

# Transaction Level Modeling: An Overview

Daniel Gajski  
Lukai Cai

**Center for Embedded Computer Systems**  
**University of California, Irvine**  
**[www.cecs.uci.edu/~{gajski,lcai}](http://www.cecs.uci.edu/~{gajski,lcai})**



# Acknowledgement

We would like to thank transaction level team at CECS that has contributed many ideas through numerous lunch discussions:

Samar Abdi

Rainer Doemer

Andreas Gerstlauer

Junyu Peng

Dongwan Shin

Haobo Yu



# Overview

- Motivation for TLM
- TLM definition
- TLMs at different abstraction levels
- TLMs for different design domains
- SL methodology = model algebra
- Conclusion



# Motivation

- SoC problems
  - Increasing complexity of systems-on-chip
  - Shorter times-to-market
- SoC solutions
  - Higher level of abstraction – transaction level modeling (TLM)
  - IP reuse
  - System standards
- TLM questions
  - What is TLM ?
  - How to use TLM ?
- This paper
  - TLM taxonomy
  - TLM usage



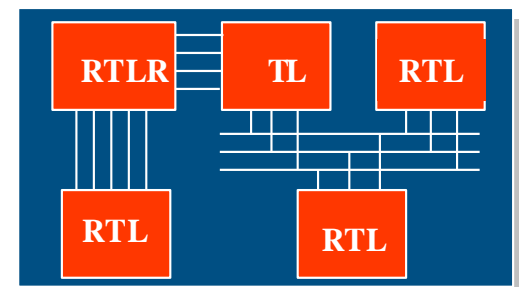
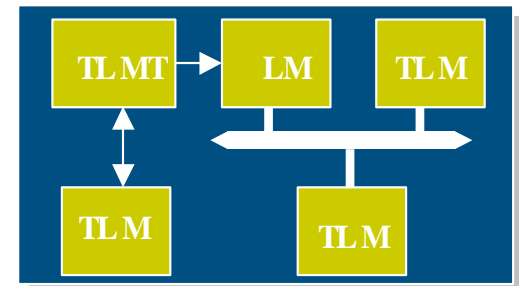
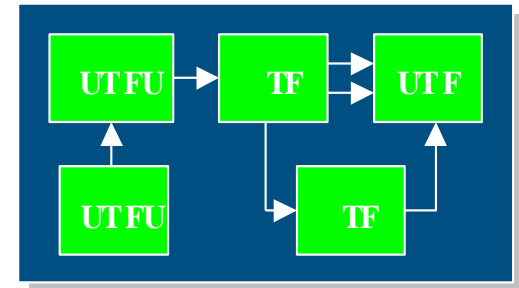
# TLM Definition

- TLM =  $\langle \{\text{objects}\}, \{\text{compositions}\} \rangle$
- Objects
  - Computation objects + communication objects
- Composition
  - Computation objects read/write abstract (above pin-accurate) data types through communication objects
- Advantages
  - Object independence
    - Each object can be modeled independently
  - Abstraction independence
    - Different objects can be modeled at different abstraction levels



# Multiple Levels of Abstraction

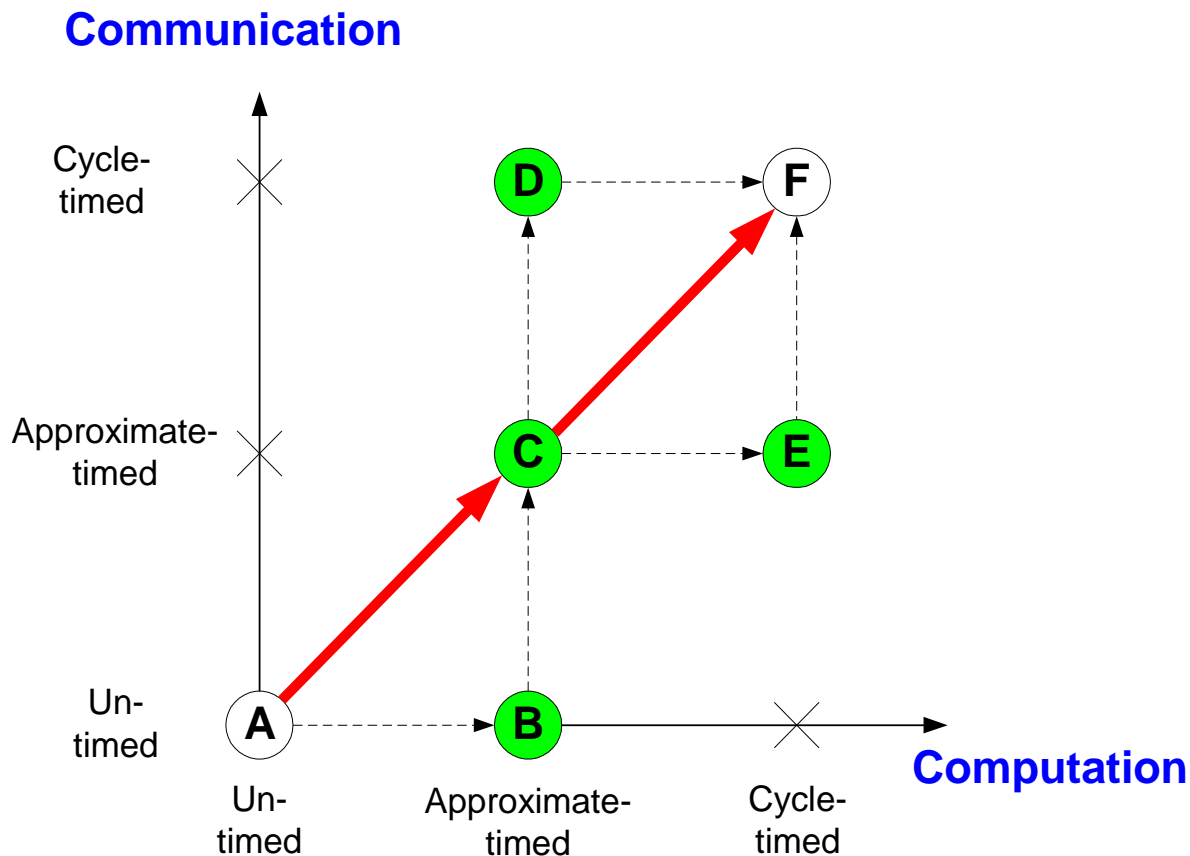
- (untimed) Functional level
  - executable specification
- Transaction level
  - analyze SoC architecture, early SW development
  - estimated timing
- Pin level
  - RTL/behavioral HW design and verification



At TLM level, concerns only focus on **mapping out data flow details**: the type of data that flows and where it is stored

# Abstraction Models

- Time granularity for communication/computation objects can be classified into 3 basic categories.
- Models B, C, D and E could be classified as TLMs.



- A. **"Specification model"**  
"Untimed functional models"
- B. **"Component-assembly model"**  
"Architecture model"  
"Timed functional model"
- C. **"Bus-arbitration model"**  
"Transaction model"
- D. **"Bus-functional model"**  
"Communication model"  
"Behavior level model"
- E. **"Cycle-accurate computation model"**
- F. **"Implementation model"**  
"Register transfer model"



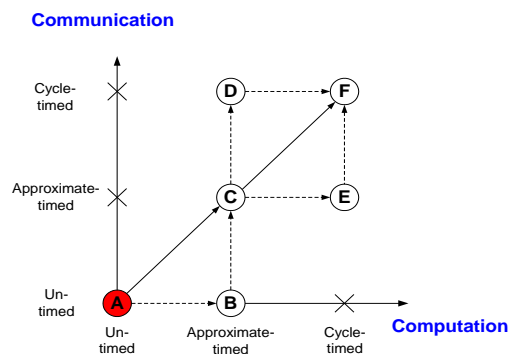
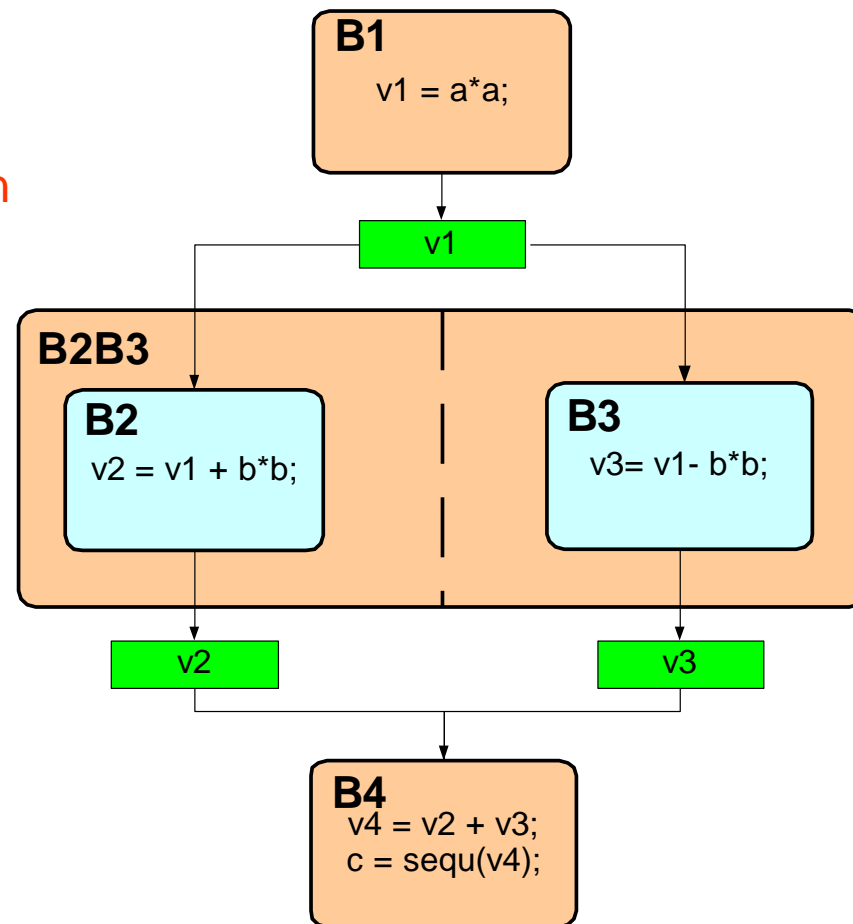
# A: “Specification Model”

## Objects

- Computation
- Behaviors
- Communication
- Variables

## Composition

- Hierarchy
- Order
- Sequential
- Parallel
- Piped
- States
- Transitions
- TI, TOC
- Synchronization
- Notify/Wait

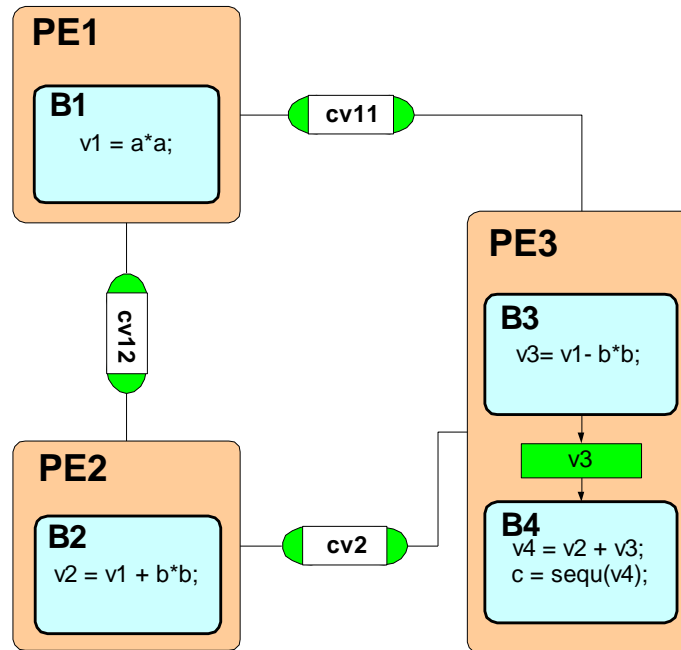




# B: “Component-Assembly Model”

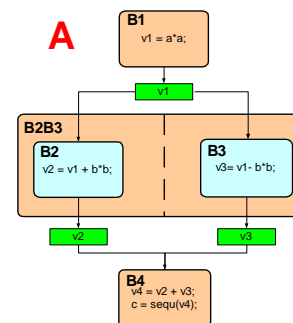
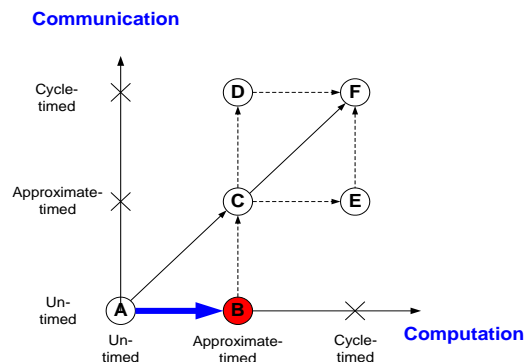
## Objects

- Computation
  - Proc
  - IPs
  - Memories
- Communication
  - Variable channels



## Composition

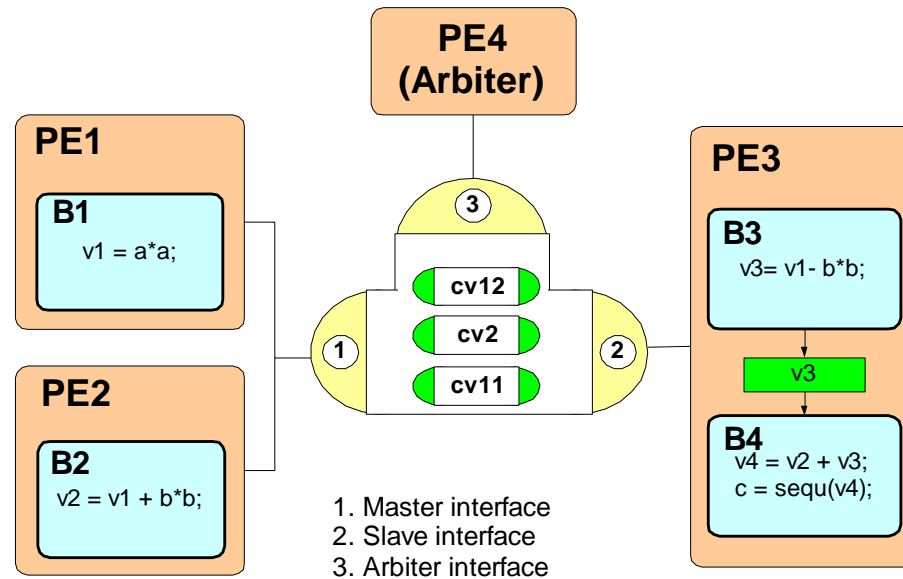
- Hierarchy
- Order
  - Sequential
  - Parallel
  - Piped
  - States
- Transitions
  - TI, TOC
- Synchronization
  - Notify/Wait



# C: “Bus-Arbitration Model”

## Objects

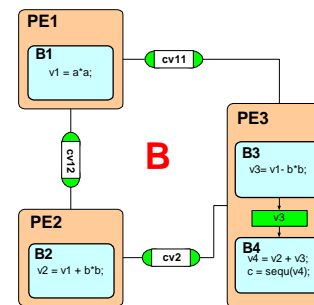
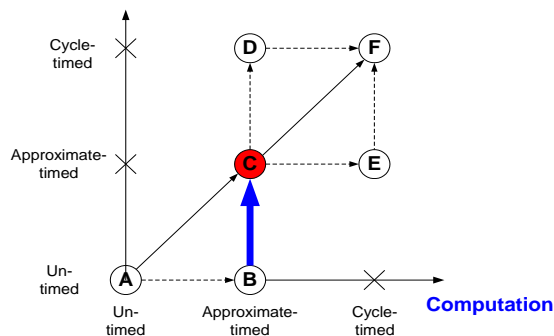
- Computation
  - Proc
  - IPs (Arbiters)
  - Memories
- Communication
  - Abstract bus channels



## Composition

- Hierarchy
- Order
  - Sequential
  - Parallel
  - Piped
  - States
- Transitions
  - TI, TOC
- Synchronization
  - Notify/Wait

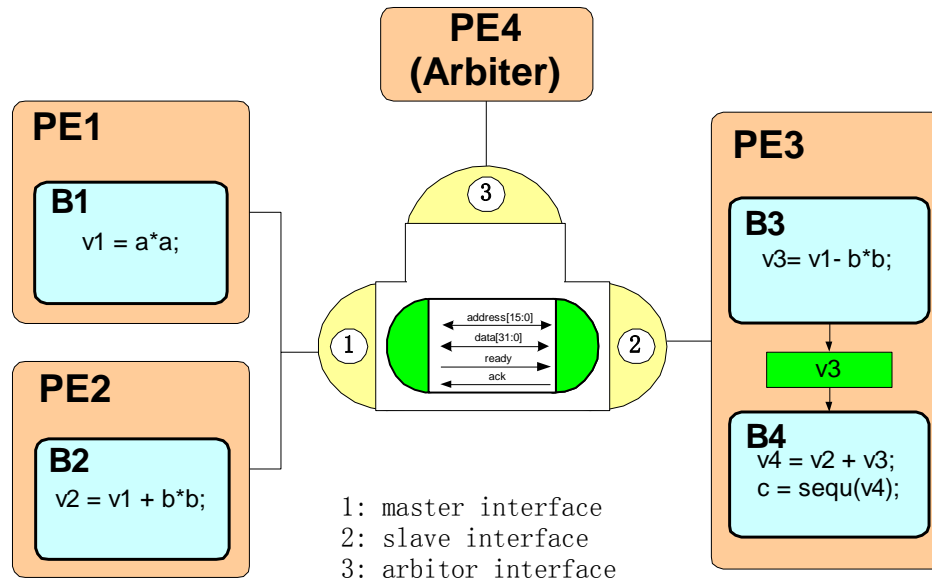
## Communication



# D: “Bus-Functional Model”

## Objects

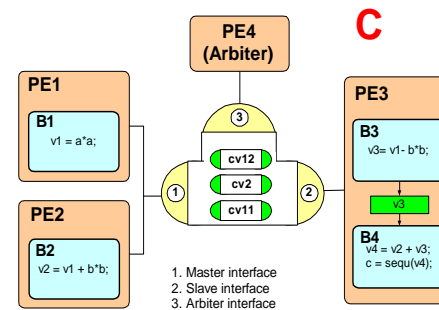
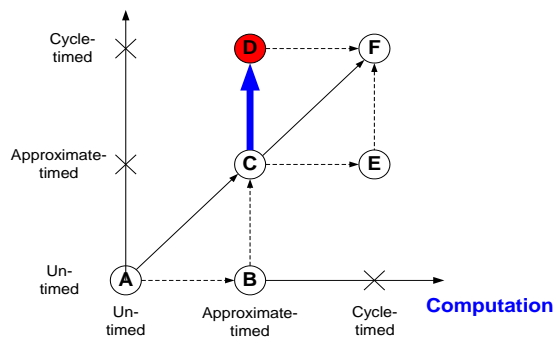
- Computation
  - Proc
  - IPs (Arbiters)
  - Memories
- Communication
  - Protocol bus channels



## Composition

- Hierarchy
- Order
  - Sequential
  - Parallel
  - Piped
  - States
- Transitions
  - TI, TOC
- Synchronization
  - Notify/Wait

## Communication



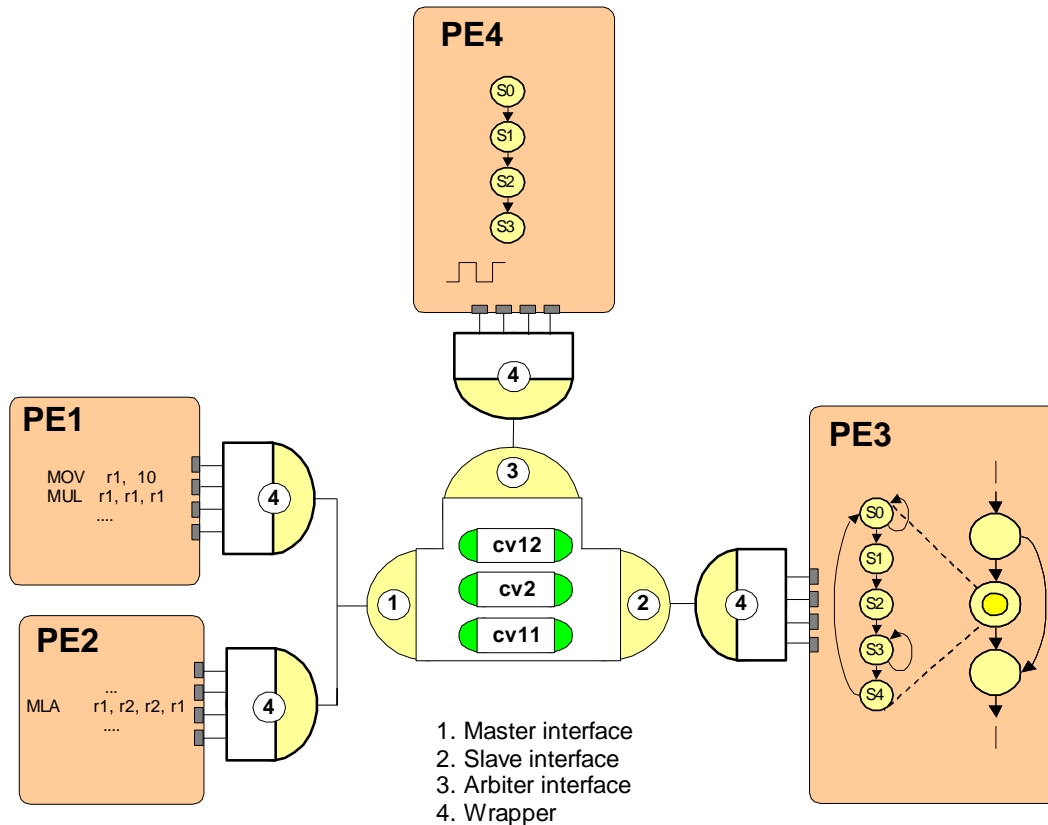
# E: “Cycle-Accurate Computation Model”

## Objects

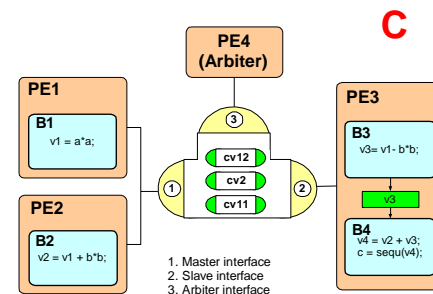
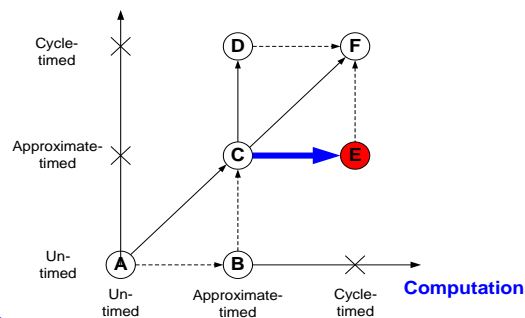
- Computation
  - Proc
  - IPs (Arbiters)
  - Memories
  - Wrappers
- Communication
  - Abstract bus channels

## Composition

- Hierarchy
- Order
  - Sequential
  - Parallel
  - Piped
  - States
- Transitions
  - TI, TOC
- Synchronization
  - Notify/Wait



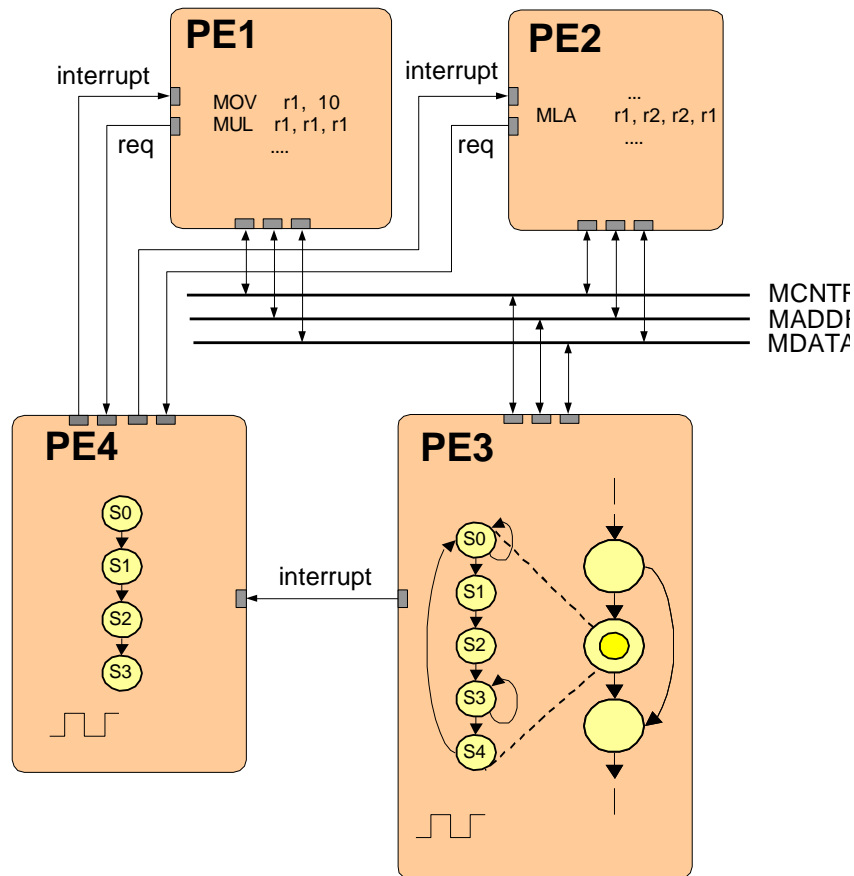
## Communication



# F: "Implementation Model"

## Objects

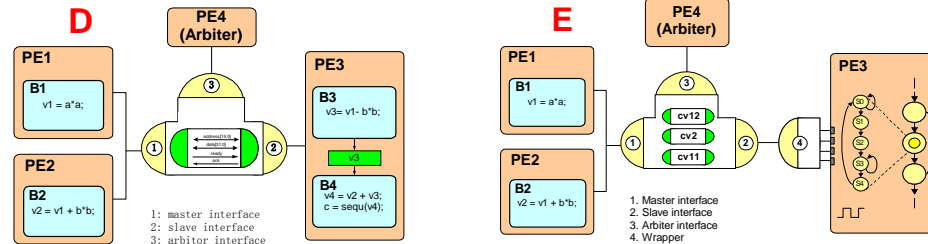
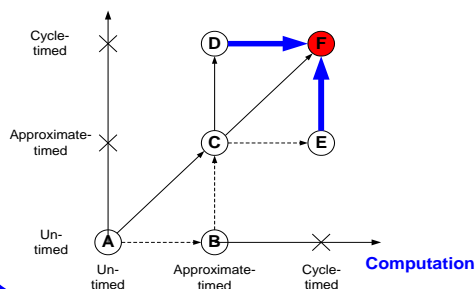
- **Computation**
  - Proc
  - IPs (Arbiters)
  - Memories
- **Communication**
  - Buses (wires)



## Composition

- **Hierarchy**
- **Order**
  - Sequential
  - Parallel
  - Piped
  - States
- **Transitions**
  - TI, TOC
- **Synchronization**
  - Notify/Wait

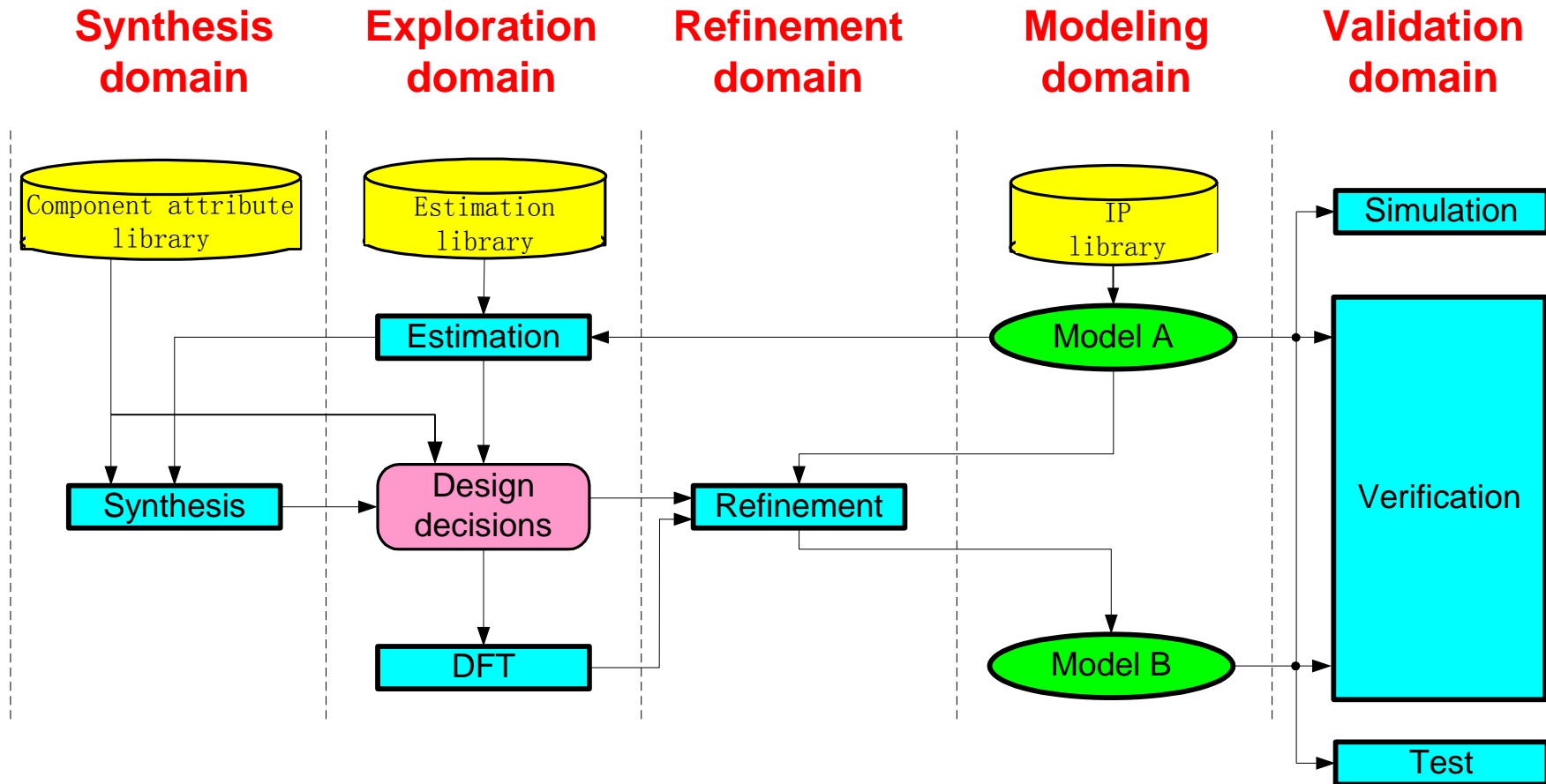
## Communication



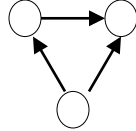
# Characteristics of Different Abstraction Models

<b>Models</b>	<b>Communication time</b>	<b>Computation time</b>	<b>Communication scheme</b>	<b>PE interface</b>
<b>Specification model</b>	no	no	variable	(no PE)
<b>Component-assembly model</b>	no	approximate	variable channel	abstract
<b>Bus-arbitration model</b>	approximate	approximate	abstract bus channel	abstract
<b>Bus-functional model</b>	time/cycle accurate	approximate	protocol bus channel	abstract
<b>Cycle-accurate computation model</b>	approximate	cycle-accurate	abstract bus channel	pin-accurate
<b>Implementation model</b>	cycle-accurate	cycle-accurate	bus (wire)	pin-accurate

# Design Domains



# Model Algebra

- Algebra =  $\langle \{\text{objects}\}, \{\text{operations}\} \rangle$  [ex:  $a * (b + c)$ ]
- Model =  $\langle \{\text{objects}\}, \{\text{compositions}\} \rangle$  [ex: 
- Transformation  $t(model)$  is a change in objects or compositions.
- Model refinement is an ordered set of transformations,  $\langle t_m, \dots, t_2, t_1 \rangle$ , such that  $model\ B = t_m( \dots ( t_2( t_1( model\ A ) ) ) \dots )$
- Model algebra =  $\langle \{\text{models}\}, \{\text{refinements}\} \rangle$
- Methodology is a sequence of models and corresponding refinements



# Model Definition

- Model =  $\langle \{\text{objects}\}, \{\text{composition rules}\} \rangle$
- Objects
  - Behaviors (representing tasks / computation / functions)
  - Channels (representing communication between behaviors)
- Composition rules
  - Sequential, parallel, pipelined, FSM
  - Behavior composition creates hierarchy
  - Behavior composition creates execution order
    - Relationship between behaviors in the context of the formalism
- Relations amongst behaviors and channels
  - Data transfer between channels
  - Interface between behaviors and channels

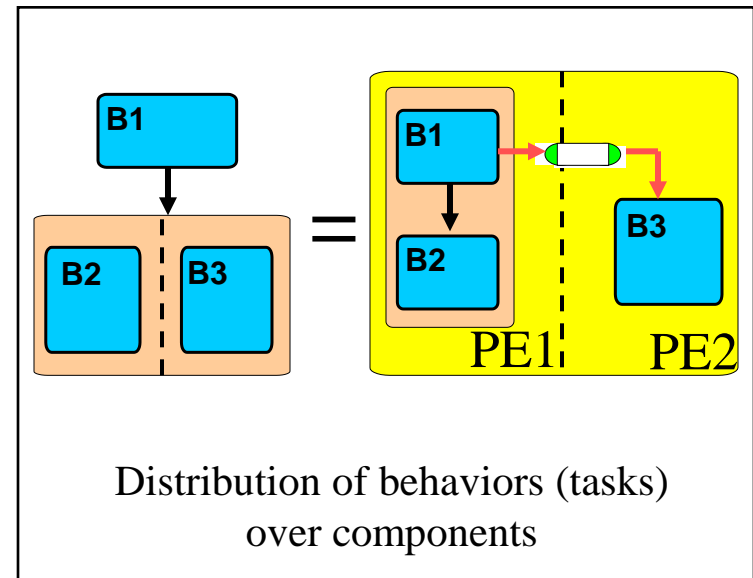
# Model Transformations (Rearrange and Replace)

- Rearrange object composition
  - To distribute computation over components
- Replace objects
  - Import library components
- Add / Remove synchronization
  - To correctly transform a sequential composition to parallel and vice-versa
- Decompose abstract data structures
  - To implement data transaction over a bus
- Other transformations
- .
- .

$$\mathbf{a}^*(\mathbf{b}+\mathbf{c}) = \mathbf{a}^*\mathbf{b} + \mathbf{a}^*\mathbf{c}$$

Distributivity of multiplication  
over addition

analogous to.....

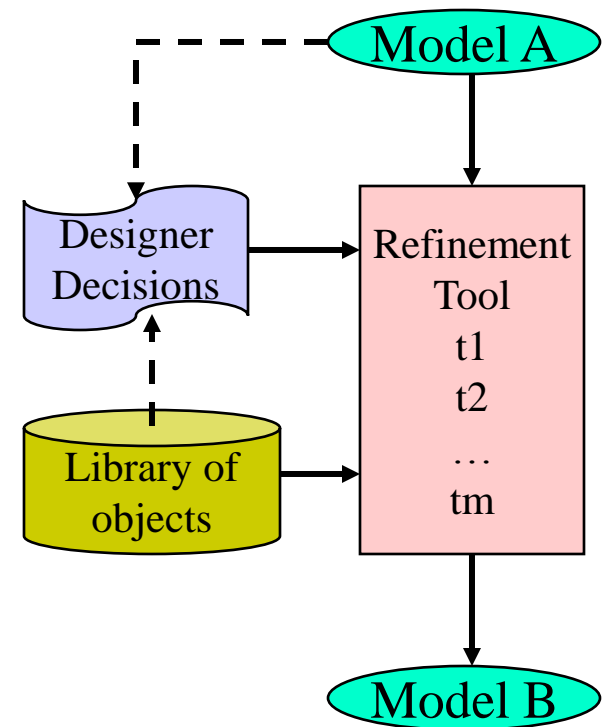


# Model Refinement

- Definition
  - Ordered set of transformations  $\langle t_m, \dots, t_2, t_1 \rangle$  is a refinement
    - $model\ B = t_m( \dots ( t_2( t_1( model\ A ) ) ) \dots )$
- Derives a more detailed model from an abstract one
  - Specific sequence for each model refinement
  - Not all sequences are relevant
- Equivalence verification
  - Each transformation maintains functional equivalence
  - The refinement is thus correct by construction
- Refinement based system level methodology
  - Methodology is a sequence of models and refinements

# Verification

- Transformations preserve equivalence
  - Same partial order of tasks
  - Same input/output data for each task
  - Same partial order of data transactions
  - Same functionality in replacements
- All refined models will be “equivalent” to input model
  - Still need to verify first model using traditional techniques
  - Still need to verify equivalence of replacements

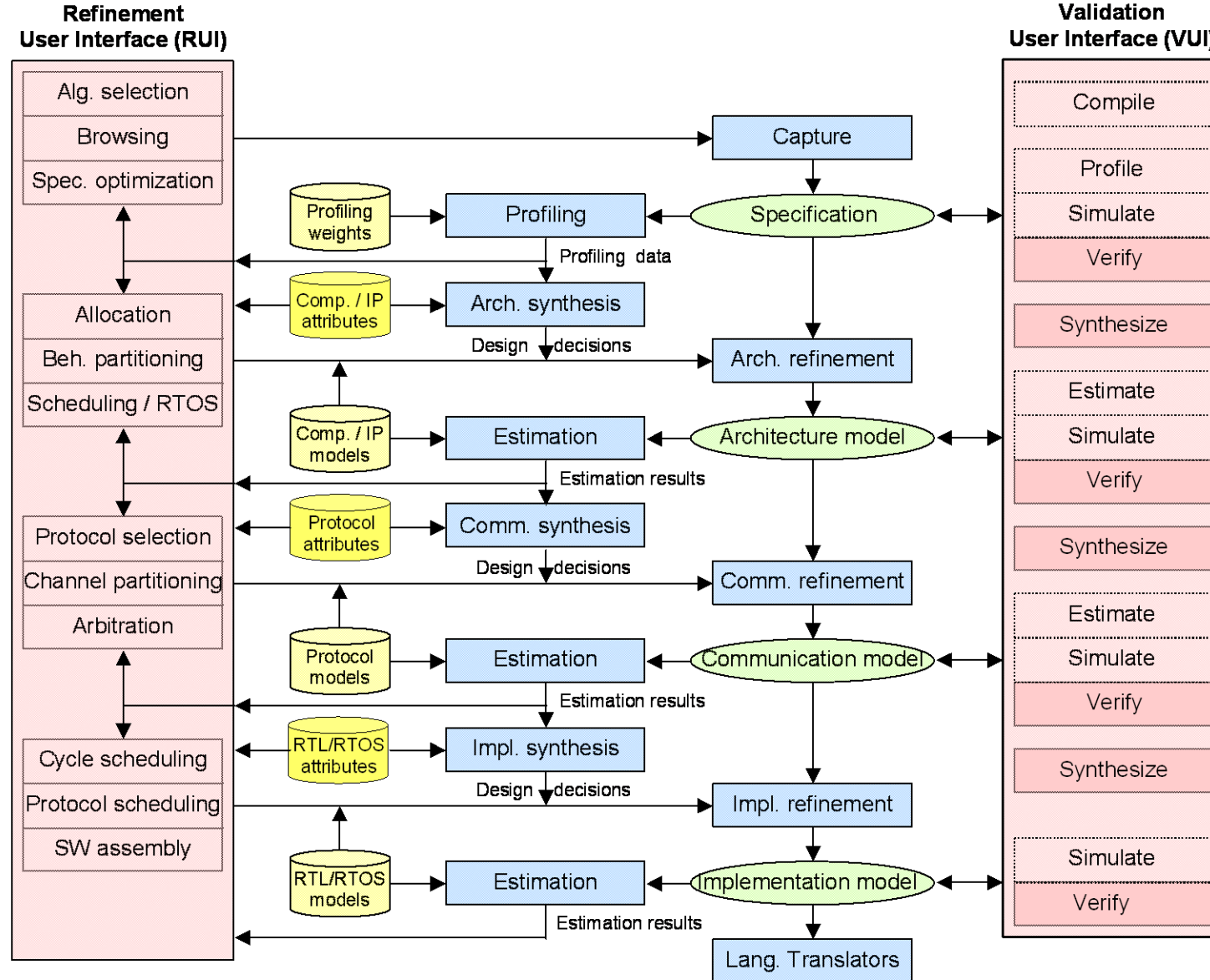


# Synthesis

- Set of models
- Sets of design tasks
  - Profile
  - Explore
  - Select components / connections
  - Map behaviors / channels
  - Schedule behaviors/channels
  -
- Each design decision => model transformation
- Detailing is a sequence of design decisions
- Refinement is a sequence of transformations
- Synthesis = detailing + refinement
- Challenge: define the sequence of design decisions and transformations



# SCE Experiment is Very Positive



Source: <http://www.cecs.uci.edu/~cad/sce.html>

# Conclusion

- Computation and communication objects of TLM are connected through abstract data types
- TLM enables modeling each component independently at different abstraction levels
- The major challenge is to define necessary and sufficient set of models for a design flow
- The next major challenge is to define model algebra and corresponding methodology for each application such that algorithms and tools for modeling, verification, exploration, synthesis and test can be easily developed
- **Opportunities are bigger than anything seen before**

