Communication/Component-Based Design
(Theme Leader: Alberto SV)

Roberto Passerone (UC Berkeley)

Radu Marculescu (CMU)

GSRC Symposium & Workshop
June 9-10, 2002

SEMICONDUCTOR INDUSTRY SUPPLIERS

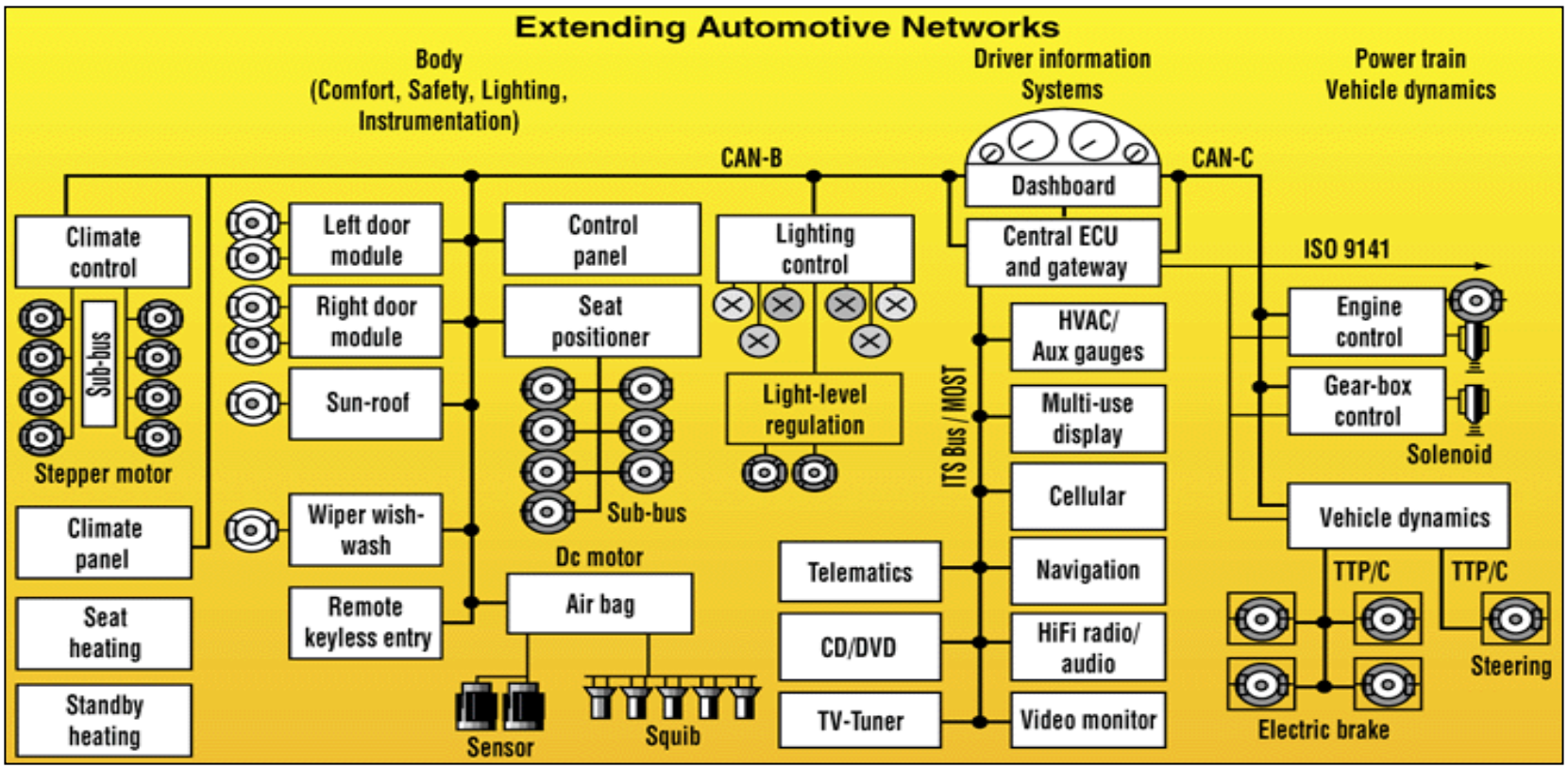
SIA SEMICONDUCTOR INDUSTRY ASSOCIATION

DUSD(Labs)

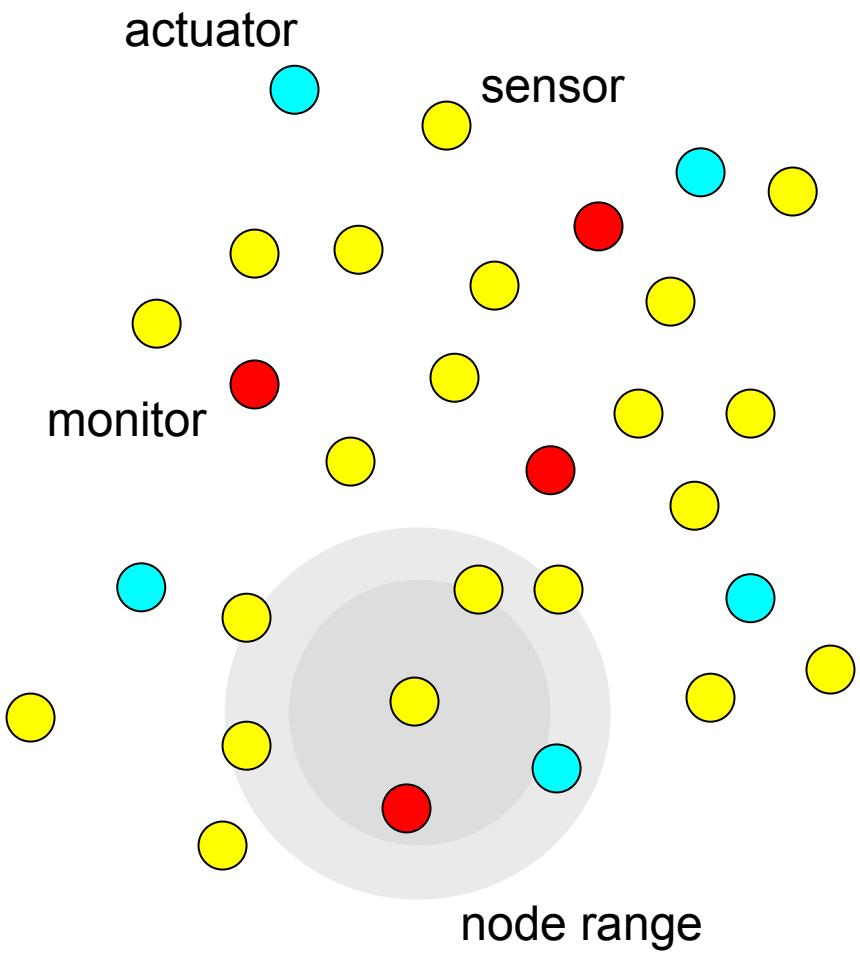
Outline

- Motivation
- Platform and Communication-based Design
  - Definition
  - Design Methodology
  - Network Platforms
  - Analog Platforms
- The Metropolis Framework
  - Metamodel
  - Verification
  - Synthesis
  - Theory of Models
  - Analysis (Radu Marculescu)
  - Communication Synthesis (Radu Marculescu)

Motivation:
Distributed Applications



The PicoRadio Networking Playground



Properties:

- system consisting of sensors (sources), monitors (controllers), and actuators (sinks)

Assumptions:

- no or minimal infrastructure
- range of any node << network size
- any node can act as repeater

Optimization Goals:

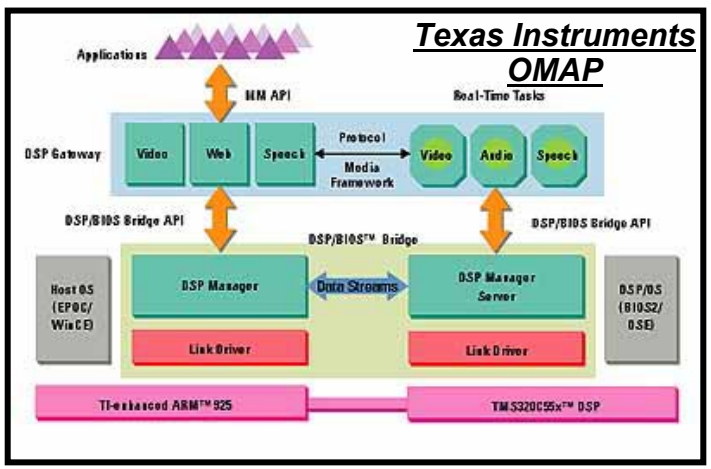
- Global energy
- System survivability
  - nodes can go down temporarily lacking energy
  - delivery of information to be ensured

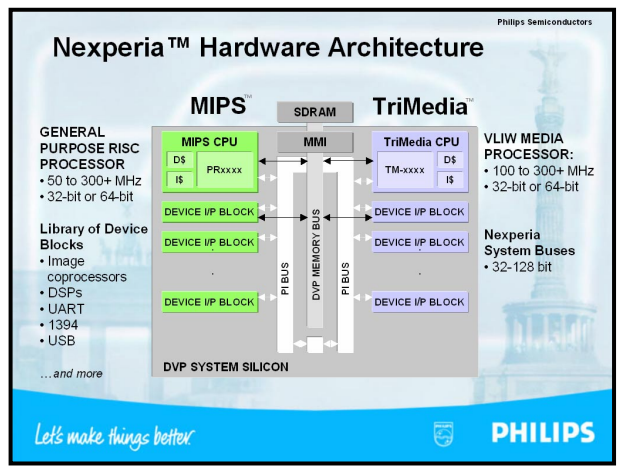
Outline

- Motivation
- Platform and Communication-based Design
  - Definition
  - Design Methodology
  - Network Platforms
  - Analog Platforms
- The Metropolis Framework
  - Metamodel
  - Verification
  - Synthesis
  - Theory of Models

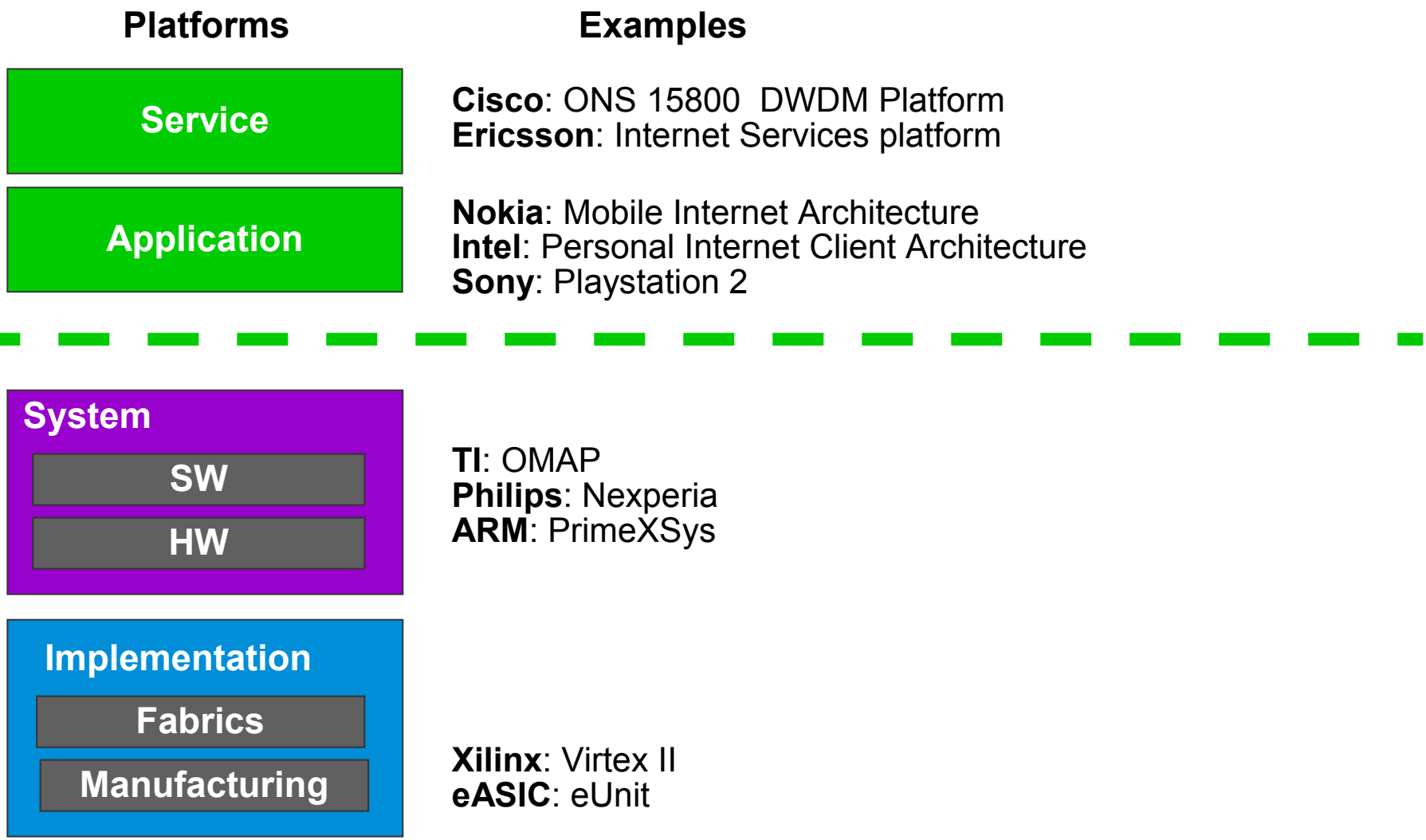
Platforms

- Texas Instruments OMAP, Philips nExperia, Infineon MGold
- Concentrates on full application
  - Delivers comprehensive set of libraries hardware and software
  - Delivers several mapping and application examples
- Hardware Platform
  - A coordinated family of architectures that satisfy a set of architectural constraints imposed to support re-use of hardware and software components





Beyond Hardware Platforms



ASV Platforms

In general, a platform is an abstraction layer that covers a ***number of possible refinements into a lower level*.**

Platform stack

Platform

Mapping Tools

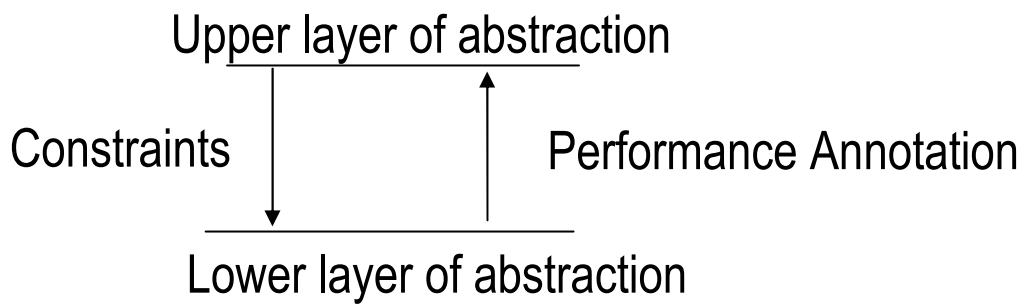
Platform

ASV Platforms

The design process is meet-in-the-middle:

- **Top-down**: map an instance of the top platform into an instance of the lower platform and propagate constraints
- **Bottom-up**: build a platform by defining the "library" that characterizes it and a performance abstraction (e.g., number of literals for tech. Independent optimization, area and propagation delay for a cell in a standard cell library)

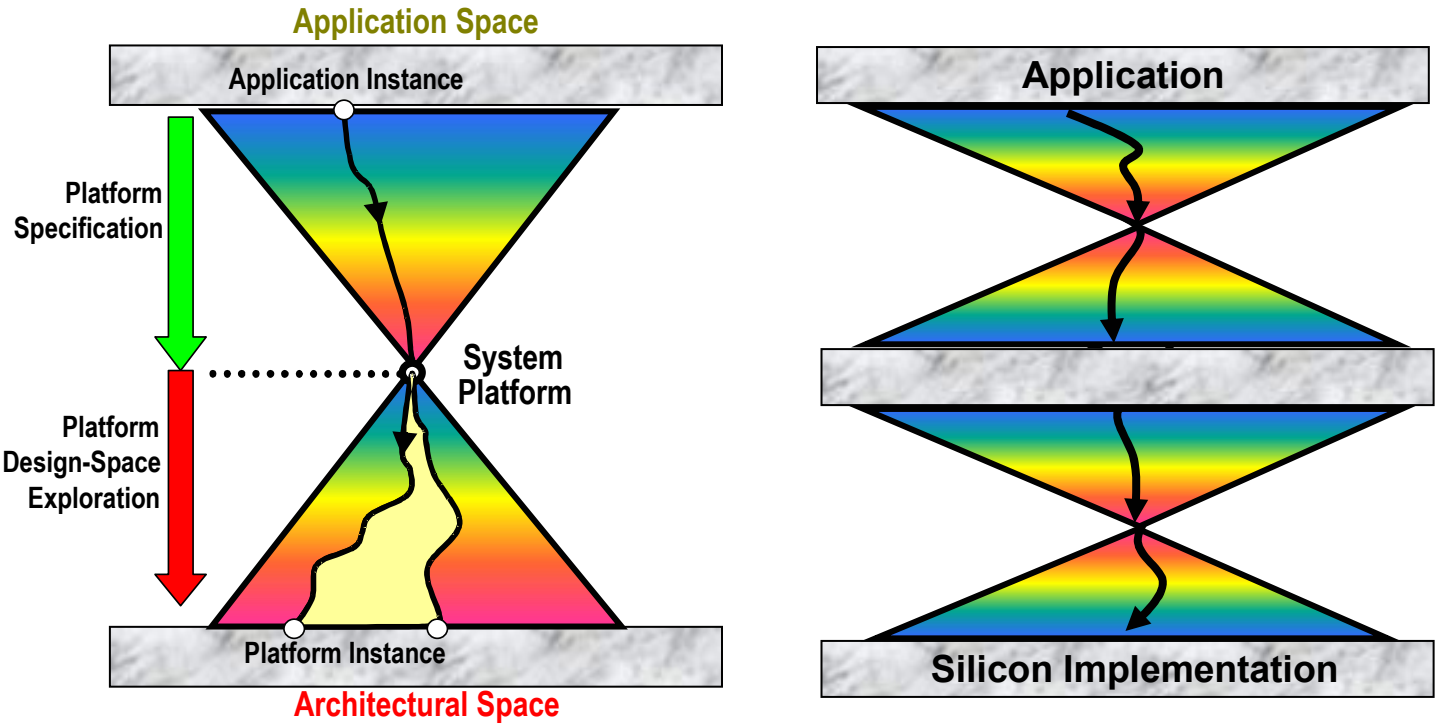
The library has elements and interconnects



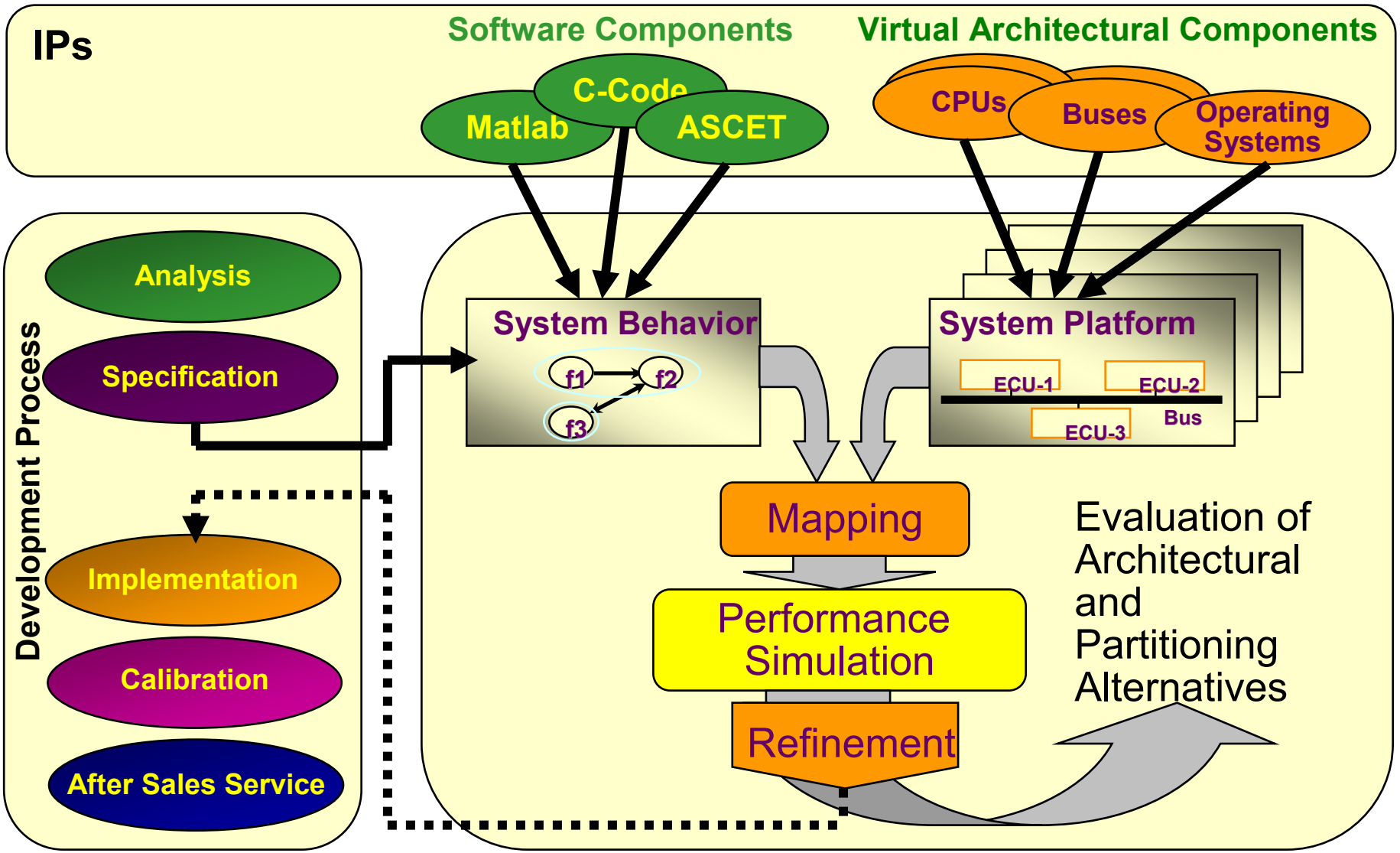
For every platform, there is a view that is used to map the upper layers of abstraction into the platform and a view that is used to define the class of lower level abstractions implied by the platform.

Platform-Based Implementation

- Platforms eliminate *large loop iterations* for affordable design
- Restrict design space via new forms of regularity and structure that surrender *some* design potential for lower cost and first-pass success
- The number and location of intermediate platforms is the essence of platform-based design



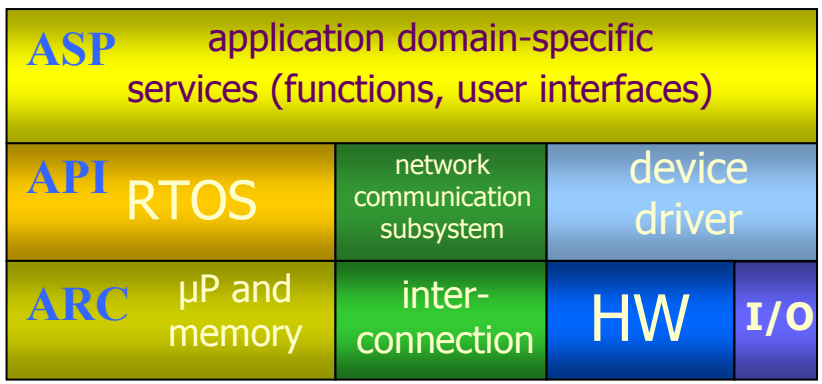
Design Methodology: Orthogonalize Concerns



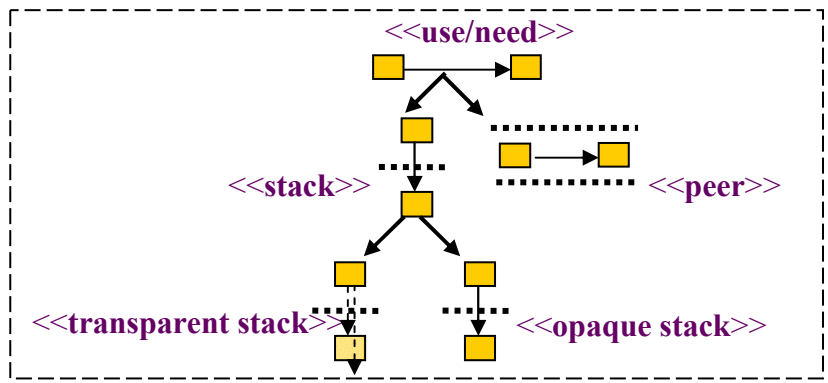
UML-Platform: Notation and Methodology for PBD

- **Overview:** – a projection of platforms into the UML notation space
- **Results:** – a new UML profile for platform-based design (PBD)
  – a methodology for representation of platform layers, relations, QoS, constraints, extension points, etc.
- **Directions:** – a front-end language for Metropolis
  – a full-fledged design methodology based on Metropolis

• **Identify platform layers**

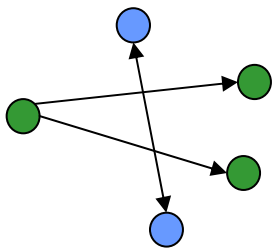


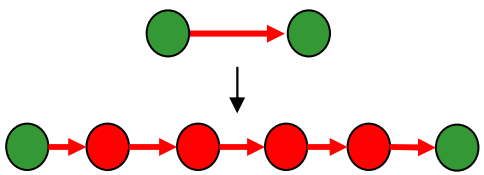
• **Build stereotypes and hierarchy**

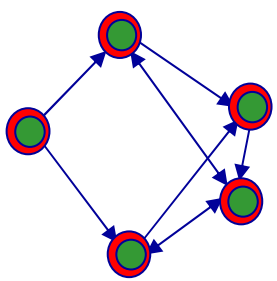


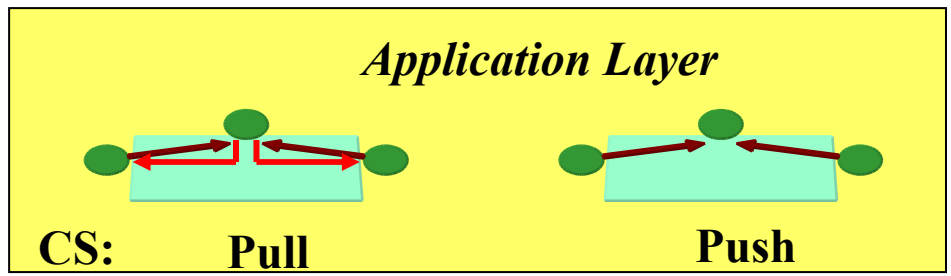
Model and Design of Network Platforms

- Formalization of Network Platforms
  - APIs: sets of Communication Services
- Application: Design of Picoradio networks
  - Communication Refinement

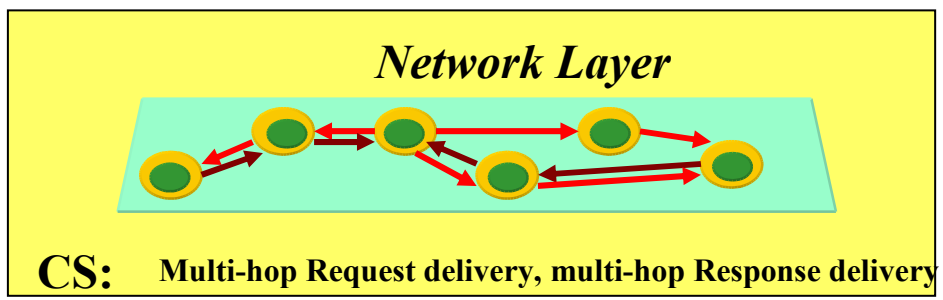






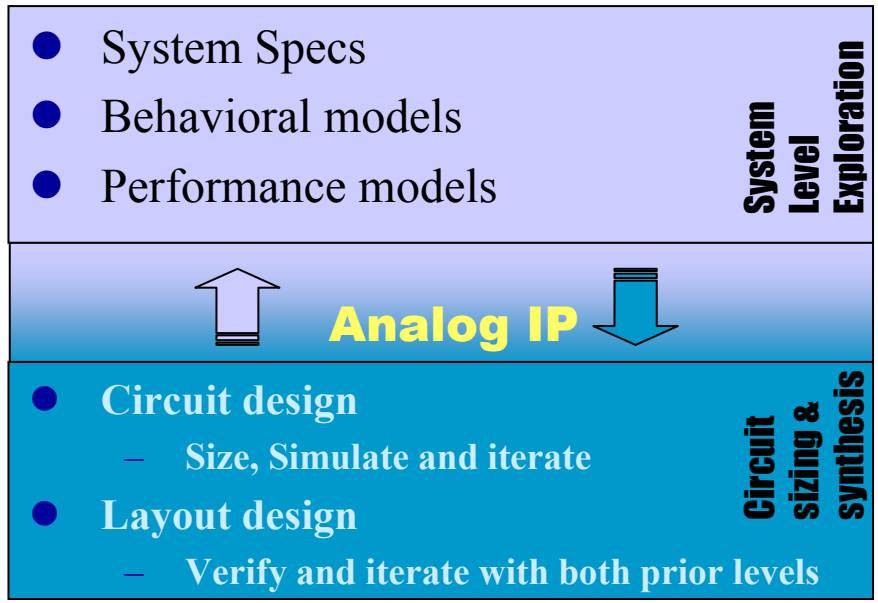


Max Power, BER



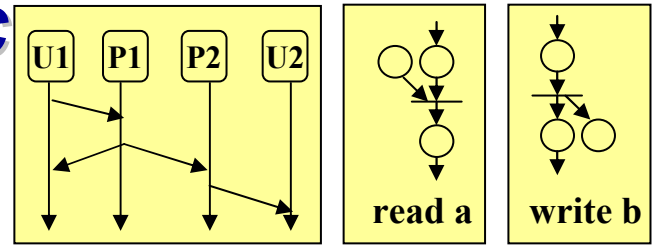
Analog Platforms

- Analog Platforms are parametrized architectural components
- Analog Platforms along with hierarchies of behavioral models define an *Analog IP*

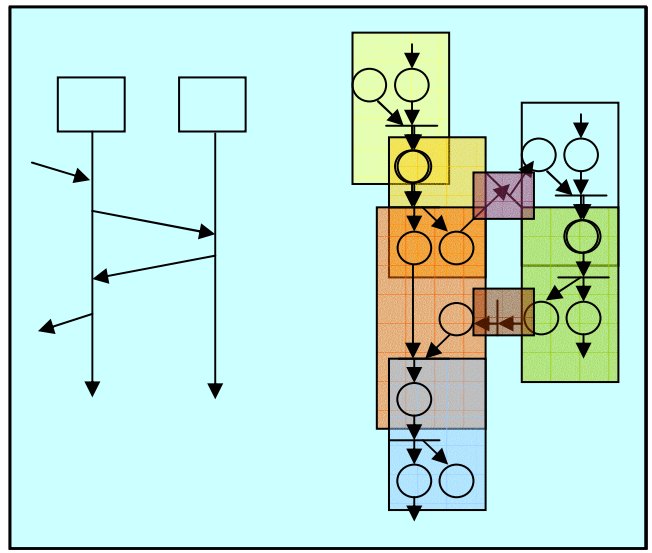


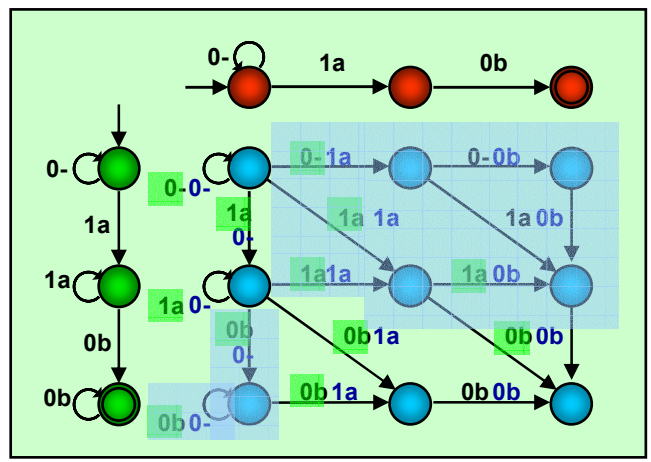
- Roles of the Analog IP
  - Separate System Level Design from Circuit Design
  - Hide all implementation details, only export performances
- The goal of Analog IPs is to support optimizations at the system level
  - Define *optimal* specs for individual blocks, thus selecting particular instances of the Analog Platforms

Communication-Network Centric



- Ulysses: Protocol Synthesis from Scenario-based Specifications
  - Avoid early partitioning into components
  - Specify scenarios independently
  - Compose scenarios
- Interface Synthesis
  - Synthesis of converters from property specification
  - Blend of synthesis and verification techniques (with T. Henzinger and L. de Alfaro)
- Directions
  - Generalization of synthesis techniques to arbitrary abstraction layer

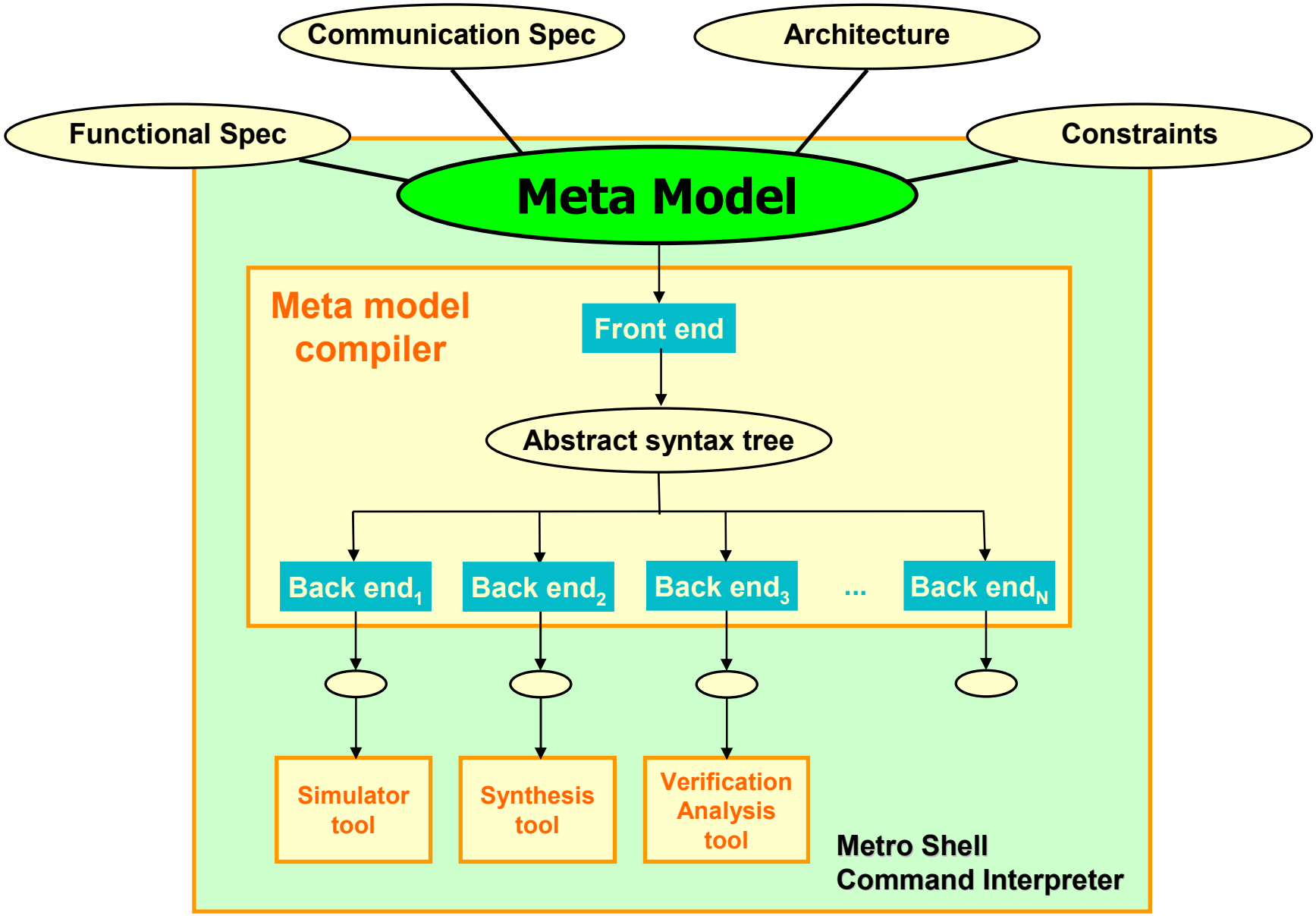




Outline

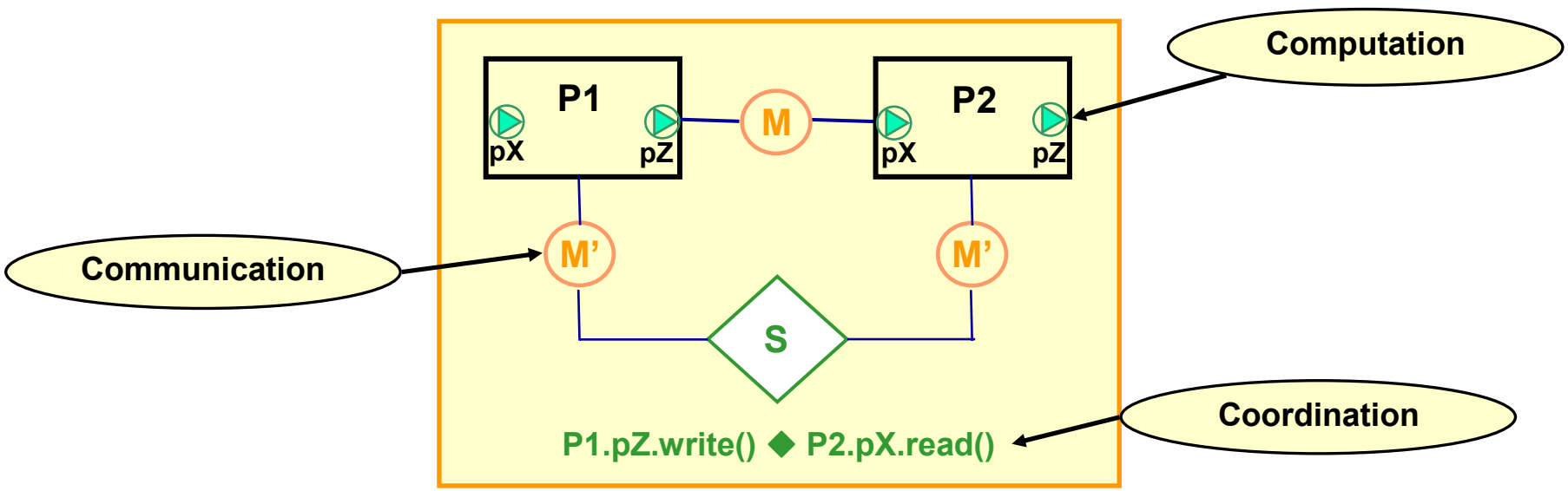
- Motivation
- Platform and Communication-based Design
  - Definition
  - Design Methodology
  - Network Platforms
  - Analog Platforms
- The Metropolis Framework
  - Metamodel
  - Verification
  - Synthesis
  - Theory of Models

Metropolis


Communication Spec
Architecture
Functional Spec
Constraints
Meta Model
Meta model compiler
Front end
Abstract syntax tree
Back end1
Back end2
Back end3
...
Back endN
Simulator tool
Synthesis tool
Verification Analysis tool
Metro Shell
Command Interpreter

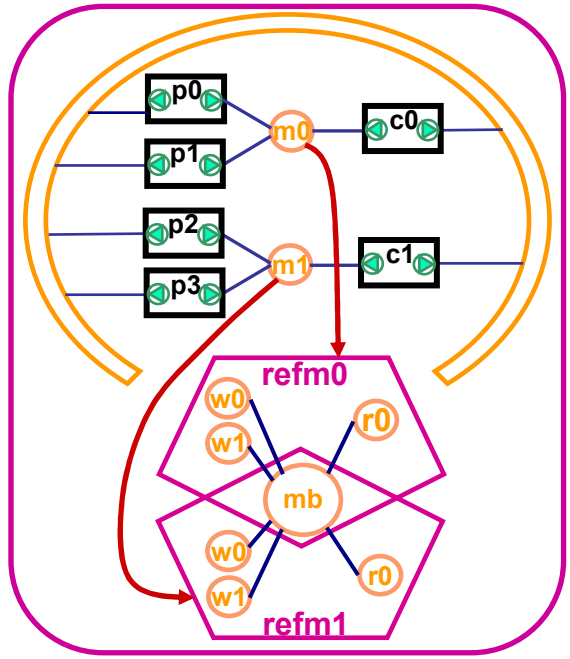
Metropolis: meta model

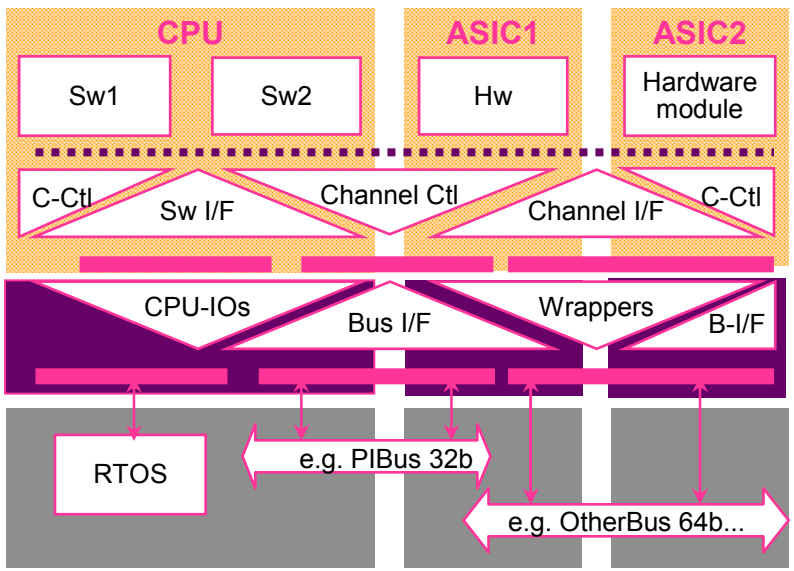
- Must describe objects at different levels of abstraction
  - Do not commit to the semantics of a particular Model of Computation
- Define a set of "building blocks":
  - specifications with many useful MoCs can be described using the building blocks
  - Processes, communication media and schedulers separate computation, communication and coordination
- Represent behavior at all design phases - mapped or unmapped



Emphasis

- Refinement
  - Functional refinement
  - Communication refinement
- Constraints
  - Quantities, Temporal logic, Schedulers
- Architecture definition





SYSTEM:

- SW modules, HW
- bounded FIFO, lossy channels
- no address, bus independent

TRANSACTION:

- address, data split in chunk
- no detailed bus protocol or width

PHYSICAL:

- specific bus protocol
- detailed RTOS characterization

Outline

- Motivation
- Platform and Communication-based Design
  - Definition
  - Design Methodology
  - Network Platforms
  - Analog Platforms
- The Metropolis Framework
  - Metamodel
  - Verification
  - Synthesis
  - Theory of Models

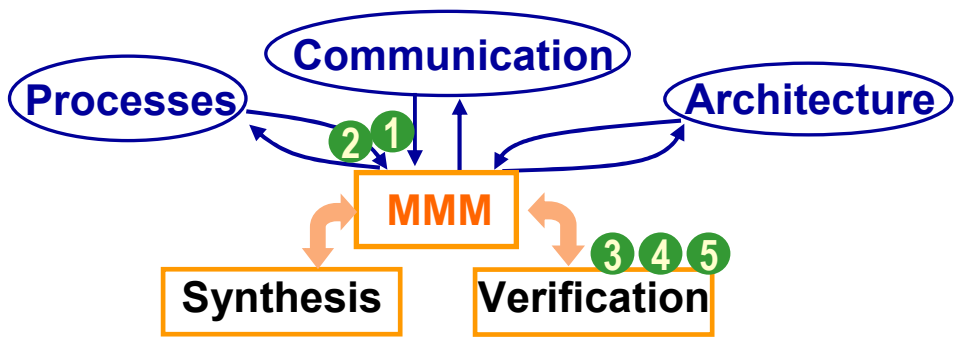
Simulating the MetaModel in the Metropolis Framework

- Multi-threaded (single or multi-processor) simulation code
  - Java™, SystemC, C++
- Extension to performance simulation
  - Architecture: netlist of blocks that provide services
  - Quantities: manage performance metrics (time, power, area, etc.)
  - Mapping: annotation of functional processes using quantities and architecture services
- Simulator Performance
  - Constructed and profiled various models in SystemC and Metropolis
  - Identified bottlenecks and implemented changes to match the performance of System C and Metropolis model simulators

  Metropolis benefits come at no extra simulation cost
- Directions:
  - Techniques for Interactive & batch based simulation
  - Simulation Coverage Enhancement [Ip, ICCAD 2000]
  - Heuristics to guide the simulator for finding bugs [Dill, DAC 1998]

Formal Specification and Analysis: Metropolis at UC-Riverside (H. Hsieh, et al)

1. Defining MOCs in MMM
   - SDF, Dataflow PN, Synchronous FSM network, SystemC subset, …
2. Translating Ptolemy/CAL designs into MMM
   - SDF, Dataflow PN
3. Formal verification of MMM designs using SPIN
   - Property verification, implementation verification
4. Conformance checking of MMM design using SPIN
   - Simulation trace containment of implementation vs. spec
5. Verifying constraint formulae with simulation
   - Simulation monitor for quantitative constraints

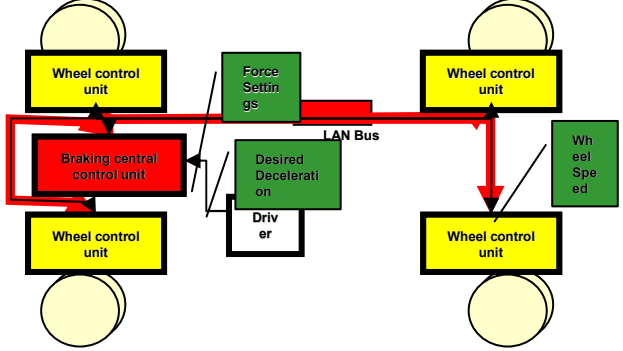


Successive Refinement in Metropolis

- **Verify properties of components : well-timedness, liveness**
  - **Properties preserved by**
    - **Composition of components (*compositionality*)**
    - **Restriction by constraints (*composability*)**
  - **Integration of the incremental modeling tool Prometheus in Metropolis (work in progress).**
  - **Case study: TinyOS networking application.**
- **Directions**
  - **Provide modeling guidelines for the meta-model to support incremental modeling.**
  - **Extend results to more “difficult” properties, e.g. schedulability of processes.**
  - **Efficiently synthesize a refinement satisfying required, more specific properties.**

Outline

- Motivation
- Platform and Communication-based Design
  - Definition
  - Design Methodology
  - Network Platforms
  - Analog Platforms
- The Metropolis Framework
  - Metamodel
  - Verification
  - Synthesis
  - Theory of Models



Application Driven Scheduling

Scheduling for real-time feedback controllers

http://www-cad.eecs.berkeley.edu/~pinello

DRAFTS: Distributed Real-time Applications Fault Tolerant Scheduling

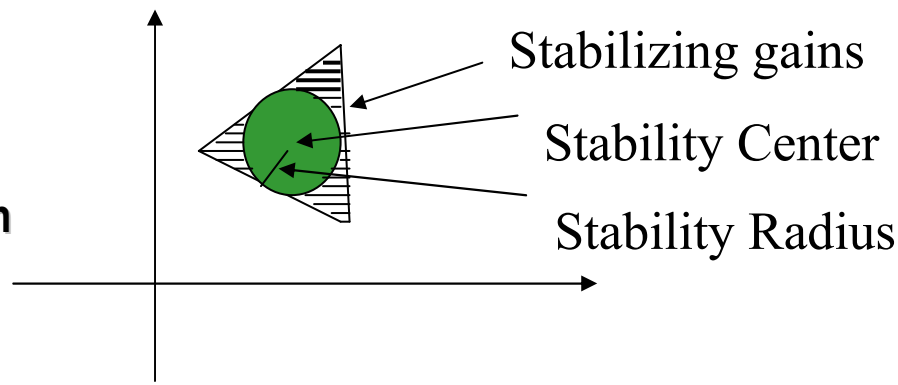
- Automatic (off-line) synthesis of fault tolerant schedules for periodic algorithms on a distributed architecture
- Automatic (off-line) verification that all intended faults are covered

RACS: Resource-Aware Control Synthesis

- Optimal Synthesis of control gains and scheduling parameters
- Performance metric: stability robustness
- Constraints: execution capacity and scheduling policy (e.g. EDF, RM)

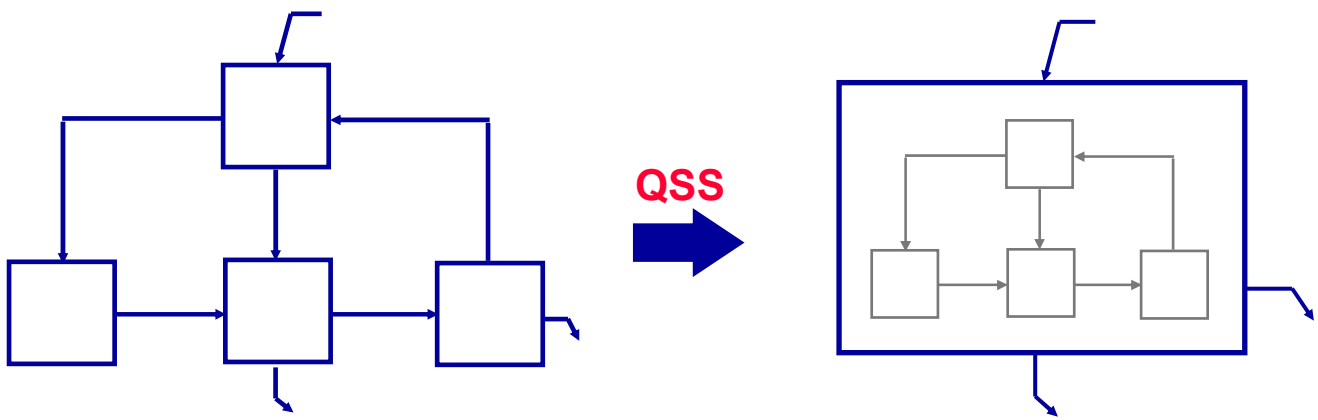
Long-term goals:

- Design Methodology for Safety Critical Distributed Systems
- Manage the design complexity of modern Drive-By-Wire applications



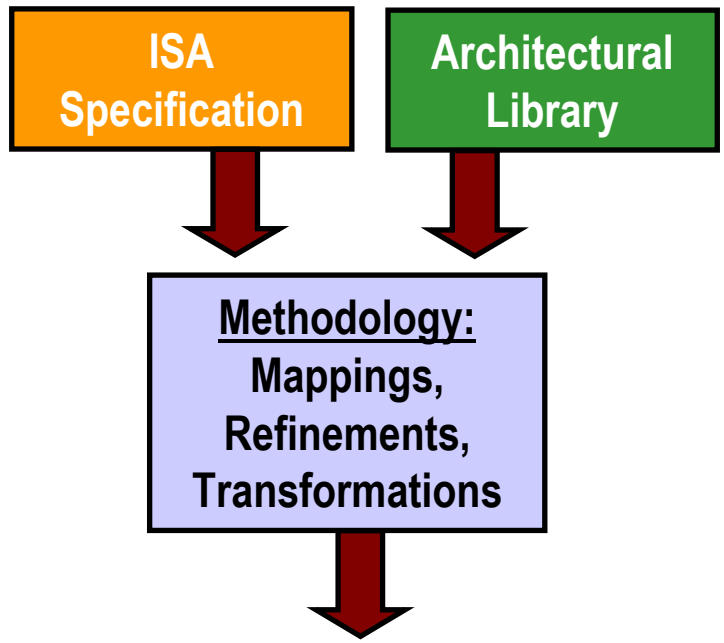
Software Synthesis: Quasi-Static Scheduling

- Sequentialize concurrent operations
  - Can handle data-dependent control, multi-rate communication
  - Better starting point for code generation
- Philips MPEG2 decoder: Performance increased by 45%
  - reduction of communication (no internal FIFOs between statically scheduled processes)
  - reduction of run-time scheduling (OS)
  - no reduction in computation
- Future directions
  - False path analysis, design partitioning, multiprocessor systems



Communication Driven HW Synthesis(CDHWSYNTH) for High-Performance Microprocessor Design

- From ISA to micro-architecture
  - Leverage Communication Based Design
  - High Performance
  - Correct by Construction Design
  - Reusability and Flexibility
- Case Study Specification of a MIPS 32
  - Developed a Trace-Driven Simulator for Multiprocessor Cache Coherence in SystemC
  - Preliminary results for
    - Representing Speculation
    - Modeling various levels of abstraction using Process Networks and Synchronous Languages
- Directions
  - Examples From Industry (Intel and Cypress)
  - Further exploration of modeling memory systems



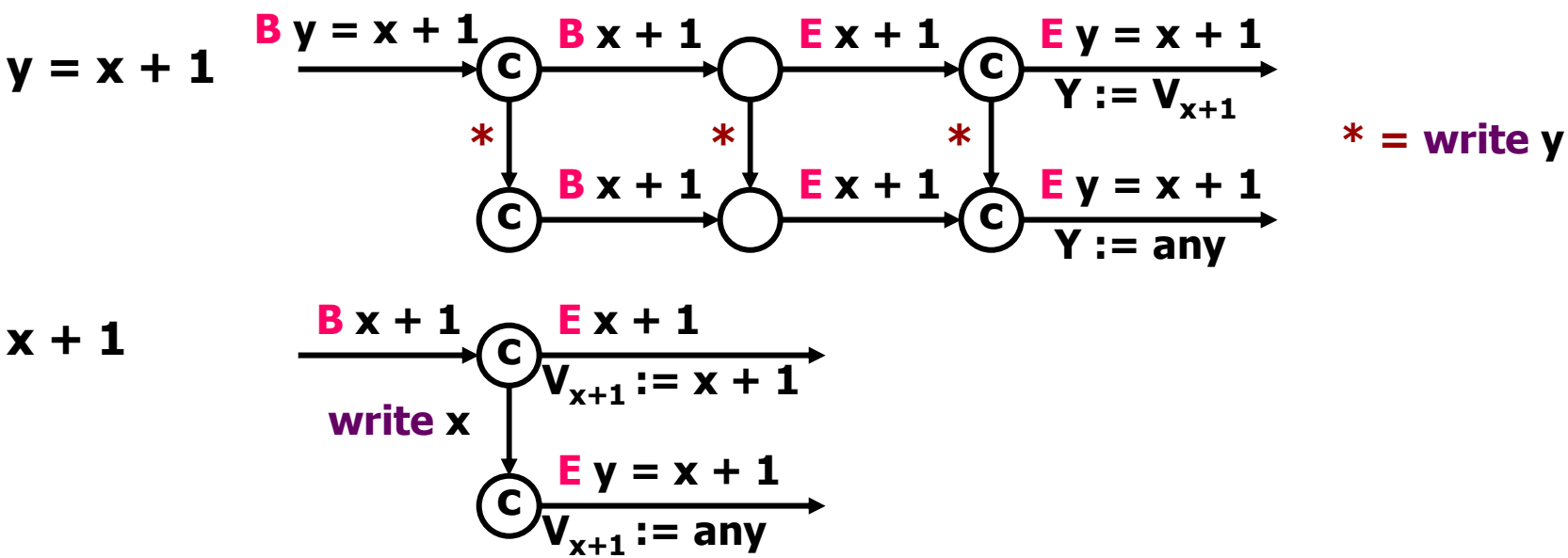
Intel Pentium IV Die (source: Intel web site)

Outline

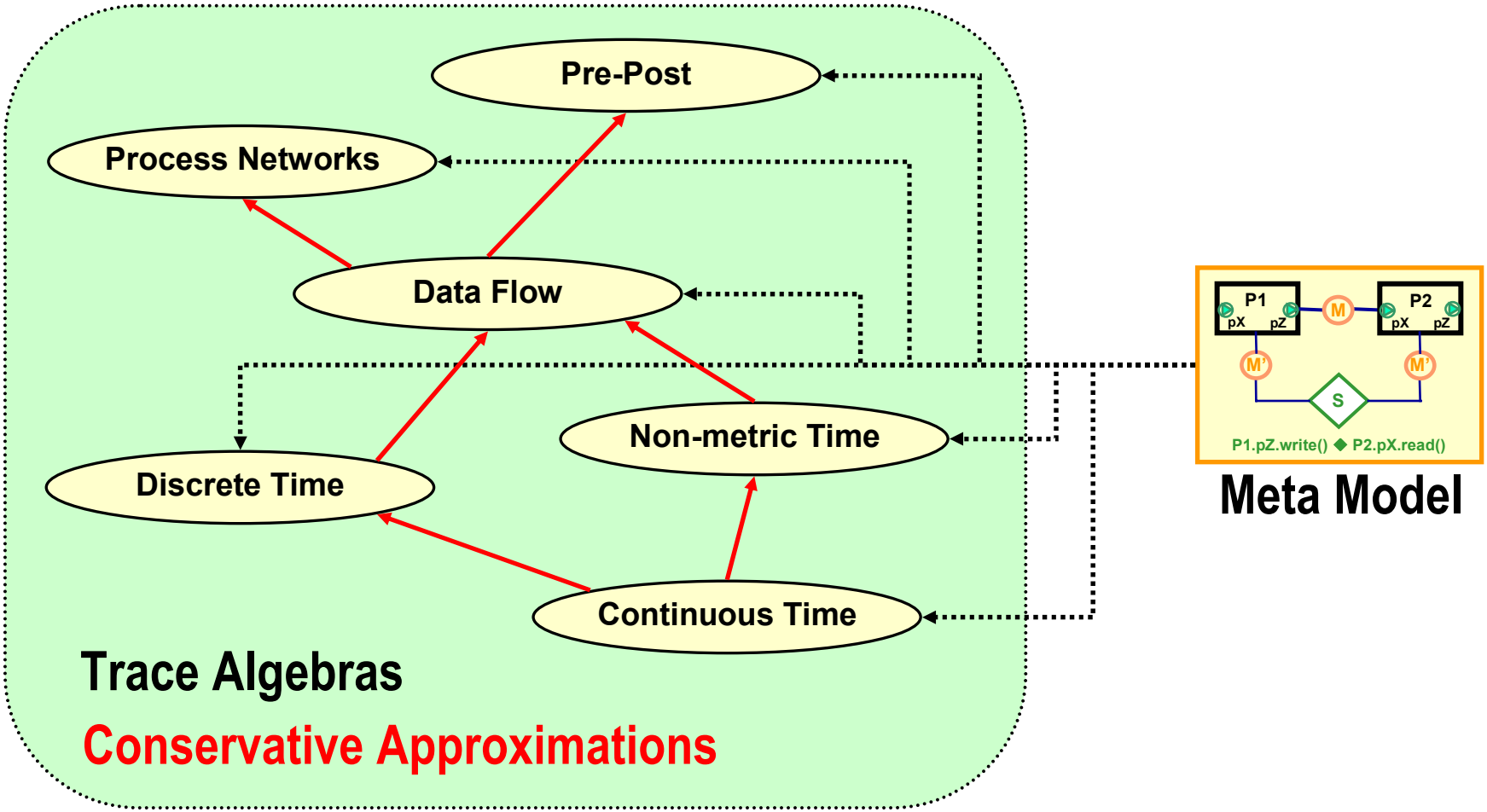
- Motivation
- Platform and Communication-based Design
  - Definition
  - Design Methodology
  - Network Platforms
  - Analog Platforms
- The Metropolis Framework
  - Metamodel
  - Verification
  - Synthesis
  - Theory of Models

Metropolis Semantics: Action Automata

- One for each action (statement, function call, expressions, etc.) of each process
- Composed synchronously
- May update shared memory variables:
  - process and media member variables
  - values of actions-expressions
- Have guards that depend on states of other action automata and memory variables



Algebraic Theory of Models

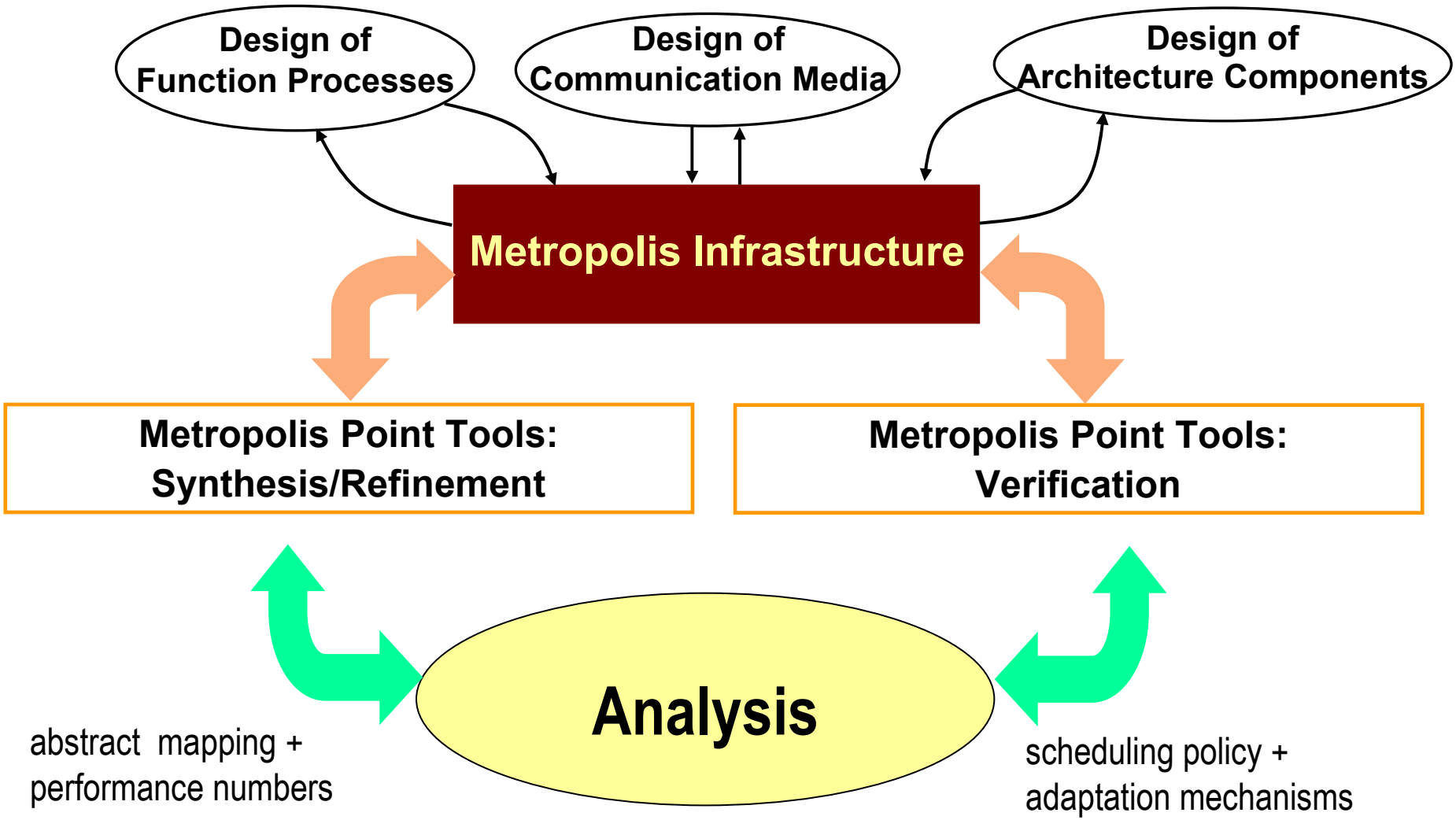


- Directions
  - Generalization to Agent Algebras
  - Generalization to sequential composition

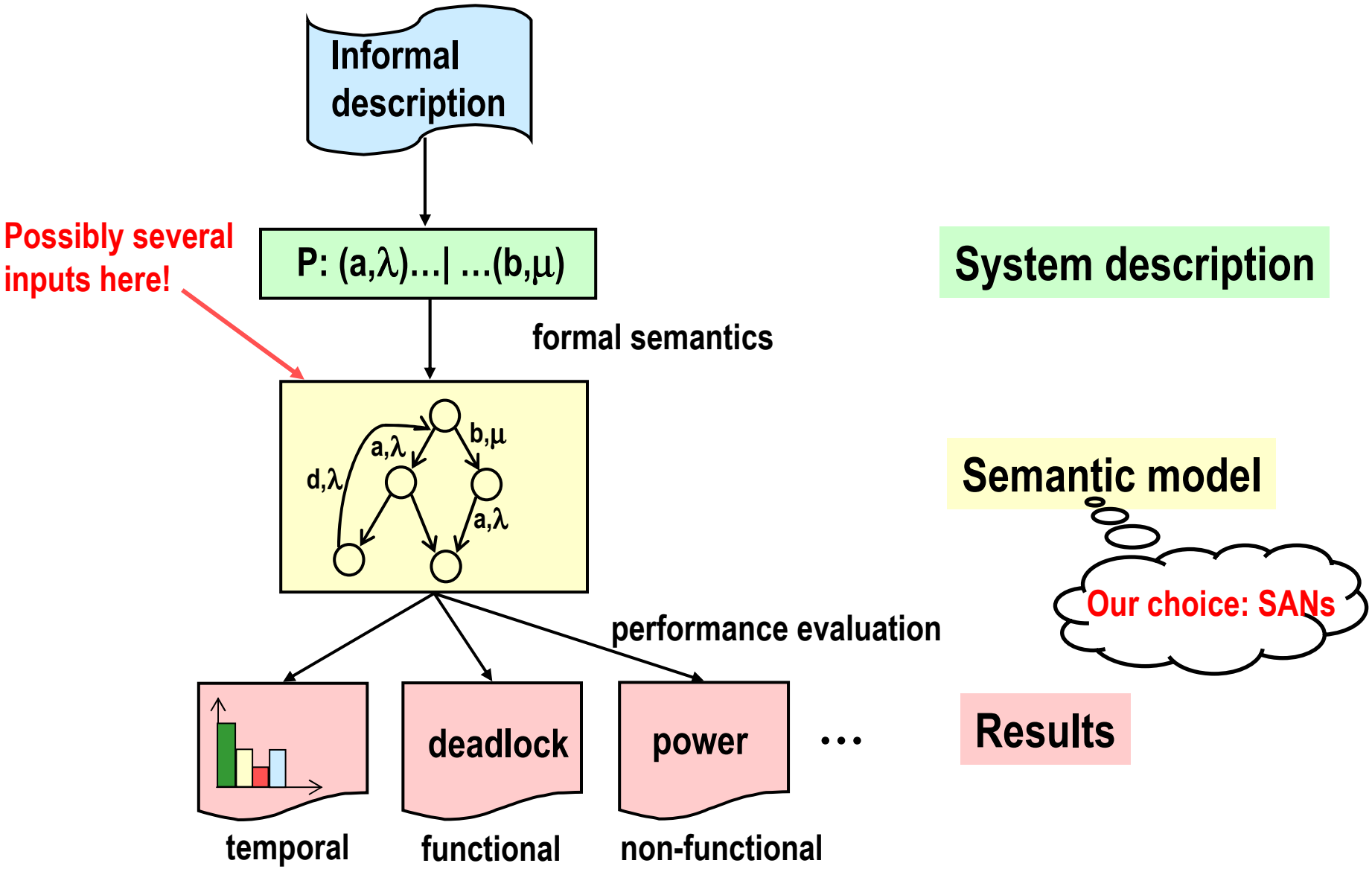
Outline

- Motivation
- Platform and Communication-based Design
  - Definition
  - Design Methodology
  - Network Platforms
  - Analog Platforms
- The Metropolis Framework
  - Metamodel
  - Verification
  - Synthesis
  - Theory of Models
  - Analysis (Radu Marculescu)
  - Communication Synthesis (Radu Marculescu)

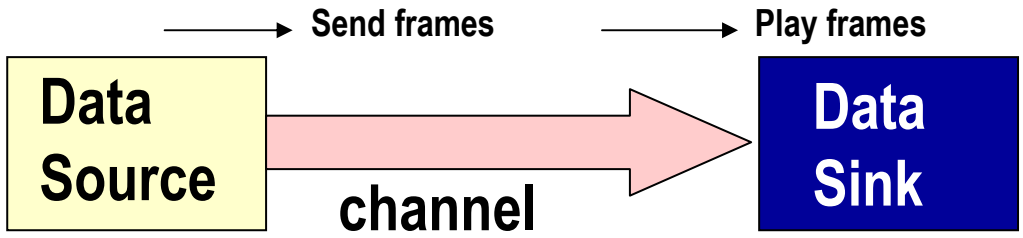
Where does the Analysis Module fit?



The Big Picture

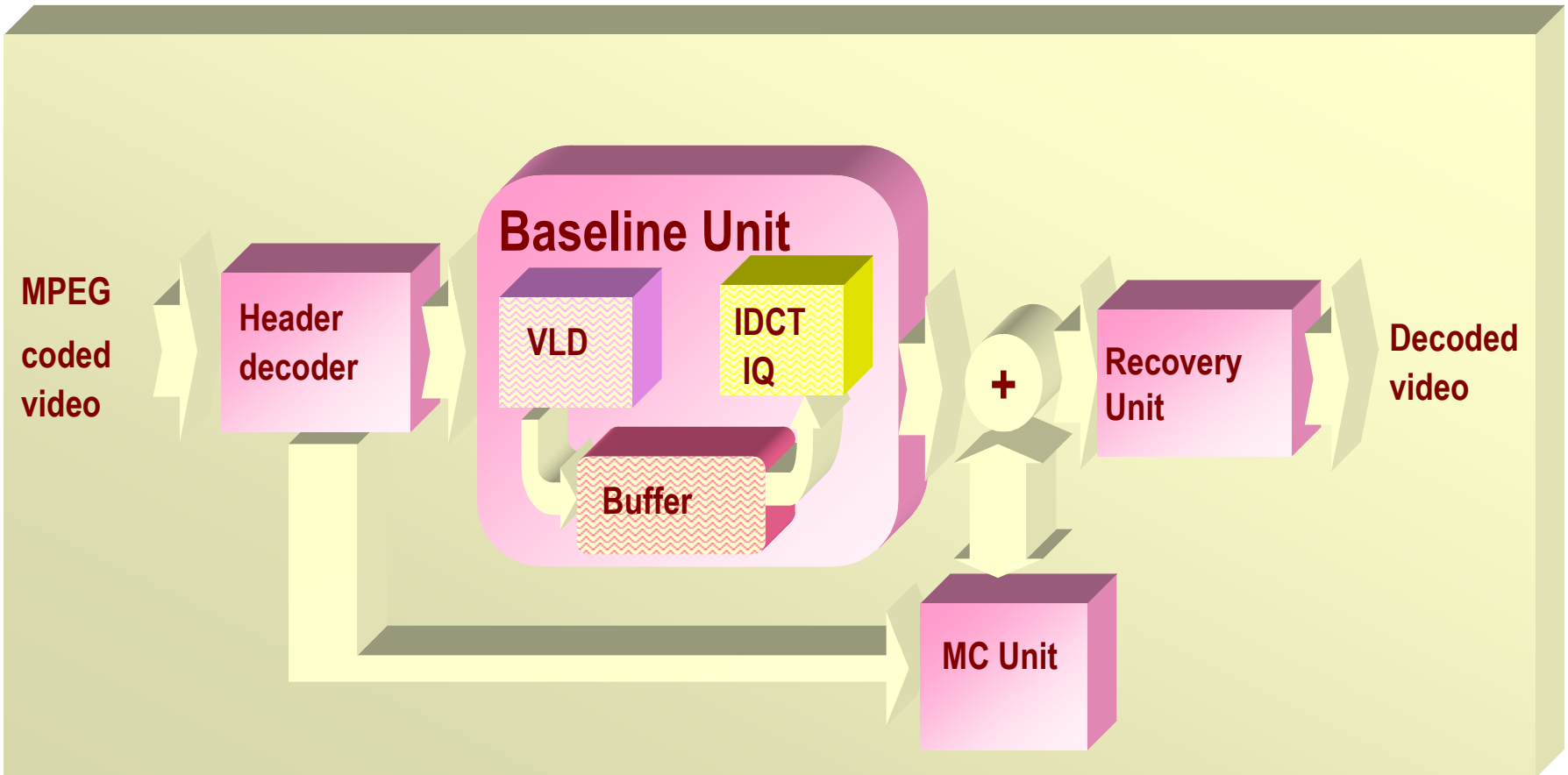


A Multimedia Stream: Informal Description



- Constraints on behavior -> QoS requirements
  - The data source repeatedly transmits data frames every 50ms (e.g. 20fps)
  - After generation of a frame, 5ms elapse before it is transmitted
  - Communication is asynchronous and channel may have errors
  - Successfully transmitted frames arrive at sink between 80ms and 90ms (latency)
  - If the number of frames arriving at the data sink is not within 15 to 20 fps (channel throughput), then an error should be reported
  - End-to-end latency should be between 100ms and 120 ms (this is the acceptable jitter on latency). A frame taking longer than 120ms is assumed to be lost
- Note
  - May change through system development (parameters that change should be easily identifiable and easy to change) (mostly a research issue)
  - Specify and work w/ probabilities (mostly a research issue)

What is our Driver Application?

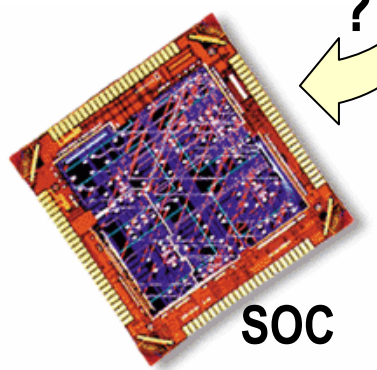


How do we Model the Application?

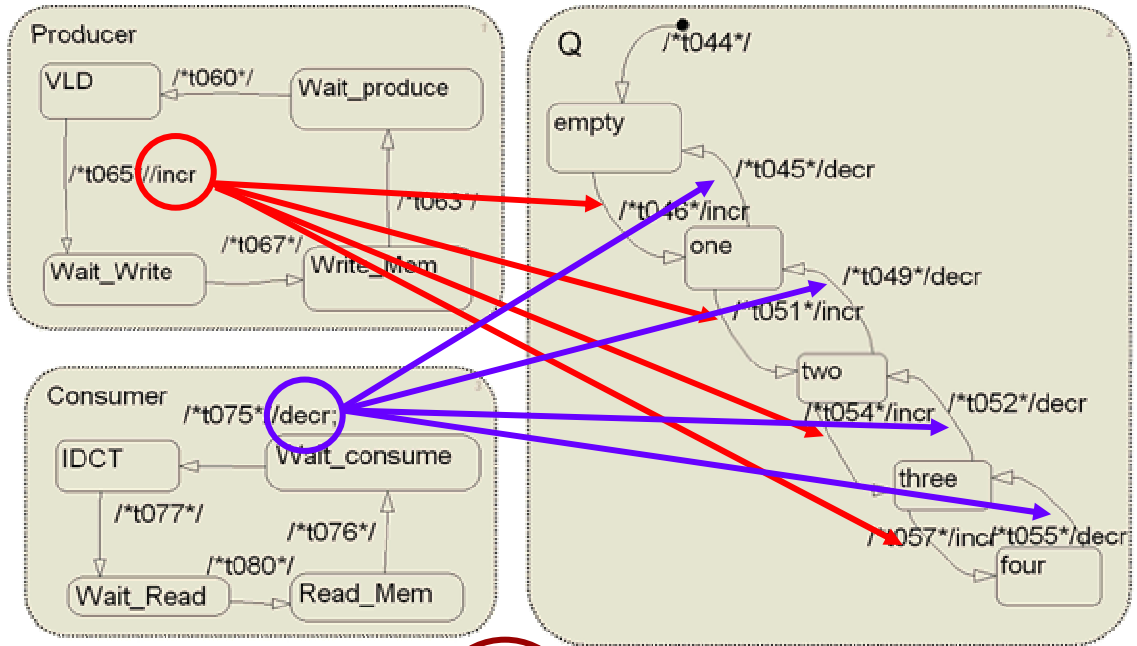
```
...
void Fast_IDCT(block)
short *block;
{
  int i;
    for (i=0; i<8; i++)
    idctrow(block+8*i);
  for (i=0; i<8; i++)
    idctcol(block+i);
}
...
```

This is *Hard*!

?



"Processes" *and* "medium" participate in communication!



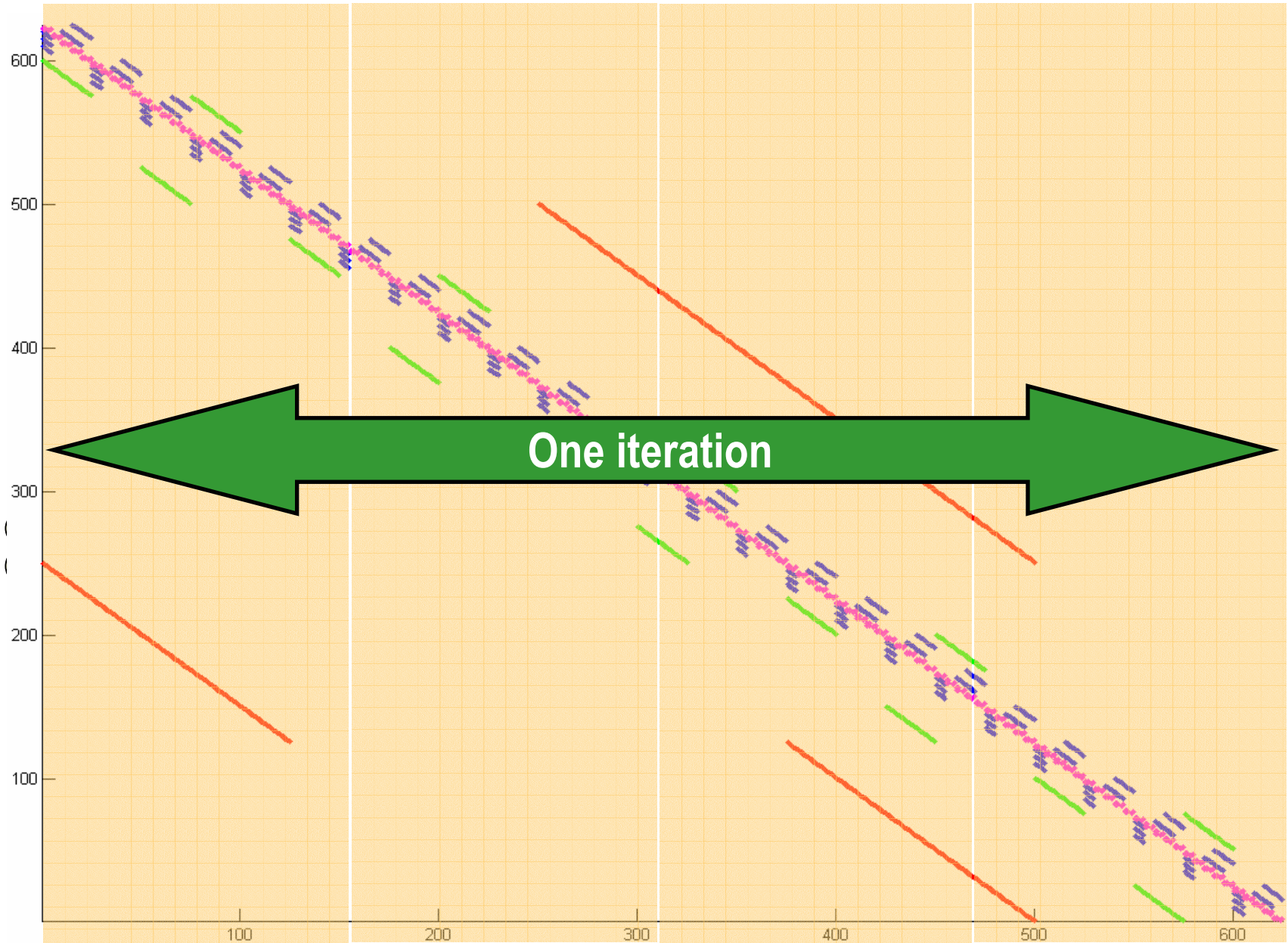
$$(C, Act, \{\xrightarrow{(\alpha,r)} : (\alpha, r) \in T\})$$

action

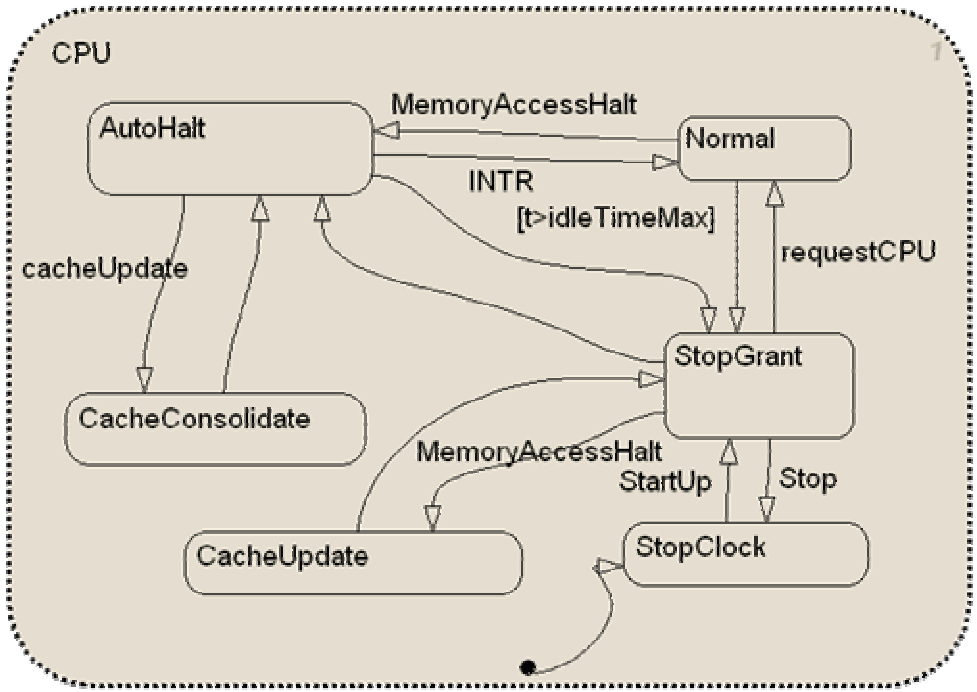
rate $\in R^+$

We talk about MCs and steady-state analysis because we assume exponentially distributed RVs (that is, $F(t) = 1 - e^{-rt}$)!

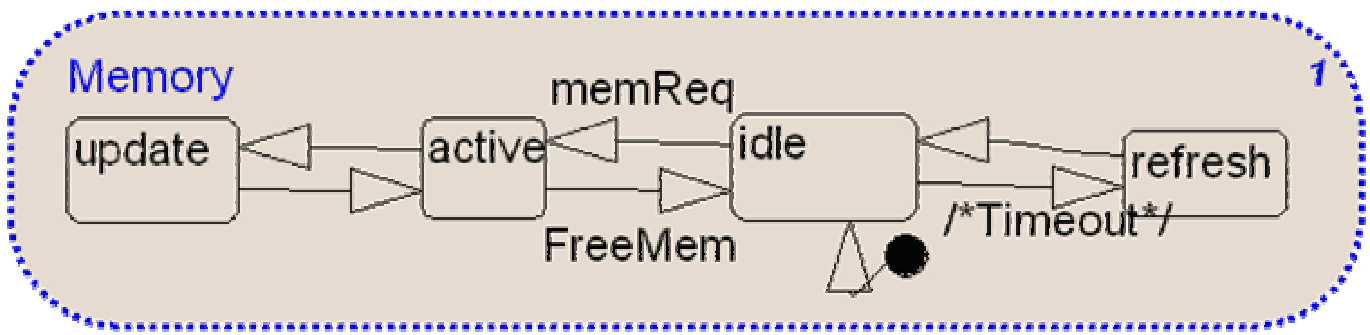
How do we Build a SAN-based Semantic Model?



How do we Model the Architecture?

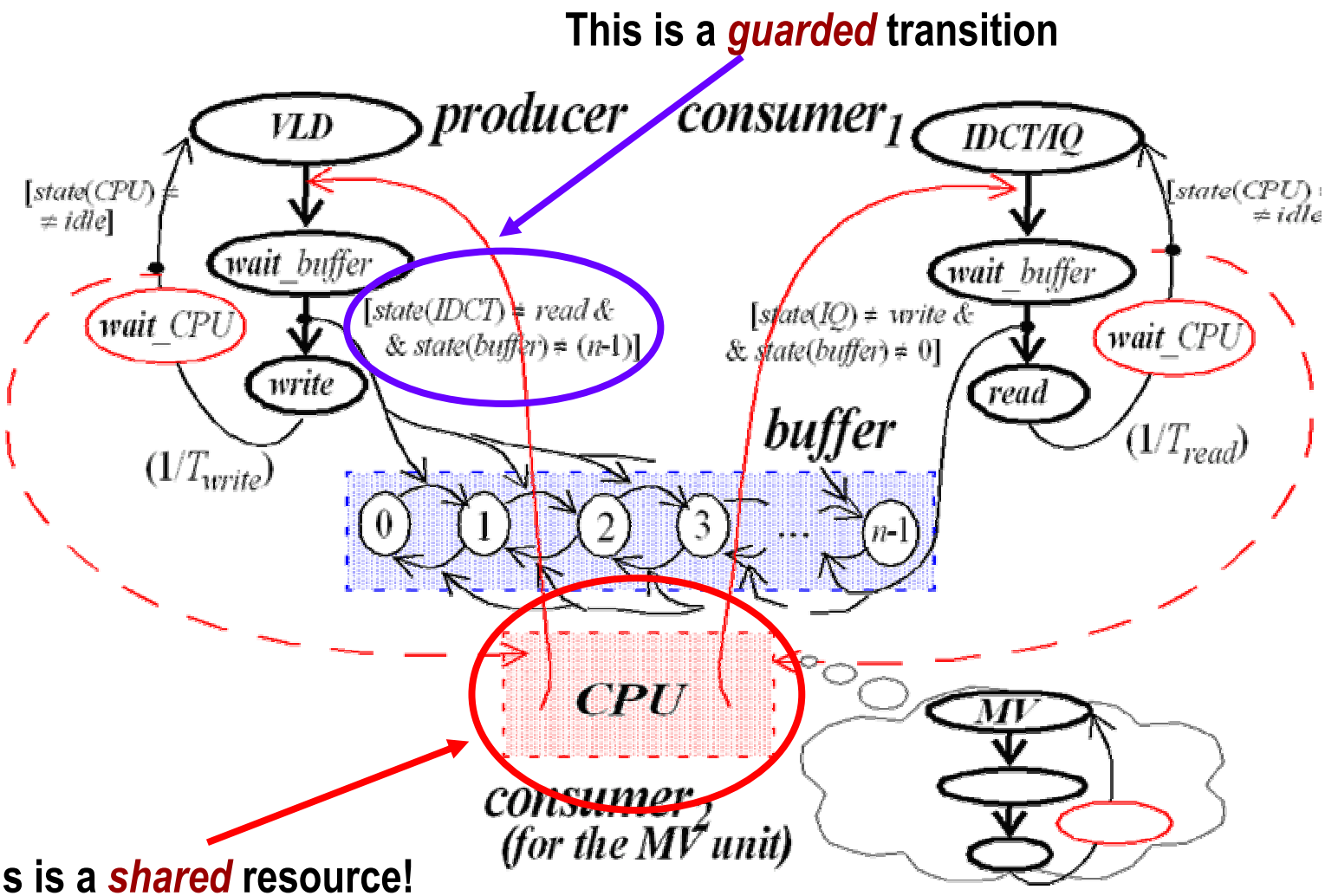


Model of a CPU

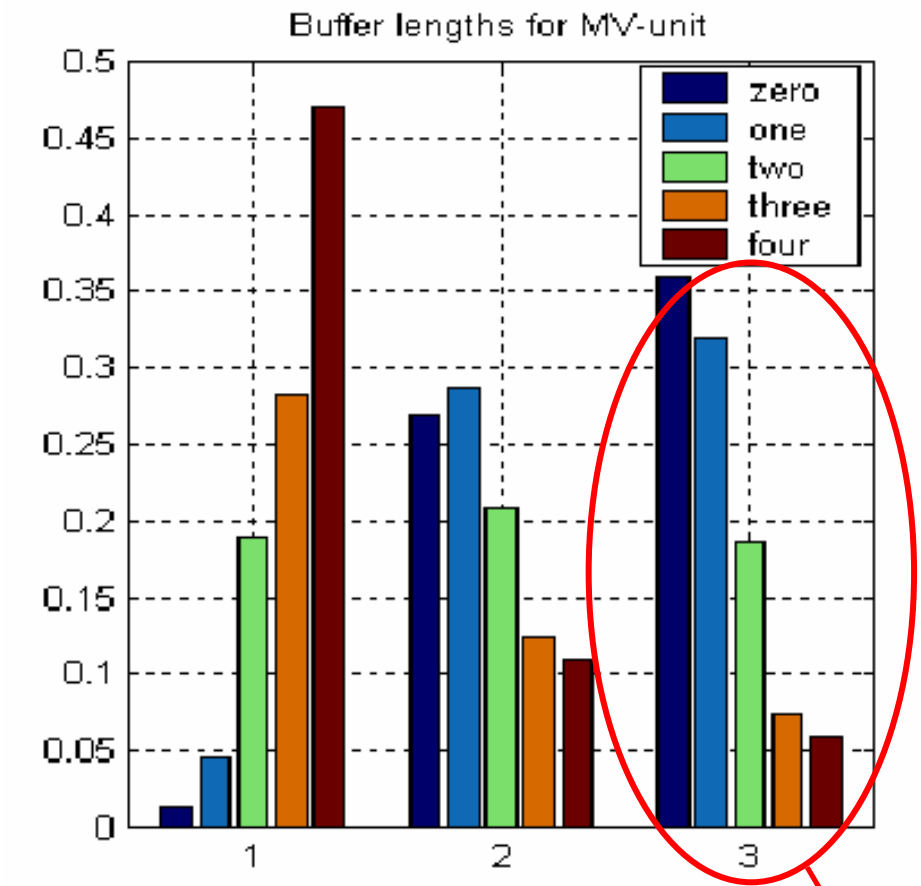


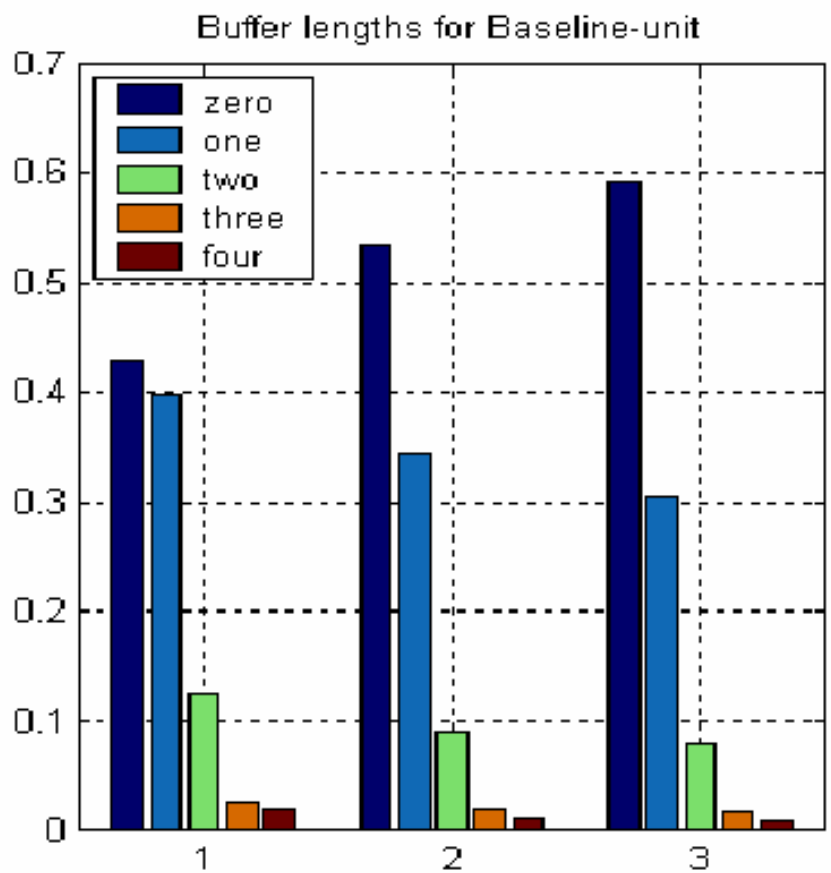
Model of the memory

Putting Everything Together



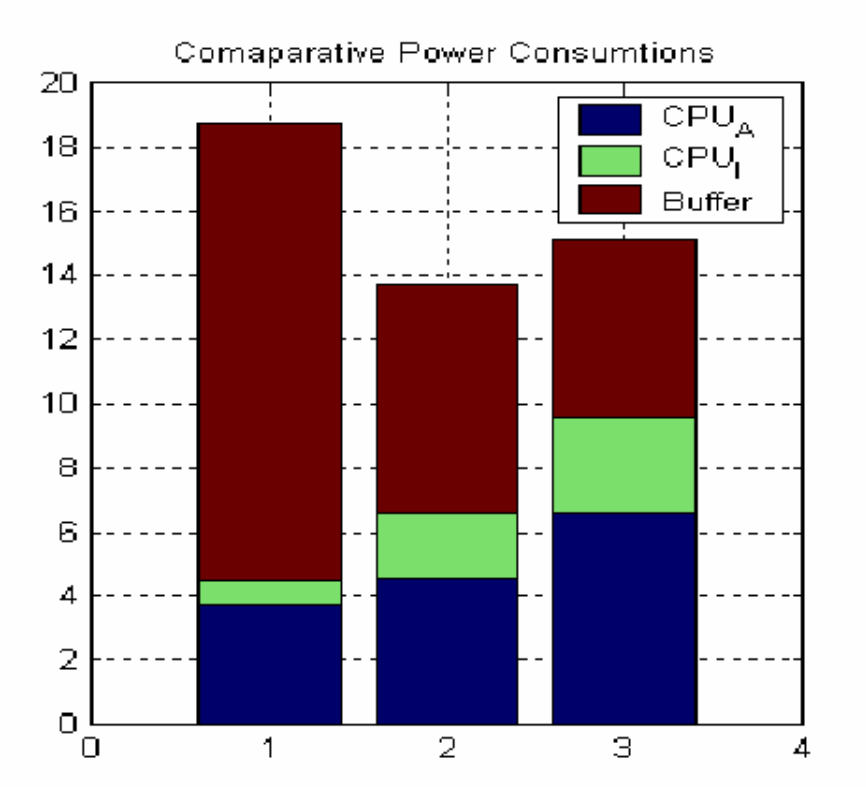
… and Getting the Results: the Node-Centric Perspective





buffer length 1.15

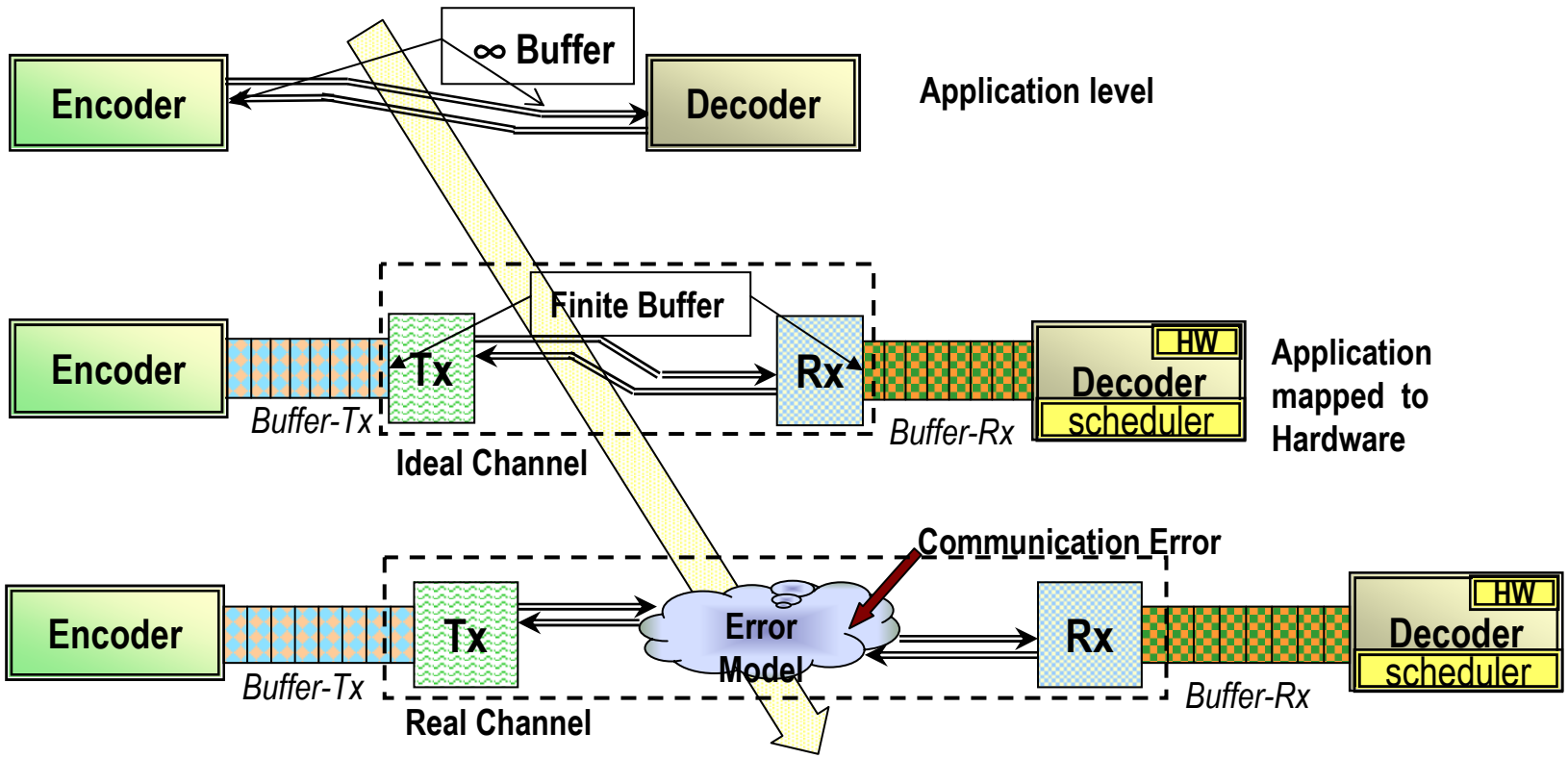
… and Getting the Results: the Node-Centric Perspective



$$P^{(k)} = \sum_{all\ \ i} \pi_i \cdot P_i + \sum_{all\ \ i,j} \lambda_{ij} \cdot P_{ij}$$

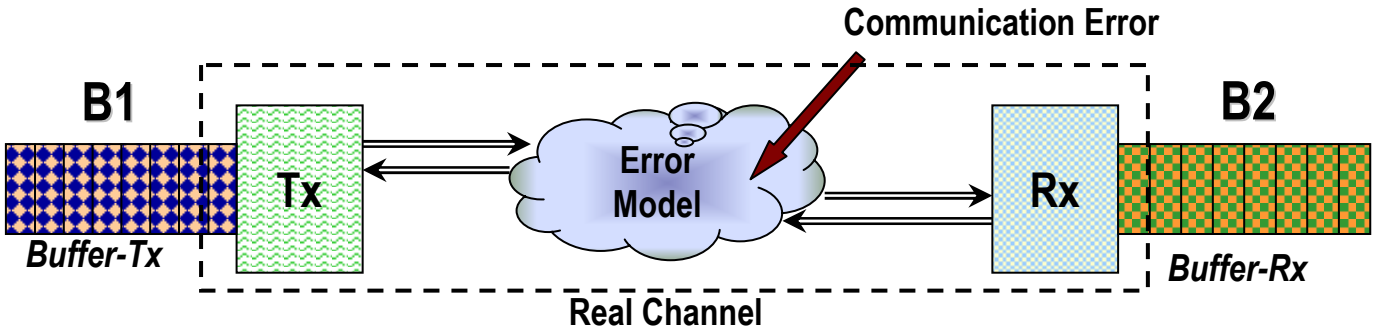
This comes from steady-state analysis

How about the Communication Channel?

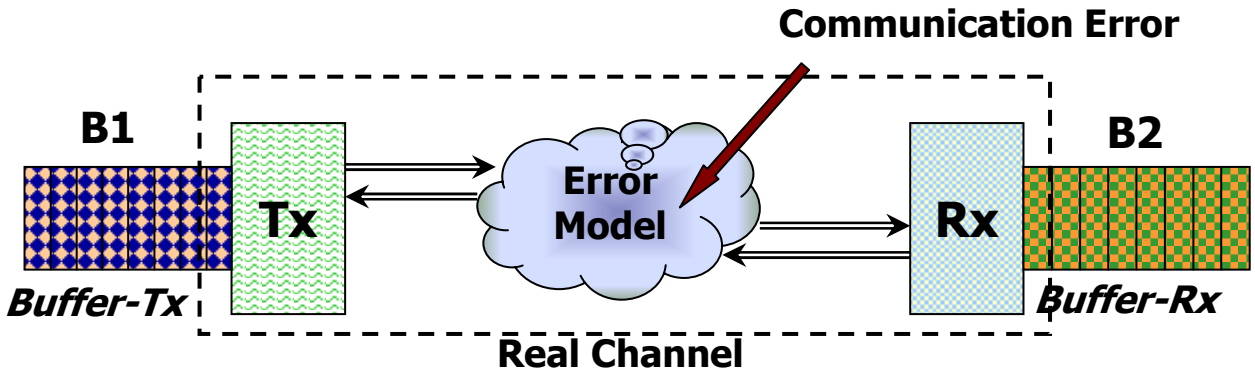


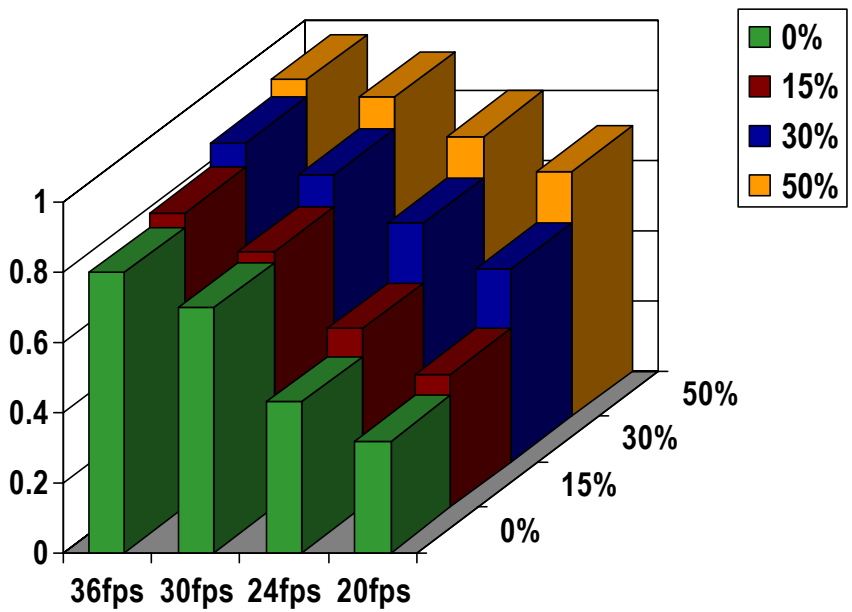
The 'node' behavior depends dramatically on channel behavior!

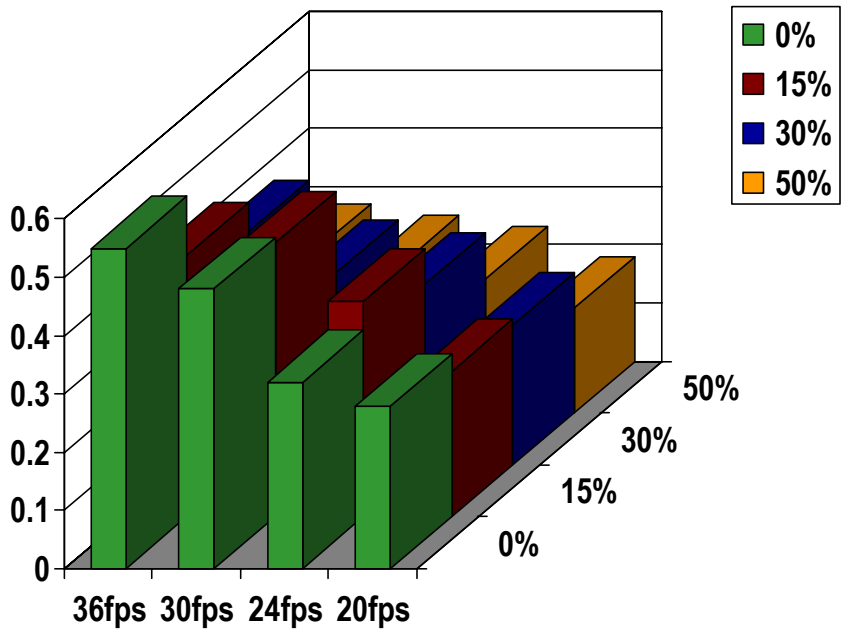
What are we Trying to Analyze?



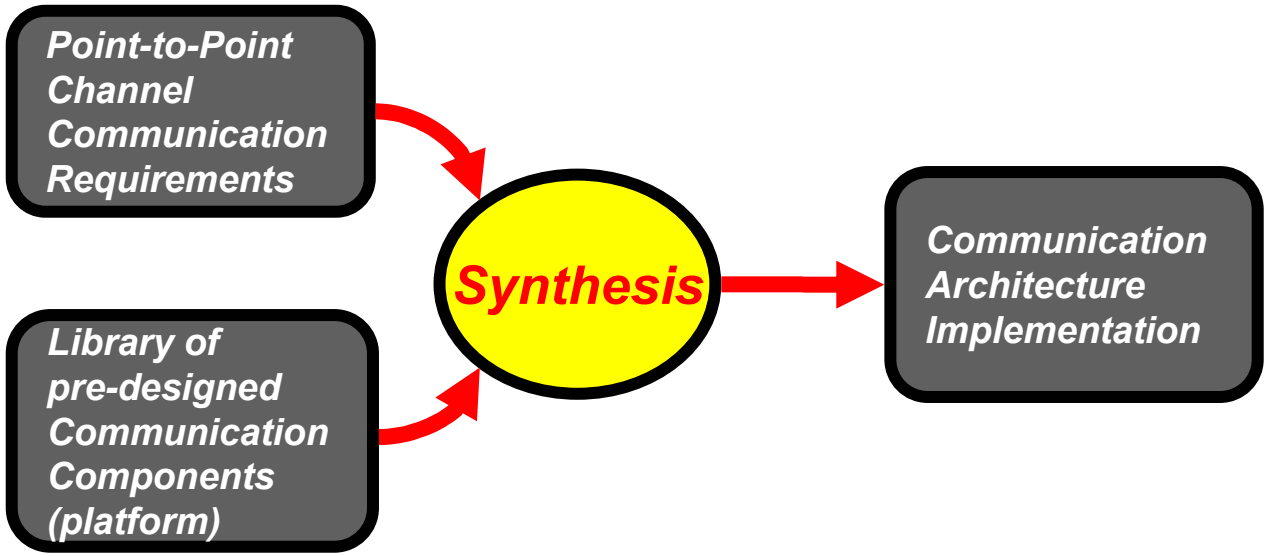
Again, the Network-Centric Perspective…





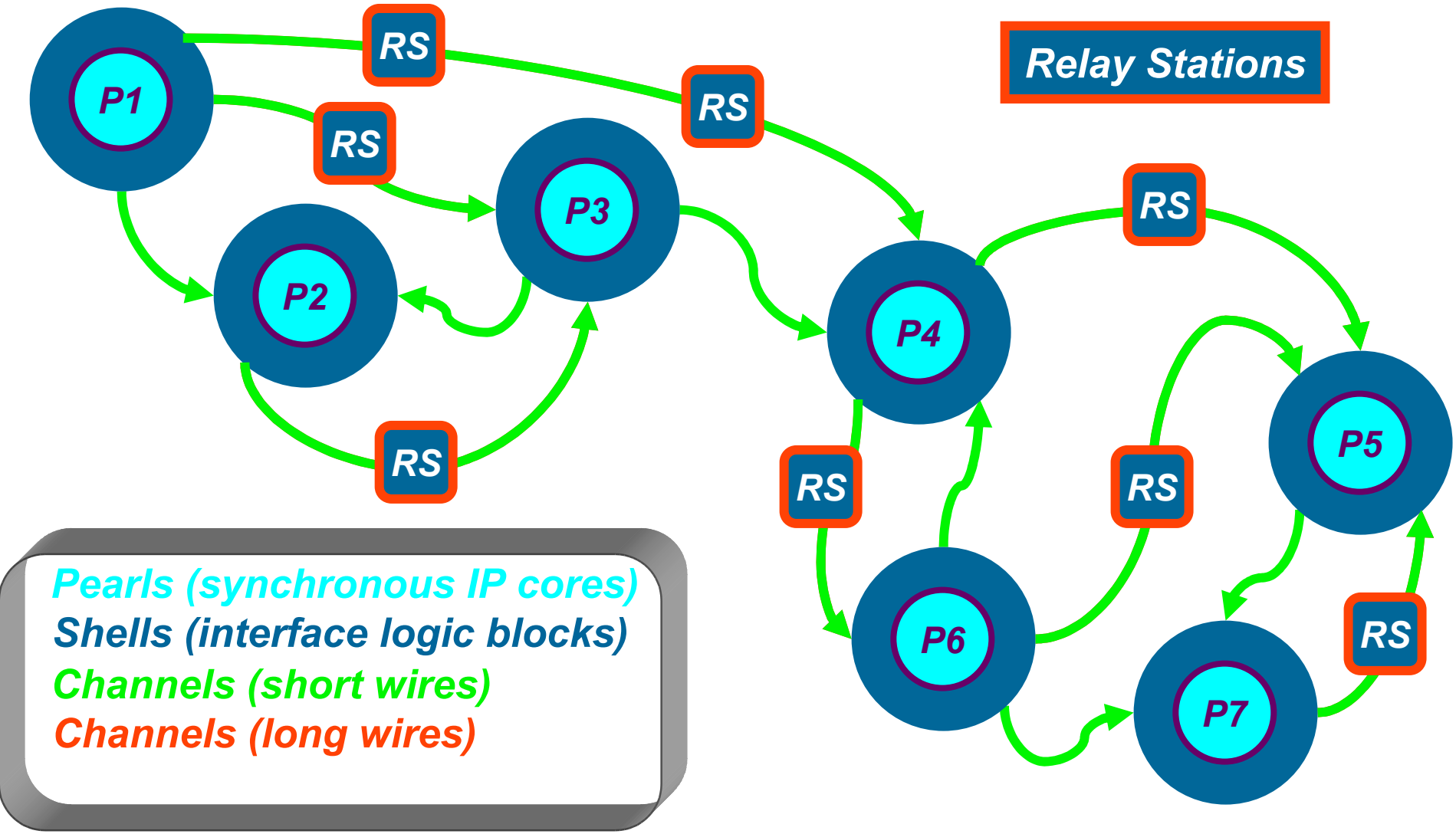


Constraint-Driven, Platform-based Synthesis of Communication Architectures



- System modules communicate by means of point-to-point channels
- High-level communication constraints for each channel in the system are captured as a Constraint Graph
- Similarly, the characteristics of all components in the Communication Library are captured as a set of feature resources together with their cost figure
- The synthesis result is represented by an Implementation Graph and obtained by solving a constrained optimization problem

Latency-Insensitive Design



Summary

- **Main ideas**
  - **Formal models emphasis**
    - **SAN analysis reduces the gap between simulation and verification**
  - **Communication architectures can be synthesized from requirements**
- **Current research**
  - **Exploiting regularity in system-level analysis**
    - **Efficient analysis enabled by symmetries (Nick Zamora)**
      - We expect orders of magnitude reduction in the complexity of the analysis
    - **Connect system-level analysis w/ lower levels of abstraction (Jingcao Hu)**
      - Efficient mapping techniques for regular architectures
  - **Communication Architectures: On- and off-chip**
    - **Analytical models for traffic analysis (Girish Varatkar)**
      - Architecture/design implications
      - Build fast and realistic simulators
    - **Communication architecture synthesis (Luca Carloni, Alessandro Pinto)**
    - **Protocol design for efficient on-chip communication (Luca Carloni, Tudor Dumitras)**

Summary

- Interdisciplinary, intercontinental project (10 institutions in 5 countries)
- Goal:
  - Design methodologies: abstraction levels, design problem formulations
  - EDA: formal methods for automatic synthesis and verification,
    a modeling mechanism: heterogeneous semantics, concurrency
- Primary thrusts:
  - Metropolis Meta Model:
    - Building blocks for modular descriptions of heterogeneous semantics
    - The internal modeling mechanism for function, architecture, and constraints
  - Design Methodology:
    - Multi-media digital systems
    - Wireless communication
    - Fault-tolerant automotive systems
    - Microprocessors
  - Formal Methods

Metropolis Project: Participants

- **UC Berkeley** (USA): methodologies, modeling, formal methods
- **CMU** (USA): methodologies, modeling, formal methods
- **Politecnico di Torino** (Italy): methodologies, modeling, formal methods
- **Universita Politecnica de Catalunya** (Spain): modeling, formal methods
- **UC Riverside** (USA): modeling, formal methods
- **Cadence Berkeley Labs** (USA): methodologies, modeling, formal methods
- **PARADES** (Italy): methodologies, modeling, formal methods
- **ST** (USA, France-Italy): methodologies, modeling
- **Philips** (USA, Netherlands): methodologies (multi-media)
- **Nokia** (USA, Finland): methodologies (wireless communication)
- **BWRC** (USA): methodologies (wireless communication)
- **Magneti-Marelli** (Italy): methodologies (power train control)
- **BMW** (USA, Germany): methodologies (fault-tolerant automotive controls)
- **Intel** (USA): methodologies (microprocessors)
- **Cypress** (USA): methodologies (network processors, USB platforms)
- **Honeywell** (USA): methodologies (FADEC)

References

- Platform-Based Design
  - Alberto Sangiovanni-Vincentelli, “Defining Platform-Based Design”, EE Design, March 5, 2002.
  - Alberto Sangiovanni-Vincentelli and Grant Martin, A Vision for Embedded Systems: Platform-Based Design and Software Methodology, IEEE Design and Test of Computers, Volume 18, Number 6, November-December, 2001, pp. 23-33
  - K. Keutzer, S. Malik, A. R. Newton, J. M. Rabaey, and A. Sangiovanni-Vincentelli, “System Level Design: Orthogonalization of Concerns and Platform-Based Design”, IEEE Transactions on Computer-Aided Design, Vol. 19, No. 12, December 2000
- Metropolis
  - F. Balarin et al., “Modeling and Designing Heterogeneous Systems”, in J. Cortadella and A. Yakovlev editors, Advances in Concurrency and System Design, Springer-Verlag, 2002.
  - F. Balarin et al., “Constraints Specification at Higher Levels of Abstraction”, in Proceedings of the IEEE International High Level Design Validation and Test Workshop, Monterey, California, November 7-9, 2001.

References

- Quasi-Static Scheduling
  - C. Passerone, Y. Watanabe, L. Lavagno, *"Generation of Minimal Size Code for Schedule Graphs"*, Proceedings of the Design Automation and Test in Europe, Munich, Germany, 2001.
  - Cortadella et al: *"Task generation and compile-time scheduling for mixed data-control embedded software"*, Proceedings of the 37th Design Automation Conference, Los Angeles, CA, June 2000.
- Application Driven Scheduling
  - L. Palopoli, C. Pinello, A. Sangiovanni Vincentelli, L. Elghaoui, A. Bicchi, "Synthesis of robust control systems under resource constraints", HSCC2002, Lecture Notes in Computer Science, March 2002.
- Algebraic Theory
  - J. Burch, R. Passerone, A. Sangiovanni-Vincentelli, "Using Multiple Levels of Abstraction in Embedded Software Design", Proceedings of the First International Workshop on Embedded Software, Tahoe City, CA, October 2001.
  - J. Burch, R. Passerone, A. Sangiovanni-Vincentelli, "Overcoming Heterophobia: Modeling Concurrency in Heterogeneous Systems", Proceedings of *Application of Concurrency to System Design*, Newcastle (UK), 2001.

References

- **Power/performance analysis for platform-based design**
  - R. Marculescu, A. Nandi, L. Lavagno, and A. Sangiovanni-Vincentelli, 'System-Level Power/Performance Analysis of Portable Multimedia Systems Communicating over Wireless Channels', in Proc. ICCAD, Nov. 2001.
  - A. Nandi, R. Marculescu, 'System-Level Power/Performance Analysis for Embedded Systems Design', in Proc. DAC, June 2001.
  - R. Marculescu, A. Nandi, 'Probabilistic Application Modeling for System-Level Performance Analysis', in Proc. DATE, March 2001.
- **On-chip communication**
  - G. Varatkar and R. Marculescu, 'Traffic Analysis for On-chip Networks Design of Multimedia Applications', in Proc. DAC, June 2002.
  - J. Hu, Y. Deng, R. Marculescu, 'System-Level Point-to-Point Communication Synthesis Using Floorplanning Information', in Proc. ASP-DAC, Jan. 2002.

References – continue

- **Constraint-Driven Communication Synthesis**
  - A. Pinto, L.P. Carloni, and A. Sangiovanni-Vincentelli, 'Constraint-Driven Communication Synthesis', in Proc. DAC June 2002.
- **Latency-Insensitive Design**
  - L.P. Carloni, K. McMillan and A. Sangiovanni-Vincentelli, 'Theory of Latency-Insensitive Design', IEEE Transactions On Computer-Aided Design, Vol. 20, No. 9, Sept. 2001.
  - L.P. Carloni and A. Sangiovanni-Vincentelli, 'Performance Analysis of Latency-Insensitive Systems', in Proc. DAC June 2000.
- **Communication Driven Hardware Synthesis**
  - SRC Technical Report, Report on problem formulation, state of the art and theory review, SRC Task 837.001, Sept. 2001
  - SRC Technical Report, Report on Case Study Specification, SRC Task 837.001, Sept 2001
  - Graduate Computer Architecture Project Report, Available at: http://www.cs.berkeley.edu/~densmore/documents/252_Final.pdf