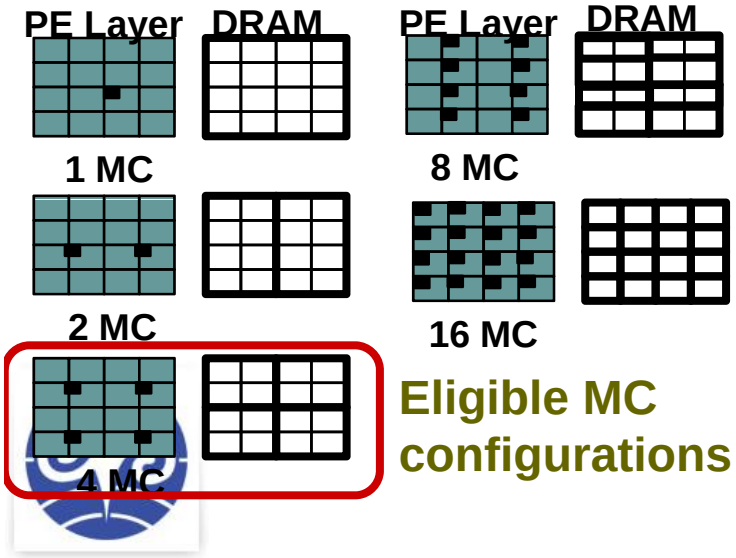
Data Flow Graph



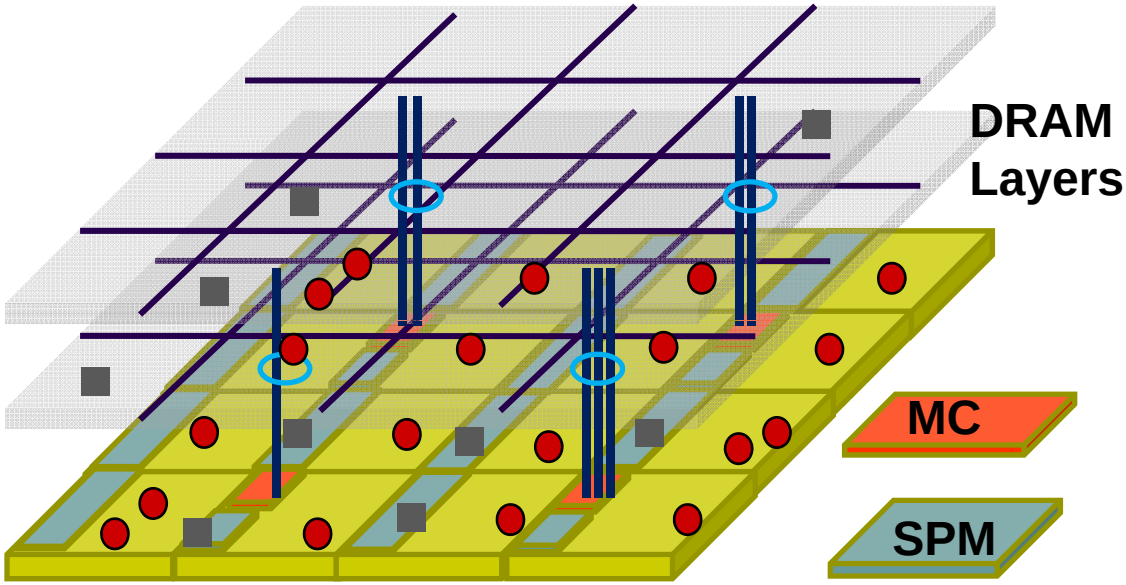
Four Synthesis Targets:

- MC Allocation
- TSV Data Bus Allocation
- Task Mapping
- Data Assignment

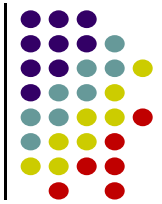
Goal: Meeting the performance constraint & minimizing the total # of TSVs



Eligible MC configurations



Proposed Synthesis Framework



- A greedy-based heuristic
- A two-stage synthesis process
  - MC Number Selection
    - Selecting the eligible MC configurations that provide sufficient # of MCs for the task set
  - TSV Reduction
    - Minimizing the # of TSVs of allocated in the system
- For a configuration of the distributed memory interface, the SW configuration is decided accordingly.

For the details of the proposed algorithm, please refer to the paper.

Experimental Setup

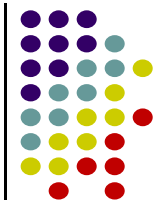
- **Benchmarks:**
  - **Synthetic: generated by TGFF**
    - **Same task graph with different average data access per data transfer**
      - **16K-bit, 32K-bit, 64K-bit and 128K-bit**
  - **Real applications from EEMBC and MPEG2**

- **Performance constraint:**
  - **Baseline architecture: All PEs has a local MC**
  - **Performance constraint is set according the percentage of degradation compared to the baseline**
    - **$T_{baseline}$ * (1 + Degradation)**
      - **$T_{baseline}$: task set execution time of the baseline**
      - **Degradation: percentage of allowable performance degradation**

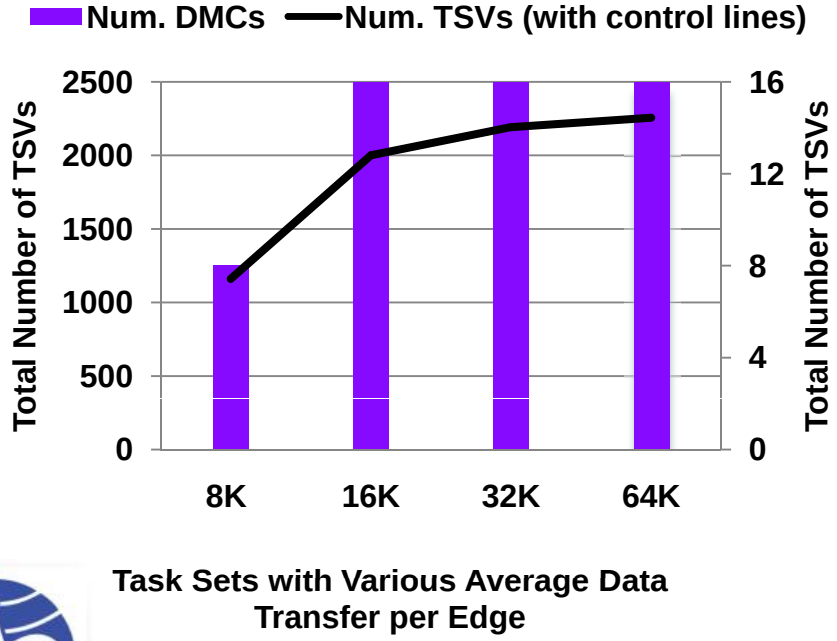
Architectural Setup

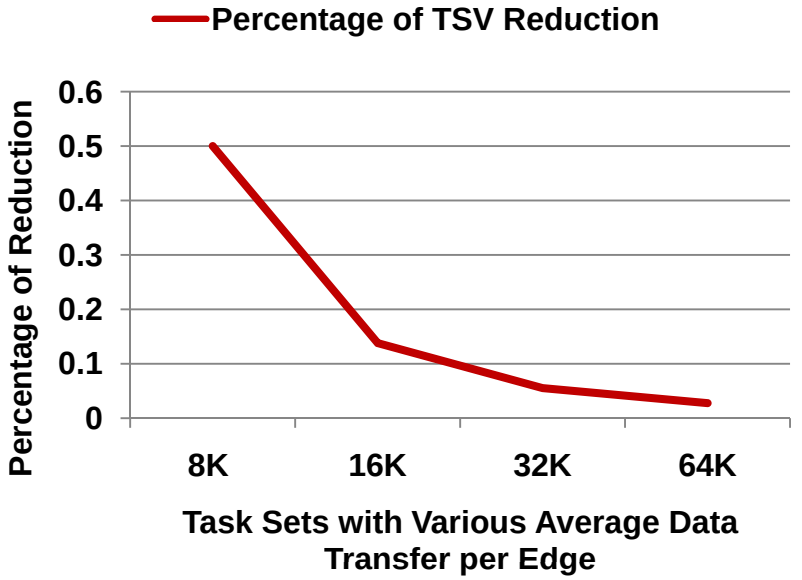
- Logic layer: 4 x 4 NoC (16 tiles)
- 5 eligible MC configurations
  - 1 MC, 2 MCs, 4 MCs, 8 MCs, 16 MCs
- Each MC can have 128-bit width data bus
  - According to [DATE'10], 64-bit data bus + control line → 4~5% of MC resource
  - Each MC can be traded for 4Kb SPM size
- On-chip DRAM size: 16Mb
- SPM size: 16Kb
- Baseline architecture
  - With 16 MCs & each MC is set to 128-bit bus width

Task Set Bandwidth vs. Memory Interface Resource Allocation

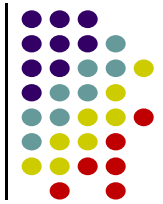


- Higher average # of bits transferred per edge → more MCs and TSVs are needed
- Task sets with the same # of MCs do not necessarily need the same total TSV data bus width.
- Achieving up to 50% TSV reduction



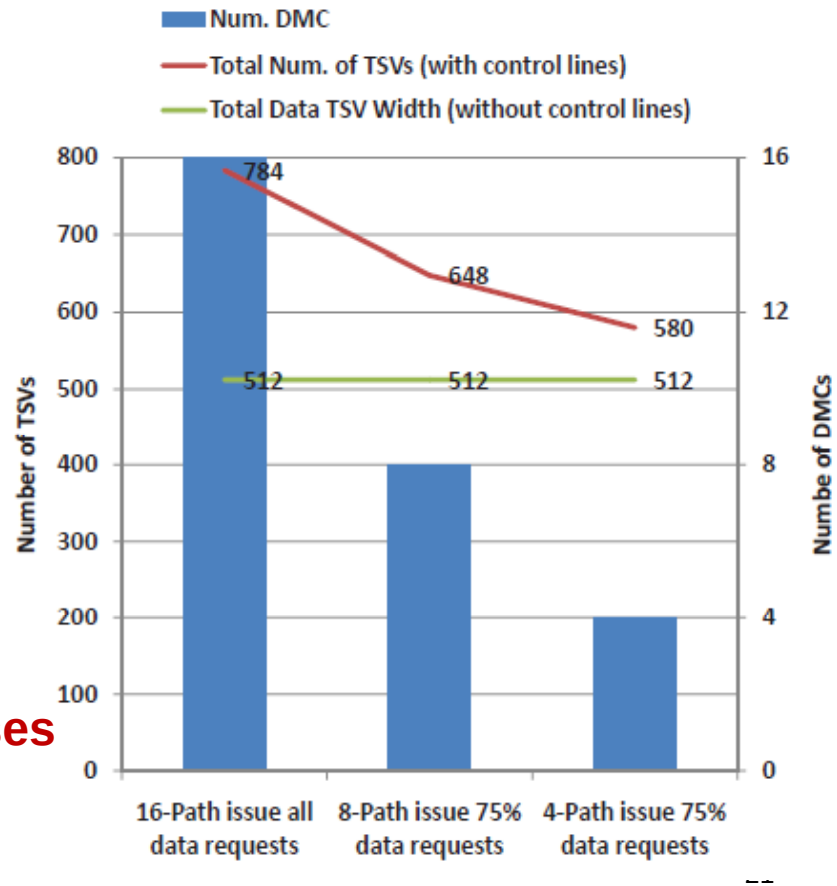


Evaluating the Correctness of the Proposed Algorithm

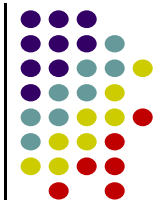


- **We used 3 task sets that have *the same total bandwidth* but *different memory access patterns* for evaluation**
  - **Task set 1:**
    - **All execution paths issue the same amount of data**
  - **Task set 2:**
    - **Half of the execution paths issue 75% of data accesses**
  - **Task set 3:**
    - **Quarter of the execution paths issue 75% of data accesse**

Task sets with less num. of execution paths issue the majority of data accesses → less num. of MCs are needed

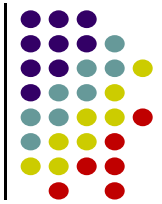


Results on Real Applications



- **Consumer+Telecom**
  - **Including applications like JPEG, auto-correlation, FFT, etc....**
  - **42 tasks in the task set**
  - **Synthesis results**
    - **Performance constraint: 1% degradation**
    - **One MC with 128-bit TSV data bus is selected.**
- **Megp2_enc+Mpeg2_dec**
  - **Utilizing 19 tasks to encode a macroblock, and 4 tasks to decode a macroblock**
  - **We assume Mpeg2 encoder and decoder can respectively process 8 macroblocks in parallel in the target platform**
  - **Synthesis results**
    - **Performance constraint: 1% degradation**
    - **2-MC configuration is selected, and each MC has a 128-bit TSV data bus**

Conclusion of this Research



- **A new dimension in the design of memory subsystem is shown in MSPoCs with 3D-stacked DRAMs**

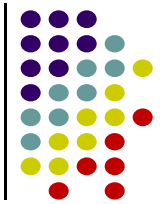
- **The first synthesis framework designed for the distributed memory interface of MPSoCs with 3D-stacked DRAMs**

- **The results show that, the cost of TSVs can be reduced provided that the performance constraint is met by customizing the distributed memory system.**

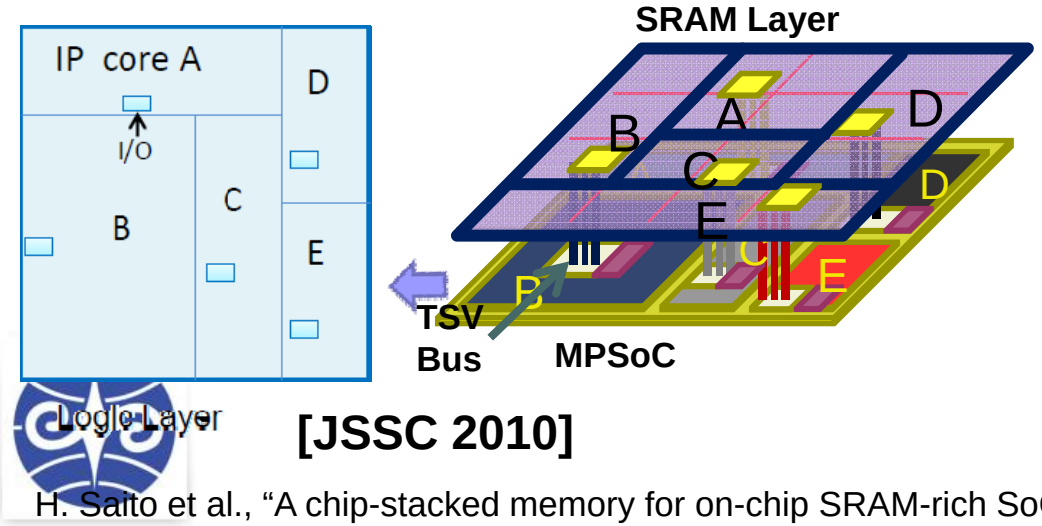
Outline

- Overview of system-level synthesis
- 3D-enabled MPSoC and Memory Integration
- Synthesis of Distribute Memory Interface for MPSoCs with 3D-stacked DRAMs
- Synthesis of Memory & Data Allocation for MPSoCs with Reconfigurable 3D-stacked SRAMs
- Conclusion

MPSoC with Reconfiguable 3D-Stacked SRAMs



- A two layer architecture that is composed of
  - Multi-Processor System-on-Chip (MPSoC) layer
    - Each IP core has at least one I/O port with Through Silicon Vias (TSVs) to access the stacked SRAMs
  - Reconfigurable 3D-stacked SRAM layer

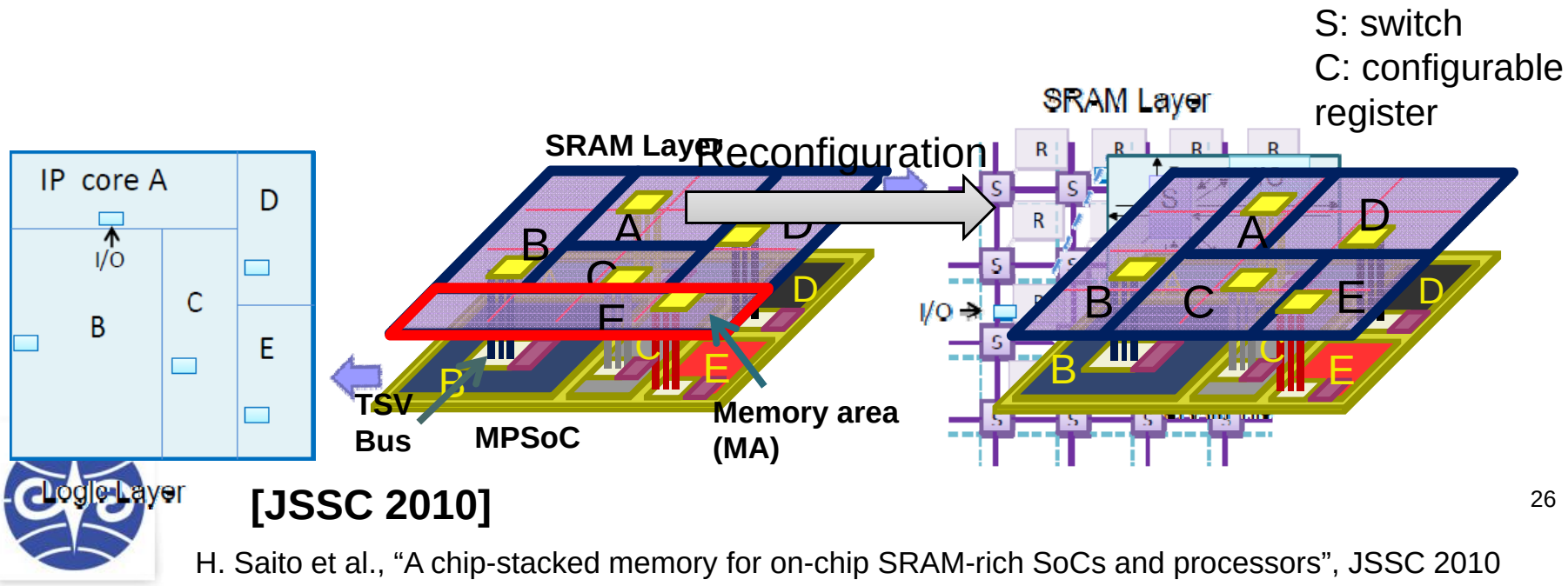


[JSSC 2010]

H. Saito et al., “A chip-stacked memory for on-chip SRAM-rich SoCs and processors”, JSSC 2010

MPSoC with Reconfiguable 3D-Stacked SRAMs

- **The reconfigurable 3D-stacked SRAM layer**
  - **SRAM tiles are connected by 2D mesh on-chip network**
  - **Contiguous SRAM tiles form a Memory Area (MA)**
    - **Each MA is only accessible by its designated IP core**
  - **Configuring MA is achieved by modifying the configurable registers, which takes one cycle time**

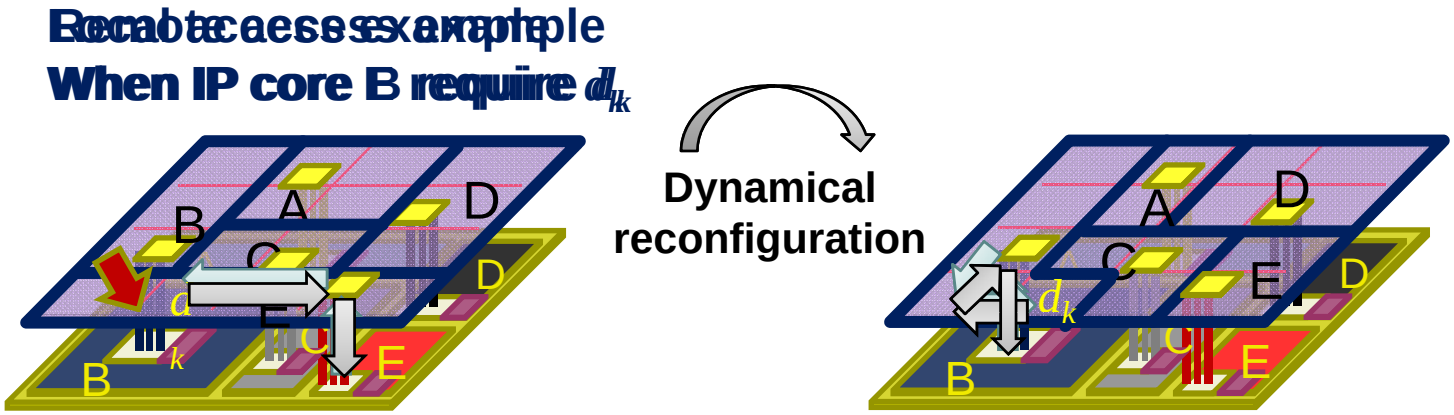


[JSSC 2010]

H. Saito et al., "A chip-stacked memory for on-chip SRAM-rich SoCs and processors", JSSC 2010

Data Accesses in the Target Architecture

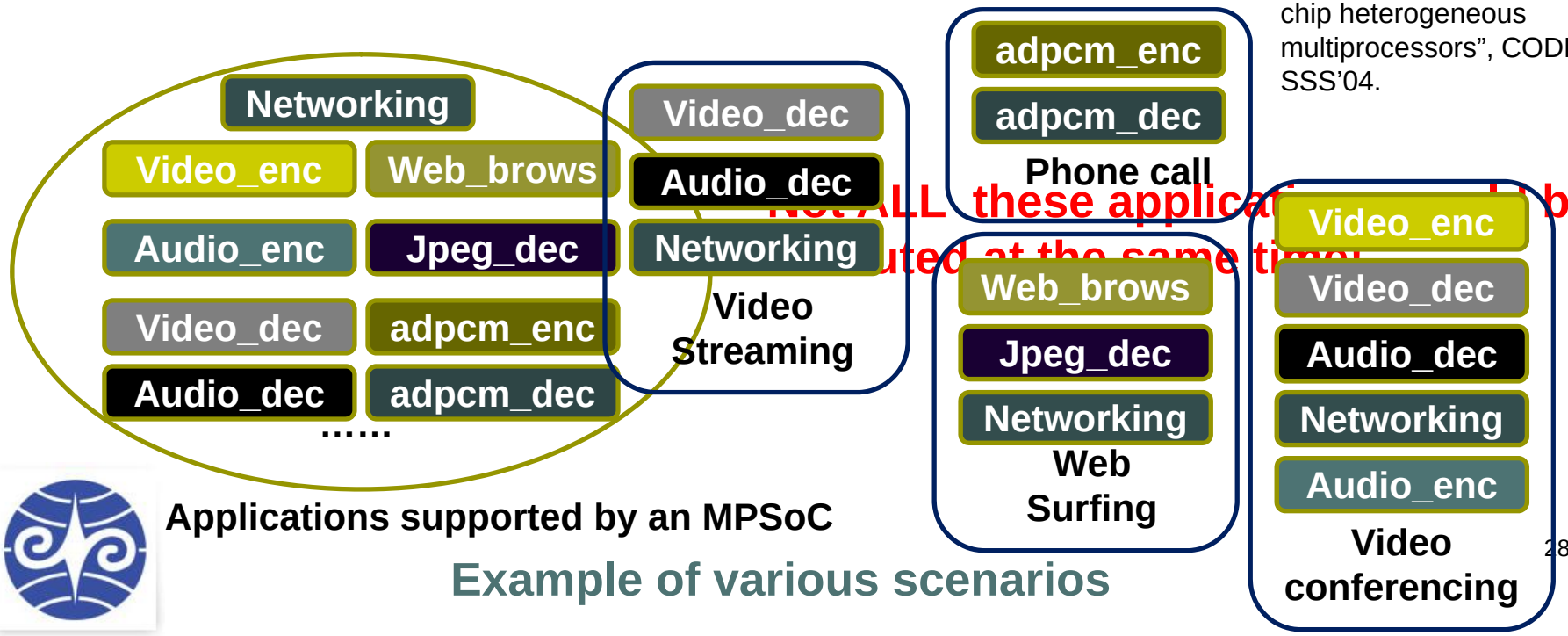
- **Local MA accesses**
  - **Data are in the SRAM tiles of the requesting IP's MA partition**
- **Remote MA accesses**
  - **Data are in the SRAM tiles belonged to the MAs of other IPs**
  - **Reconfigurations are needed to include the target SRAM tile to the requesting IP's local MA**



Timing of Reconfigurations

- Change of runtime behavior
  - MPSoC executions can be divided into different *scenarios* [CODES+ISSS'04]
    - Only a specific set of applications are executed concurrently at a time
  - Change of scenario → change of memory area allocation
- An IP invokes remote MA accesses

J. M. Paul et al., "Benchmark-based design strategies for single chip heterogeneous multiprocessors", CODES+ISSS'04.



Applications supported by an MPSoC

Example of various scenarios

To improve memory system performance, we have to consider…

- Memory area allocation of each scenario
  - Sufficient memory space for each active IP core
  - Considering the Non-Uniform Memory Access (NUMA) nature of the system to perform data placement

- Reconfiguration overheads
  - In addition to the 1-cycle overhead of writing the configurable registers, data accesses to the related SRAMs must be stalled until the reconfiguration is done
  - Reducing reconfigurations due to remote MA accesses is a must for improving system performance

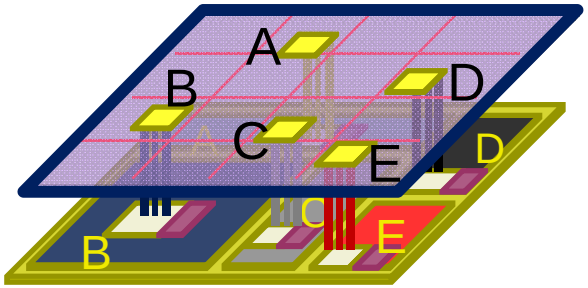
In this research, we propose…

- A scenario-aware memory resource and data allocation synthesis algorithm
  - Synthesis targets
    - Memory area allocation & data placement for each scenario
  - Optimization goal
    - System performance
- Considering the behavior both within a scenario and across all scenarios to minimize
  - The average data access latency
  - The number of reconfigurations
- This part of research is published in DATE'14

M.-L. Tsai, Y.-J. Chen, Y.-T. Chen, and R.-H. Chang, "Scenario-aware data placement and memory area allocation for MPSoCs with reconfigurable 3D-satcked SRAMs", DATE'14.

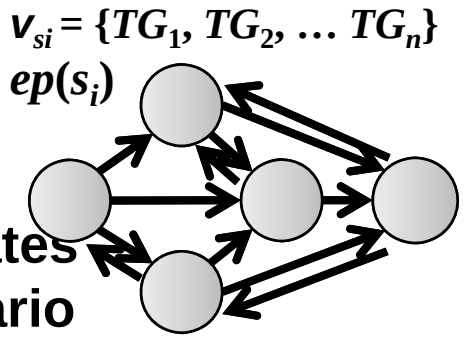
System Model – Hardware Model

- **Hardware models represent**
  - **Set of IP Cores $P = \{p_0, p_1, \ldots, p_l\}$**
  - **Set of SRAM tiles $T = \{t_0, t_1, \ldots, t_{mxn}\}$**
  - **I/O port position of each IP core**
  - ***size*(SRAM): capacity of each SRAM tile**

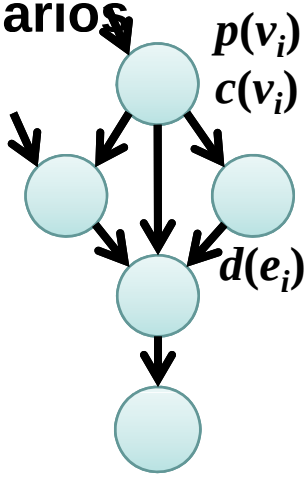


System Model – Software Model

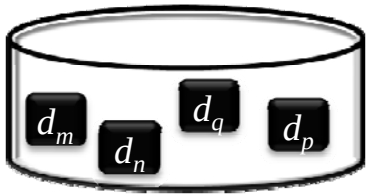
- **Software models represent**
  - **Scenario graphs**
    - **A vertex represents a scenario, which indicates the set of applications executed in the scenario**
      - $ep(s_i)$**: execution probability of scenario** s_i
    - **An edge represents a possible transition of scenarios**
  - **Task graphs**
    - **A vertex represents a task or kernel function of the application**
    - **An edge represents the data transfer among tasks**
  - **Data block library**
    - **The collection of data blocks utilized by the system**



Scenario graph



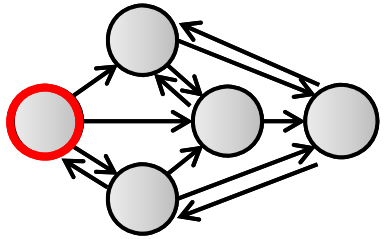
Task graph



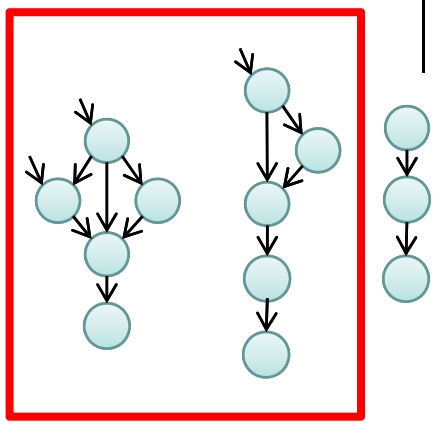
Data block library

Problem Formulation

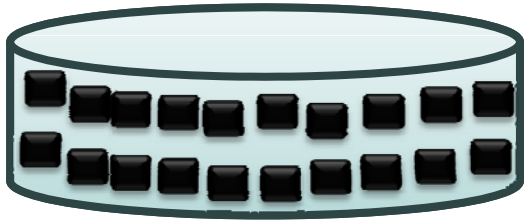
- Given
  - Hardware models
  - Software models
- For each scenario, the algorithm synthesizes
  - Data placement
    - represented by the function $D \rightarrow T$
  - memory allocation accordingly
    - represented by the function $T \rightarrow P$
- Goal
  - Minimizing the execution time



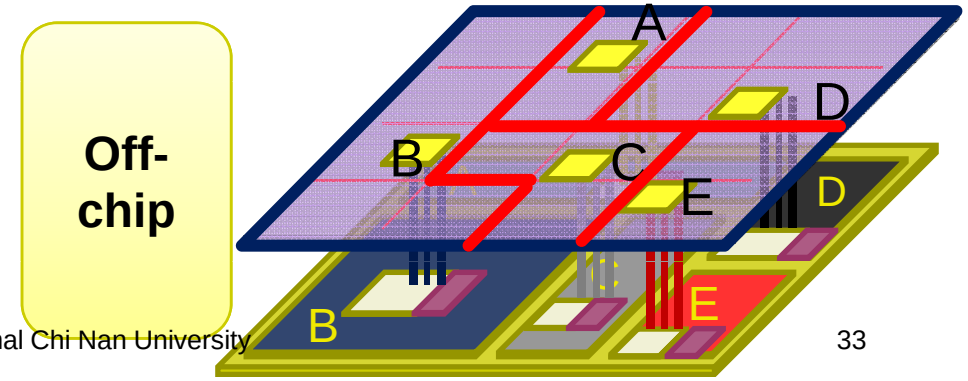
Scenario graph



Task graphs



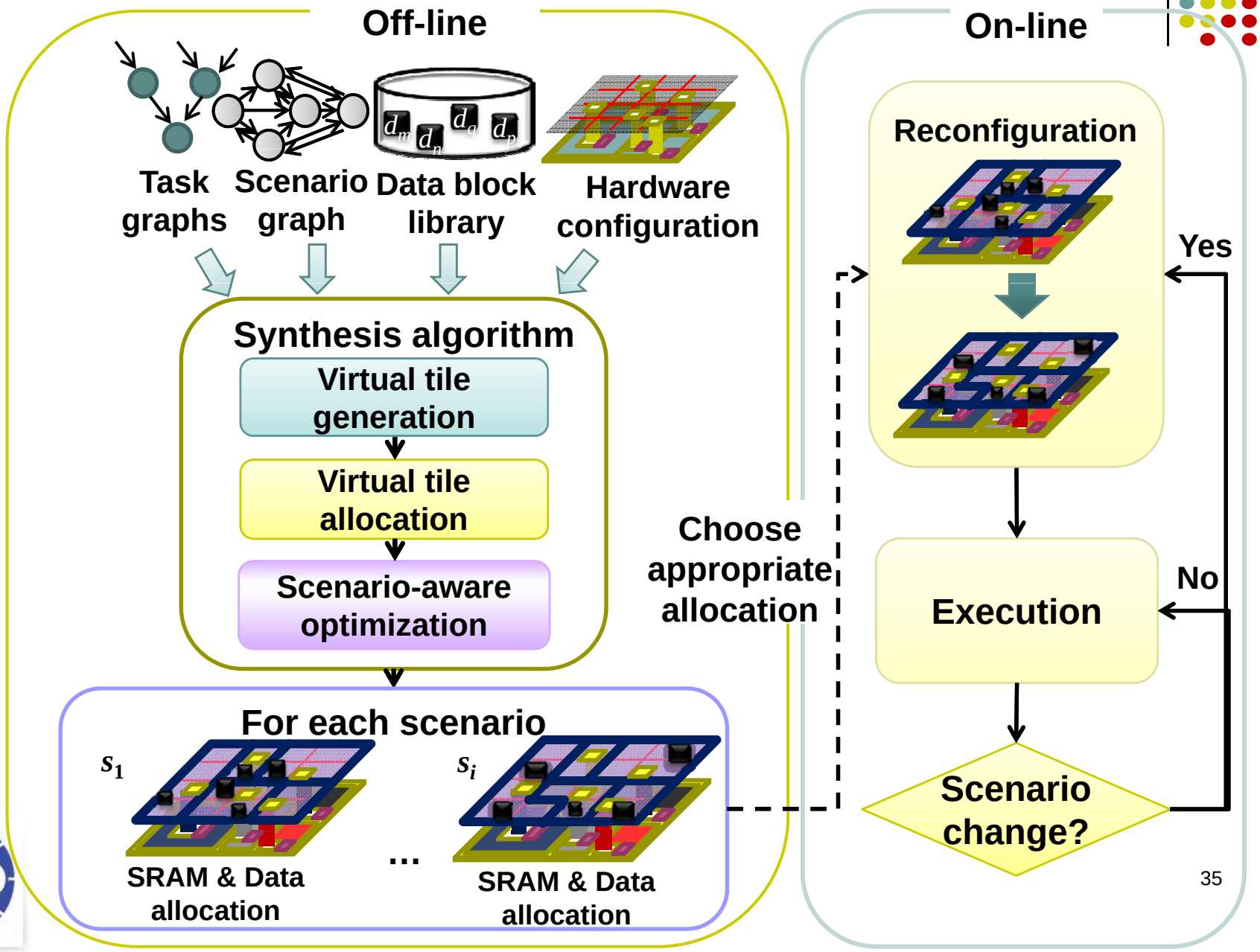
Data block library



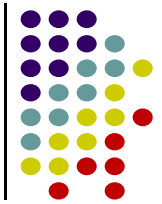
Algorithm Overview

- A heuristic approach
- Optimizing memory system performance by considering
  - Operations within a scenario to place data in on-chip SRAMs so that
    - # of reconfigurations due to remote MA accesses can be minimized
    - Average access latency can be minimized
  - Data accesses among all scenarios to place data in on-chip SRAMs so that
    - # of retrieving data from the off-chip memory can be minimized

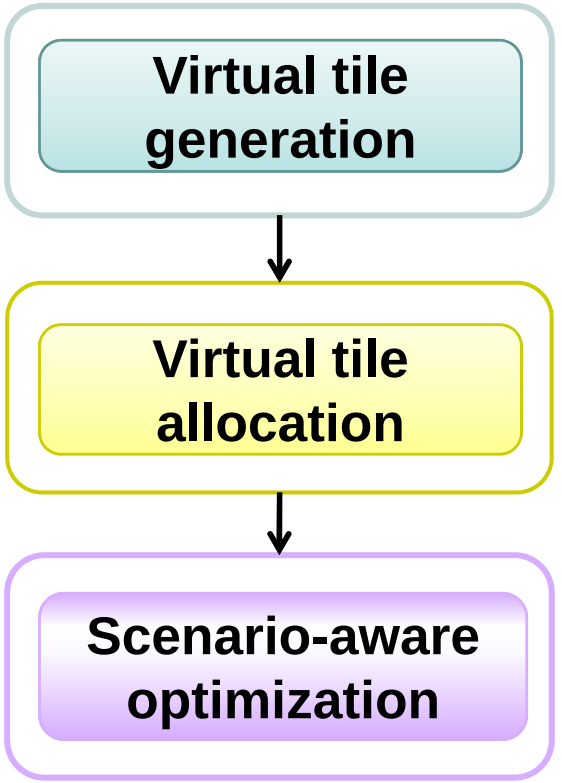
Execution Flow



Flow of the Algorithm



- Performed for each scenario
  - Virtual tile generation
    - Deciding the data blocks to be allocated in the same SRAM tile
    - Goal: reducing contention of an SRAM tile
  - Virtual tile allocation
    - Deciding the physical tile position of each virtual tile
    - Goal: reducing average data access latency and MA interferences
  - Scenario-aware optimizations
    - Adapt the data allocation to consider the data accesses behavior among all scenarios
    - Goal: reducing the number of off-chip memory accesses



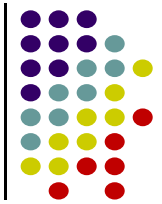
For details of the synthesis algorithm, please refer to the paper.

Experimental Methodology

- **Simulator**
  - **Based on Hornet [ISPASS '11]**
    - **A highly configurable, cycle accurate network-on-chip simulator**
- **Target platform**
  - **Logic layer has**
    - **Dual core Cortex A9, Audio/Video processor, and an image processor**
  - **4 x 4 SRAM tiles, each of 64KB capacity**
  - **Each I/O port has 256-bit TSV bus**

M. Liz et al., "Scalable, accurate multicore simulation in the 1000-core era", ISPASS 2011.

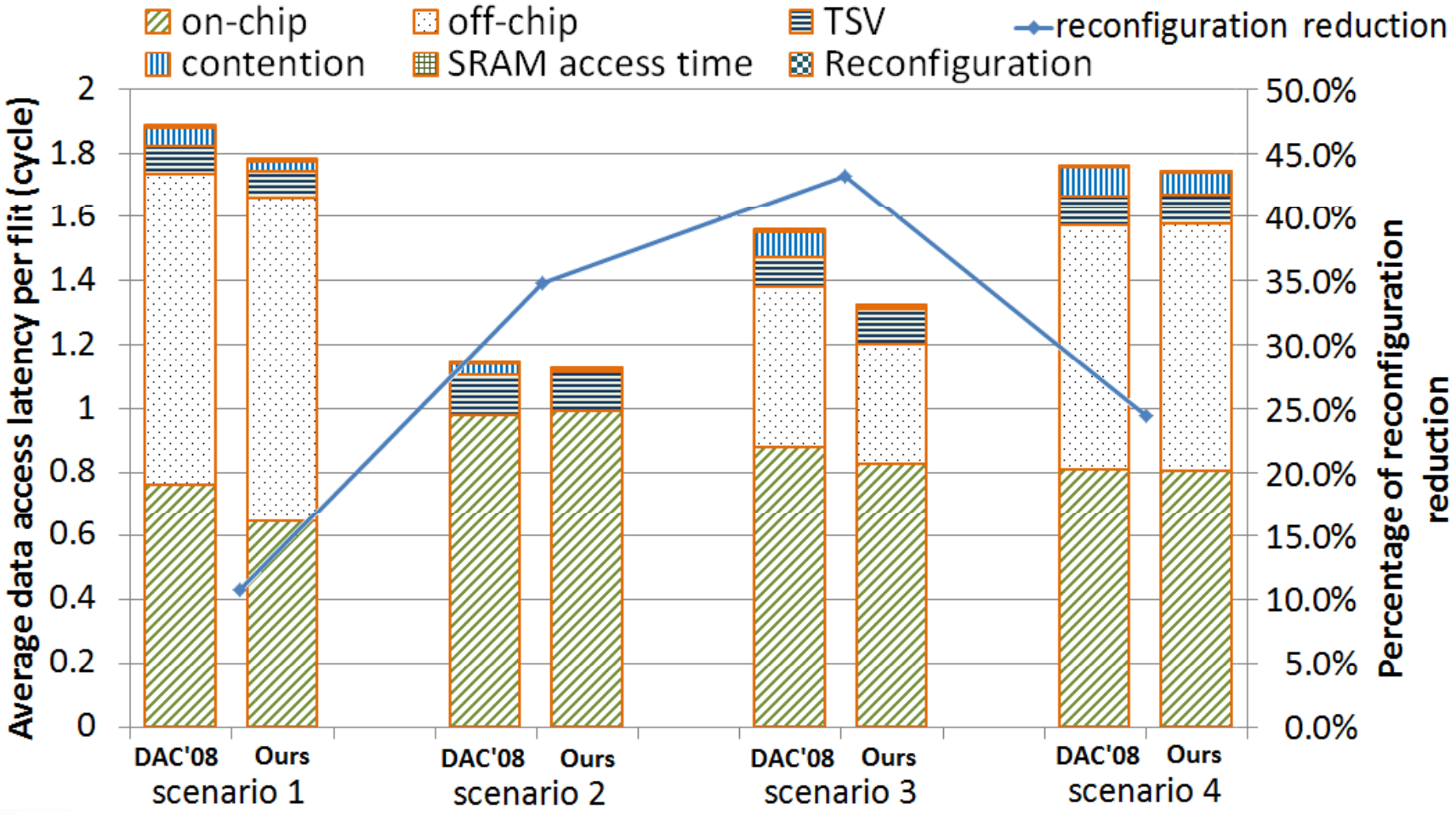
Experimental Methodology (cont.)



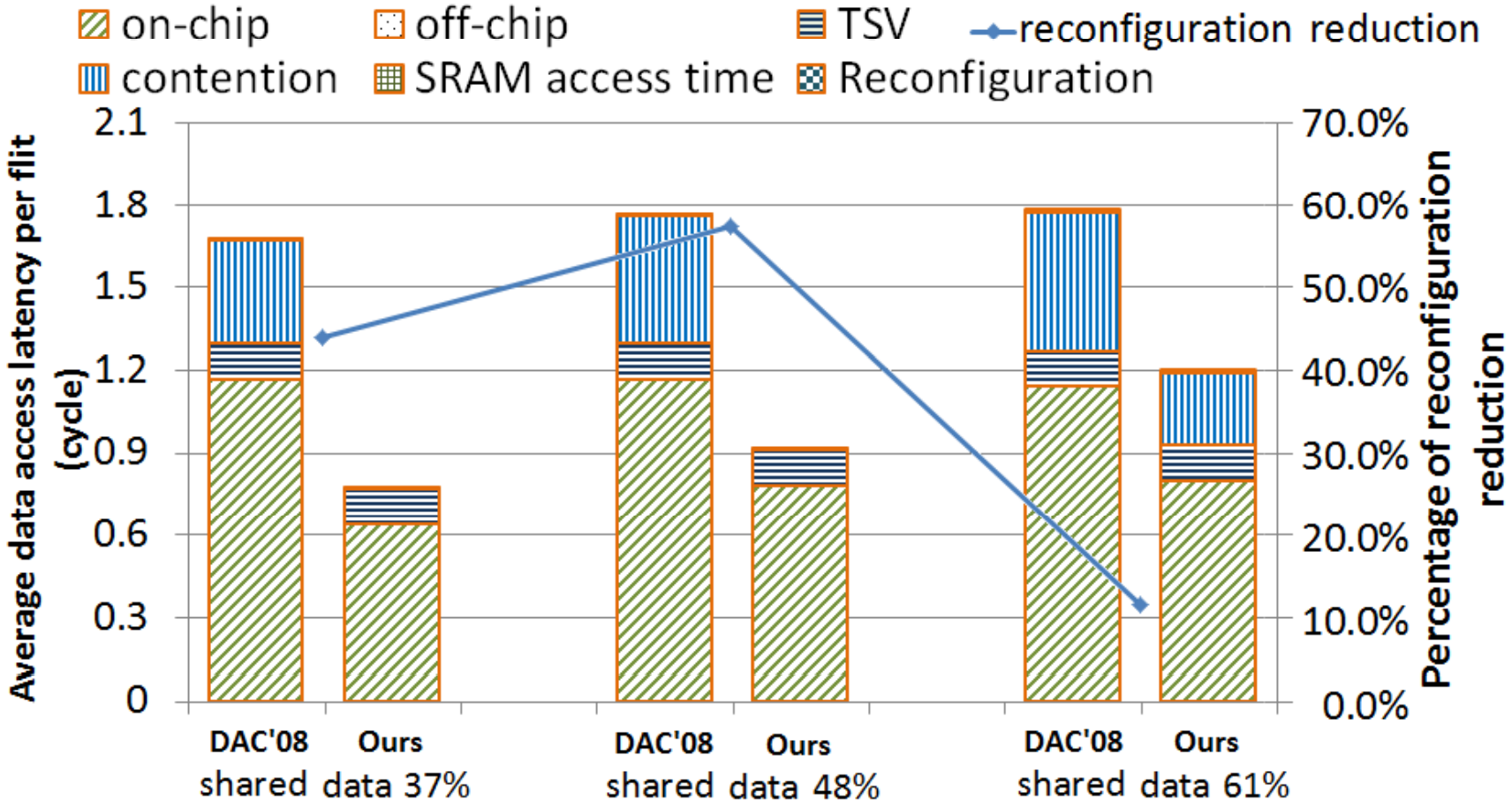
- **Workloads**
  - **Synthetic workloads**
    - **Task graphs are generated by TGFF**
    - **Scenarios and their execution probabilities are randomly generated**
  - **Real applications**
    - **Selected task graphs from E3S**
      - **Task graphs of EEMBC**
    - **Scenarios and their execution probabilities are set according to [CASES'12]**
- **All results are compared to the data placement algorithm proposed for distributed memory architecture [DAC'08]**

L. Schor et al., "Scenario-based design flow for mapping streaming applications onto on-chip many-core systems", CASES 2012.
G. Chen et al., "Application mapping for chip multiprocessors", DAC 2008.

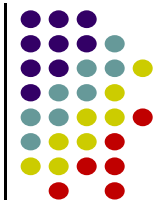
Average Access Latency – Synthetic Workloads



Average Access Latency for Scenarios w/ Various amount of Shared Data



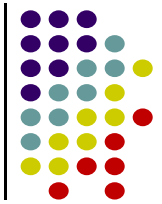
Reduction in Access Latency



- For synthetic workloads running 30 different scenarios
  - With scenario-aware optimizations
    → further reduce access latency by 9.72%

- For real world applications
  - Compared to the method proposed in [DAC'08], our method reduces up to 15.40% of access latency for each scenario
  - When running 30 scenarios, reduces up to 11.72% of access latencies.

Conclusion of this Research



- **We proposed a memory and data allocation algorithm for MPSoCs with 3D-stacked reconfigurable SRAMs**

- **The proposed algorithm reduces memory access latency by**
  - **Reducing the # of reconfigurations by considering data access patterns within a scenario**
  - **Reducing the # of off-chip memories by considering data accessed among all scenarios**

Conclusion

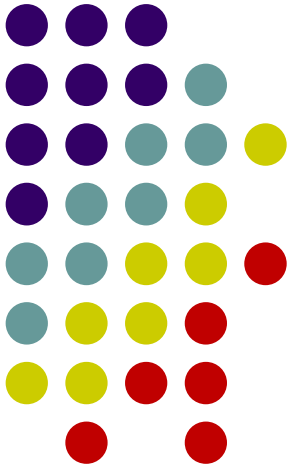
- Two research topics are presented
  - Memory interface synthesis for Multiprocessor System-on-Chips (MPSoCs) with 3D-stacked DRAMs
  - Memory resource allocation and data placement for MPSoCs with reconfigurable 3D-stacked SRAMs

- Research overview
  - Problem formulation is the key of the research
  - Synthesis problems are mostly NP-complete
    → heuristics are utilized
  - If we can map the existing problems to known geometry or graph algorithms
    → utilizing existing algorithm to find exact solution

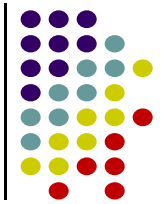
Thank you for Listening Questions?

Yi-Jung Chen

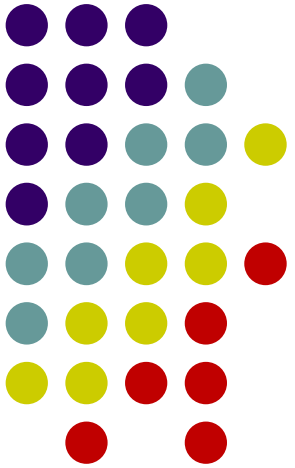
yjchen@ncnu.edu.tw



國立暨南國際大學
National Chi Nan University



Backup Slides

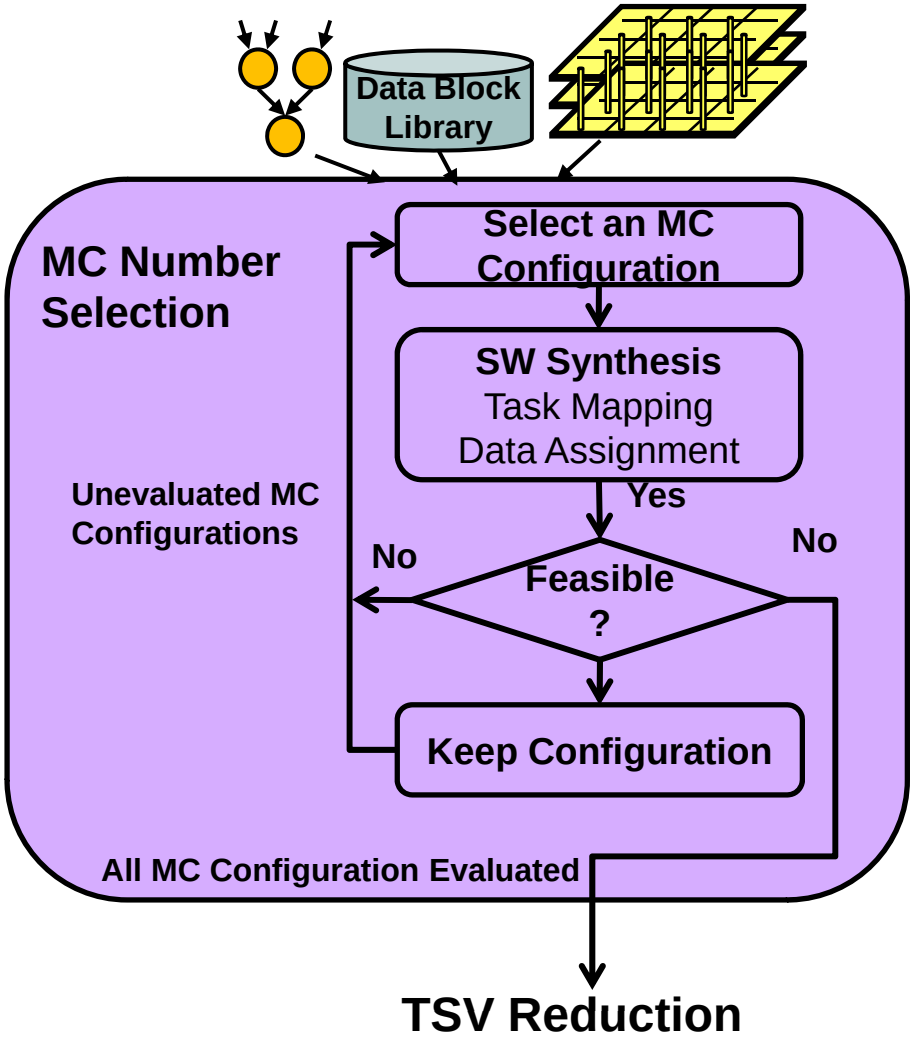


國立暨南國際大學
National Chi Nan University

Hardware Synthesis Process – MC Number Selection

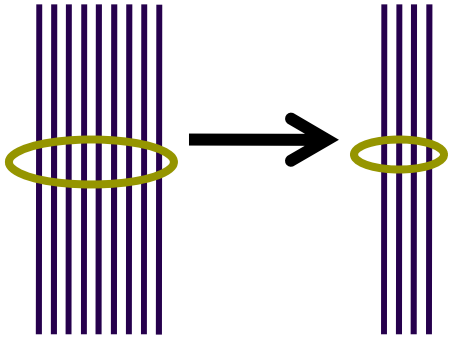
- **Assumption of this process:**
  - **TSV data bus width of each MC is set to B_{max}**
    → evaluating the performance degradation due to insufficient # of MCs only

- **Only the feasible configurations are evaluate in the next stage**

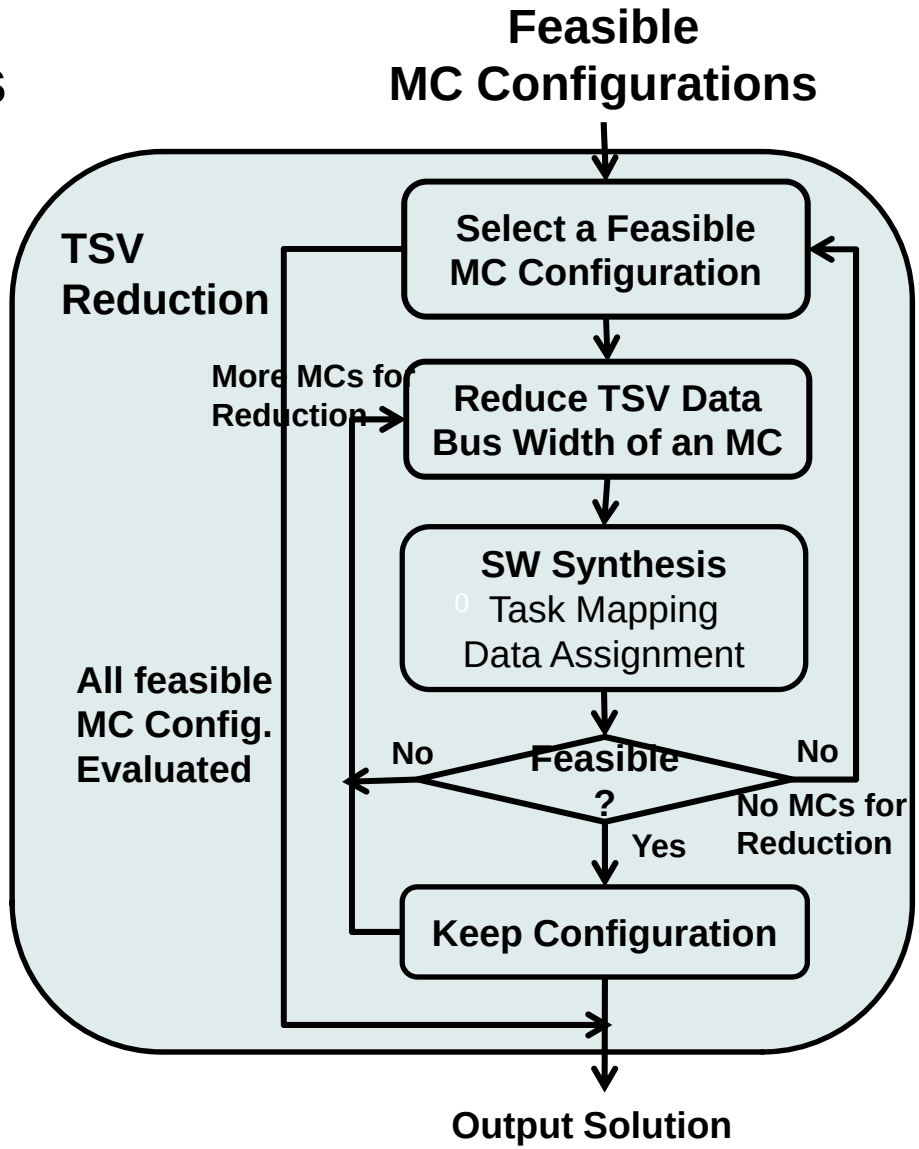


Hardware Synthesis Process – TSV Reduction

- To reduce the TSV-data-bus width of an MC, set to half of the original width



Reduce to half

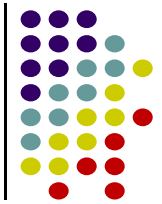


The Software Synthesis Process

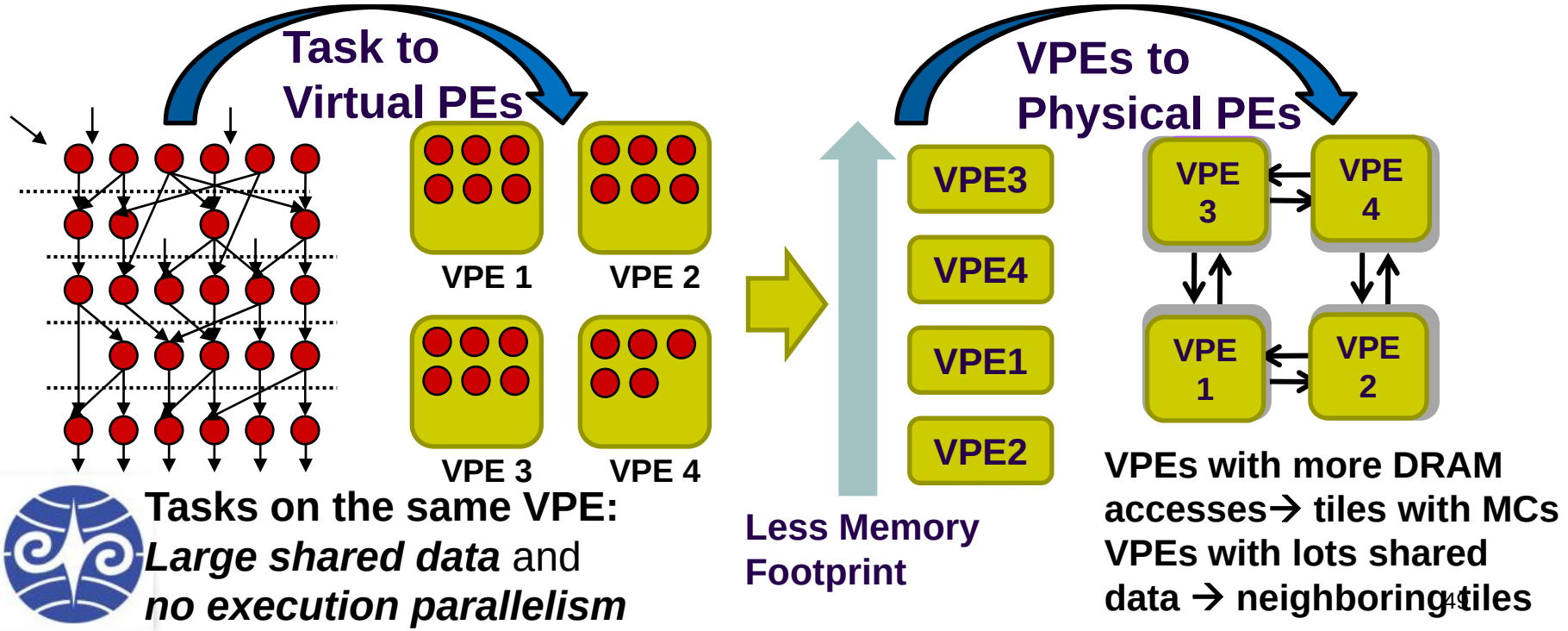
- **Software synthesis process includes**
  - **Task mapping**
    - **Mapping a task to a specific PE for execution**
  - **Data assignment**
    - **Assigning a data block to a specific memory module**
      - **Two types of memory modules can be selected in our target platform: DRAMs and SPMs modules**

- **Goal: maximize performance of the selected hardware configuration**

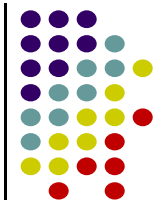
Task Mapping



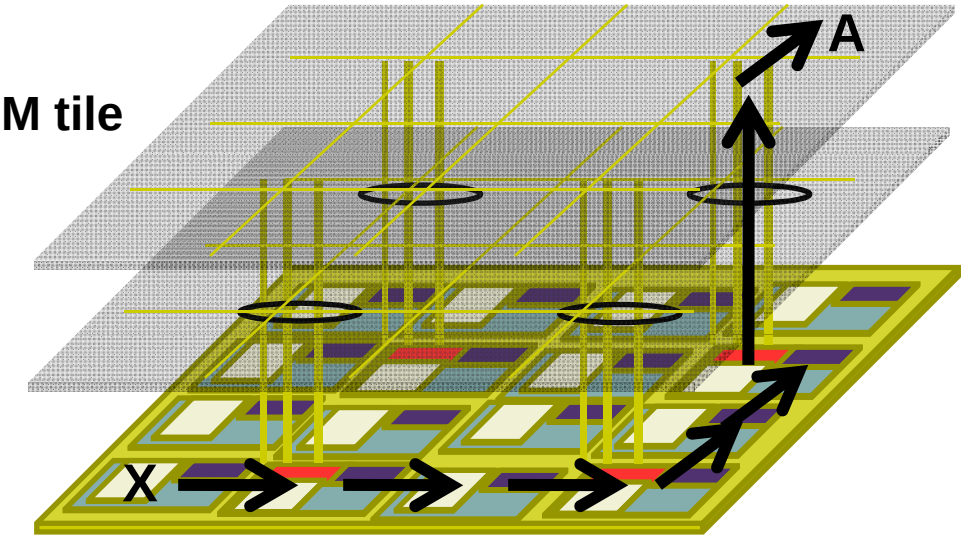
- **Goal:**
  - **Exploiting the parallelism in the task set as much as possible**
  - **Improving the locality of data accesses**
  - **Considering the requirements of DRAM accesses**
- **The Task Mapping process:**
  - **Similar to the one proposed in [DAC 2008]**



Data Assignment

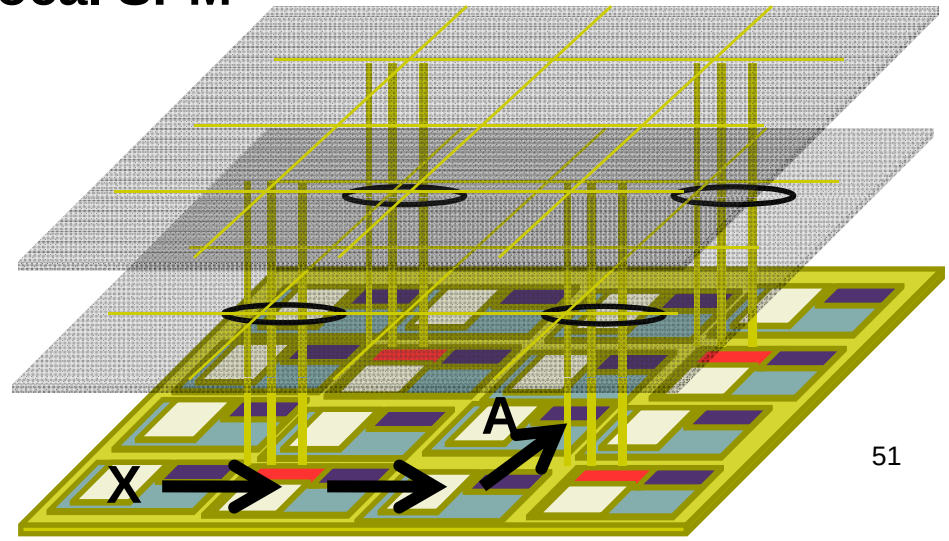


- **Goal:**
  - **Assigning data blocks to memory modules so that the overall time spent in data accesses is minimized as much as possible**
- **Challenge: Estimation of access time, which can be divided into tow cases**
  - **Accessing data blocks in stacked DRAM modules**
    - **PE → MC**
    - **MC → DRAM layer**
    - **DRAM layer → target DRAM tile**
  - **Accessing data blocks in local SRAMs**



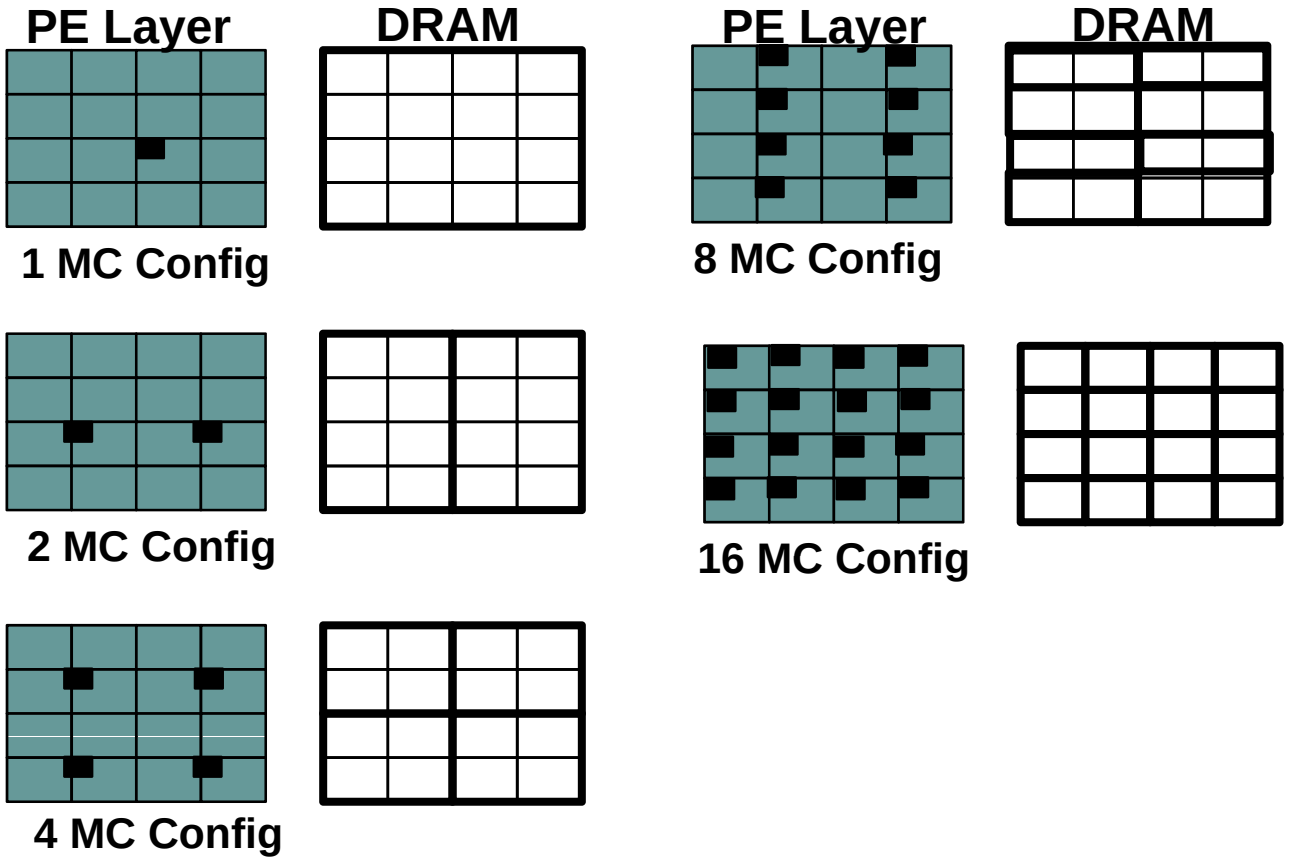
Data Assignment

- Goal: Find a data assignment that minimizes the average data access latencies
- Challenge: Estimation of access time, which can be divided into tow cases
  - Accessing data blocks in stacked DRAM modules
    - PE → MC
    - MC → DRAM layer
    - DRAM layer → target DRAM tile
  - Accessing data blocks in local SPM
    - Source tile → target tile



Architectural Setup

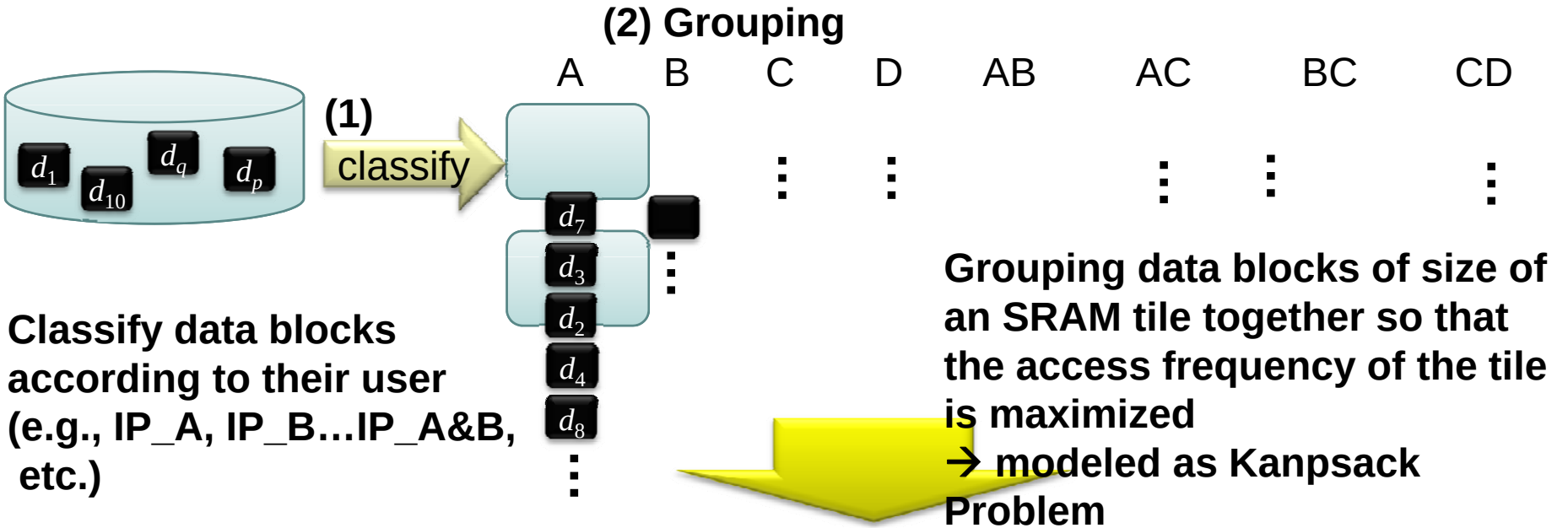
- Logic layer: 4 x 4 NoC (16 tiles)
- 5 eligible MC configurations
  - 1 MC, 2 MCs, 4 MCs, 8 MCs, 16 MCs

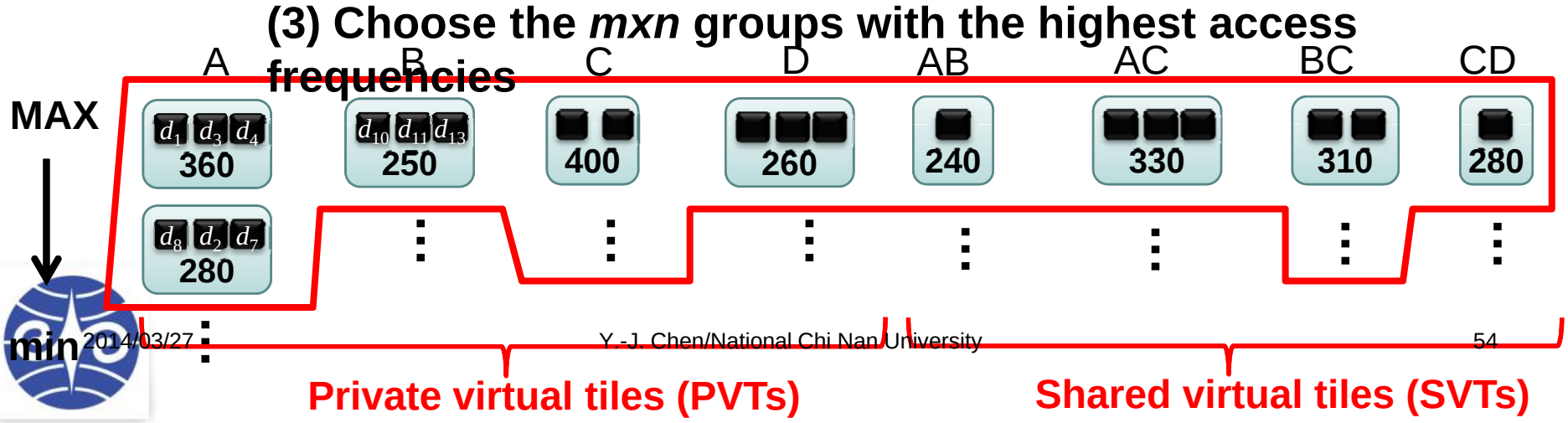


Virtual Tile Generation

- Virtual tile
  - The set of data blocks to be placed in the same SRAM tile
  - No physical SRAM tile indicated yet
- Goal
  - Maximizing the # of on-chip memory accesses
    - Selecting the most frequently accessed data blocks
  - Minimizing the contention of an SRAM tile
    - More than one IP accesses an SRAM tile
      → invoking of MA reconfigurations
    - Data blocks with the same users should be allocated in the same SRAMs

Process of Virtual Tile Generation





Virtual Tile Allocation

- Allocating virtual tiles to physical tile positions
- Goal
  - Minimizing the average data access latency
    - Placing the most frequently virtual tiles to the position that is close to the I/O port
  - Minimizing the # of reconfigurations
    - Placing the virtual tiles to avoid interferences among MAs

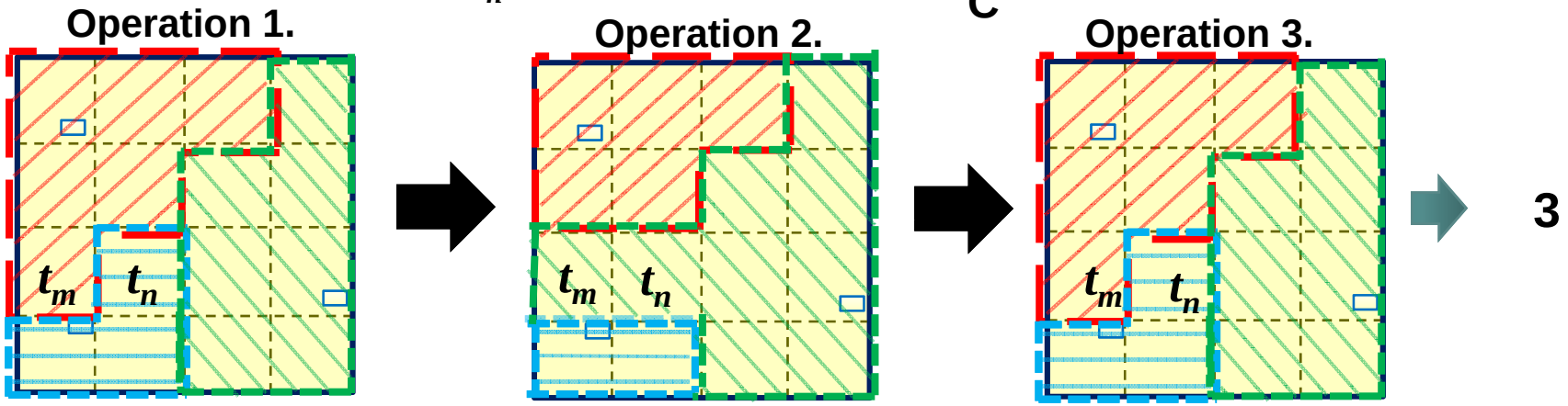
Assume the following operations are executed

1. Core A accesses tile t_m
2. Core B accesses tile t_m
3. Core C accesses tile t_n

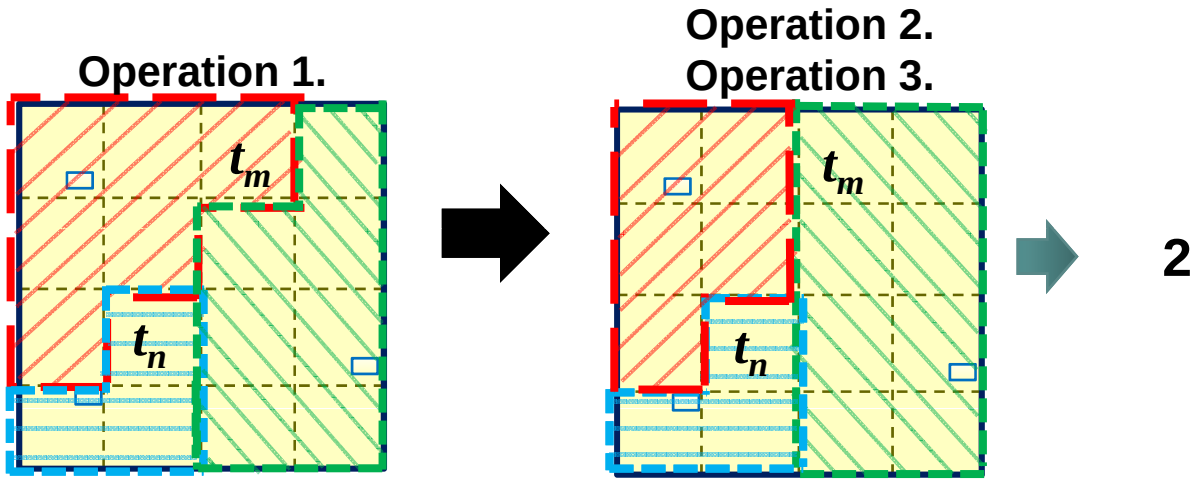
Memory area used by core A

Memory area used by core B

Memory area used by core C



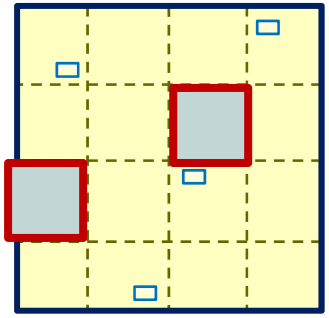
(a) Tile t_m is allocated without distance consideration

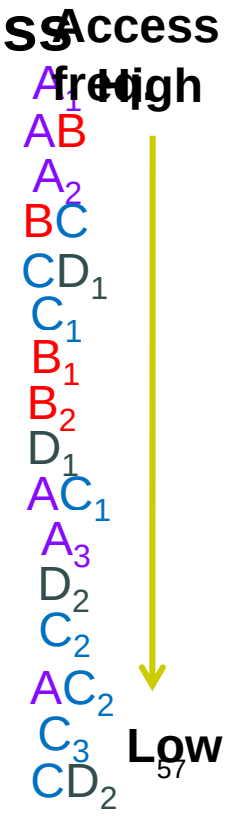


(b) Tile t_m is allocated with distance consideration

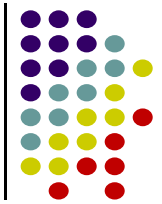
Process of Virtual Tile Allocation

- Sorting the virtual tiles according to their access frequencies
- Private virtual tiles have the highest priority to get the tiles with I/O port
- Starting from the virtual tile with the highest access frequency
  - For private virtual tiles, select the tile position that yields the least access latencies
  - For shared virtual tiles, select the tile position
    - Yields the least access latencies
    - Away from the I/O ports of the IPs that do not access the tile





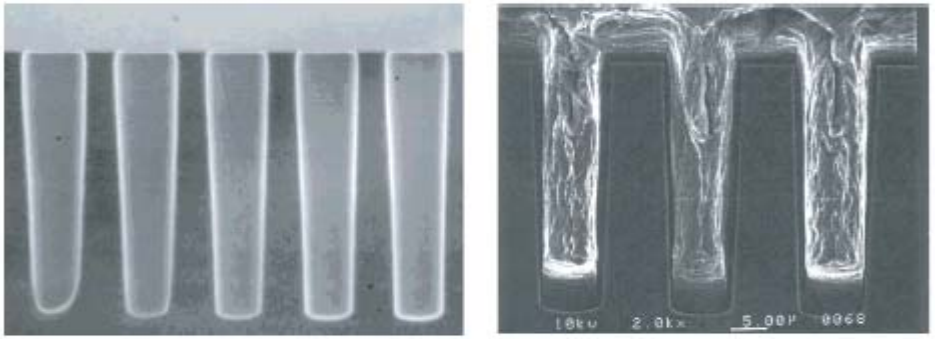
Scenario Optimizations



- Keep the data blocks accessed by several scenarios in the on-chip SRAMs to reduce # of off-chip accesses
- The process
  - Calculate the execution frequency of a data block
    - Across all scenarios
  - Select data blocks with execution frequencies > threshold
    - Threshold is set to the 80% of the maximum execution frequency
  - Remove the data blocks with the lowest access frequency in a scenario for the data blocks utilized in several scenarios

3D IC Process Flow

- **Major steps:**
  - **Wafer thining: grinding, wet etching …**
  - **Via drilling: laser drilling, DRIE …**
  - **Via filling : Electropolating…(tungsten or copper)**
  - **Wafer bonding with thermocompression**
    - **Chip to wafer**
    - **Wafer to wafer**
- **TSVs are easy to fail during the process**



(a) (b)

Wafer preparation process: (a) through-hole fabrication, (b) electropolating [JJAP 01]

Takahashi et al., “Current status of research and development for three-dimensional chip stack technology”, JJAP 2001.