

Design Methodology for Systems-on-Chip

What is needed and what is not

Daniel D. Gajski

Center for Embedded Computer Systems

University of California, Irvine

www.cecs.uci.edu/~gajski



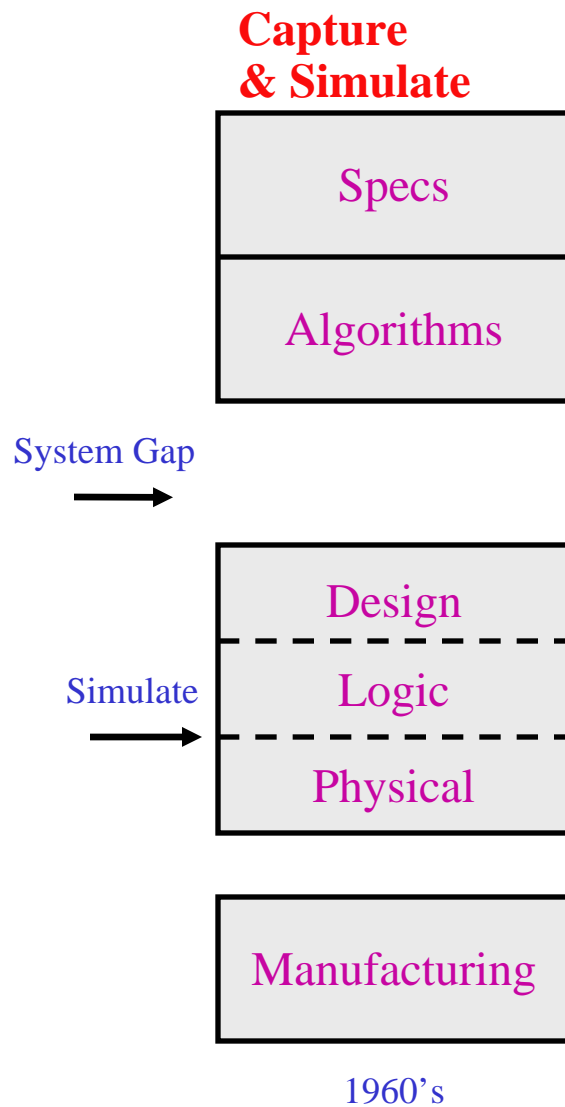
Who we are?

- **Center for Embedded Computer Systems**
(www.cecs.uci.edu)
- **Independent Research Organization in UC**
- **15 faculty and over 60 Ph.D. students**
- **Methodology group:**
 - first system contract 1989 from SRC
 - over 50 person years in system methodology since 1990
 - over 10 Ph.D. in system design flow since 1990
 - published first books on:
 - RTL Sythesis 1992
 - Embedded Systems 1994
 - System Design 2001
 - developed SpecC language
 - leaders in SpecC Open Technology Consortium (www.specc.org)
 - “Inspired” SystemC

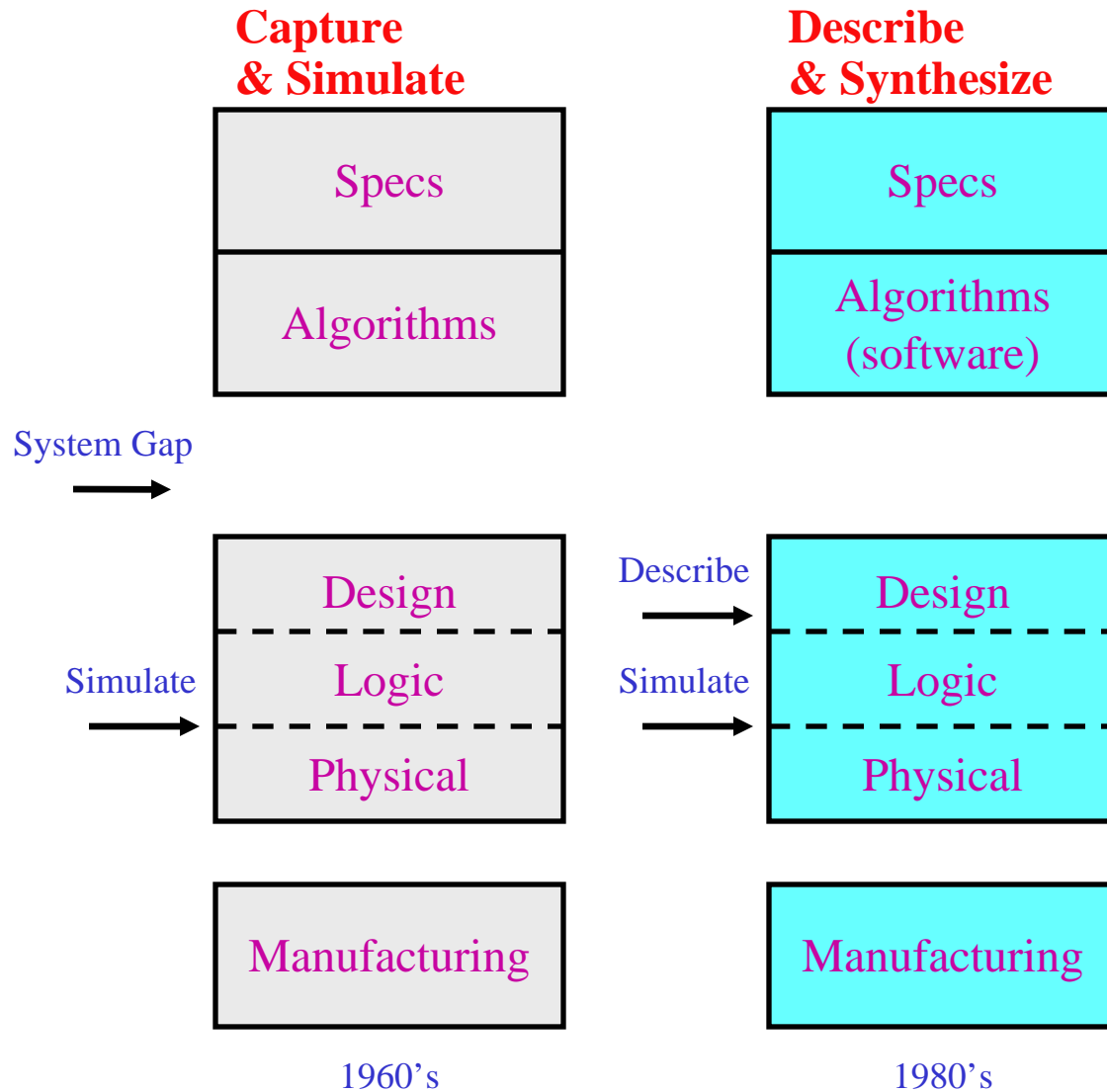
Outline

- **System gap**
- **Semantics, styles and refinements**
- **RTL Semantics**
- **System-Level Semantics**
- **Where are we going?**
- **Conclusion**

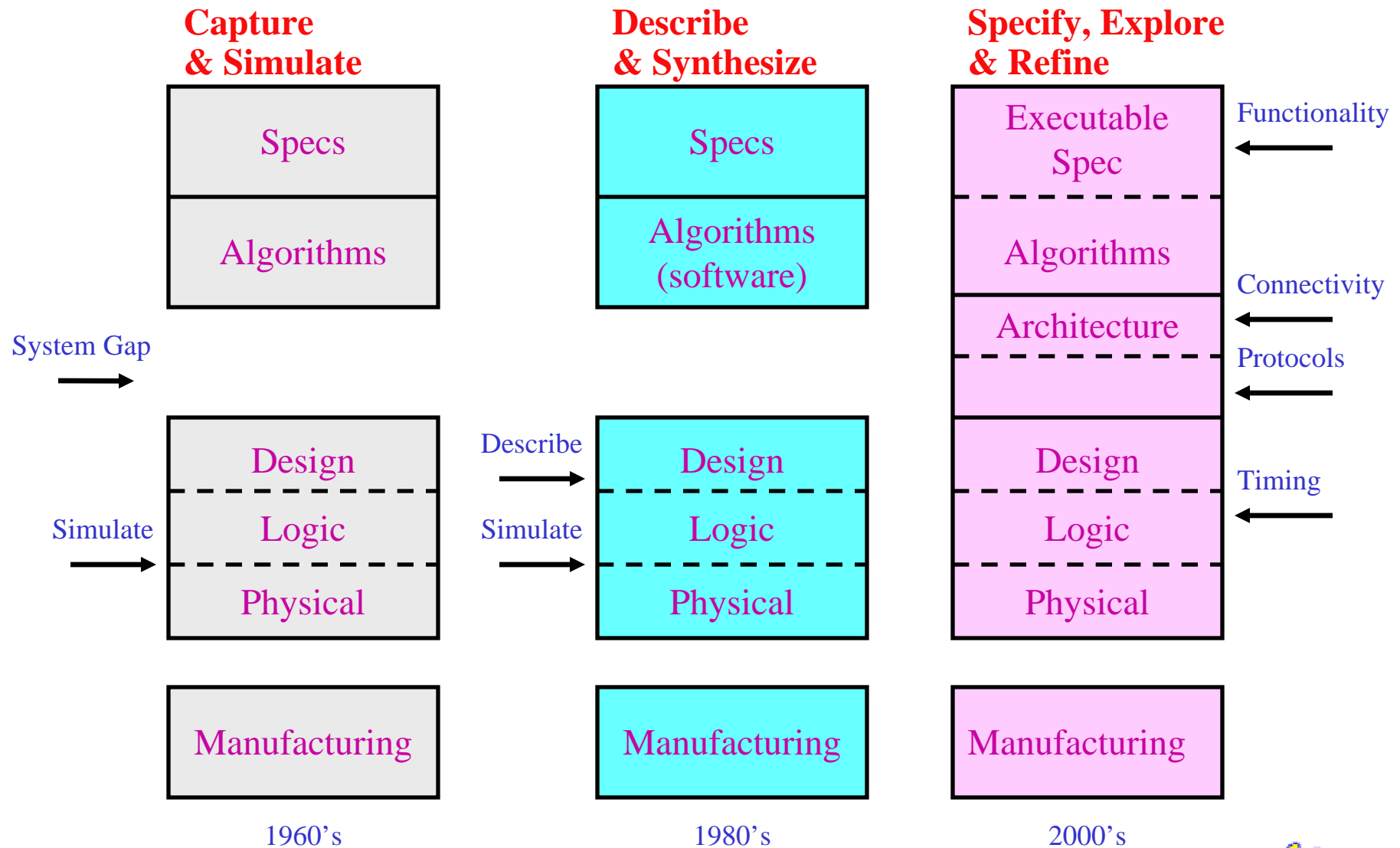
Past Design Flow



Past and Present Design Flow

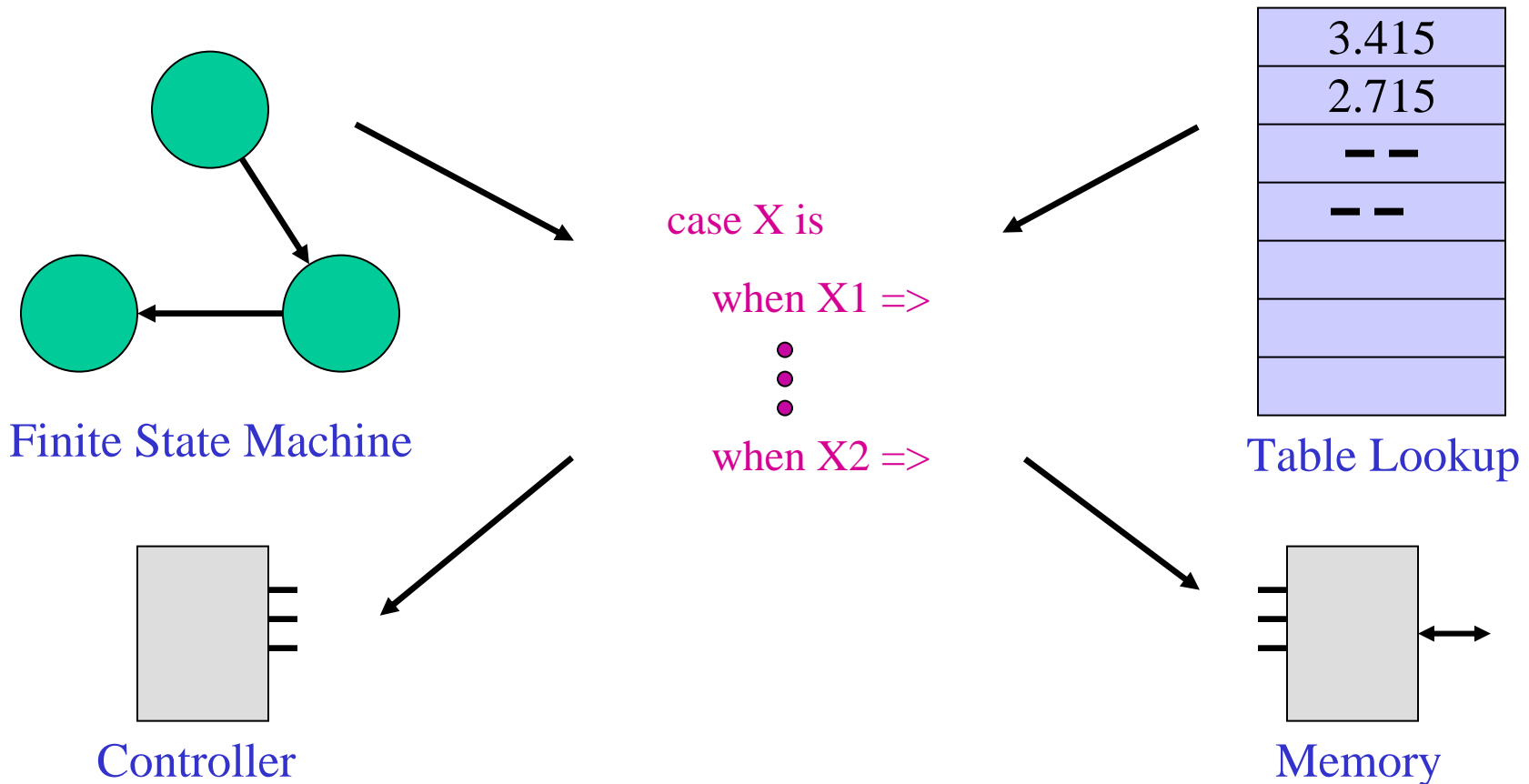


Past, Present and Future Design Flow



Missing Semantics: Simulation Dominated Design Flow

- **Simulatable but not synthesizable**

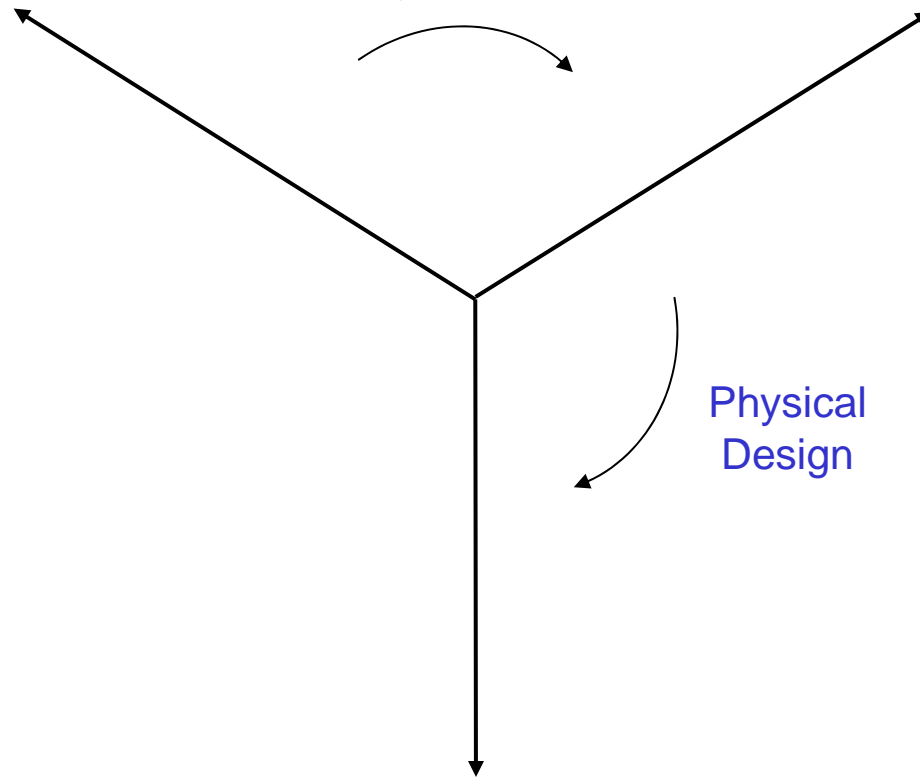


Y Chart

Behavior

Structure

Synthesis

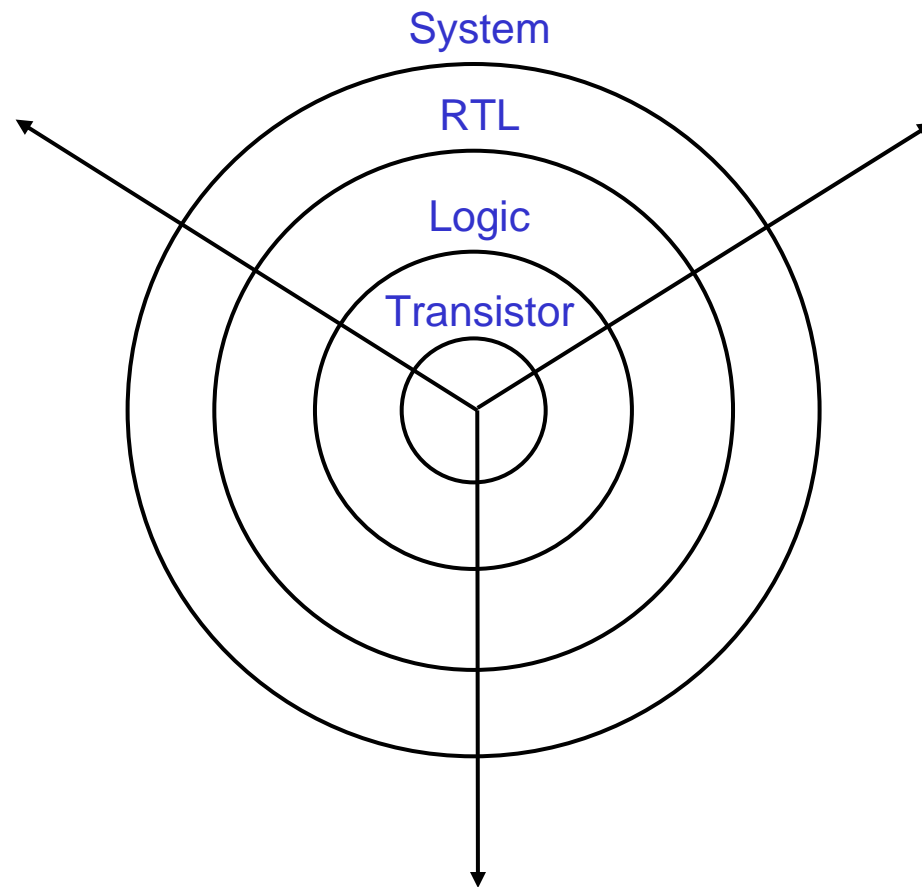


Physical

Y Chart

Behavior

Structure

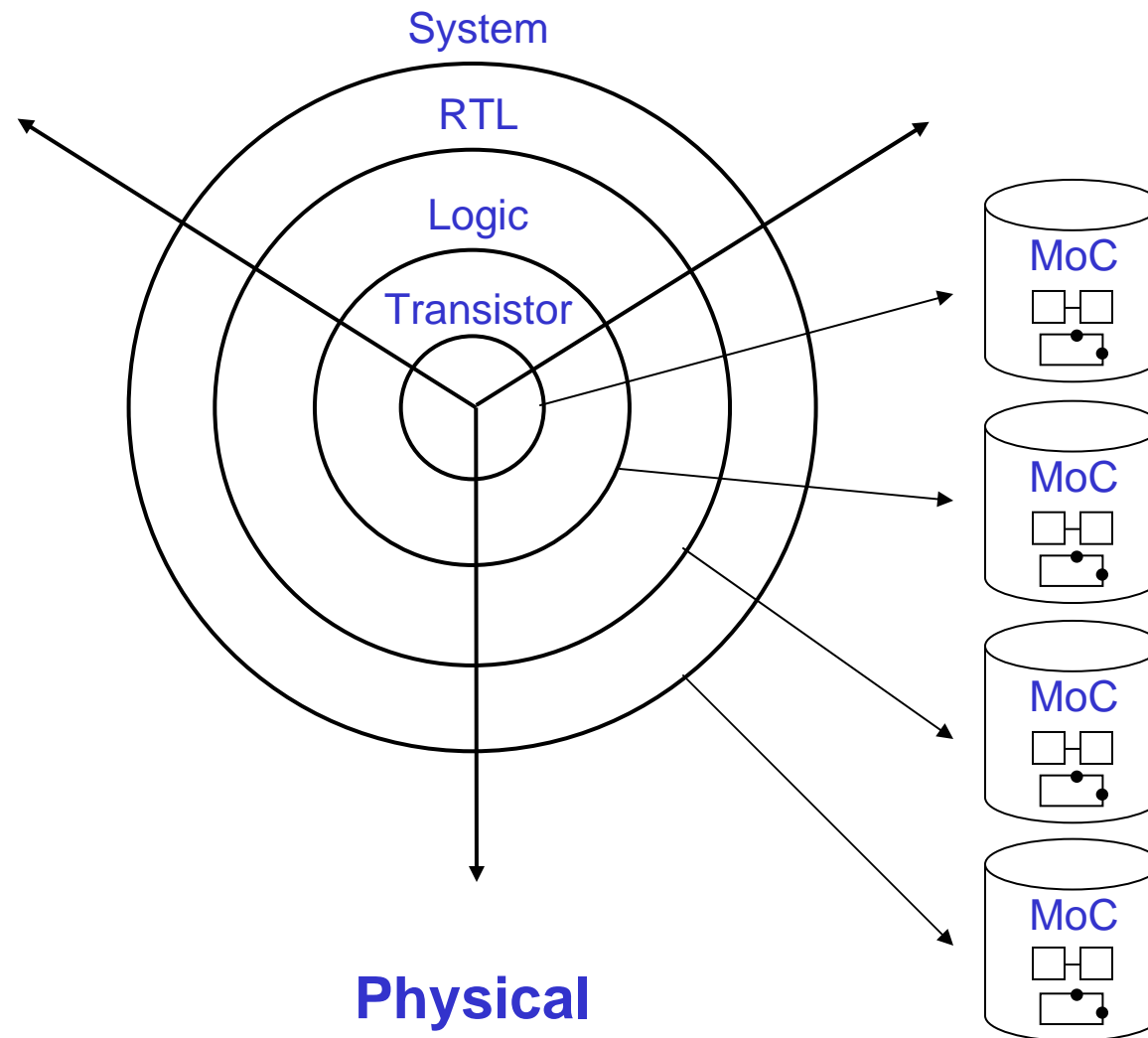


Physical

Y Chart

Behavior

Structure



Physical

Abstraction Algebra

Algebra := $\langle \{\text{objects}\}, \{\text{operations}\} \rangle$

SoC Algebra := $\langle \{\text{models}\}, \{\text{transformations}\} \rangle$

Ordered set of transformations $\langle t_m, \dots, t_2, t_1 \rangle$ is a refinement iff

$$\text{model } B = t_m(\dots (t_2(t_1(\text{model } A))) \dots)$$

Question: $\{ \text{models} \} ? ; \{ \text{transformations} \} ?$

Why Abstraction Algebra?

1. Enabling standard for ESDA
2. Discover truth behind system-level myths
3. Define system-level field (abstract semantics)
4. Introduce interoperability
5. Identify system-level methodology
6. Apply system-level methodology to SystemC, SpecC and others.

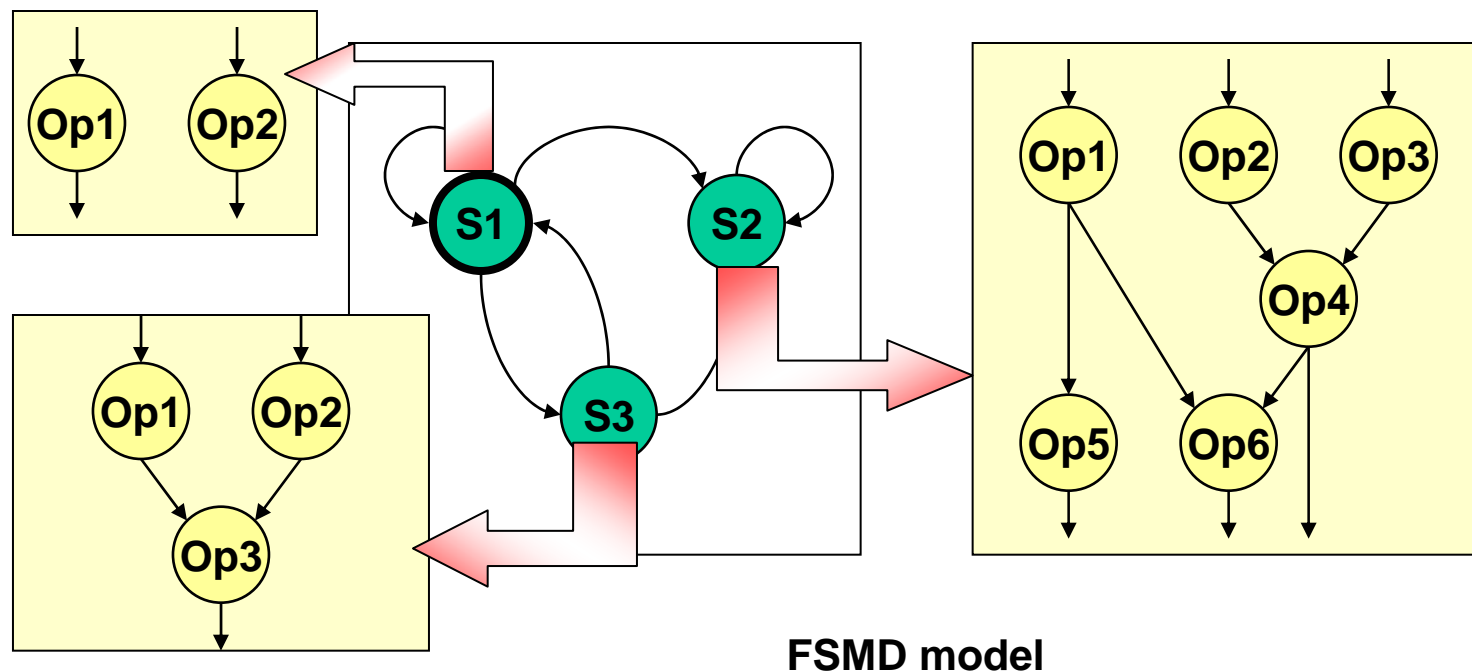
Semantics, Styles & Refinements

- Each model uses well defined semantics
- Each model has simple style
- Each style uniquely expressed
 - no syntactic variance or semantic ambiguity
- Each model needs style checker

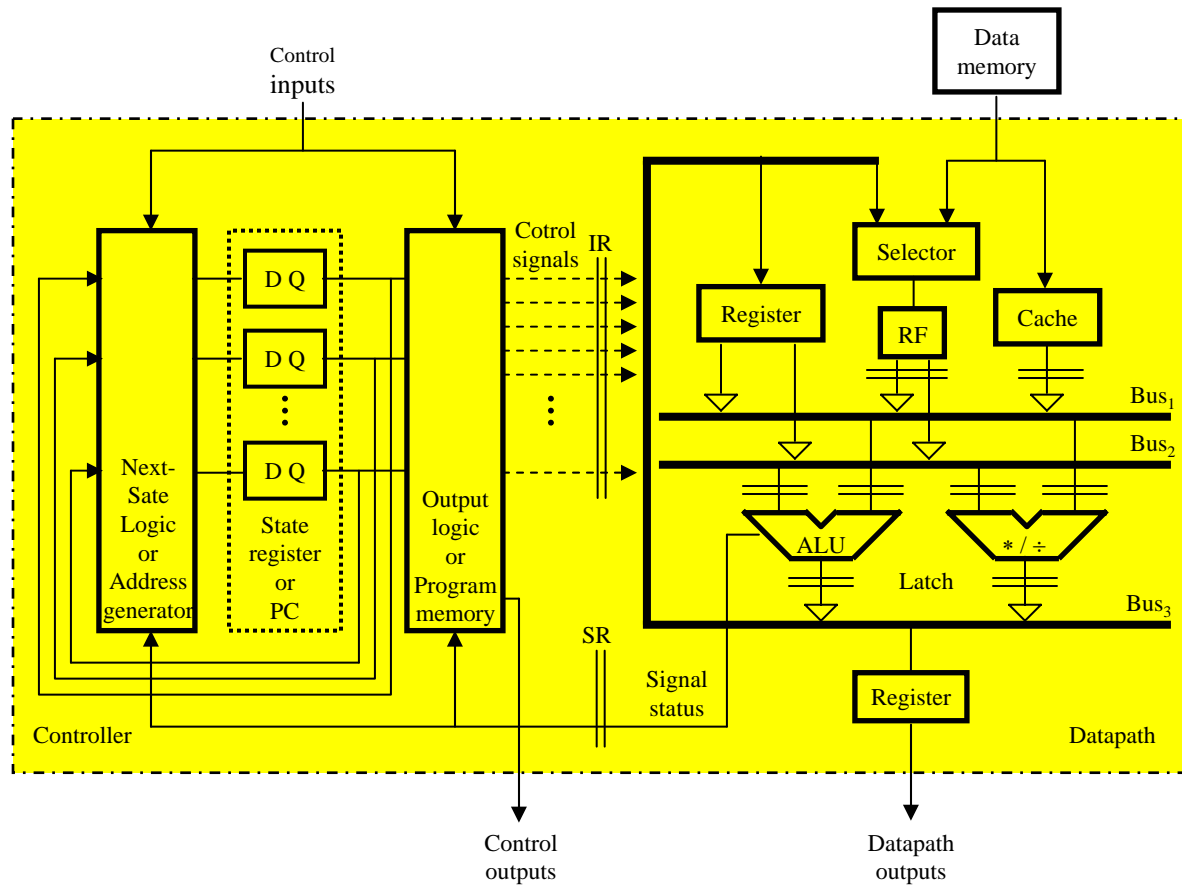
- Each model can be refined from its predecessor
- Clear refinement rules
- Clear application order of refinement rules
- Model refinements are verifiable

RTL Computational Models

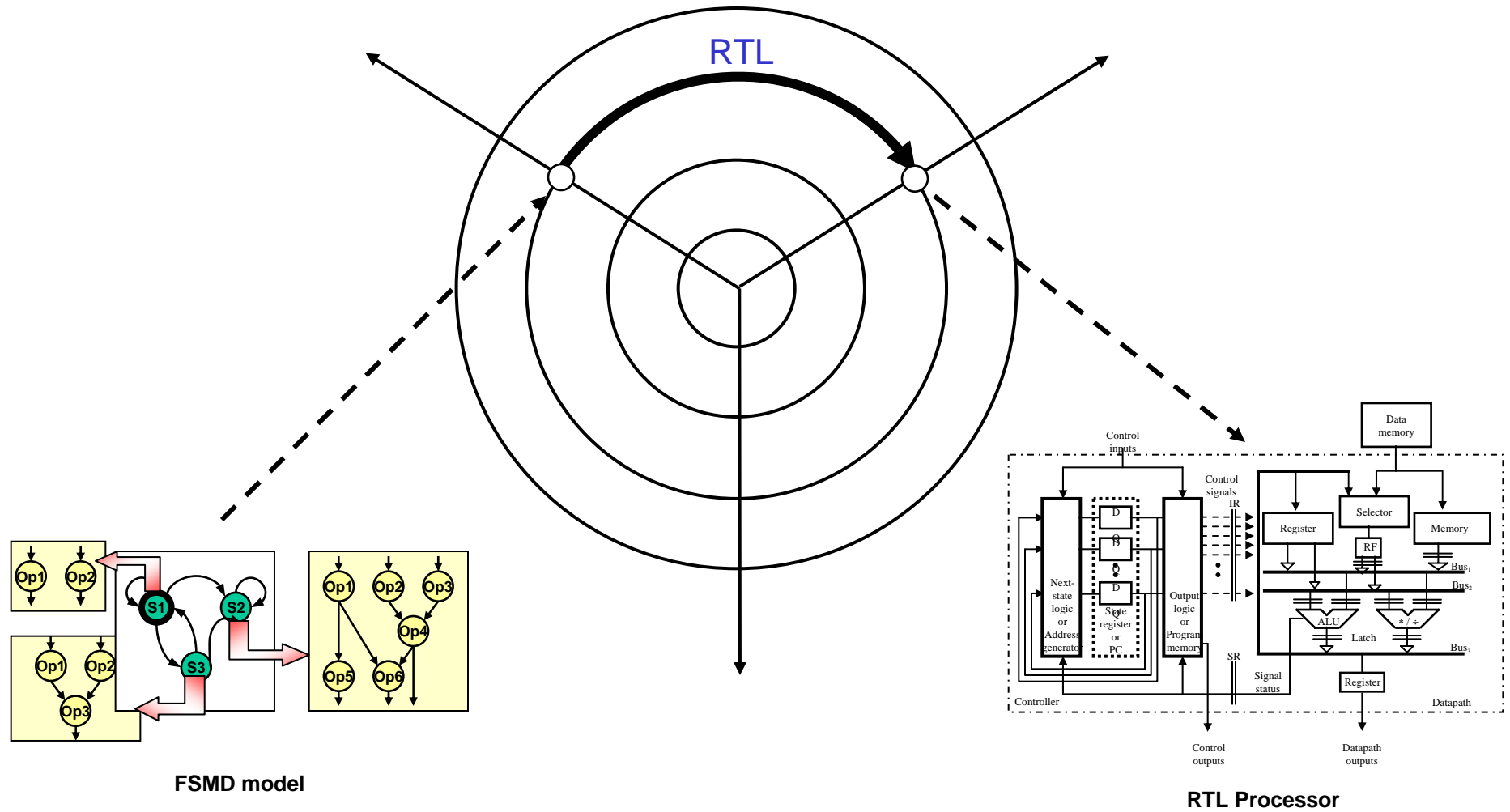
- **Finite State Machine with Data (FSMD)**
 - Combined model for control and computation
 - FSMD = FSM + DFG
 - Implementation: controller plus datapath



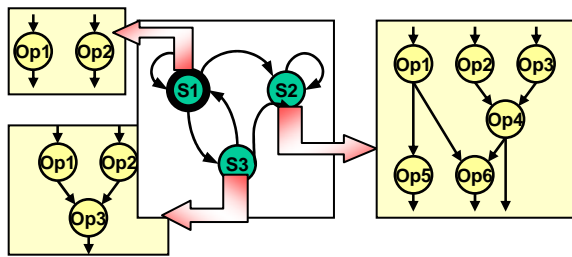
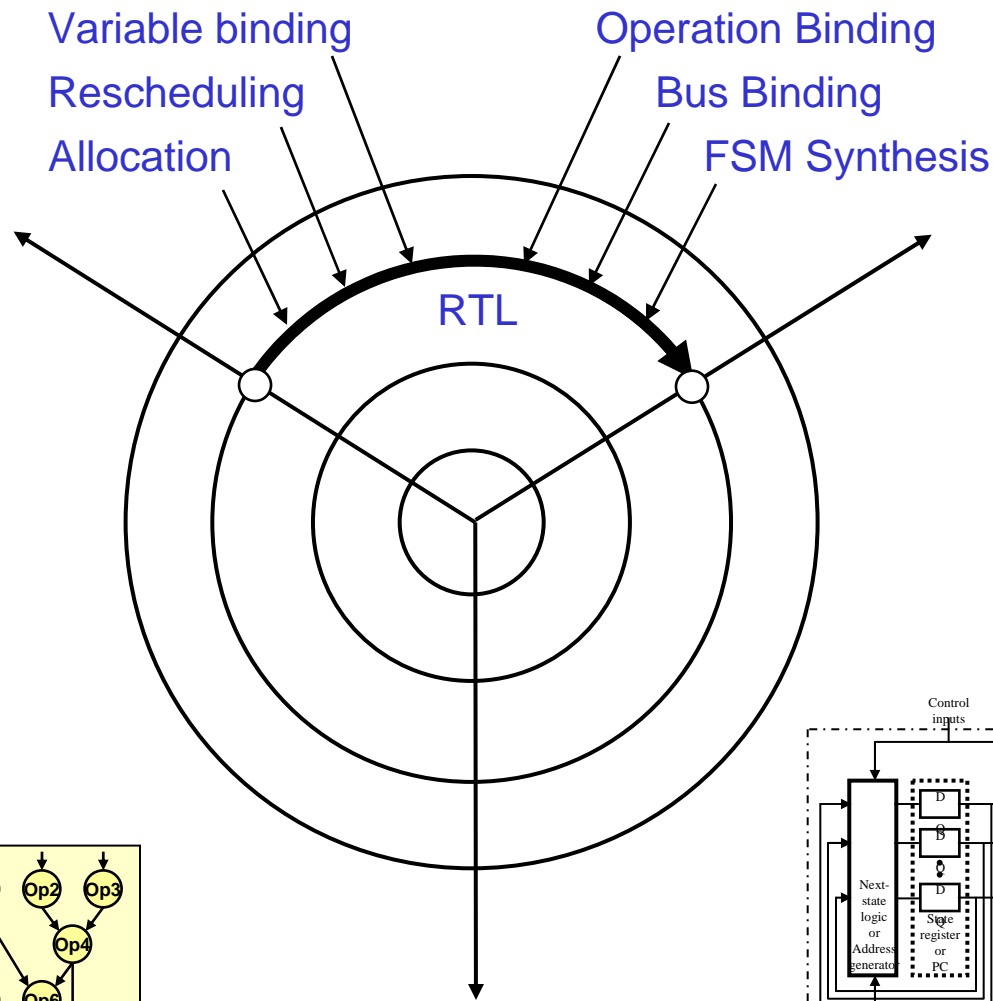
RTL Processor



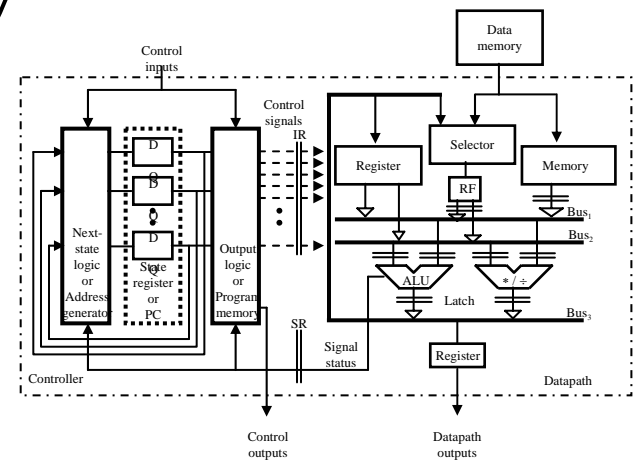
RTL Synthesis



RTL Synthesis



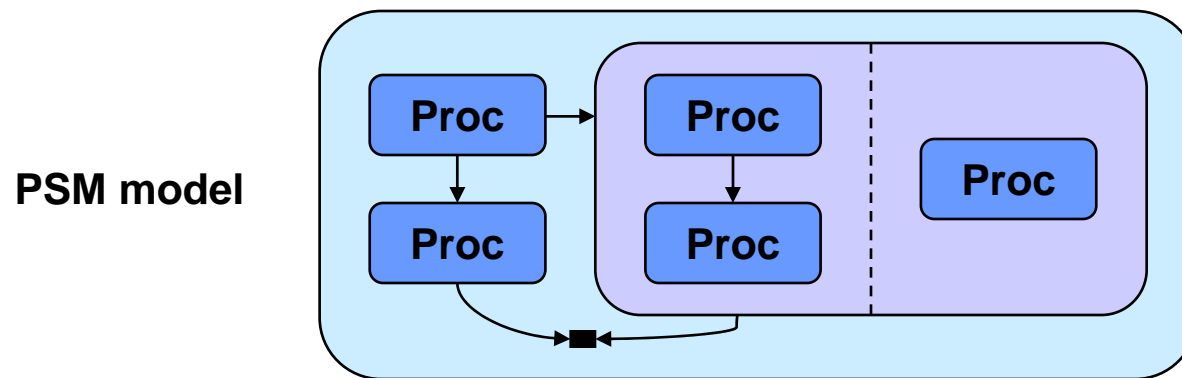
FSMD model



RTL Processor

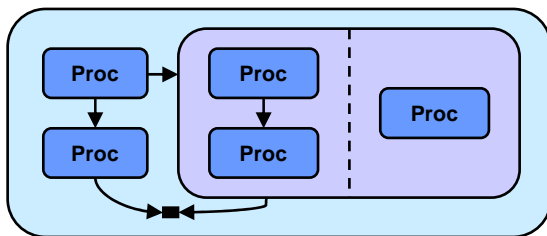
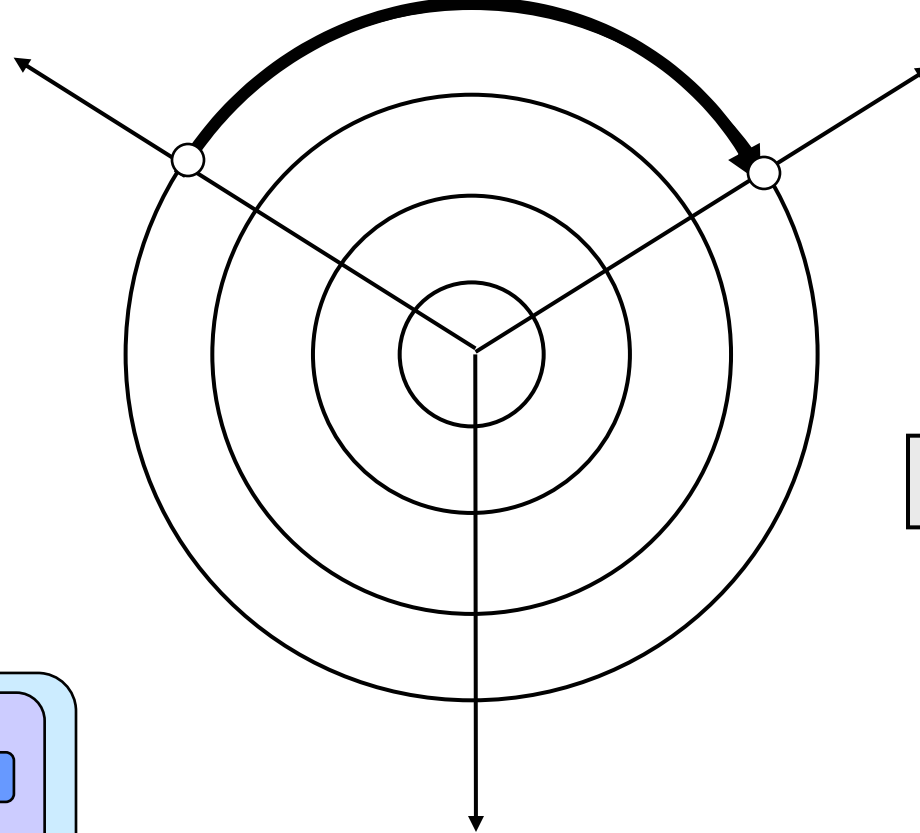
System Computational Models

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

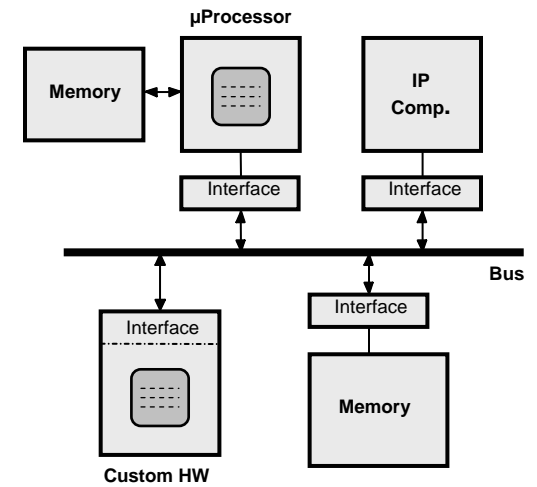


System Synthesis

System



PSM model



System architecture

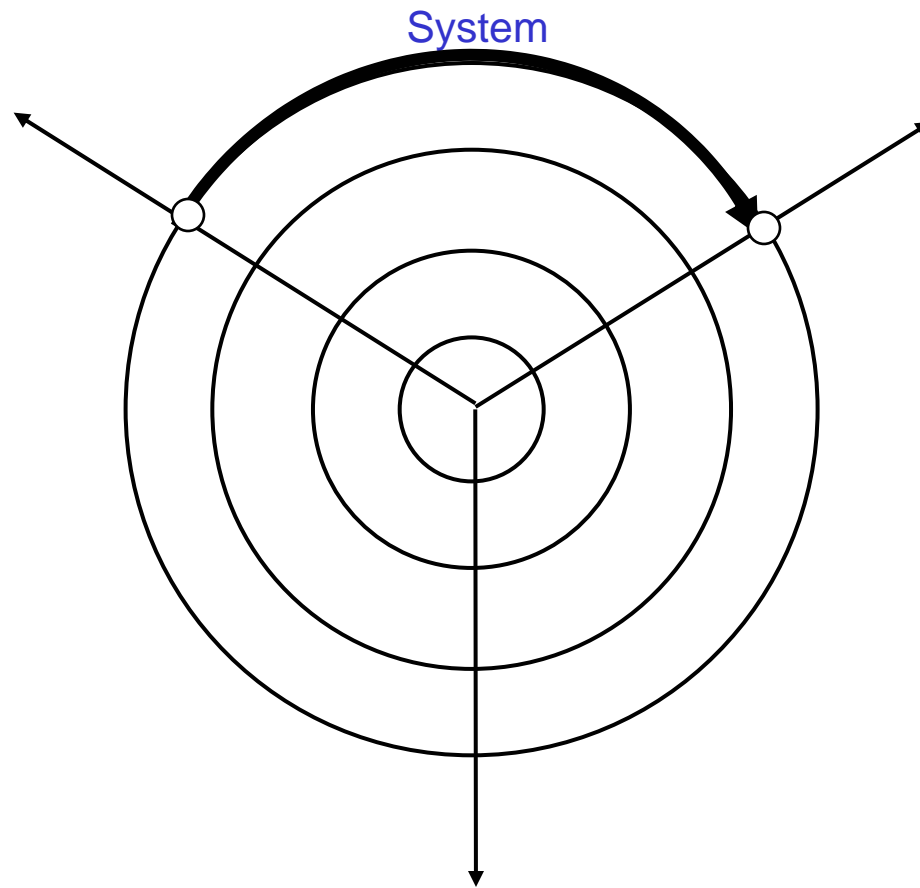
System Semantics

Objects:

- Behaviors
- Channels

Composition:

- Hierarchy
- Order
 - Sequential
 - Parallel
 - Piped
 - States
- Transitions
 - TI
 - TOC, TOS, ...
- Synchronization



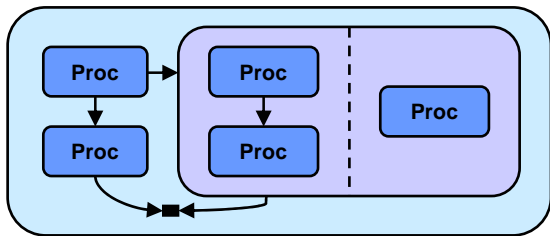
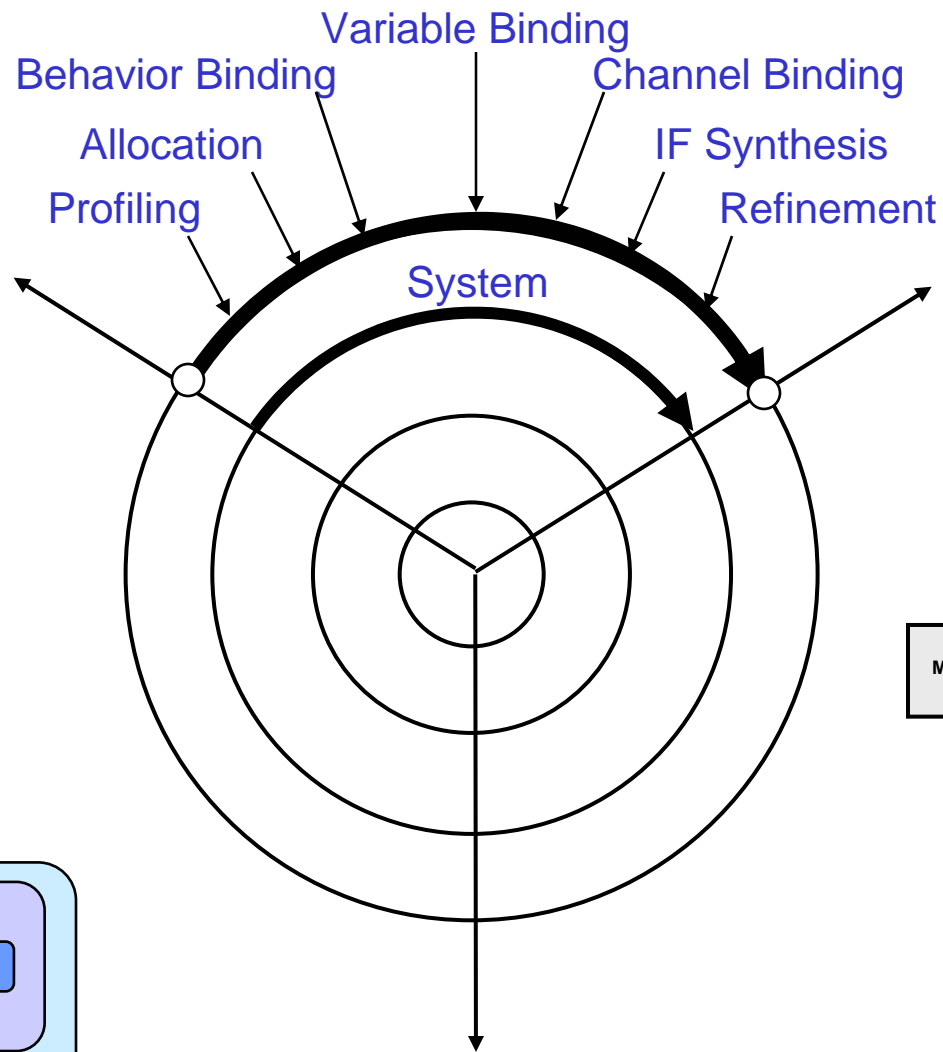
Objects:

- Components
 - Proc
 - IP
 - Memories
 - IF
- Connections
 - Buses
 - Wires

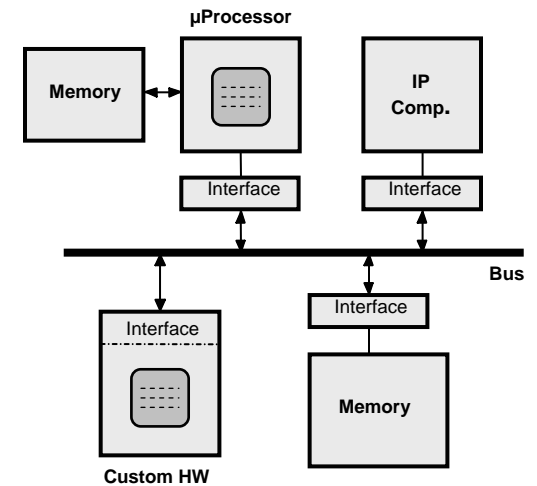
Composition:

(same as in Behavior Model)

System Synthesis

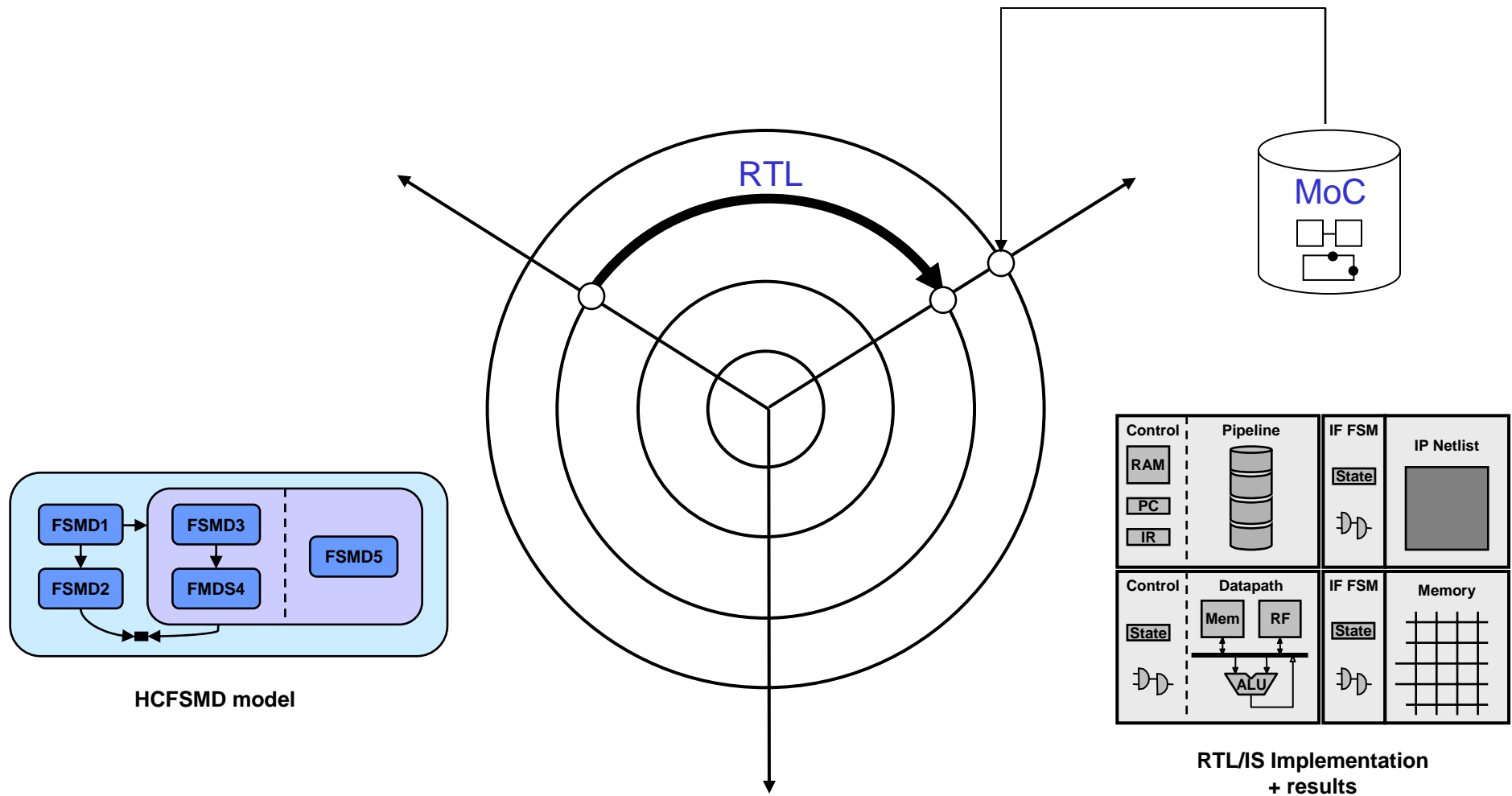


PSM model

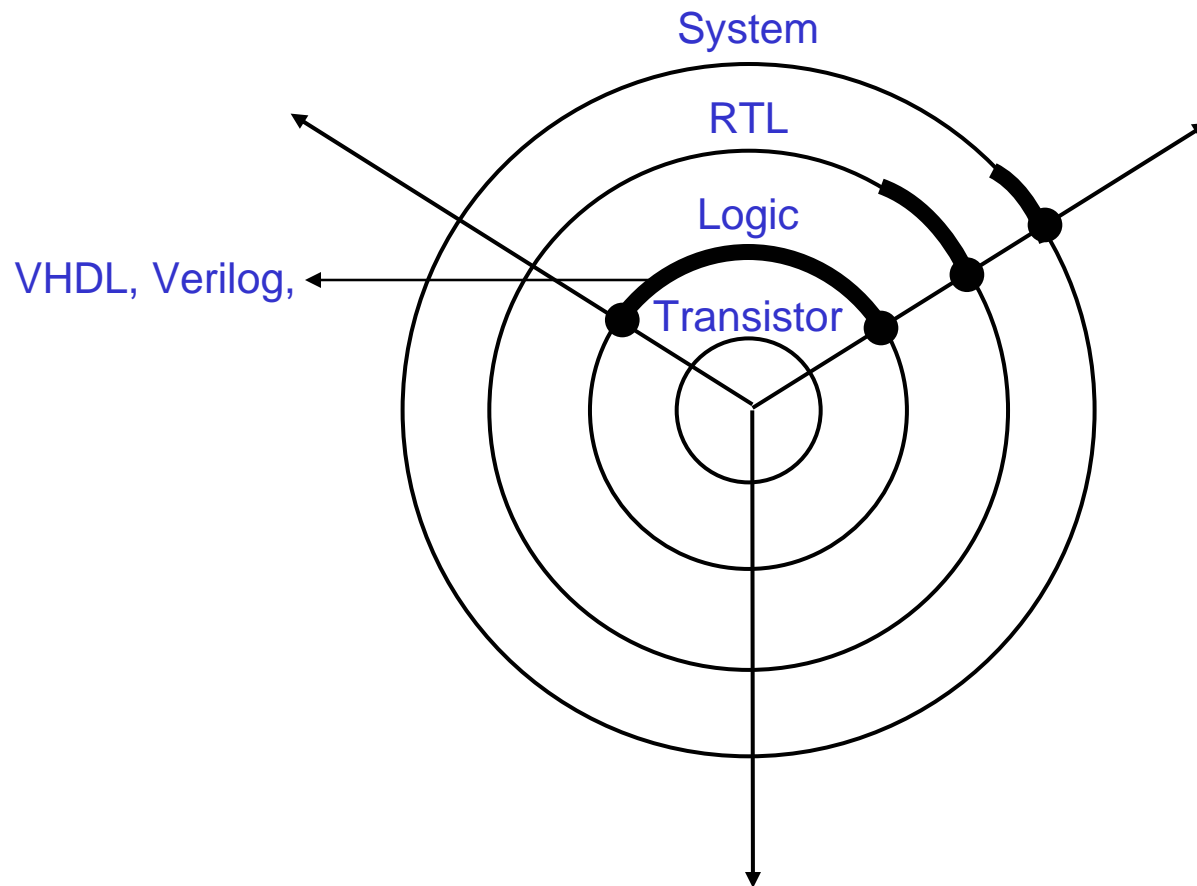


System architecture

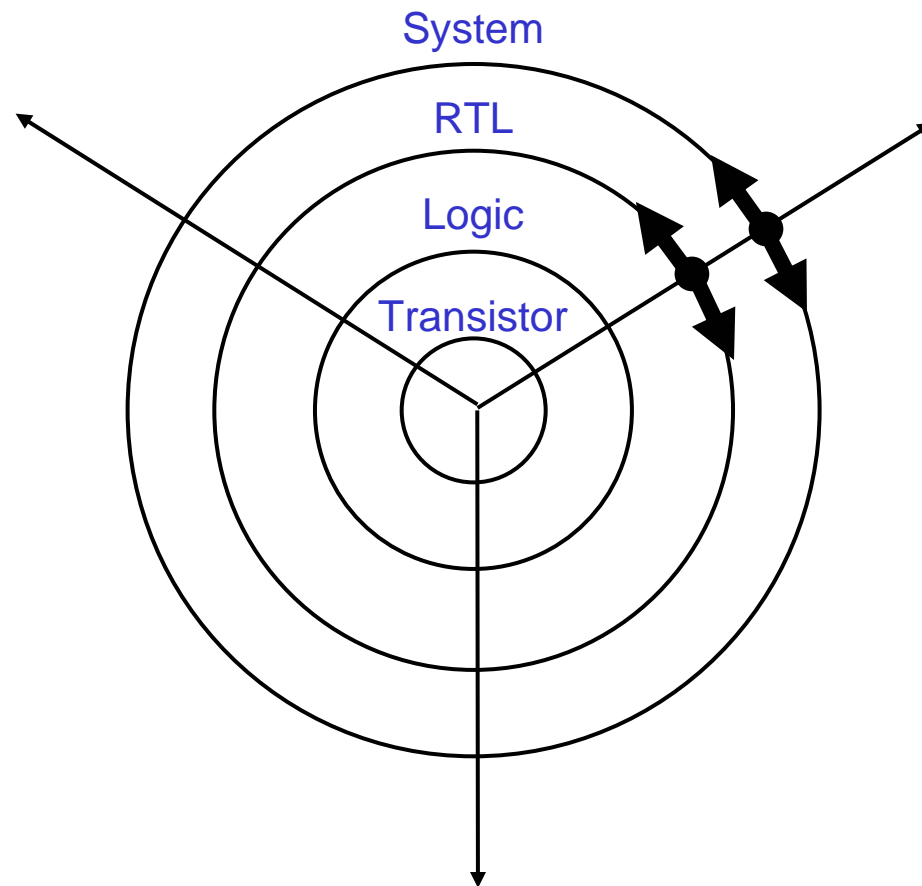
System Synthesis (continued)



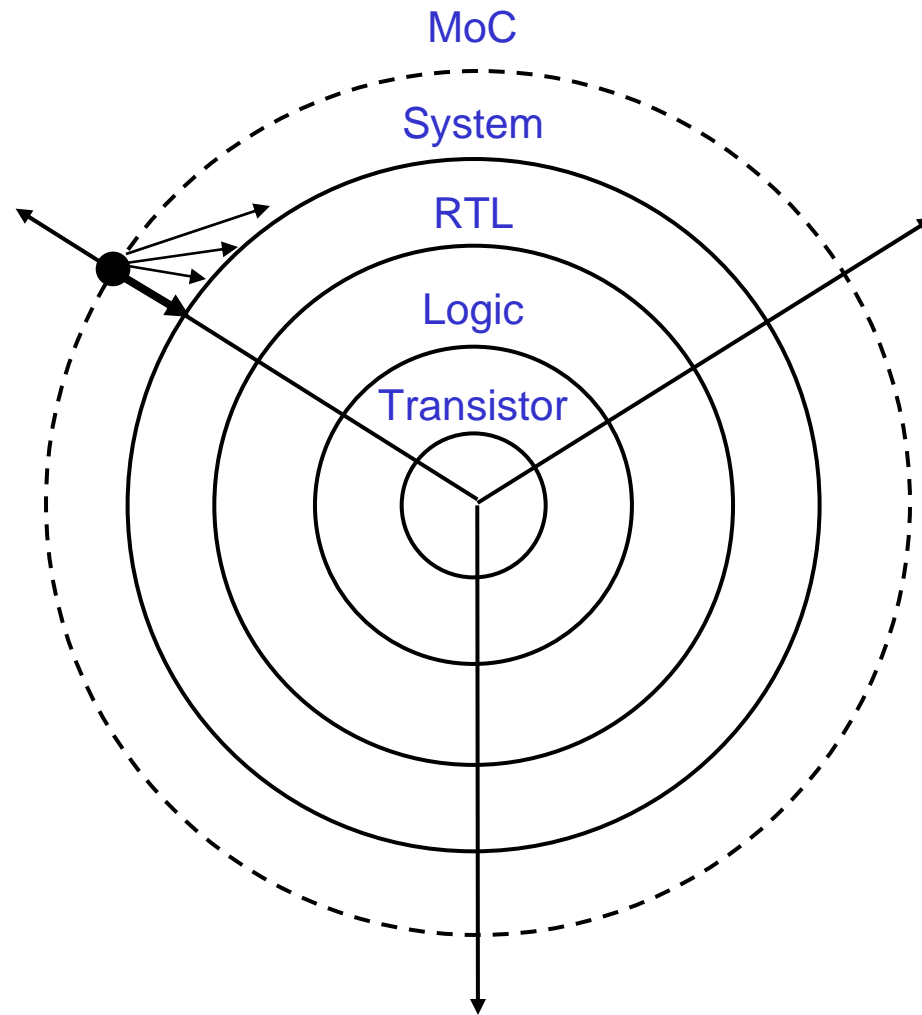
EDA Approach: Simulation



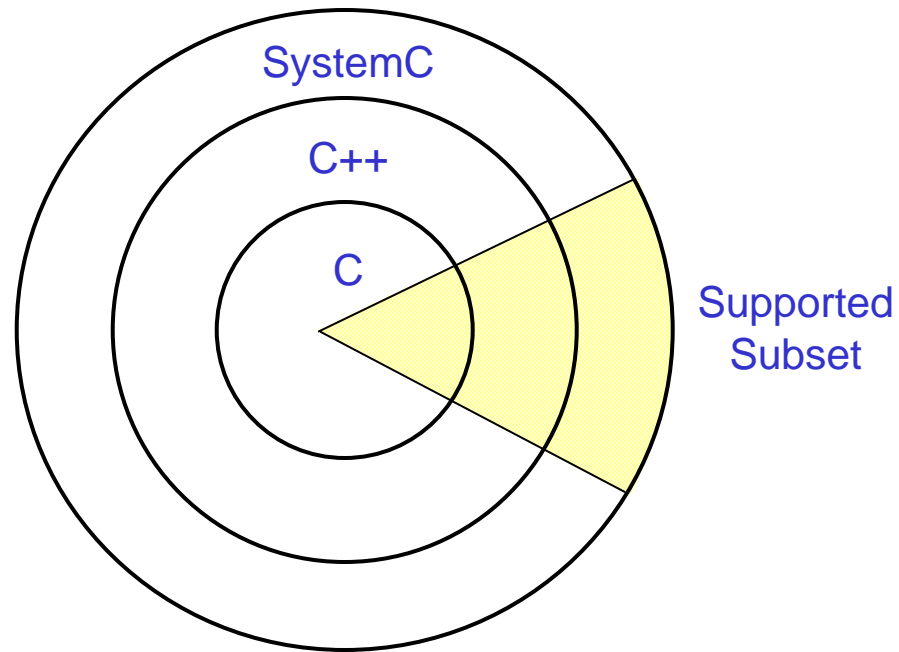
C++ Approach: Syntax



MoC Approach: Diversity

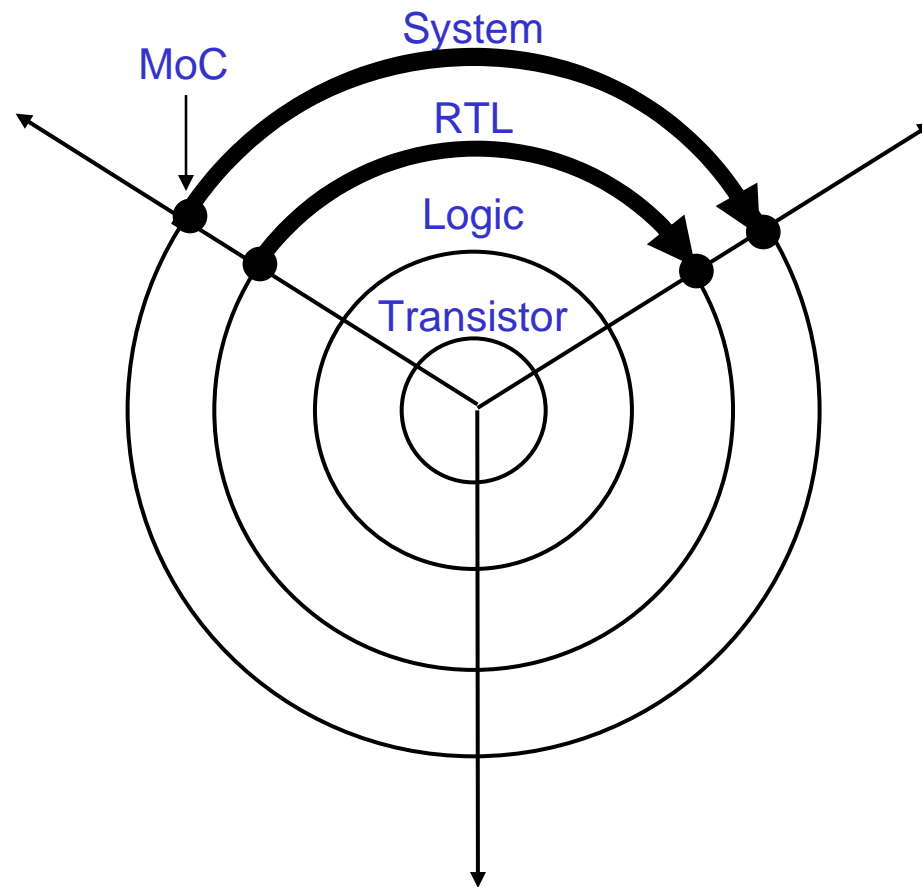


SystemC Approach: Language First

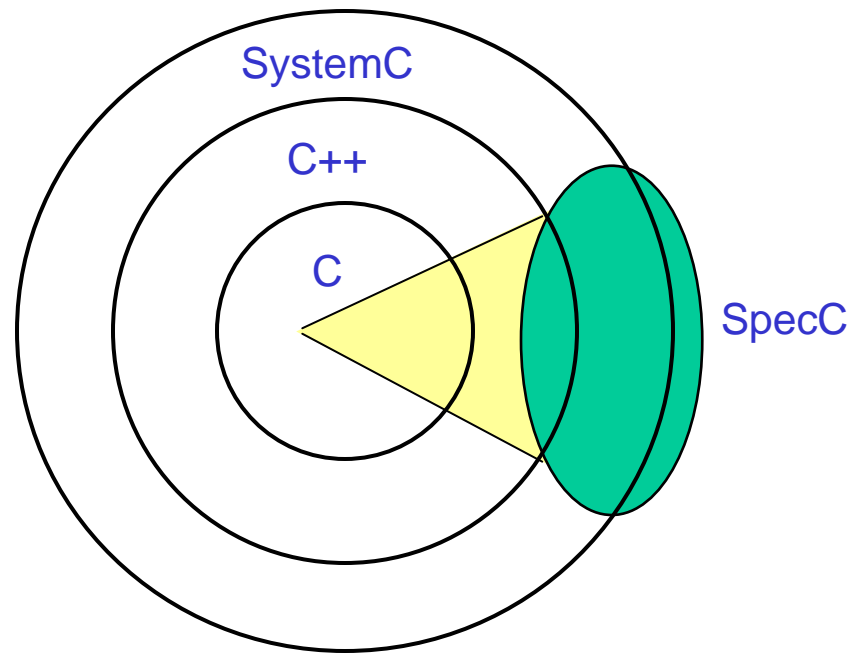


Source: J. Kunkel, VP Synopsis, (CODES, May 2002)

SpecC Approach: Semantics First



SystemC/SpecC



Quote from SystemC

FUNCTIONAL SPECIFICATION FOR SYSTEMC 2.0

Version 2.0-M
January 17, 2001

1.8 ACKNOWLEDGEMENTS

“Many companies and individuals have contributed time and resources in the development of both SystemC 1.0 and SystemC 2.0. Some of these contributors are listed in the contributors section of this specification and in the SystemC 1.0 Users’s Guide.

It should be noted that the fundamental mechanisms used to model communication and synchronization in SystemC 2.0 - interfaces, channels, and events - were inspired by similar constructs in Professor Daniel Gajski’s SpecC language. (For further information, see “SpecC: Specification Language and Methodology” at www.wkap.nl)”

Conclusion

Work to be done:

1. Abstraction Levels
2. Model Semantics
3. Refinement Rules
4. Methodology
5. Language
6. Simulation, Synthesis, Verification Tools
7. ESDA Market/Community Emergence

Prediction: No success in 7 without 1-6