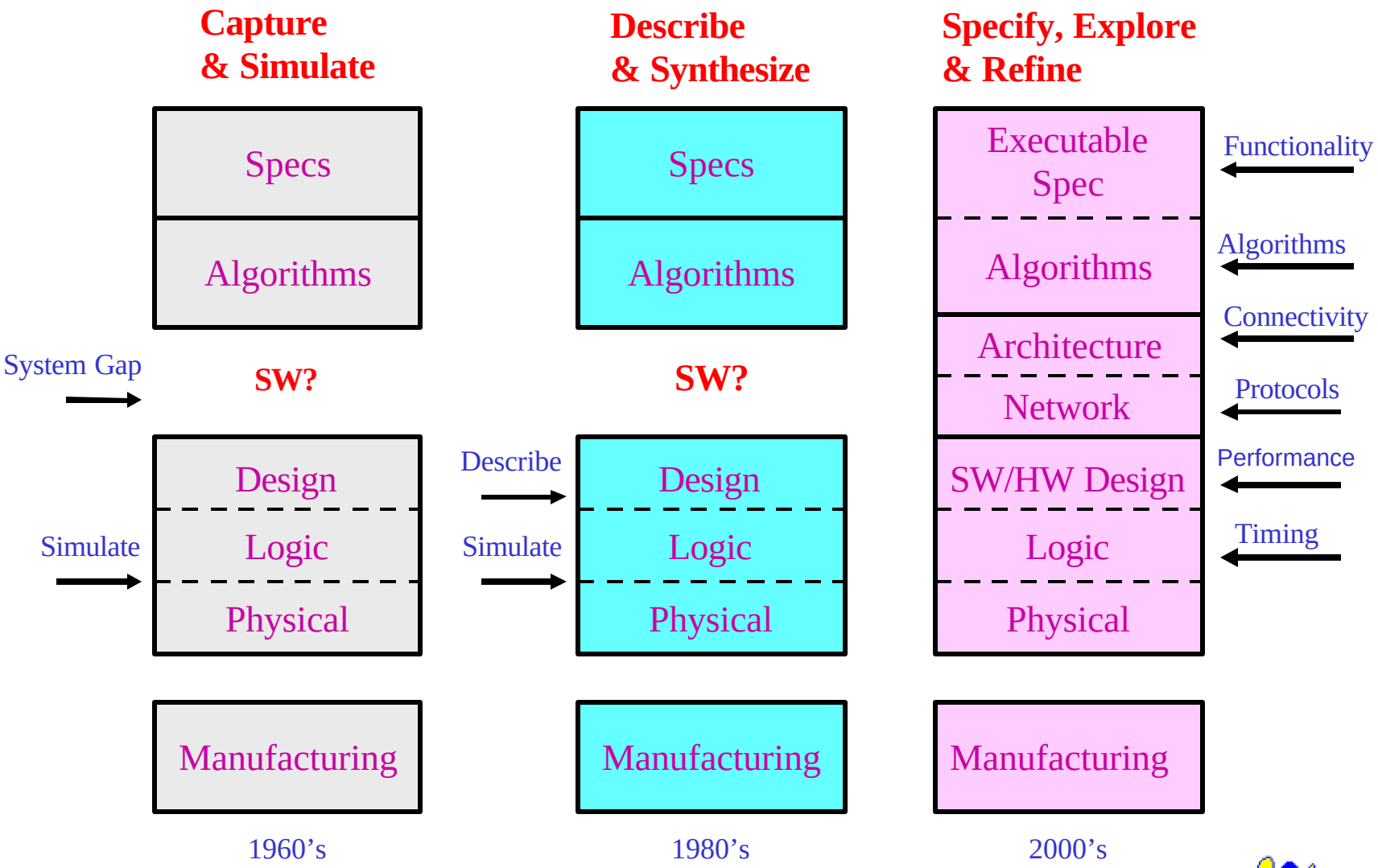
SoC Design for the New Millennium

Daniel D. Gajski
Center for Embedded Computer Systems
University of California, Irvine
www.cecs.uci.edu/~gajski

Outline

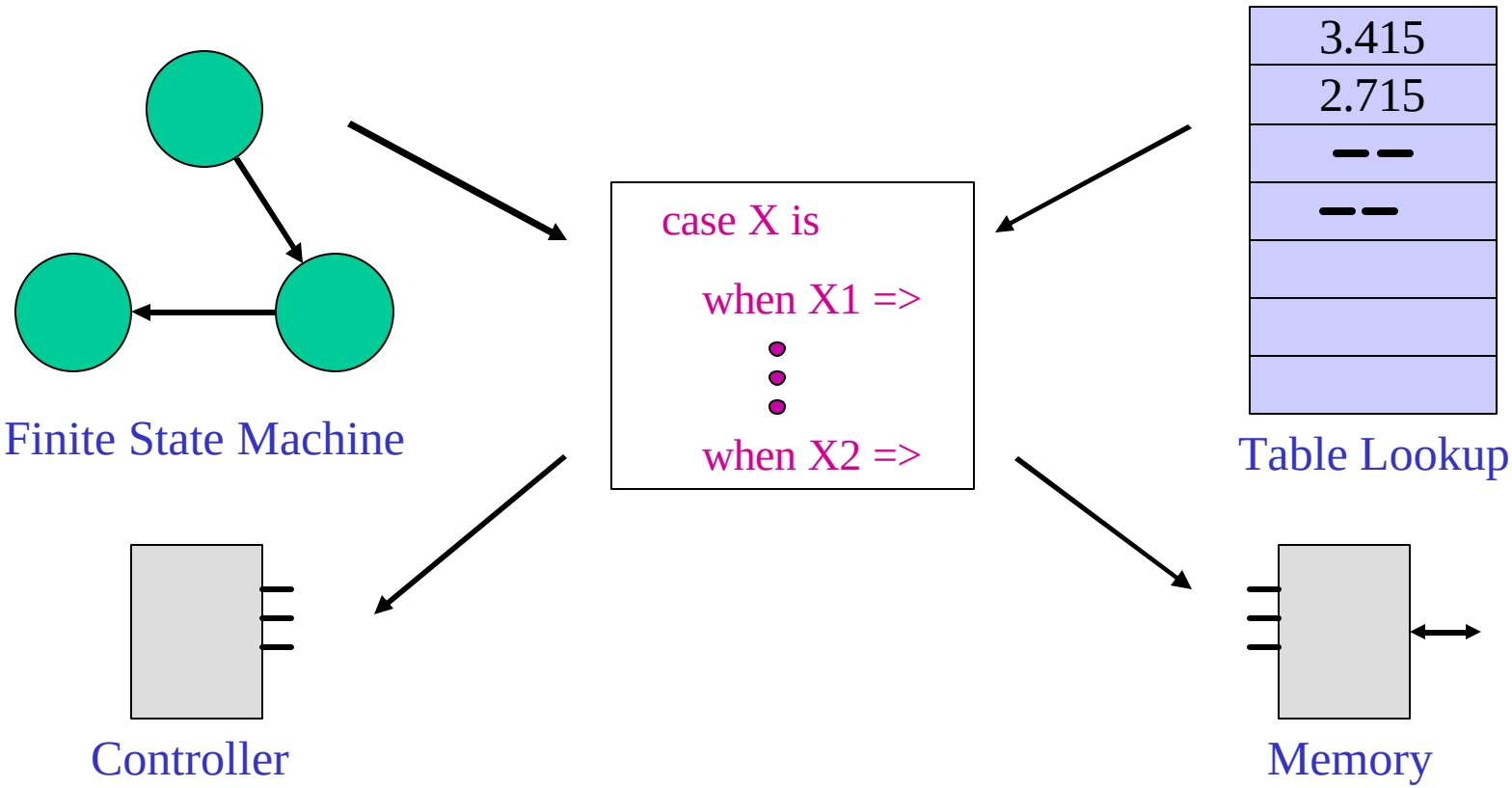
- System gap
- Design flow
- Model algebra
- System environment
- Vision
- Conclusion

Past, Present and Future Design Flow



Missing Semantics: Simulation Dominated Design Flow

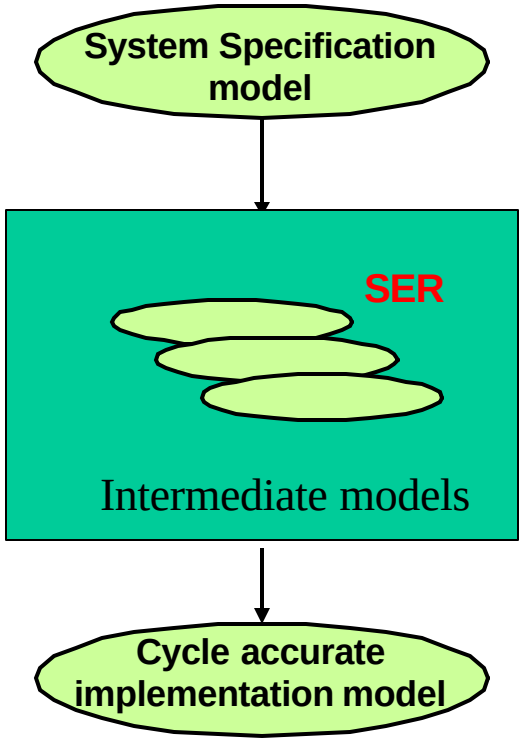
- **Simulatable but not synthesizable/verifiable**



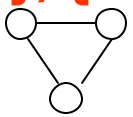
(VHDL, Verilog, C, SystemC, …)

Principles of Design Methodology

- **Well defined specification**
  - Complete
  - Just another model
- **Well defined system models**
  - Several possible models
  - Well defined semantics
  - Formal representation
- **Design synthesis**
  - Design decisions => transformations
  - Automatic model generation
- **Model verification**
  - Formally defined transformations
  - Provable equivalence



Modeling as Algebra

- **Algebra = < {objects}, {operations} >**
  Set of all possible expressions [ex: a * (b + c)]
- **Model Algebra = < {objects}, {compositions} >**
  Set of all possible models [ex: ]
- **Model Transformation** *t* is a change in objects or composition
- **Model refinement** is an ordered set of transformations
- **Methodology** is a sequence of models and corresponding refinements

Model Definition

- **Model Algebra = < {objects}, {composition rules} >**
- **Objects**
  - Behaviors (representing tasks / computations / functions)
  - Channels (representing communication between behaviors)
- **Composition rules**
  - Sequential, parallel, pipelined, FSM
  - Behavior composition creates hierarchy
  - Behavior composition also creates execution order
- **Relations**
  - Relations define a specific model [a*((b+c)*(b-c))]
  - Relations define the model structure (composition and connectivity)
  - Relations **?** {objects} x {compositions}

Model Transformations

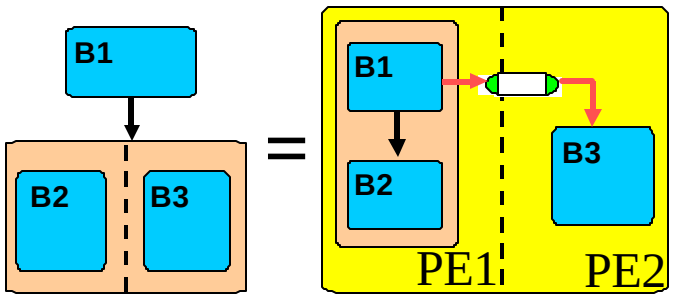
(Rearrange and Replace)

- **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 …**

a*(b+c) = a*b + a*c

Distributivity of multiplication over addition

analogous to……



Distribution of behaviors (tasks) over components

Model Refinement

- **Definition**
  - Ordered set of transformations $< t_m, \ldots, t_2, t_1 >$ is a refinement
    - *model B* $= t_m(\ldots (t_2(t_1($ *model A* $))) \ldots)$
- **Derives a more detailed model from an abstract one**
  - Specific sequence for each model refinement
  - Not all sequences are relevant
- **Automatic model generation**
  - Each transformation can be automated
  - Each refined model can be automatically generated
- **Equivalence verification**
  - Each transformation maintains functional equivalence
  - Refinement is thus correct by construction

Refinement based Methodology

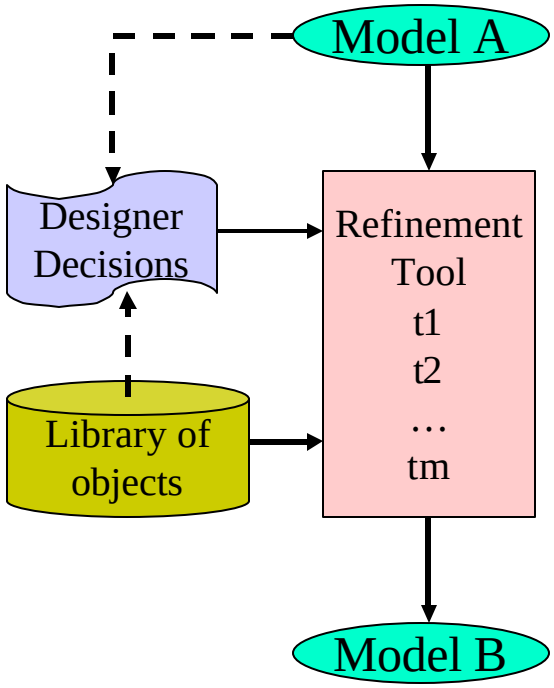
- **Refinement based system level methodology**
  - Methodology := < {models}, {refinements} >
- **Each transformation is uniquely defined**
- **Each transformation can be automatic**
- **No need for designer modeling**
- **No need for modeling languages**
- **Designers only make design decisions**

System Synthesis through Refinement

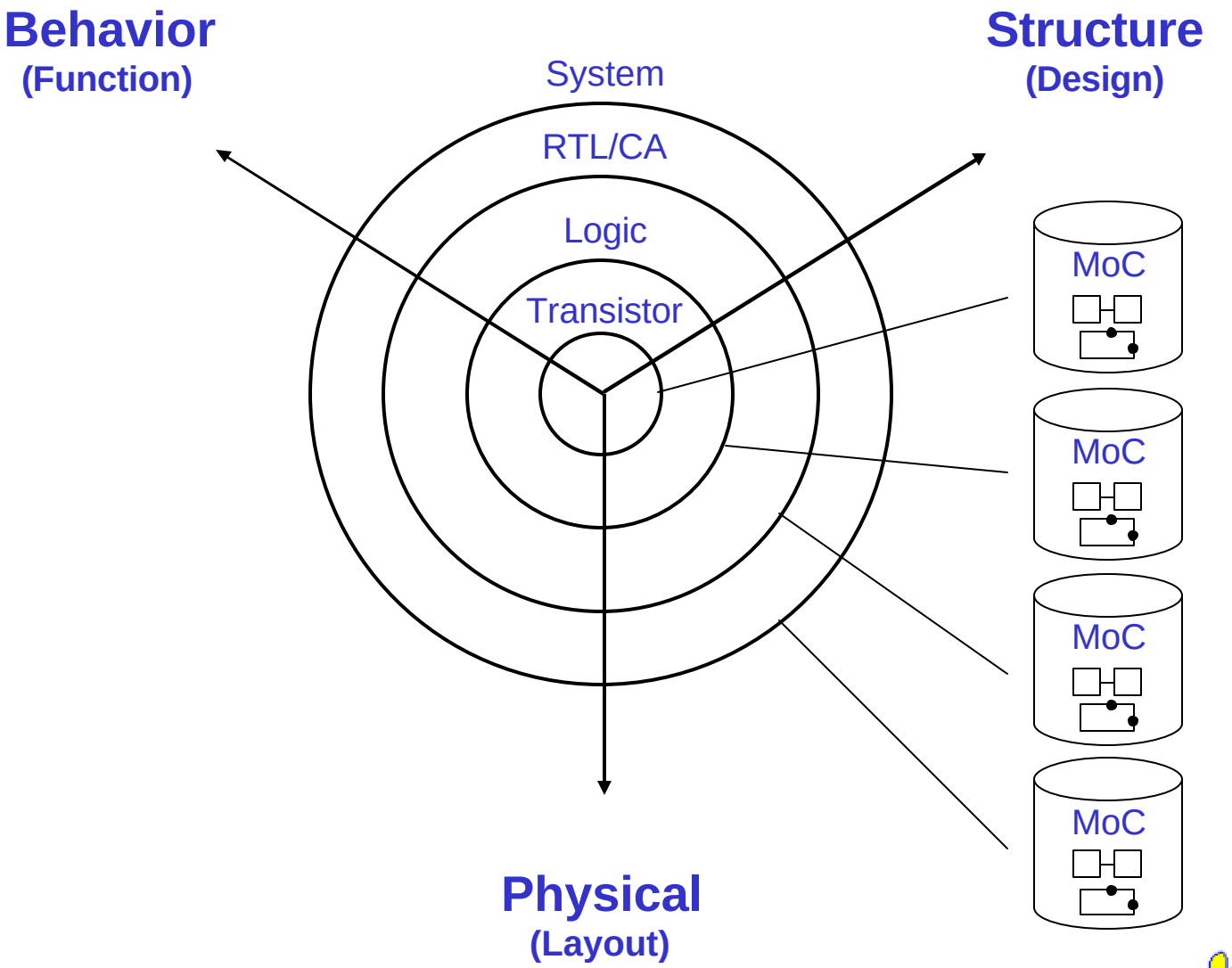
- **Set of models**
- **Each design decisions => one model transformation**
  - Select components / connections/ IPs
  - Map behaviors / channels
  - Schedule behaviors/channels
  - Add new objects (IF, IC, Arbiters, …)
  - Synchronize to preserve equivalence
- **Design is a sequence of decisions**
- **Refinement is a sequence of model transformations**
- **Synthesis = design + refinement**
- **Challenge: define the design and refinement sequences**

System Verification through Refinement

- **Set of models**
- **Transformations preserve equivalence**
  - Same partial order of tasks
  - Same input/output data for each task
  - Same partial order of data transactions
  - Equivalent replacements
- **All refined models will be "equivalent" to input model**
  - Still need to verify
    - First model
    - Correctness of replacements

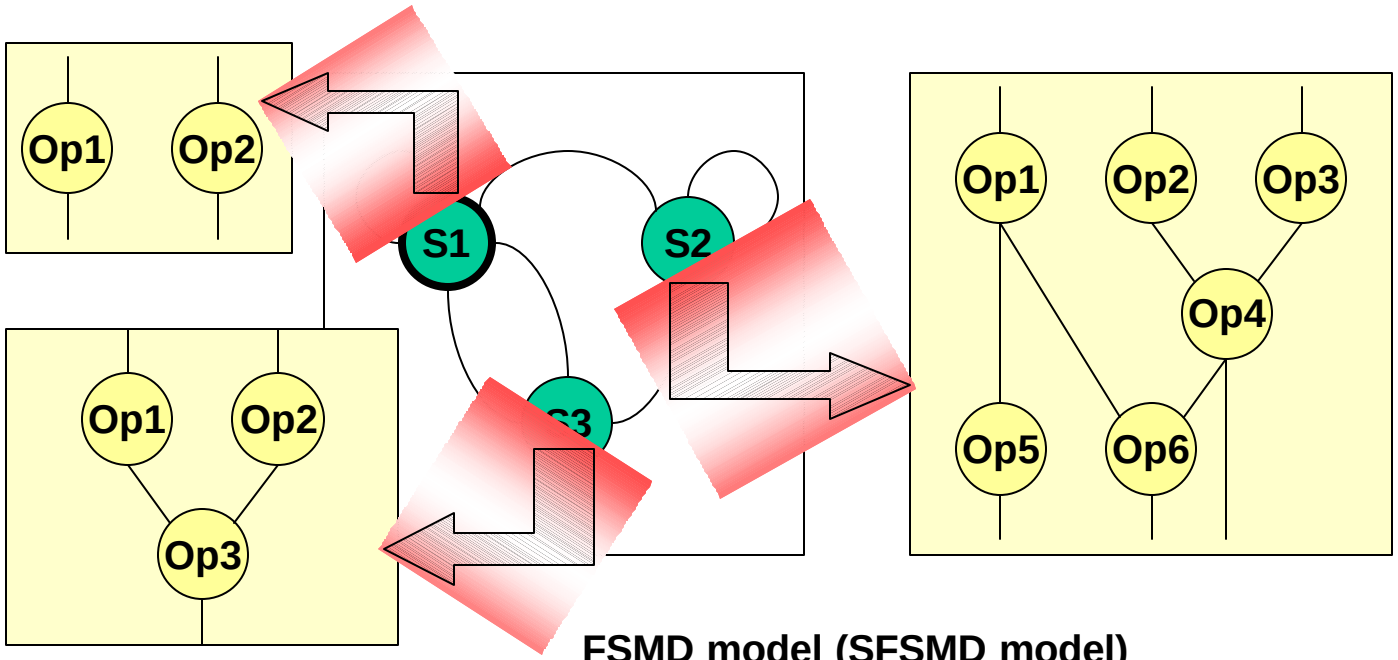


Y Chart



RTL Functional Model

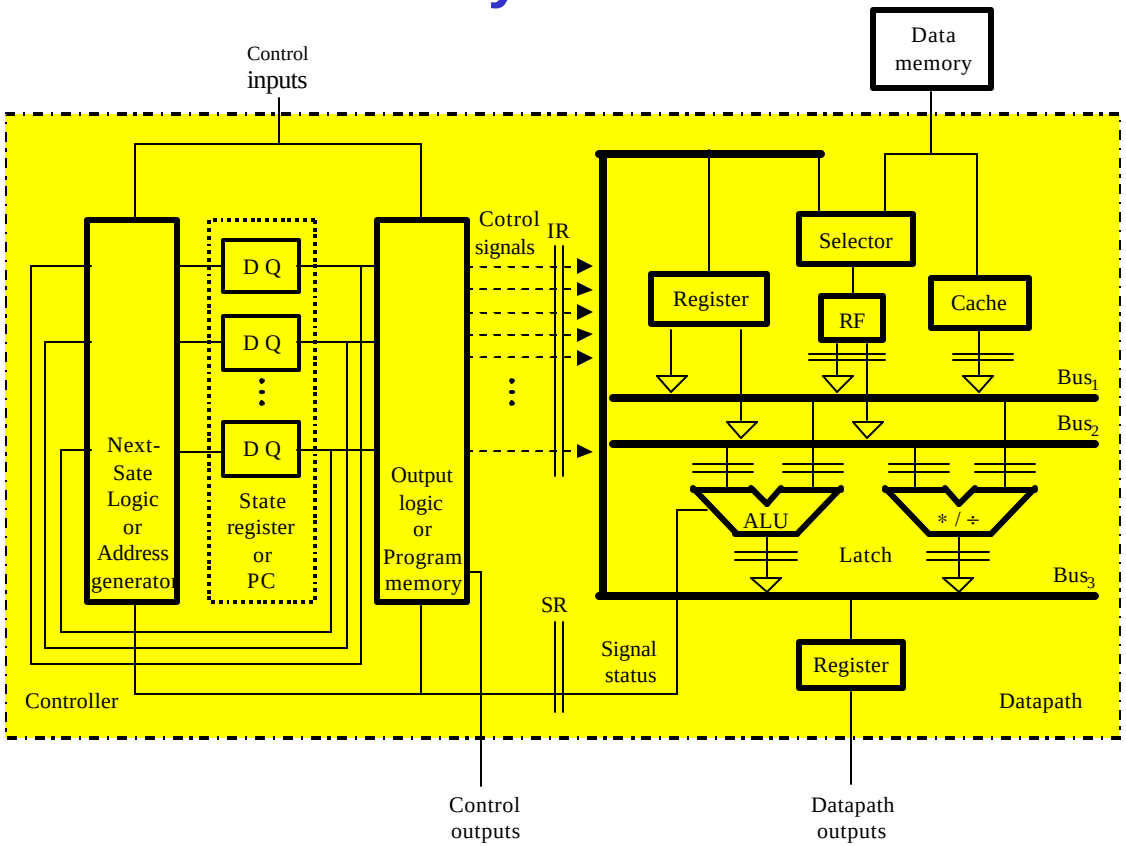
- **Finite State Machine with Data (FSMD)**
  - Combined model for control and computation
    - FSMD = FSM + DFG
  - Implementation: controller plus datapath



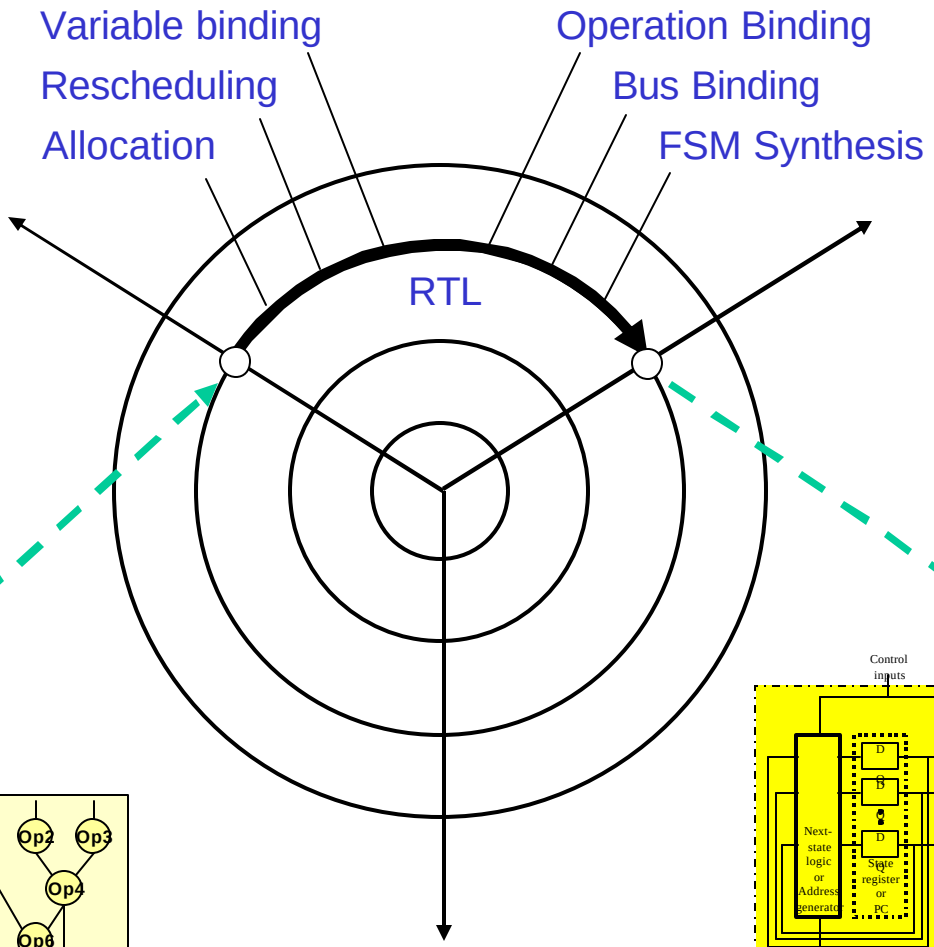
FSMD model (SFSMD model)

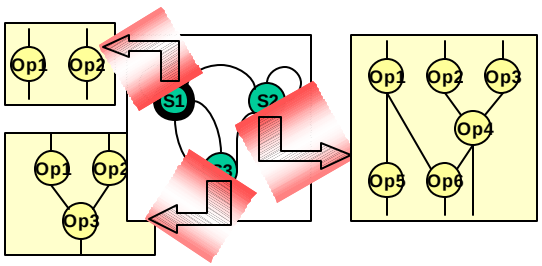
RTL Structural Model

- Multi-pipelined data path
- Hardwired or programmable controller
- RISC or NISC style

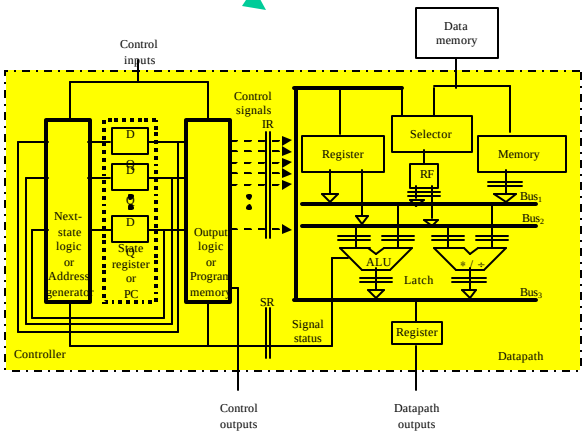


RTL Synthesis





FSMD model

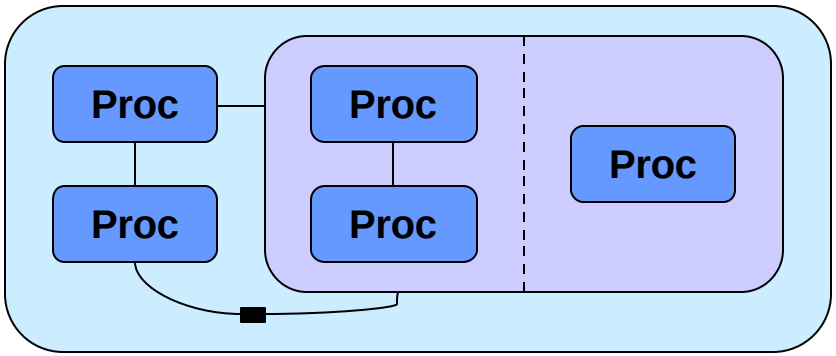


RTL/NISC structural model

System Functional Model

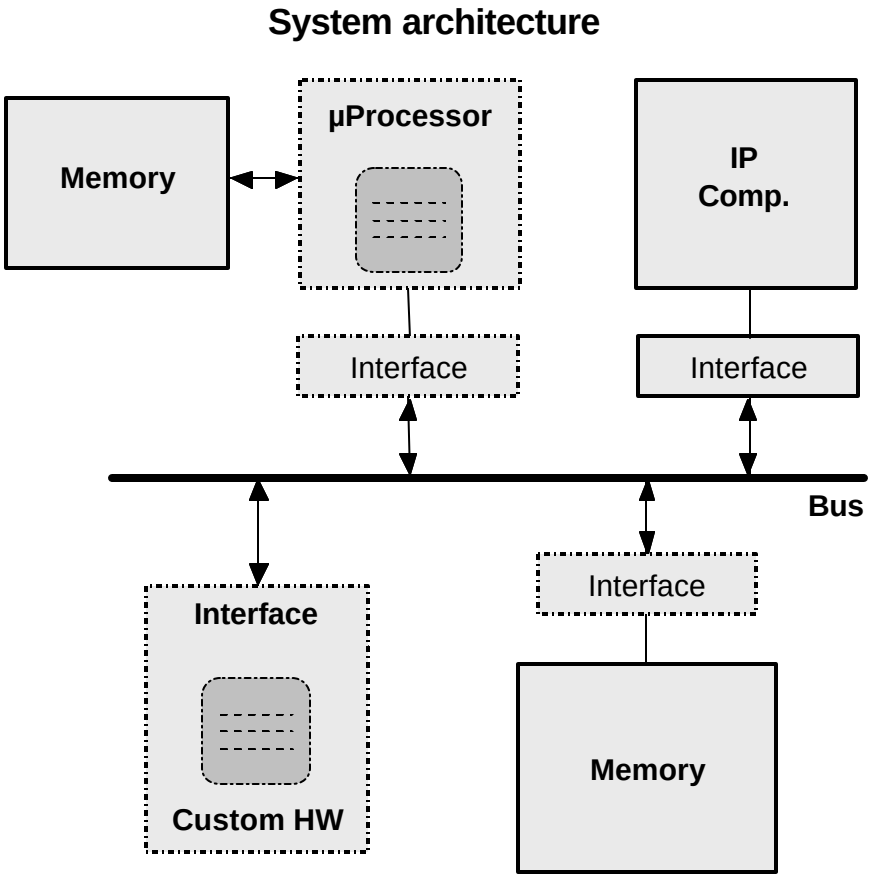
- **Program State Machine**
  - States described by procedures in a programming language
  - Example: SpecC! (SystemC subset?!)

PSM model

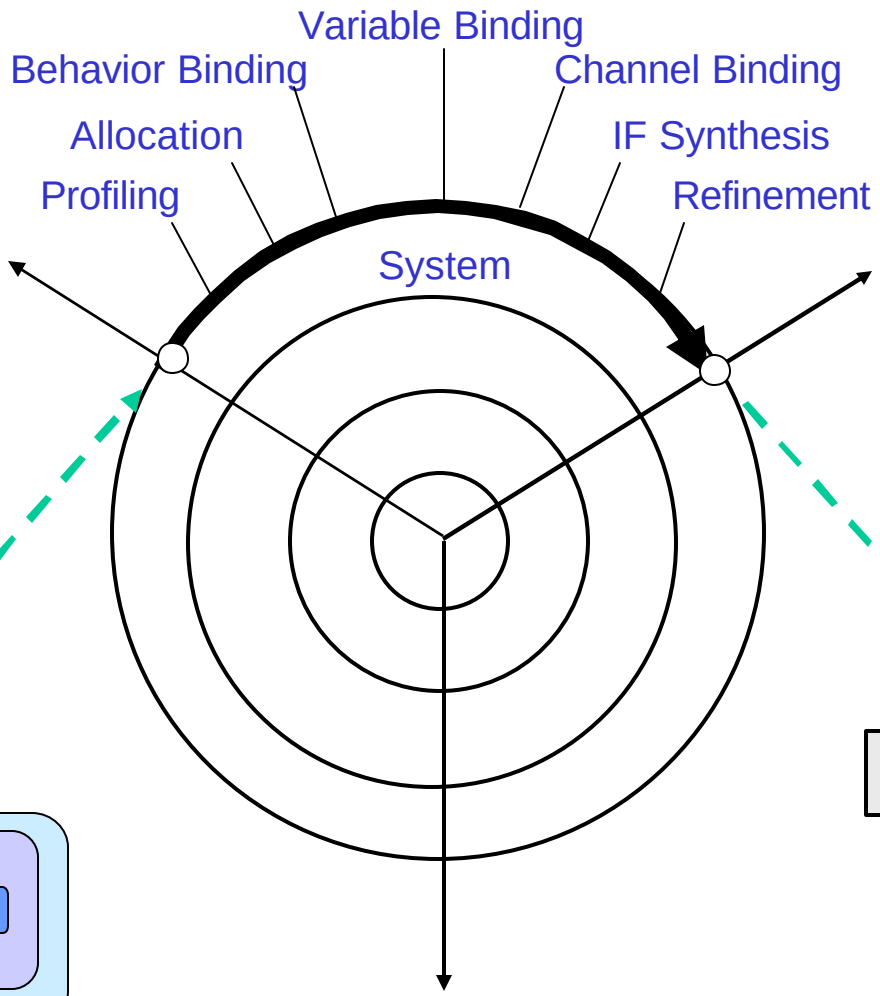


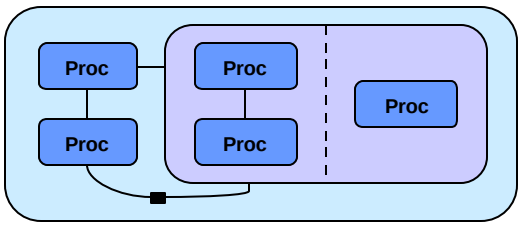
System Structural Model

- Arbitrary components (Proc., IP, Memory)
- Arbitrary connectivity (buses, protocols)



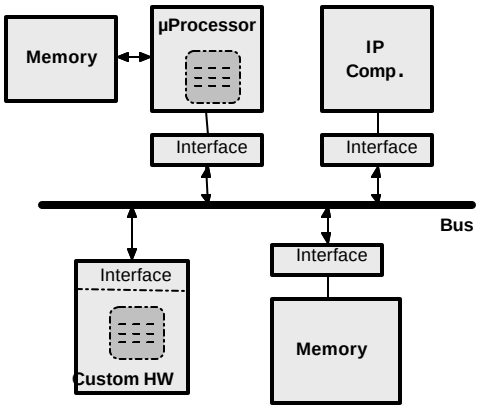
System Synthesis



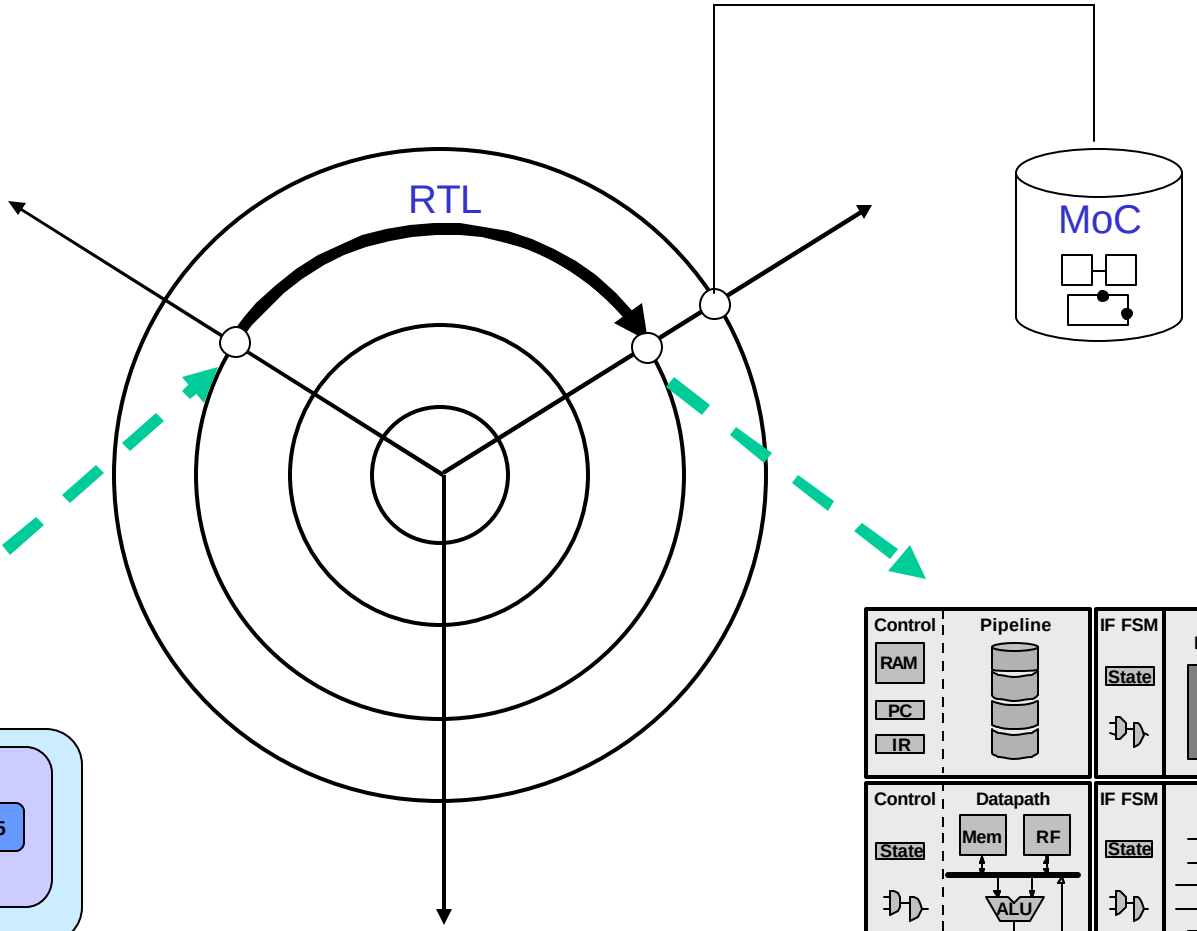


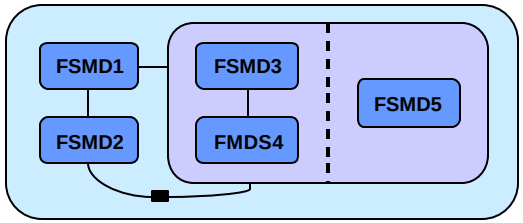
PSM model

System architecture



System Synthesis (continued)



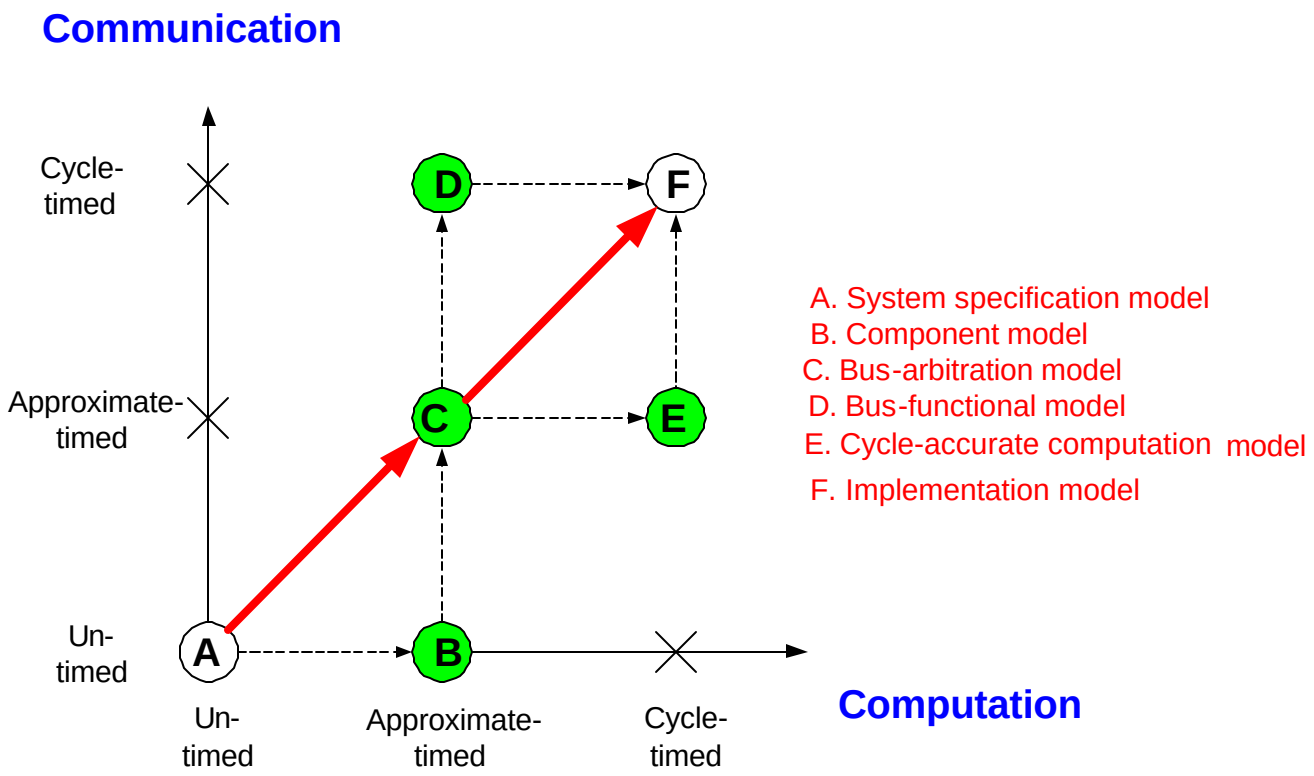


HCFSMD model

RTL/IS Implementation
+ results

System Level Models

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

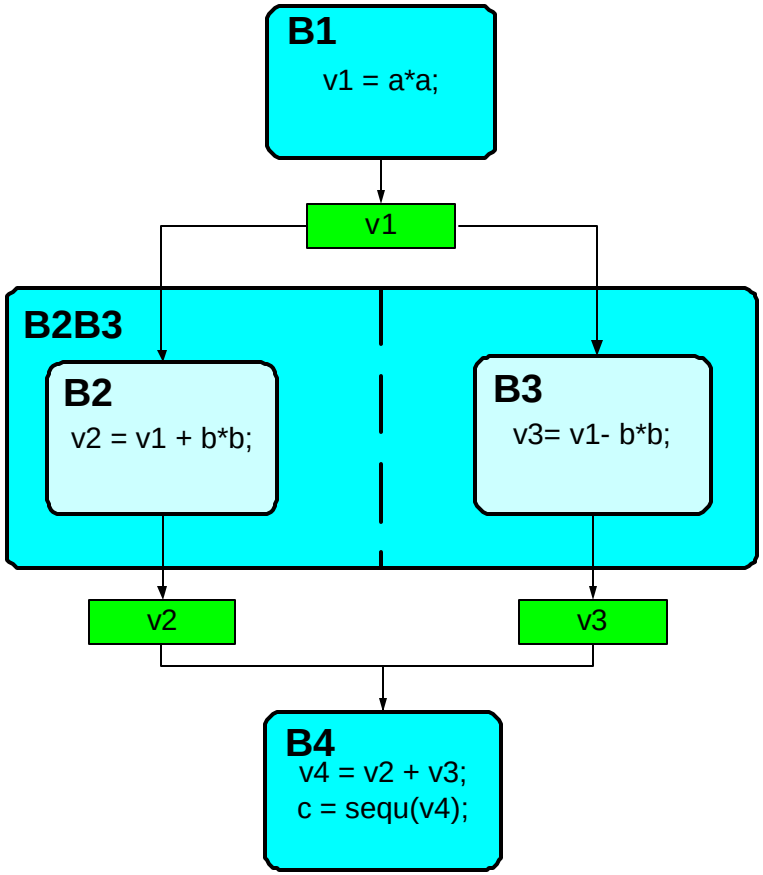


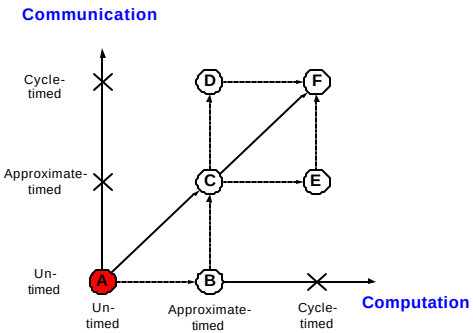
Source: Lukai Cai, D. Gajski. "Transaction level modeling: An overview", ISSS 2003

A: “Specification Model”

Objects

- Computation
  - -Behaviors
- Communication
  - -Variables





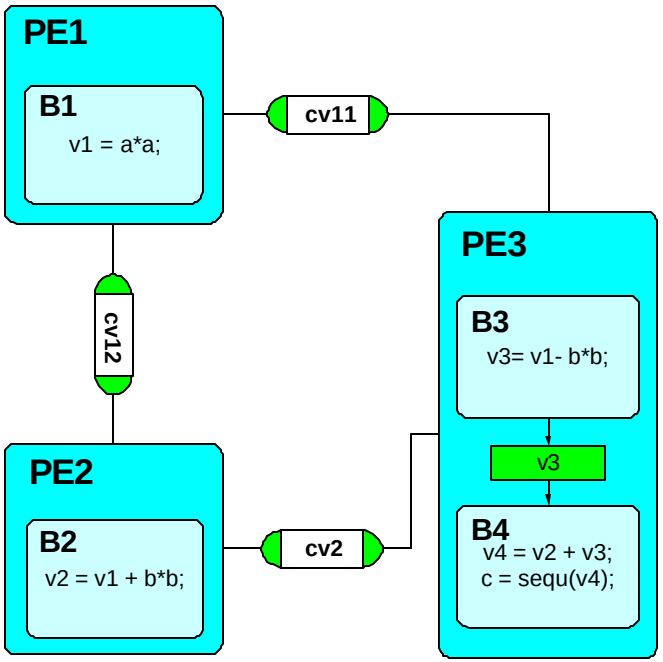
Composition

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

B: “Component-Assembly Model”

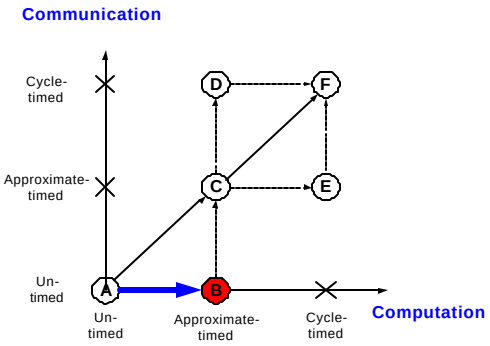
Objects

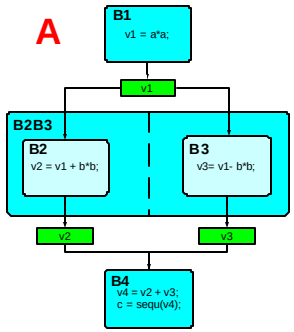
- Computation
  - Proc
  - IPs
  - Memories
- Communication
  - Variable channels



Composition

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

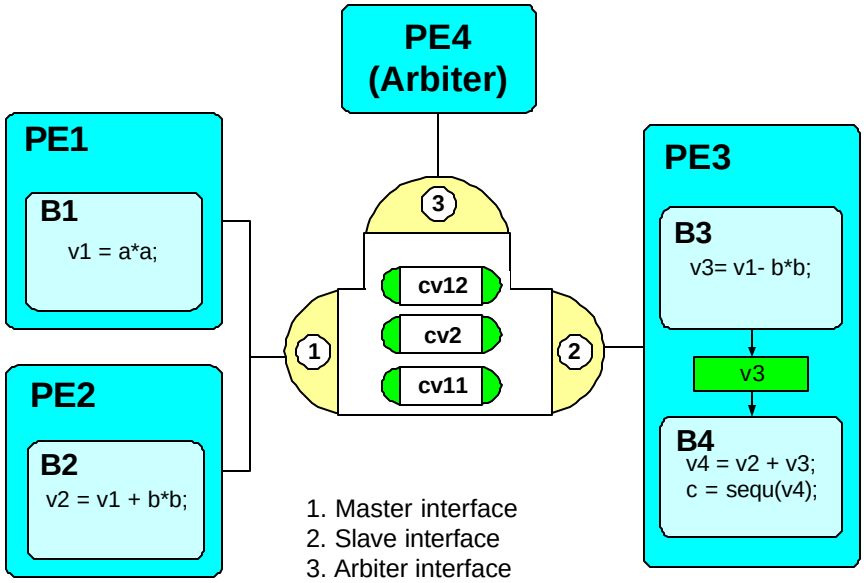




C: “Bus-Arbitration Model”

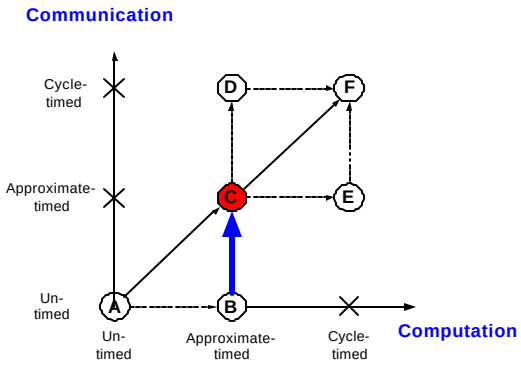
Objects

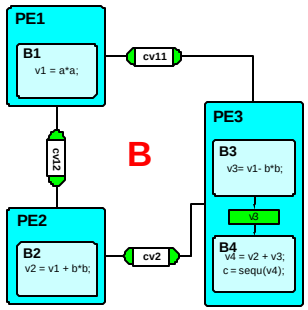
- Computation
  - Proc
  - IPs (Arbiters)
  - Memories
- Communication
  - Abstract bus channels



Composition

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

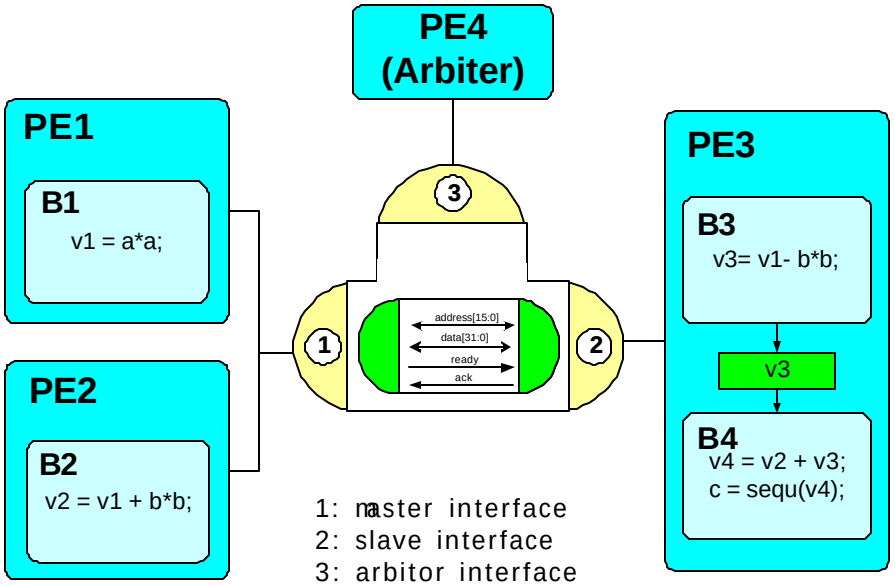




D: “Bus-Functional Model”

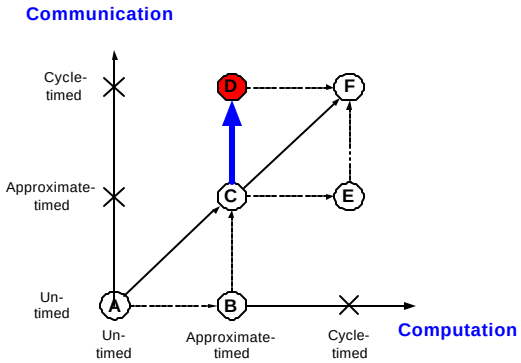
Objects

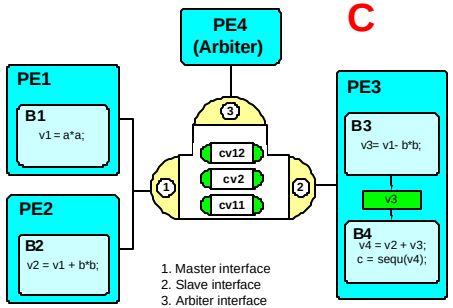
- Computation
  - Proc
  - IPs (Arbiters)
  - Memories
- Communication
  - Protocol bus channels



Composition

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

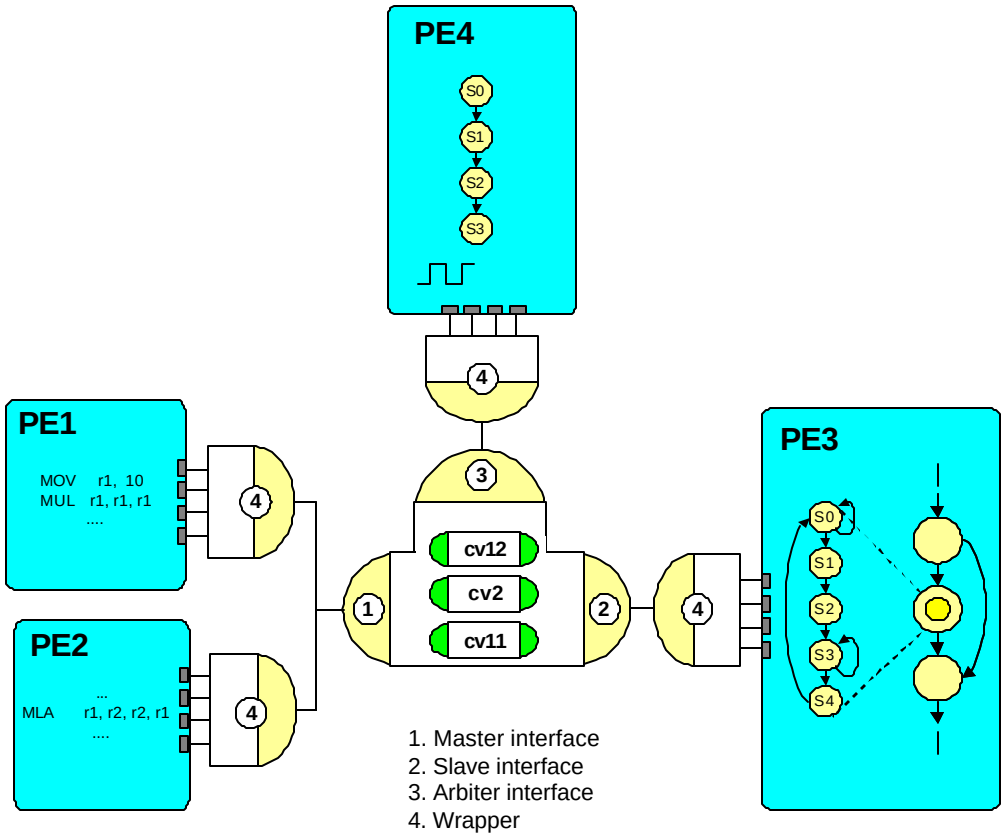


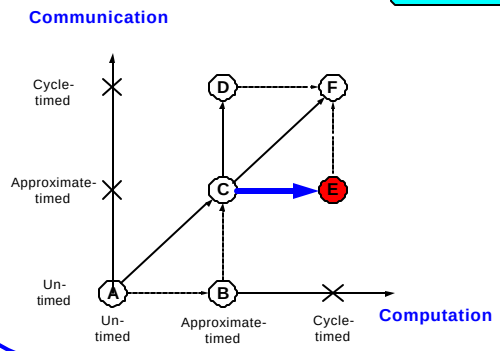


E: “Cycle-Accurate Computation Model”

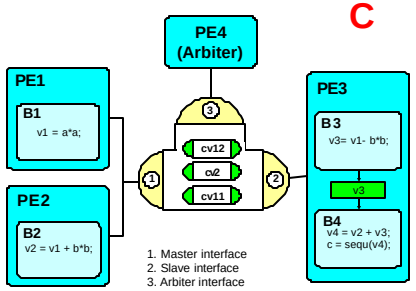
Objects

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





C



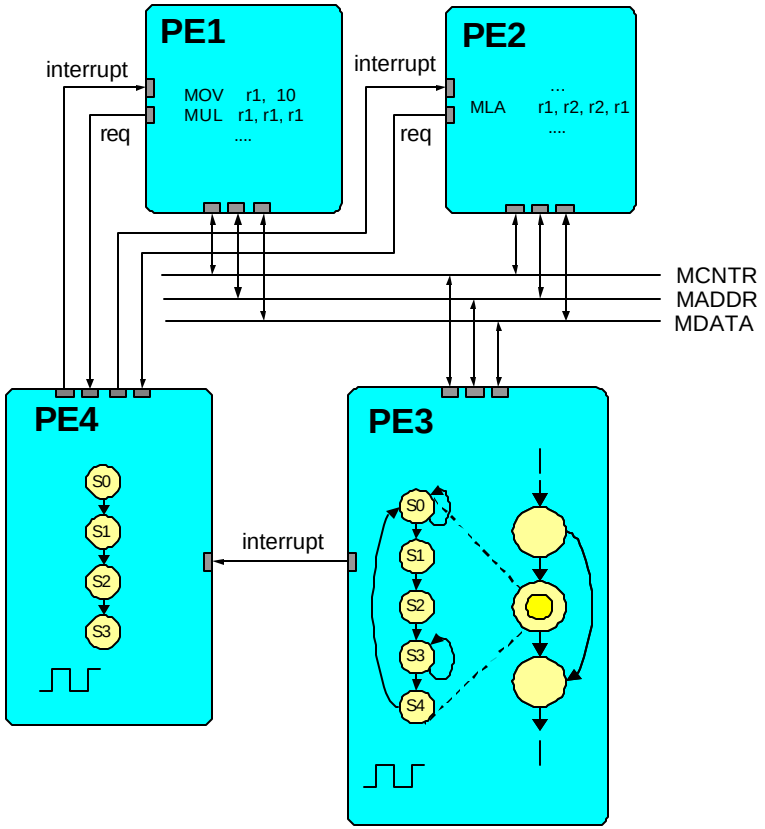
Composition

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

F: “Implementation Model”

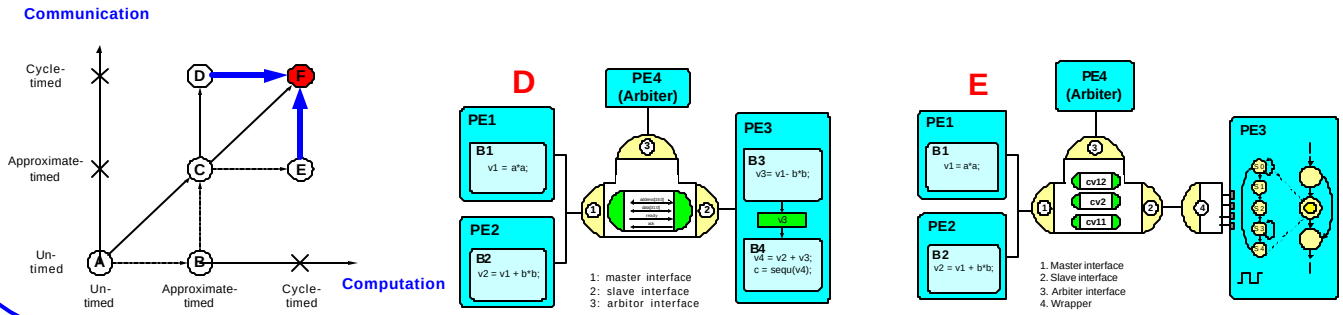
Objects

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

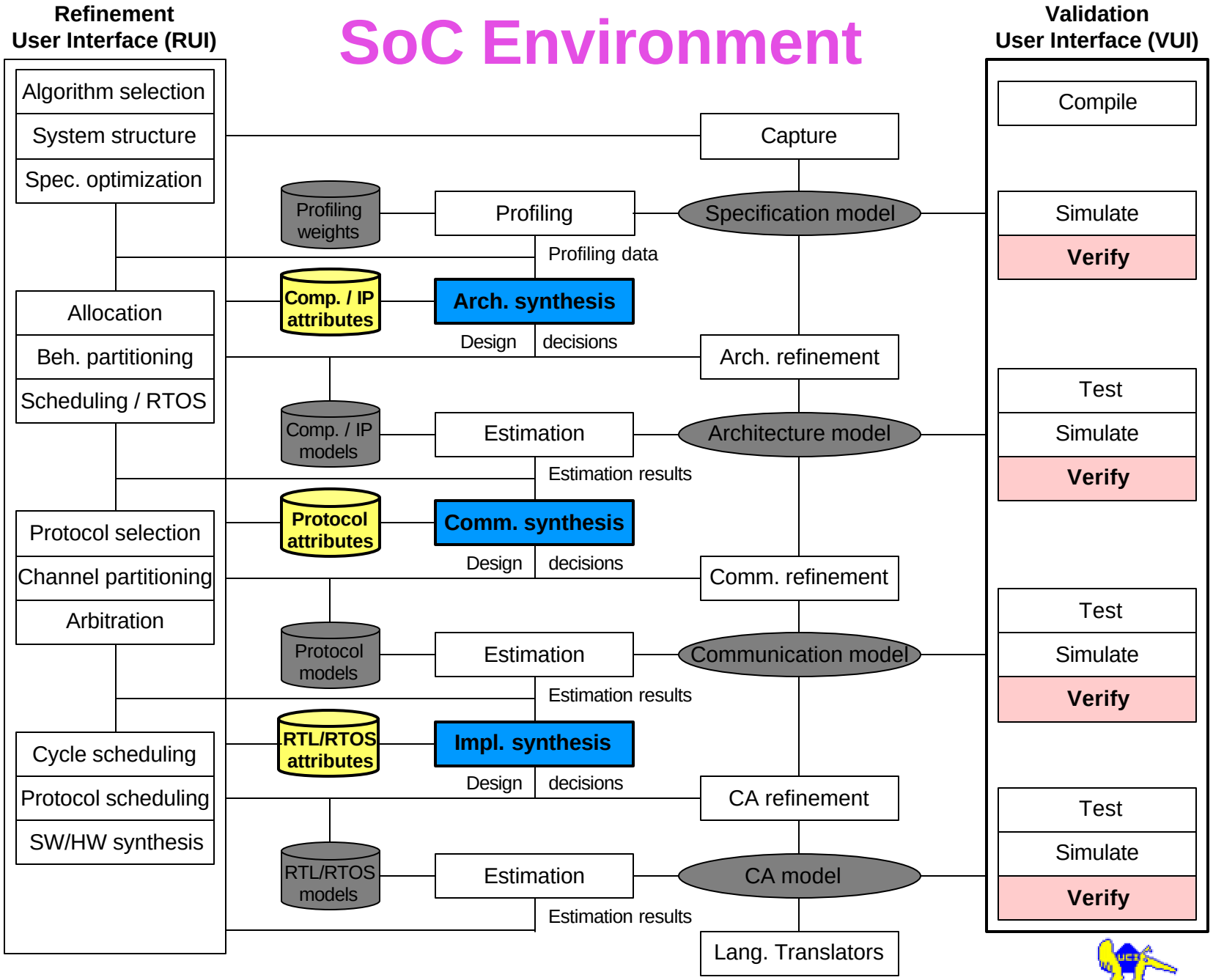


Composition

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



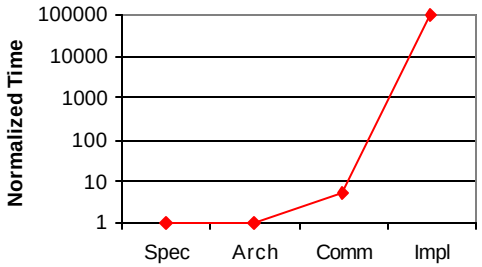
SoC Environment



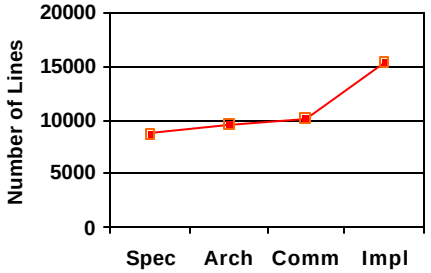
Vocoder Results

- **Experiment on GSM Vocoder design (10K lines of code)**

Simulation Speed



Code Size



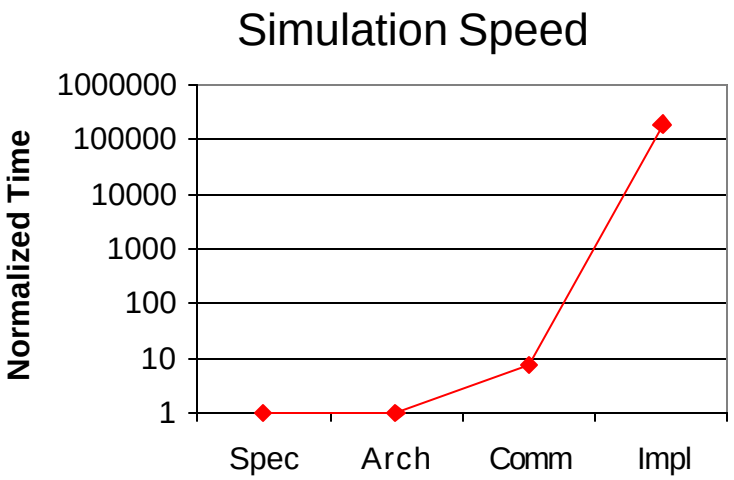
Refinement Effort

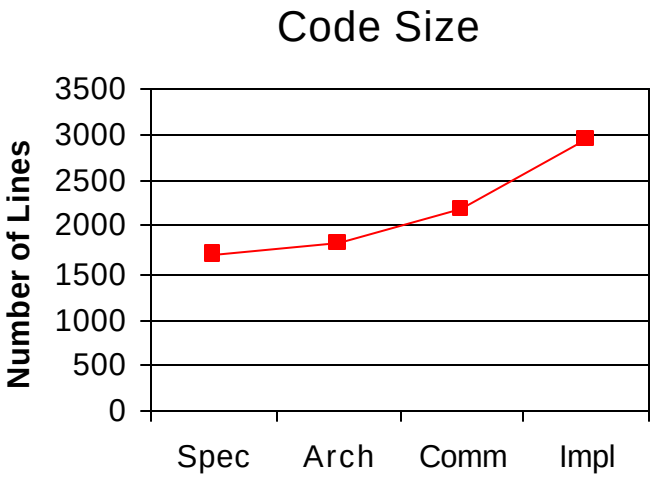
	Modified lines	Manual	Automated User / Refine
Spec→Arch	3,275	3~4 mons.	15 mins / <1 min
Arch→Comm	914	1~2 mons.	5 mins/ <0.5 min
Comm→Impl	6,146	5~6 mons	30 mins / <2 mins
Total	10,355	9~12 mons	50 mins / <4 mins

- **Conclusion**
  - Productivity gain >2,000X for industrial strength designs
  - Compare 9-12 months (manual refinement) vs. 50+4 minutes (user decisions + automatic refinement)
  - Enables extensive design exploration (60/day)

JPEG Results

- **Simulation Speed & Code size**





- **Refinement Effort**

Refinement Effort

	Modified lines	Manual	Automated User / Refine
Spec® Arch	751	1~2 mons.	5 mins / <0.5 min
Arch® Comm	492	~1 mons.	3 mins/ <0.5 min
Comm® Impl	1,278	3~4 mons	20 mins / <1 mins
Total	2,521	5~7 mons	28 mins <2mins

- **Compare 5-7months (manual refinement) vs. 28 minutes (user decisions) + 2 minutes (automatic refinement)**

Conclusion

- New design flow paradigm based on SER methodology
- Semantics before syntax (language does not matter)
- Model algebra before semantics
- System verification, synthesis, modeling simplified
- SW = HW = computation is the inevitable truth
- SoC environment gives 1000X productivity gain
- Challenge: Define models, refinements, methodology