

Estimation and Exploration Automation of System Level Design

Lukai Cai

Center for Embedded Computing Systems
University of California, Irvine

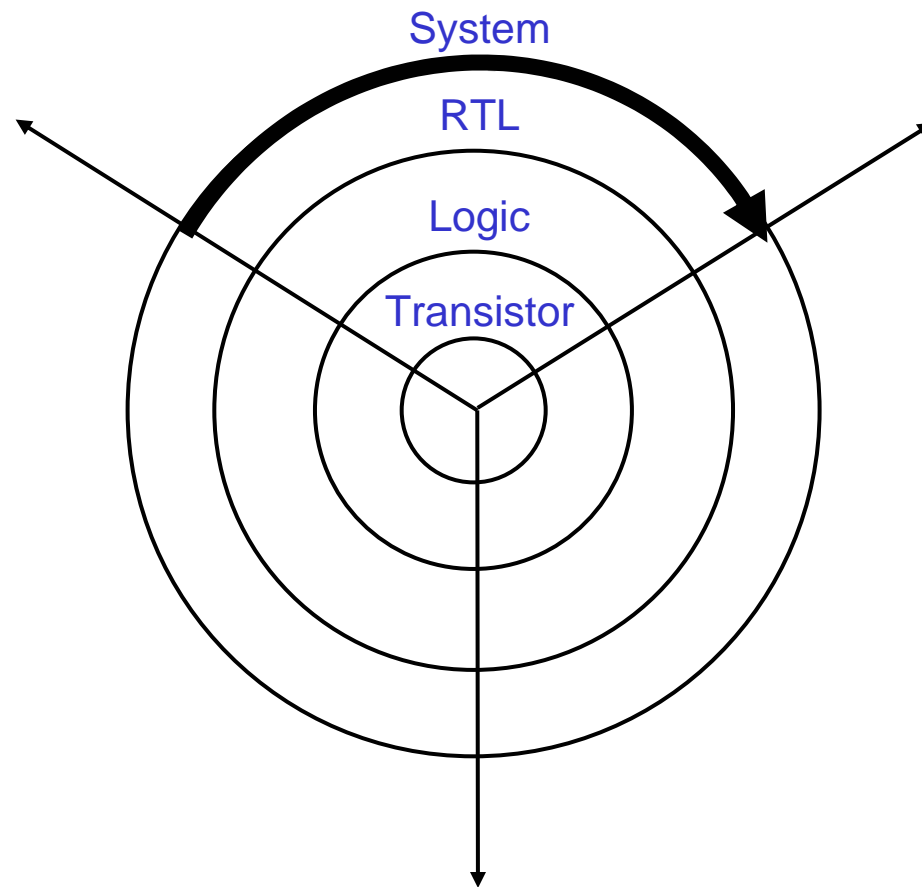
Outline

- Motivation
- Design flow and problem definition
- Re-targetable profiling
- Specification tuning
- Variable mapping
- Conclusions

Y-Chart

Behavior
(Function)

Structure
(Design)



Physical
(Layout)

Challenges & Goals

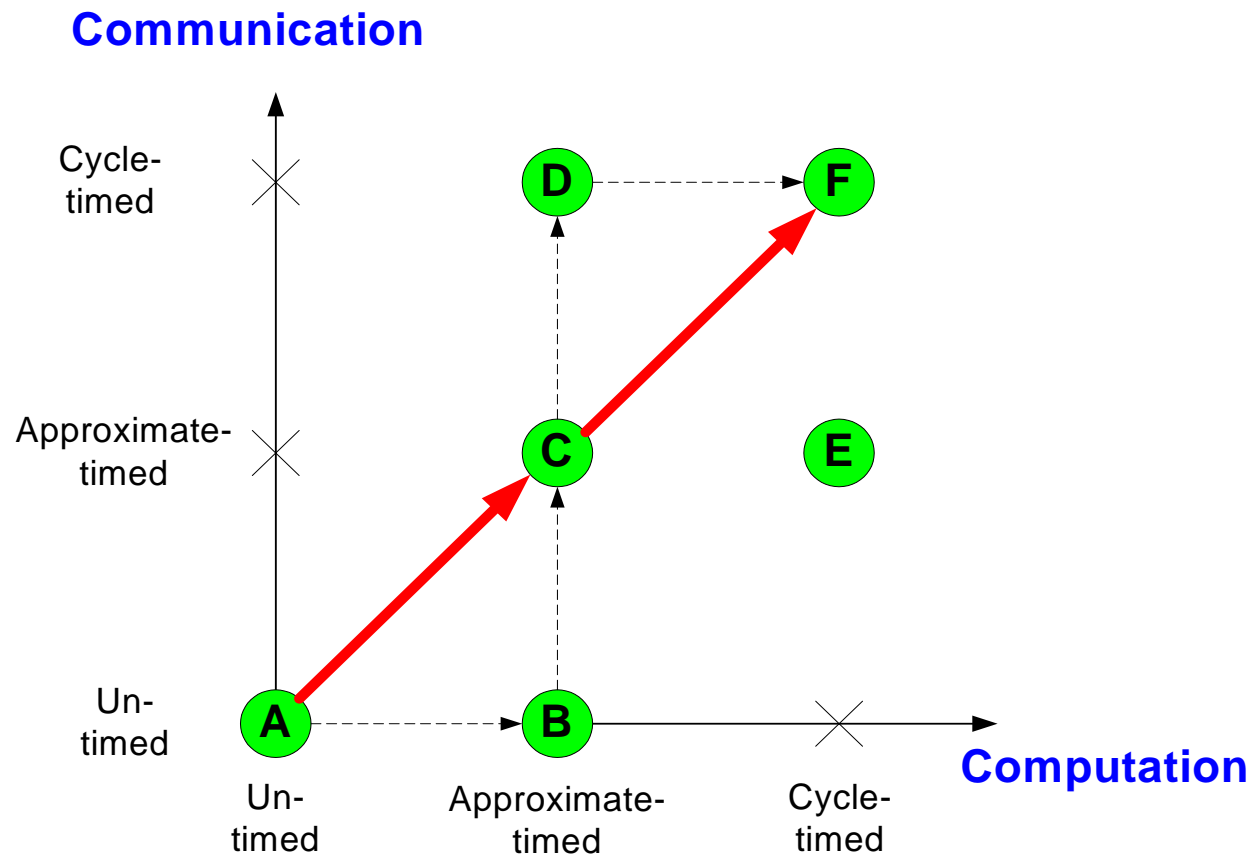
- **Challenge 1: increasing complexity of design**
 - System behavior (application)
 - Thousands of lines of code
 - Hierarchical and concurrent model
 - System architecture
 - Hundreds of PEs with different attributes (speed, protocol)
 - Exploration
 - System behavior → system architecture
- **Challenge 2: decreasing time to market**
- **Goal 1: move the design to the high levels of abstraction**
- **Goal 2: speed up design automation**

Outline

- Motivation
- Design flow and problem definition
- Re-targetable profiling
- Specification tuning
- Variable mapping
- Conclusions

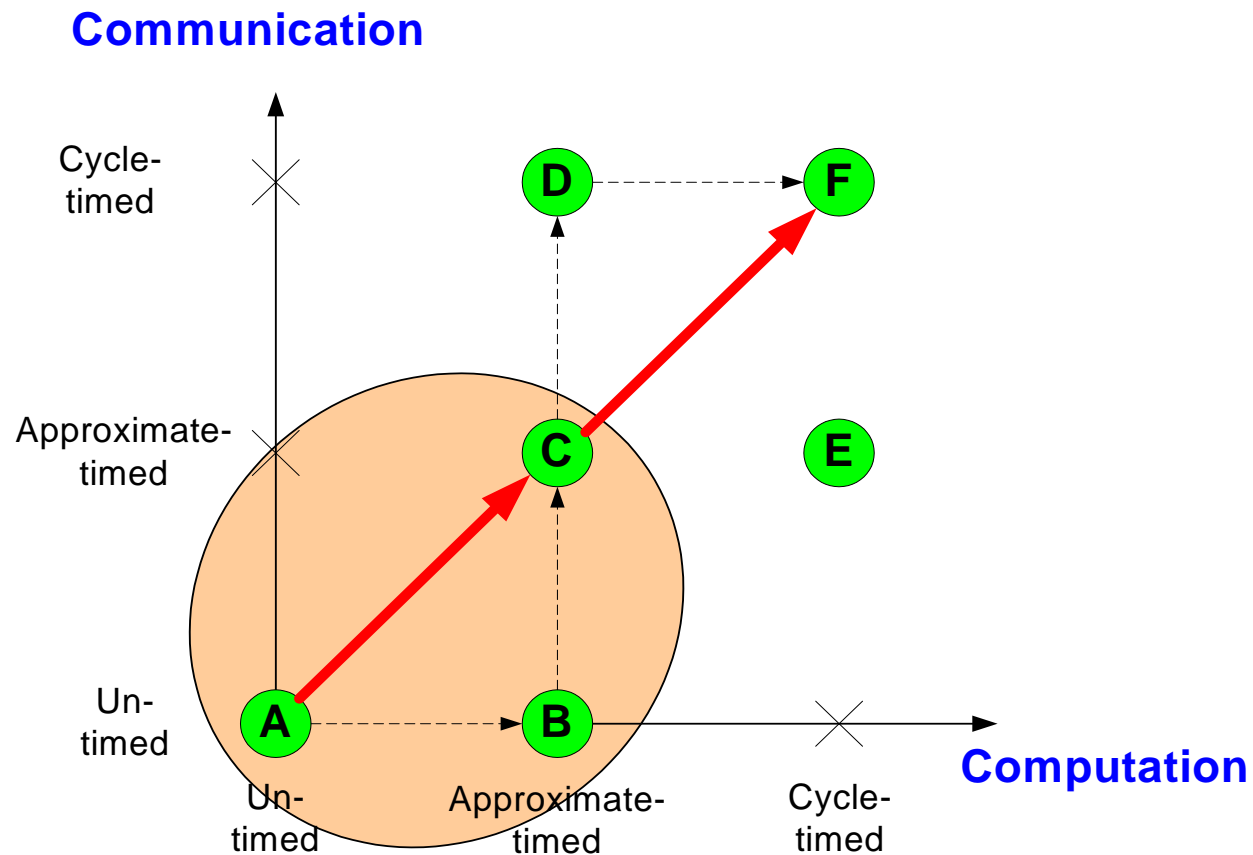
System Level Models

- Models based on time granularity of computation and communication
- A system level design flow is a path from model A to F

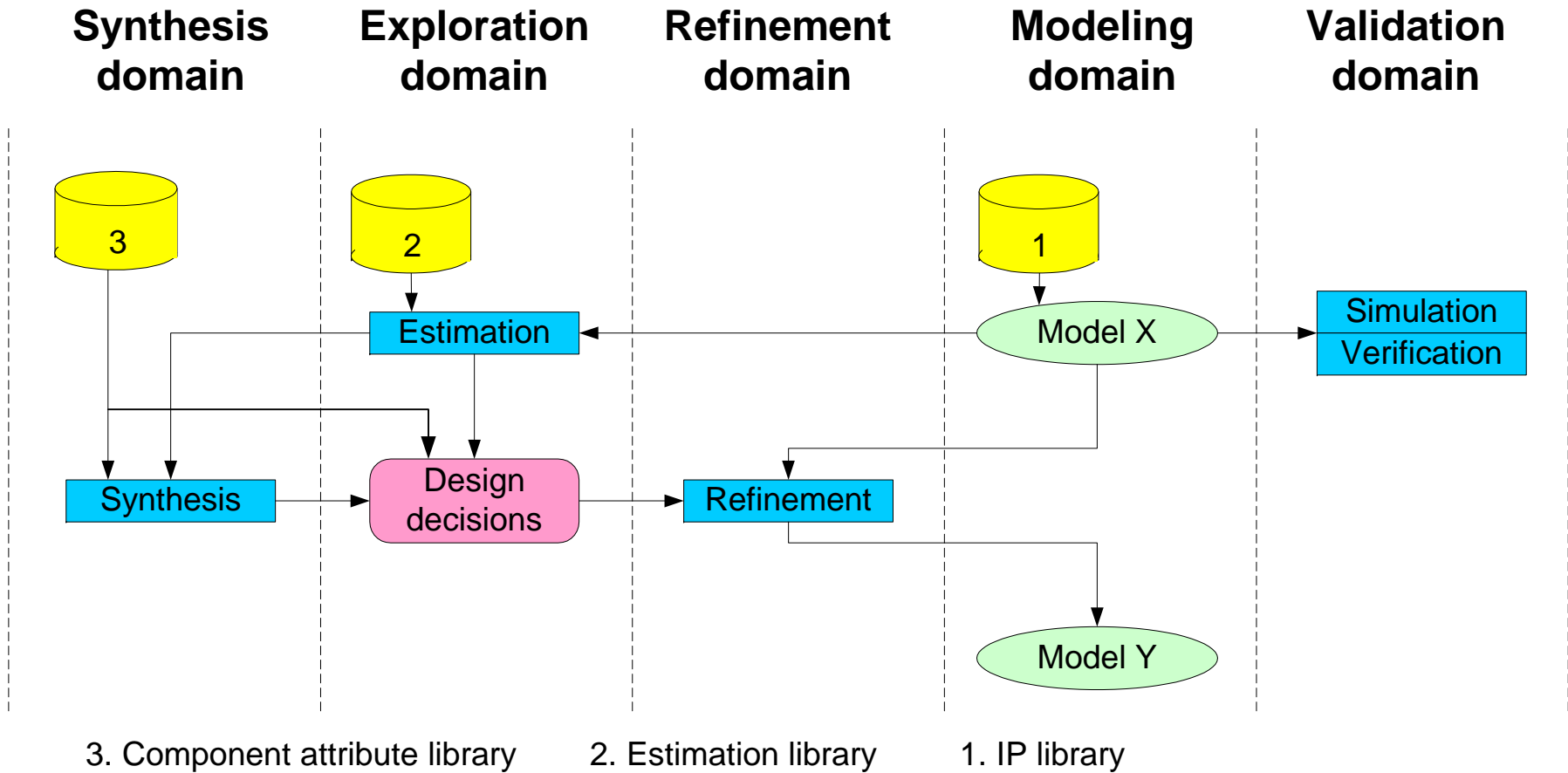


System Level Models

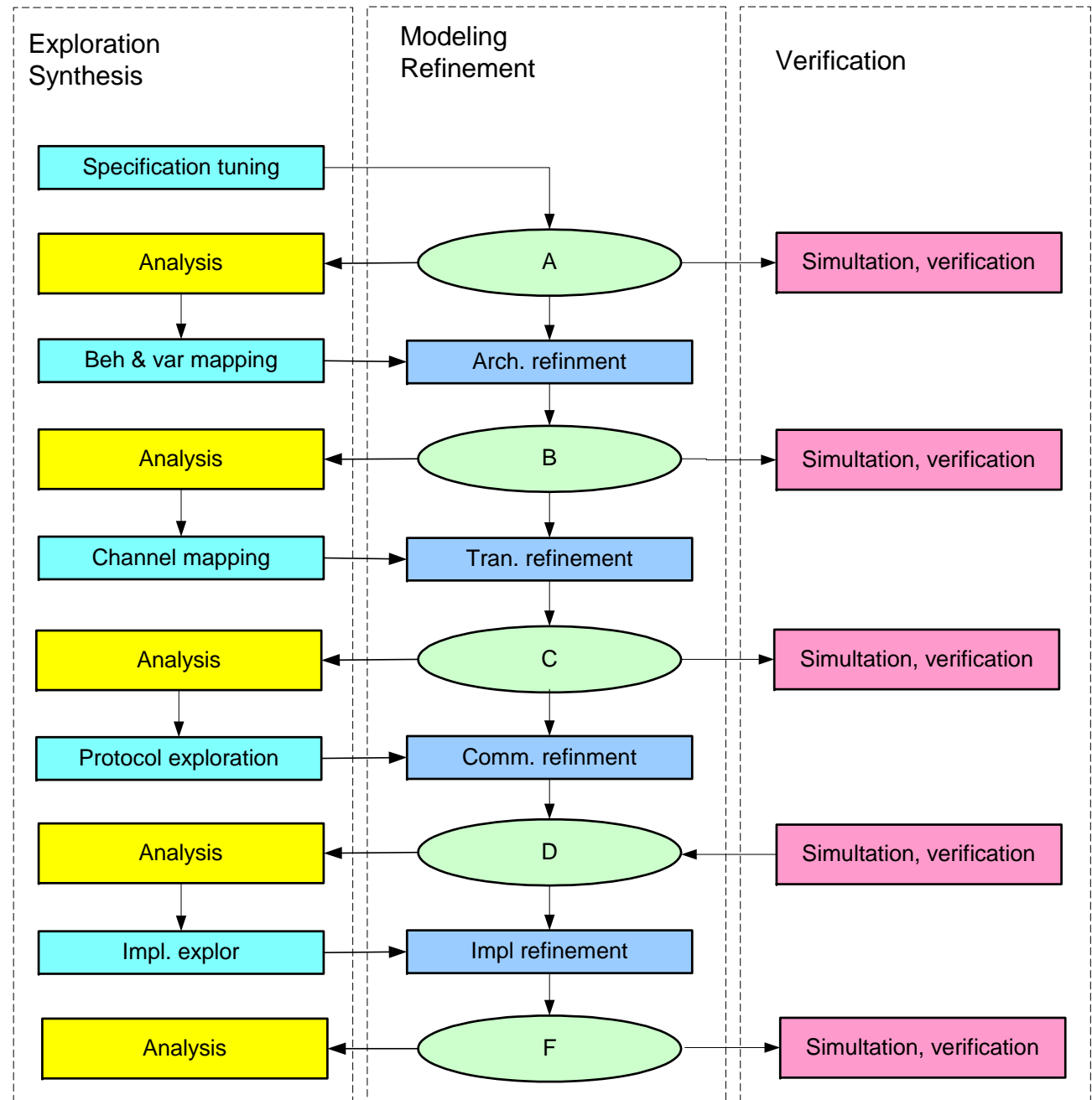
- Models based on time granularity of computation and communication
- A system level design flow is a path from model A to F



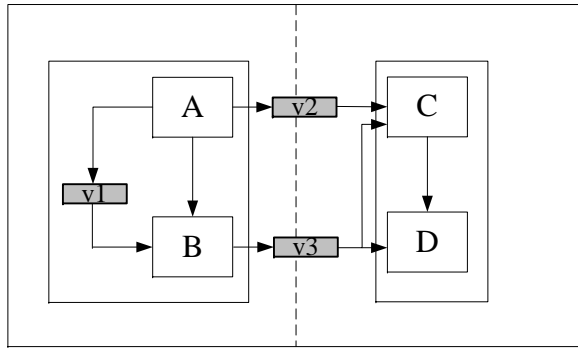
Five Design Domains



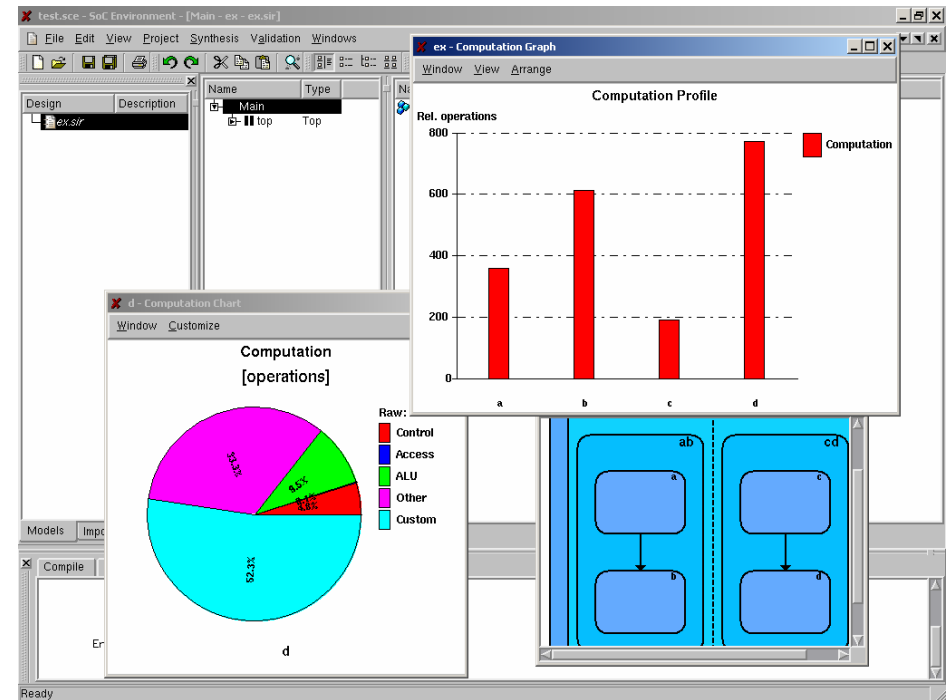
System Level Design Flow



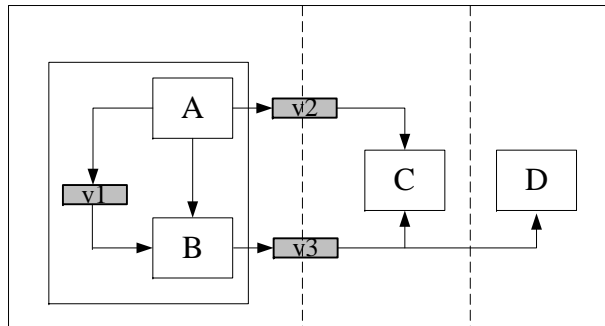
Design Flow Example



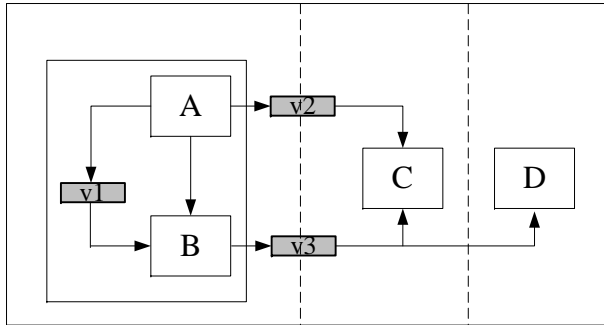
Analysis



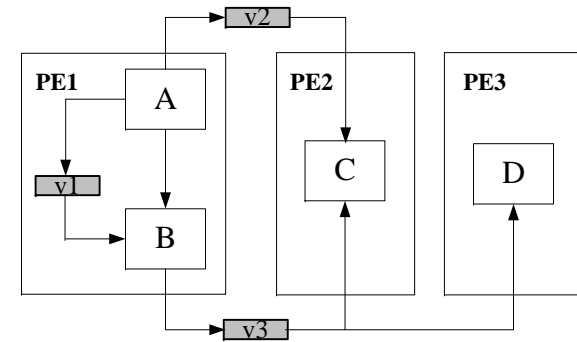
Spec Tuning



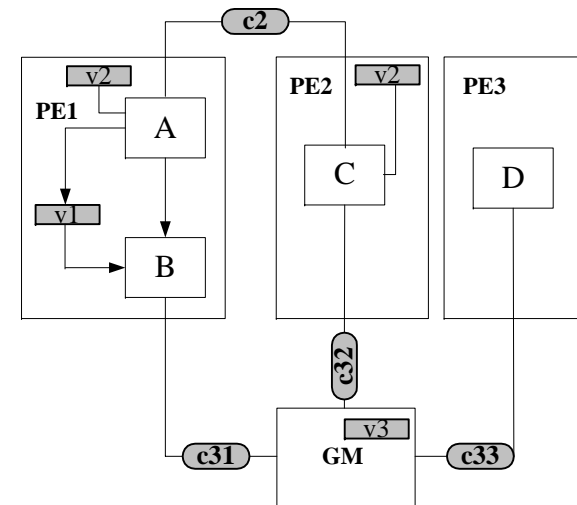
Design Flow Example (cont.)



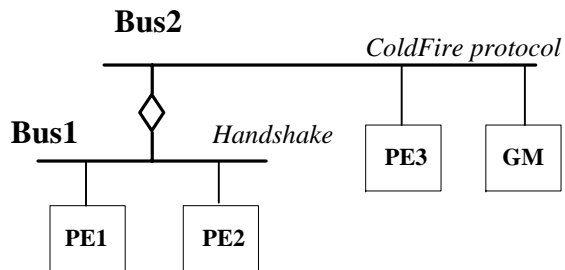
Behavior Mapping



Variable Mapping



Channel Mapping



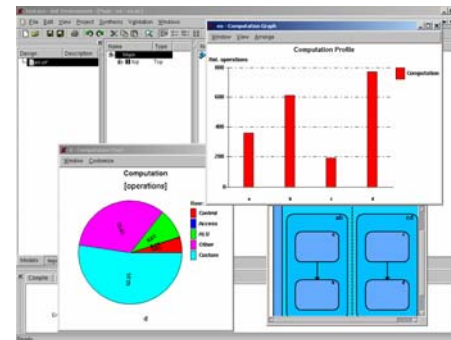
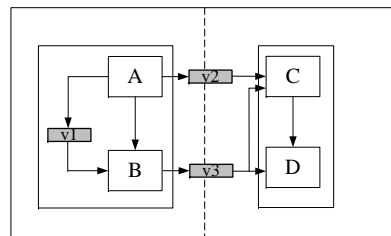
Channel	Mapped buses
c2	bus1
c31	bus1, bus2
c32	bus1, bus2
c33	bus2

Outline

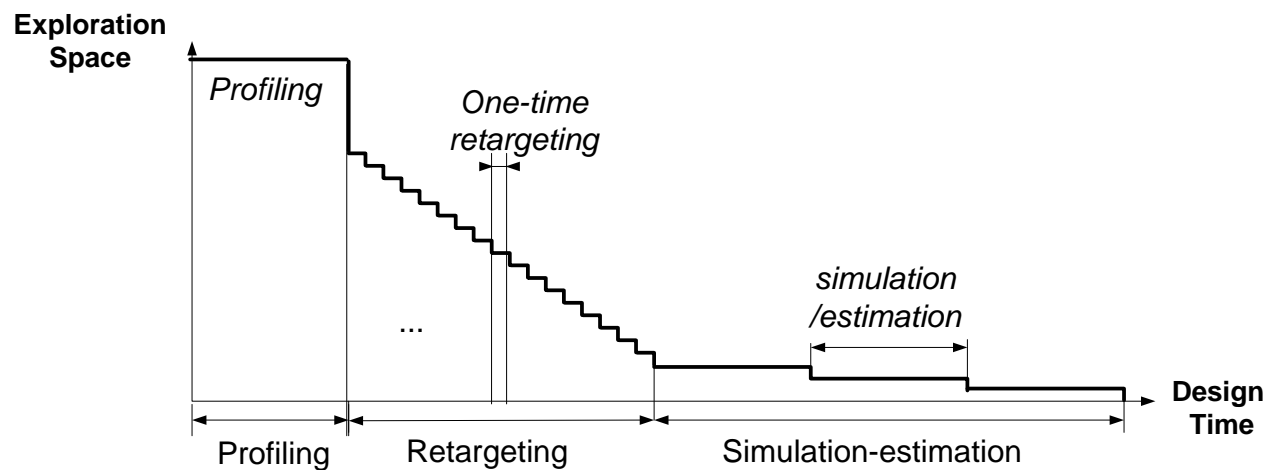
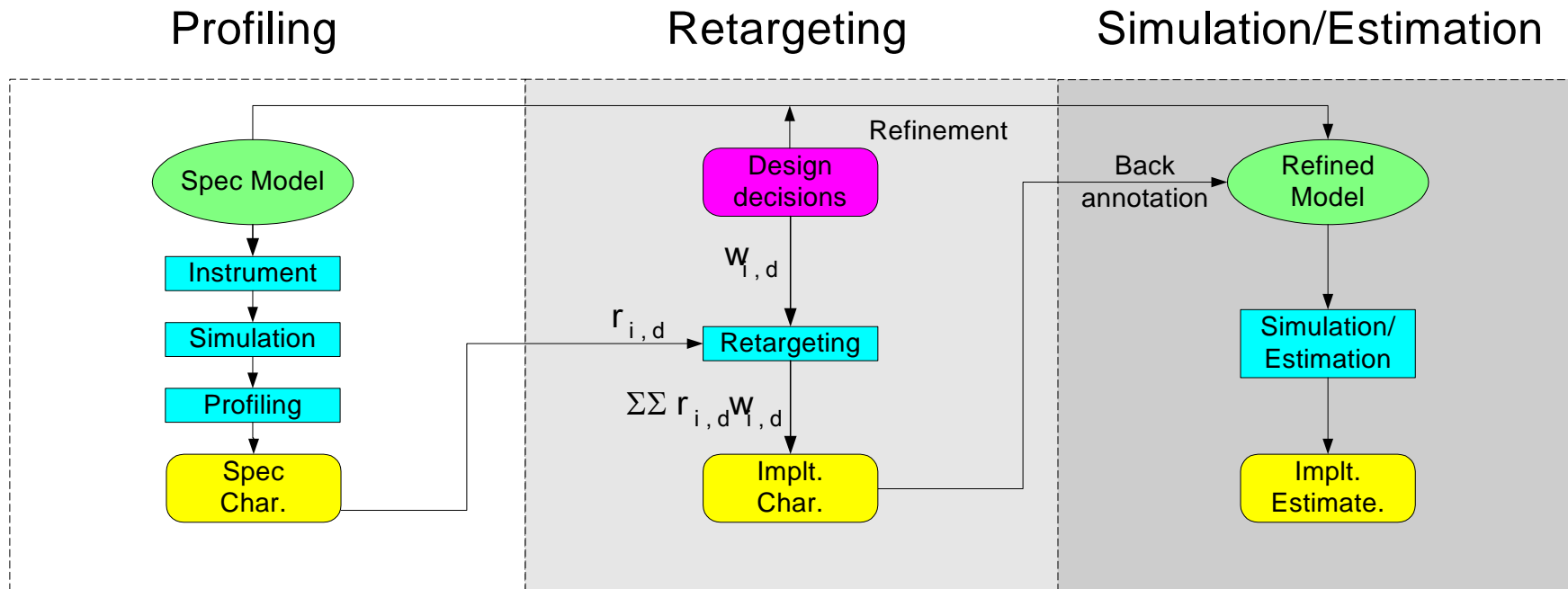
- Motivation
- Design flow and problem definition
- **Re-targetable profiling**
- Specification tuning
- Variable mapping
- Conclusions

Overview

- Motivation at system level
 - Multi-level evaluation
 - Multi-metrics evaluation
 - Rapid evaluation of a large number of design
- Proposed approach
 - Static profiling + dynamic retargeting.



Profiling/Estimation Flow

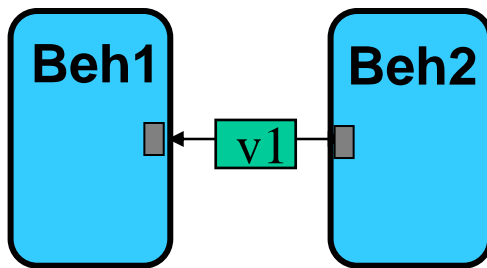
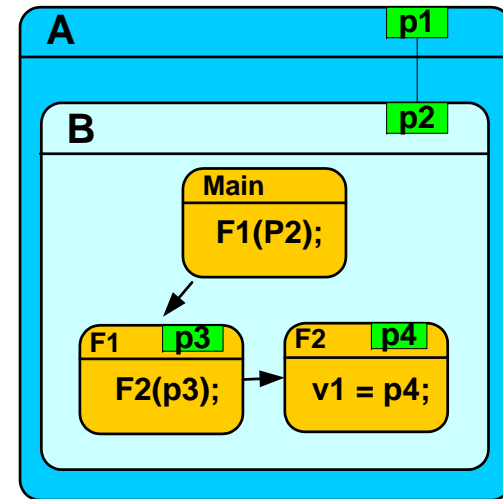


Multi-Metrics: Computed Characteristics

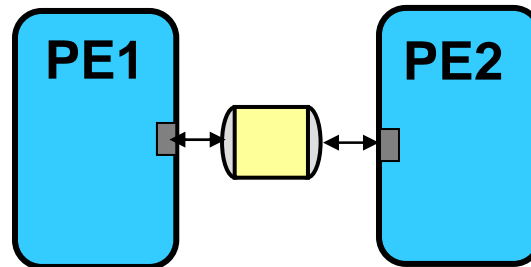
		Specification Char.	Implementation Char.
Operation	Static	Code complexity	Program memory size (SW) Custom hardware controller (HW)
	Dynamic	Computational complexity	Execution time Power consumption
Traffic	Static	Connectivity complexity	Communication delays
	Dynamic	Access complexity	Communication delays
Memory	Static	Static mem requirements.	Static memory size
	Dynamic	Stack & heap requirements	Stack & heap size

Deriving Traffic Characteristics

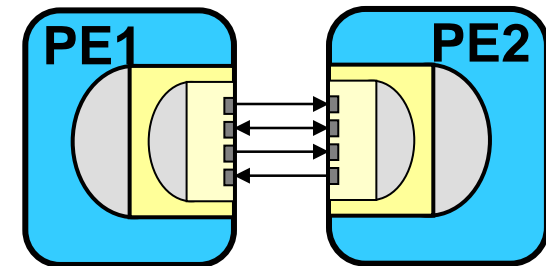
- Profiling for the hierarchically instantiated behaviors and recursively called functions
 - $A.p1 \rightarrow B.p2 \rightarrow B.F1.p3 \rightarrow B.F2.P4$
- Behavior, channel, variable, port at each hierarchical level
- Different abstraction levels



- Beh.port \rightarrow Var



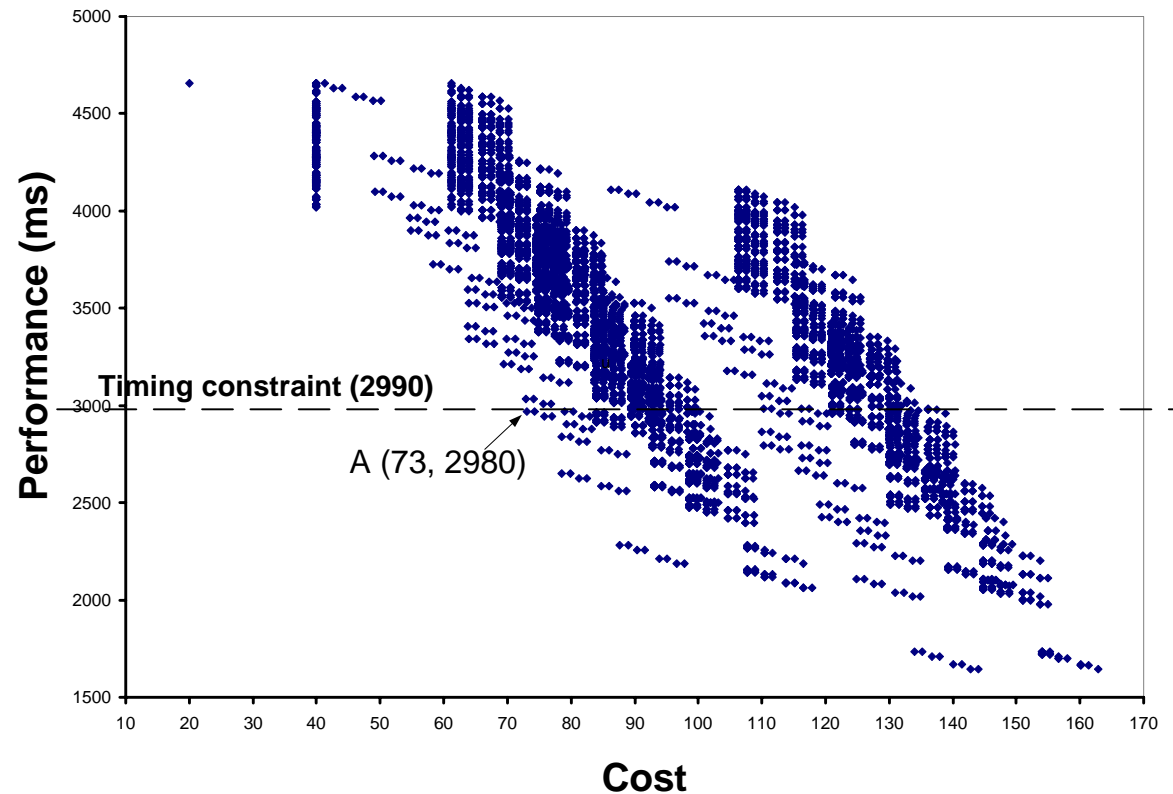
- Beh.port \rightarrow Chan
- Chan in Chan



- Beh.port \rightarrow Pin
- Chan.port \leftrightarrow Chan

Experimental Result

- 8 behaviors → 3 PEs
- Total $3^8 = 6561$ design alternatives
- Evaluation time: 3:15 hour
 - 1 simulation (2.23s)
 - 1 profiling (8.41s)
 - 6561 retargeting (0.8s)
 - 6561 mapping (0.97s)



Contributions

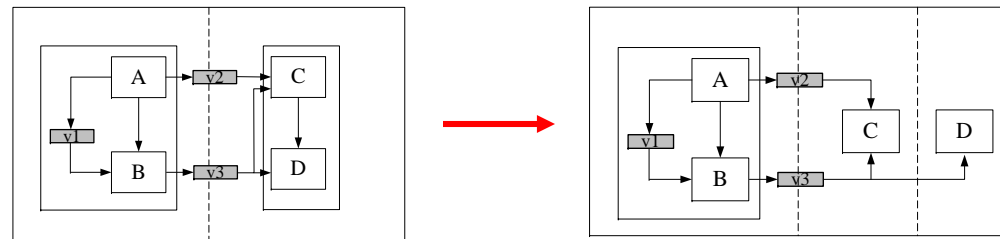
- Retargeting is ultra-fast
- Compute operation, traffic, and memory characteristics in both dynamic and static fields.
- Analyze for different abstraction levels
 - Implementation-independent
 - Implementation-dependent

Outline

- Motivation
- Design flow and problem definition
- Re-targetable profiling
- **Specification tuning**
- Variable mapping
- Conclusions

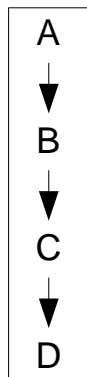
Overview

- Motivation at system level
 - Design from C/C++ code: ex. lack concurrency
 - Modeling the specification manually is inefficient
- Concurrency optimization
 - Explicitly model all the concurrency implied in the original specification

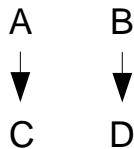


Concurrency Optimization

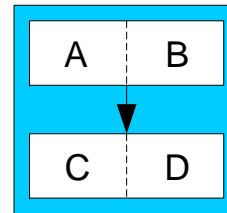
- Goals
 - Minimize the length of the critical path (LCP)
 - Minimize the number of added dependencies among behaviors (NAD)
- Proposed constructive algorithm
 - Extract the concurrency
 - Construct the behaviors hierarchically and concurrently



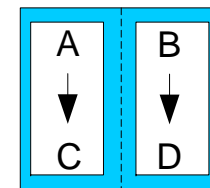
(a) Original sequence



(b) Dependency graph



(c) Solution 1



(d) Solution 2

Experimental Result

- Compare our algorithm with ASAP algorithm
 - LCP(our) is 11% longer than LCP(ASAP)
 - NAD(ASAP) is 325% larger than NAD(our)

Contributions

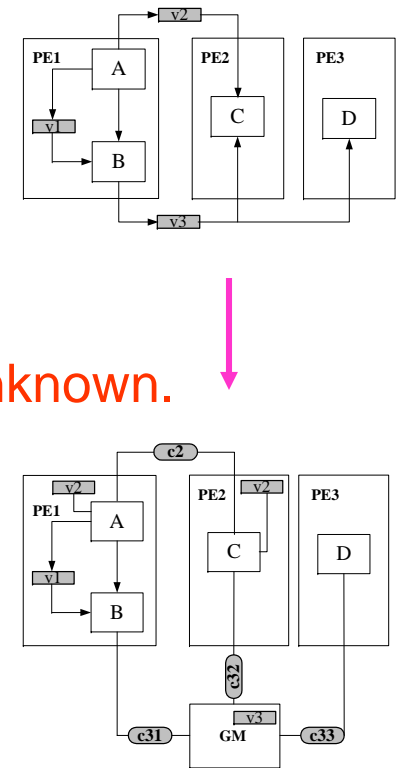
- New algorithm to explicitly model all the concurrency hierarchically
- Ease the ANSI C to SpecC conversion for system design
- Provide suitable input model for system level mapping

Outline

- Motivation
- Design flow and problem definition
- Re-targetable profiling
- Specification tuning
- **Variable mapping**
- Conclusions

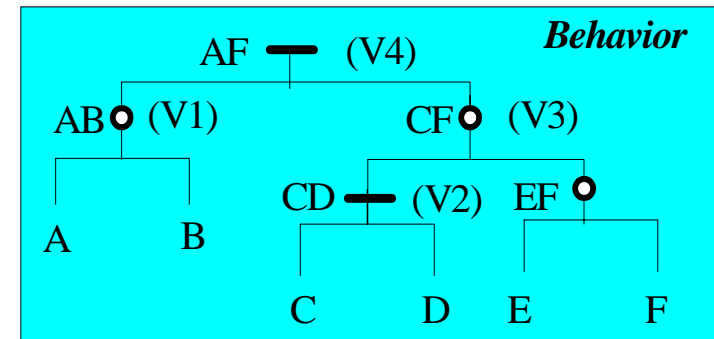
Variable Mapping Overview

- Definition
 - Select global memories for the system architecture, and map the behavior variables to the global memories or local memories of PEs.
- Motivation at system level
 - Life time analysis at system level
 - Concurrency and hierarchy.
 - No basic unit: state (in FSM) and instruction.
 - Variable size variety.
 - Interconnection topology and bus information are unknown.
- Proposed approach
 - Life time analysis at system level
 - Variable mapping algorithms
 - Minimize the required memory size
 - Minimize generated traffic among PEs and global memories



Life Time Analysis – Mapping Independent

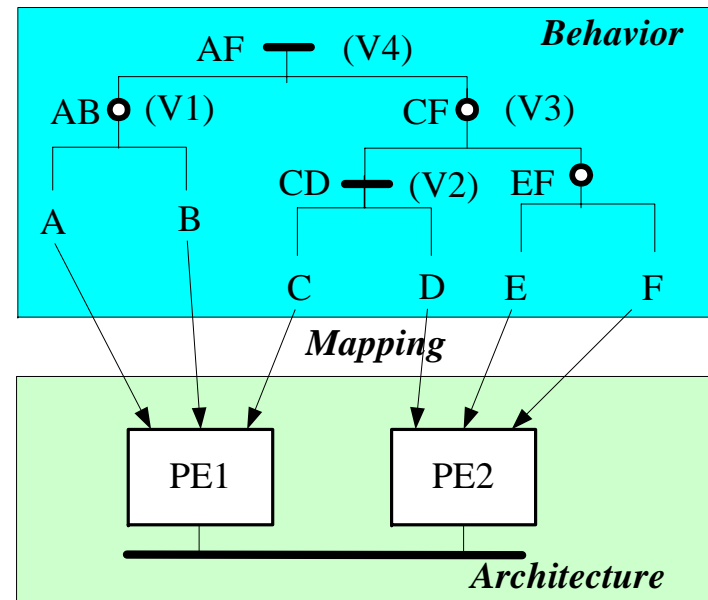
- Lifetime of any variable equals to the lifetime of the behavior in which the variable is declared.
- By analyzing the overlapping relation among behaviors, we derive the overlapping relation among variables.



Life Time Analysis – Mapping Dependent

- For (b,m) , where b is a behavior and m is a local/global mem, we compute:
 - Total-used memory size $t(b,m)$
 - Un-used memory size $u(b,m)$

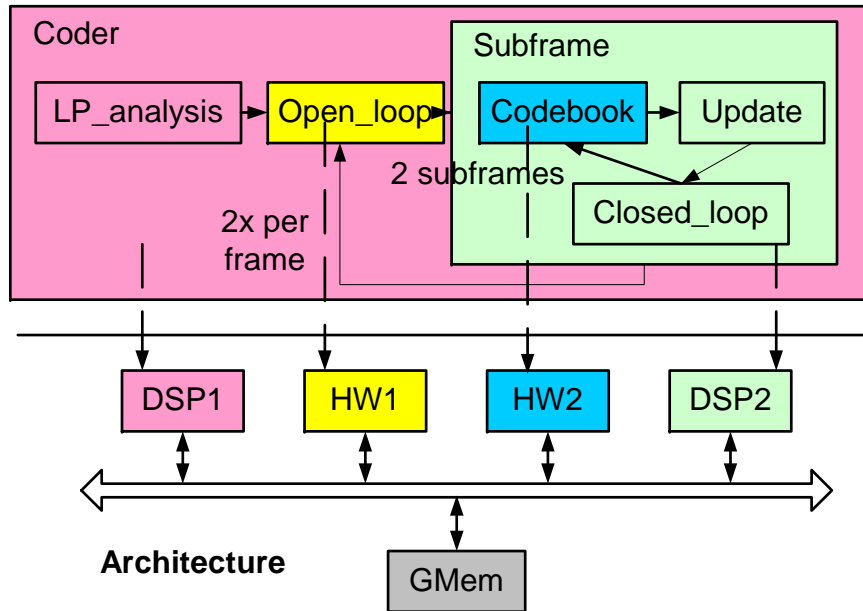
- Attributes of b & m



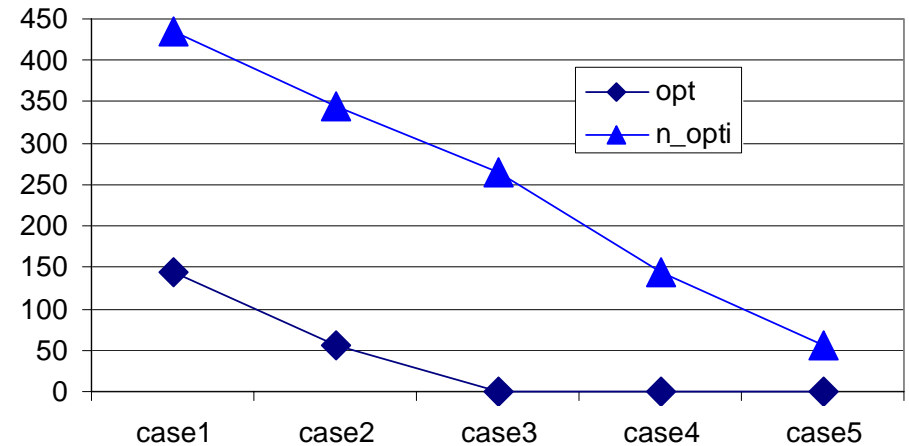
- After mapping a number of variables, we compute
 - $size(m) = t(\text{root}, m)$
 - For an unmapped variable $v(b)$. if $u(b, m) \geq size(v, m)$, then v can be mapped to m .

Experimental Result

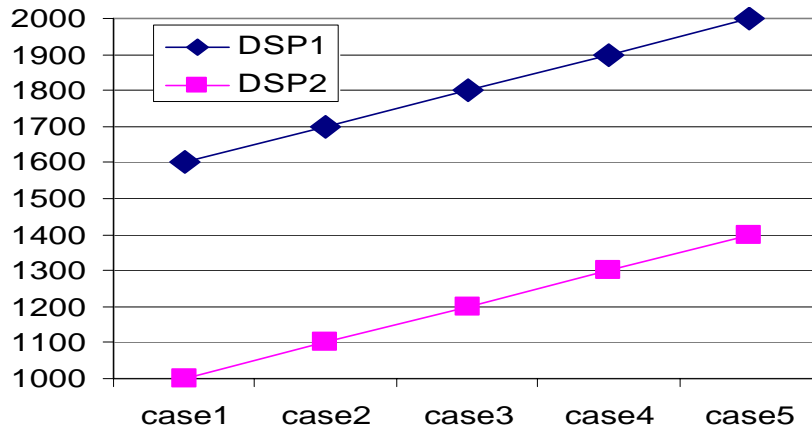
Behavior



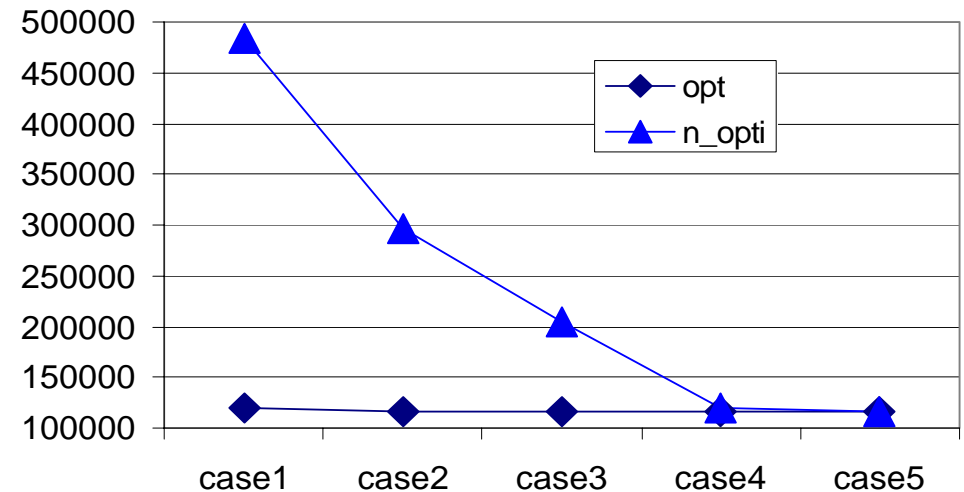
Global Memory Size (Byte)



Local Memory Size (Byte)



Traffic (Byte)



Contributions

- Analyze the variable lifetime at system level based on the attributes of behaviors and PEs
- Move variable mapping to higher abstraction level, even before the interconnection topology is known.

Outline

- Motivation
- Design flow and problem definition
- Re-targetable profiling
- Specification tuning
- Variable mapping
- **Conclusions**

Overall Conclusions

- Summary
 - Attempt to solve some problems for exploration and synthesis above approximated computation and communication model.
- Contributions
 - Move the design decision to higher abstraction levels.
 - Solve some new problems
 - Traffic profiling for system level design
 - Concurrency optimization
 - Behavior mapping based on computational characteristics
 - Lifetime analysis for variable mapping
 - Channel mapping considering transducer related issues.
 - Speed up the design process.
 - Free designers from tedious and error-prone tasks.

Publication

• Book chapters

- "C/C++ Based System Design Flow Using SpecC, VCC, and SystemC". Authors: L. Cai, M. Olivarez, P. Kritzinger, D. Gajski. In E Villar. Editors. "System Specification and Design Languages" Kluwer. 2003
- "The Guidelines and JPEG Encoder Study Case of System-Level Architecture Exploration Using the SpecC Methodology". Authors: L. Cai, M. Olivarez, D. Gajski. In A. Mignotte, Editors. "System On Chip Design Languages". Kluwer. Boston 2002

• Conferences

- "A Novel Memory Size Model for Variable-Mapping In System Level Design" Authors: L. Cai, H. Yu, D. Gajski. Asia and South Pacific Design Automation Conference Yokohama, Japan 2004
- "Transaction Level Modeling: An Overview" Authors: L. Cai, Daniel Gajski. First IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, Newport Beach CA, 2003.
- "Introduction of System Level Architecture Exploration Using the SpecC Methodology" Authors: L. Cai, M. Olivarez, D. Gajski. IEEE International Symposium on Circuits and Systems 2001, Sydney, Australia.
- "C/C++ Based System Design Flow Using SpecC, VCC, and SystemC" Authors: L. Cai, M. Olivarez, P. Kritzinger, D. Gajski. Forum on specification and Design Languages 2002, Marseille, France
- "The Guidelines and JPEG Encoder Study Case of System Level Architecture Exploration Using the SpecC Methodology". Authors: L. Cai, M. Olivarez, D. Gajski. Forum on specification and Design Languages 2001, Lyon, France