Discussion on: "Quo Vadis, SLD? Reasoning About the Trends and Challenges of System Level Design"

by Alberto Sangiovanni-Vincentelli

By Alberto Puggelli

Outline

- SLD challenges
- Platform Based Design (PBD)
  - Case study: Wireless Sensor Network
- Leveraging state-of-the-art CAD
- Metropolis
  - Case study: JPEG Encoder

SLD Challenge

- Establish a further layer of abstraction
  - Behavioral layer (mathematical background)
- Expand the market towards different domains
  - Heterogeneity: Mechanical, Health Care, Chemical
- New design methodology
  - Orthogonalization of concerns
  - Formal design process
  - Plug and play among different design methods
  - Synthesis

Behavioral Layer

- Functionality and time-to-market
  - Explore the design space → Partitioning
  - Cost → yield, power consumption, etc.
  - Formal Verification - Certification
  - Reusability
- Smoothness among players
  - Mobile
  - Automotive (possible shift of added value → instability) OEM, Tier1, Tier2

New Markets

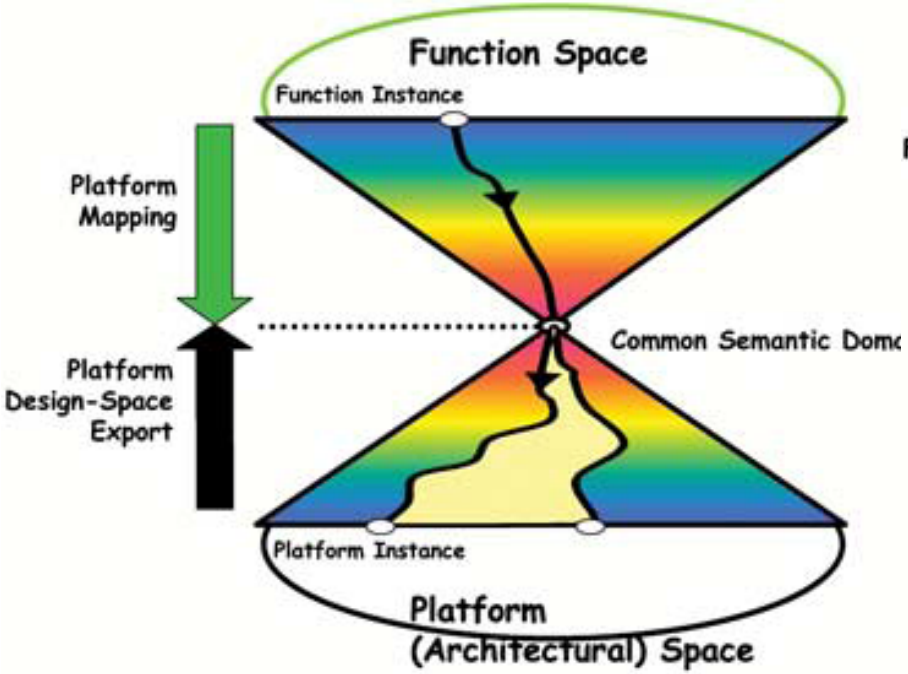
- Increase of benefits might drive the methodology shift (EDA companies)
- Embedded system are the added value (OEMs)

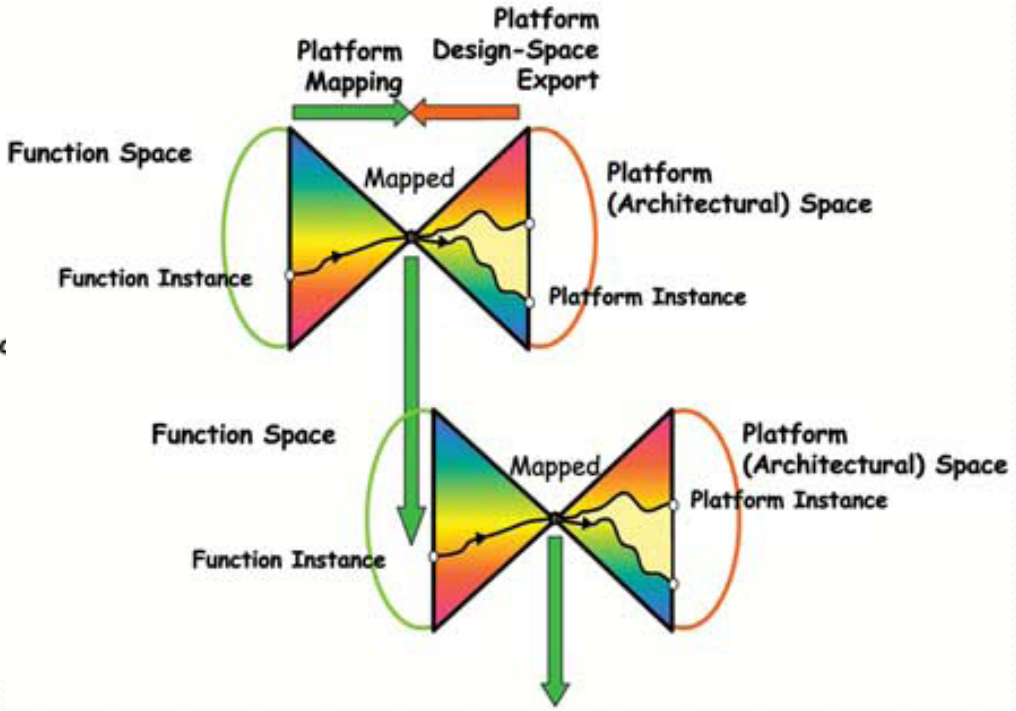
Why a new design methodology?

- SW & HW kept separately
- SW
  - Millions of customized lines of codes
  - Verification is extremely difficult
  - Certification of the process, not of the functionality!
- HW
  - Various design methodology
  - Various semantics
- Extremely slow and costly process with no chance of reusability

Platform based design: PBD

- Orthogonalize functionality and platform representations





Platform

- Library of components
  - Computational
  - Communication
- Each component has performance metrics + supported functionality
- Possibility of having place holders
- Platform instance: specific selection of components in a library to get a given functionality (set the parameters of components)

Wireless Sensor Network

- Functional Model: SNSP
  - Formally describes the interaction between controllers, actuators and sensors.
- Protocol Platform: SNAPP
- Sensor Platform: SNIP
- Step 1: check for functional support + derivation of constraints
- Step 2: select sensors and a protocol that are able to implement the functional description

Related strategies

- Model Driven Development (MDD)
  - Orthogonalization of concerns
  - Mapping functionalities
- Domain Specific Languages (DSL)
  - Use of a metamodels
- MILAN framework
  - Orthogonalize computation and communication
- Already implemented in HW and SW
- Need for a common semantic and a mathematical description of the interactions for embedded systems.

Wrap up

- PBD represents a set of contracts among different players along the supply chain
  - Supplier defines performance metrics to be used by the client
  - The client may ask the supplier for specific performance metrics
- More degrees of freedom → design space exploration
- Open issue: finding the right layers

Leveraging state-of-the-art CAD

- Functional
  - HW
    - SystemC different semantics to account for hardware concurrency and execution time. It allows verification but it has to be manipulate to be synthesized
    - HDL abstraction of RTL → synthesis is possible
  - Embedded SW
    - Need for concurrency, multiprocessing, multithreading
    - Verification by construction → synchronous languages
    - Computation and communication don't overlap ("critical path" is defined)

Leveraging state-of-the-art CAD (2)

- Models of Computation (MoC)
  - Need for orthogonalization from architecture
  - Trade expressivity with ease of synthesis (DE, FSM)
  - Need for mixing different MoC (heterogeneous systems)
    - Interface automata: shift the problem to interface and see whether 2 interfaces can communicate
    - Behavioral description: formal sets of behaviors for each subsystems that can be intersected to get the system description.
  - Ptolemy II: more MoC are supported
    - Each subsystem is a thread that can communicate with other systems by sending messages.

Leveraging state-of-the-art CAD (3)

- Architecture
  - Netlist of connections among components
  - Capabilities of each component
  - Cost: performance metrics of each component and communication medium
  - SW
    - UML, ADL, ECLIPSE

Leveraging state-of-the-art CAD

- Architecture (2)
  - HW
    - TLM
    - Simulation Engine (SystemC): need for a library of models and of interconnect, each with cost metrics
    - Communication-based-design: NoC to verify whether the behavior of two components is preserved when connected together → assume-guarantee approach

Leveraging state-of-the-art CAD

- Mapping
  - Scheduling
  - Giotto: enforce common semantic language
  - Automatic Optimization
    - Common mathematical description between functionality and platform (boolean algebra)
    - Look for primitives (NAND2)
    - Covering problem

Metropolis

- Support different MoC → Metamodel (MMM)
- Support different layers of abstraction → PBD
  - Functional model
  - Library
- Evaluation of the cost of a mapped design
- Functionality + Constraint driven mapping

Metropolis Meta Model (MMM)

- Support different MoC →more language for different applications
- Functionality description
  - Processes, communication media, netlists (refinement)
  - Constraint specification
  - Set of executions, each consisting of events (call for services)
- Architectural model
  - Scheduled Netlist: computational and communication components
  - Scheduling Netlist: performance metrics (quantity managers)

MMM (cont.)

- Mapping
  - Synchronize components to coordinate their interfaces
  - The system does what is modeled in the functional model according to the constraints given by the architectural model.
- Costraints
  - Define quantities that are not explicit in the MoC semantic (e.g. time)
  - Propagation
  - Specify constraints or check for validity in a specific implementation (LOC)

Tools

- Parser
  - Check the MMM and creates an abstract syntax tree
- Simulator
  - Enforce LTL and LOC (prevent or check illegal behaviors)
- Refinement verification tool
  - Check whether model B is a refinement of model A

Methodology

- Design flow
  - Specify behaviors
  - Execute abstractions by using constraints
  - Use best synthesis algorithm for a given domain
- Common semantic background
  - Plug-in different subsystems
  - Incorporate external tools

JPEG Encoder

- Map the encoding algorithm on Intel MXP5800 architecture
- Step 1:
  - Behavioral Model of the algorithm with a specific semantic
  - Describe the architecture in terms of processes, media and quantity managers
    - running time is the main concern → global time manager

JPEG Encoder (cont.)

- Step 2: Mapping
  - Synchronization constraints among read, write and execution + memory allocation and register location
  - Execution order between different tasks