Designing Multi-Processor and Multi-Core Systems-on-Chip

Andreas Gerstlauer
Electrical and Computer Engineering
University of Texas at Austin
http://www.ece.utexas.edu/~gerstl

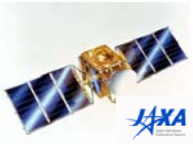
with contributions from

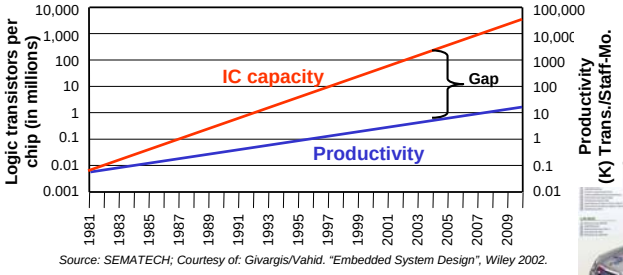
Christian Haubelt
University of Erlangen-Nuremberg

Additional source material from the book:
D. Gajski, S. Abdi, A. Gerstlauer, G. Schirner. "Embedded System Design: Modeling, Synthesis, Verification," Springer 2009.

Embedded System Design

- **Embedded systems**
  - Ubiquitous, application-specific
  - Tight constraints, market pressures
- **Growing system complexities**
  - Increasing demands & technological advances
  - ➢ Multi-Core/Multi-Processor System-On-Chip (MPSoC)

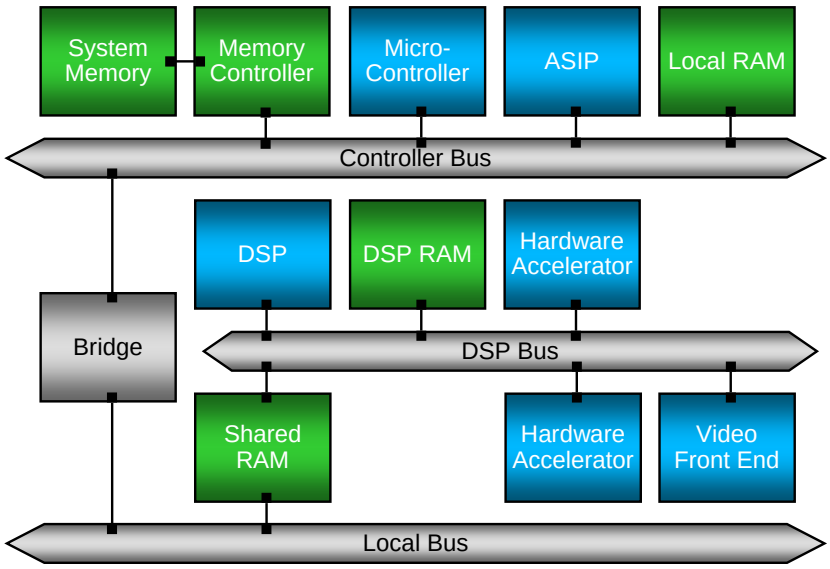




Source: SEMATECH; Courtesy of: Givargis/Vahid. "Embedded System Design", Wiley 2002.

➢ **Raising the level of abstraction [ITRS07]**

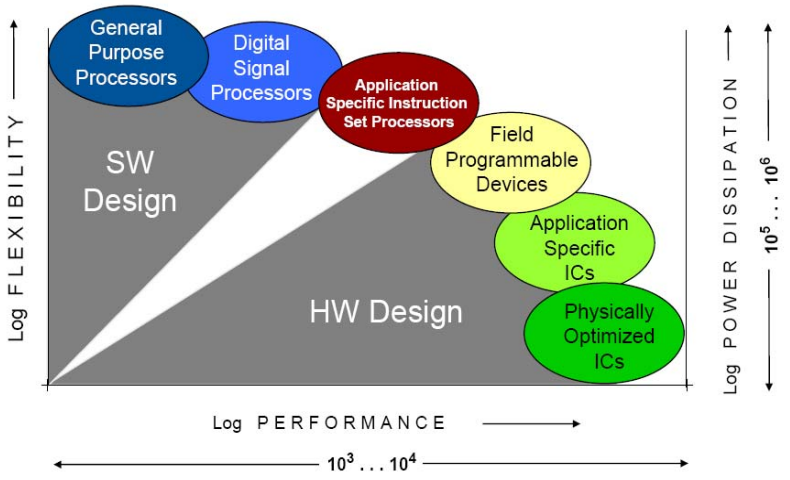
MPSoC Platform



MPSoC Terminology

- **Multi-processor**
  - Heterogeneous, asymmetric multi-processing (AMP)
  - Distributed memory and operating system
- **Multi-core**
  - Homogeneous, symmetric multi-processing (SMP)
  - Shared memory and operating system
  - ➢ Multi-core processors in a multi-processor system
- **Many-core**
  - > 10 cores per processor…

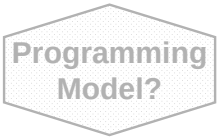
Processor Implementation Options

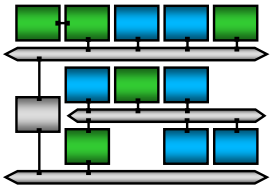


Source: T. Noll, RWTH Aachen, via R. Leupers, "From ASIP to MPSoC", Computer Engineering Colloquium, TU Delft, 2006

MPSoC Challenges

- **Complexity**
  - High degree of parallelism at various levels
- **Heterogeneity**
  - Of components
  - Of tools
- **Low-level communication mechanisms**
- **Programming model**

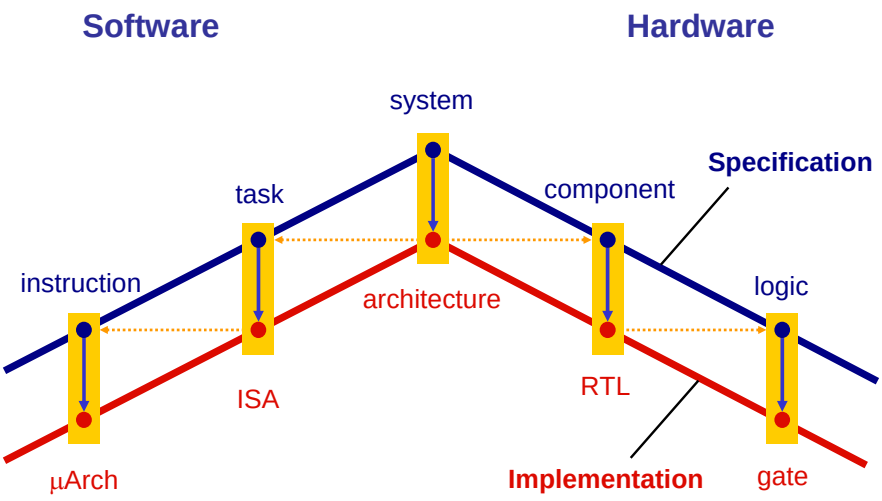




Outline

- ✓ **Introduction**
- **System design flow**
  - Electronic system-level (ESL) design
- **ESL design**
  - Modeling
  - Synthesis
  - Verification
- **ESL landscape**
- **Summary and conclusions**

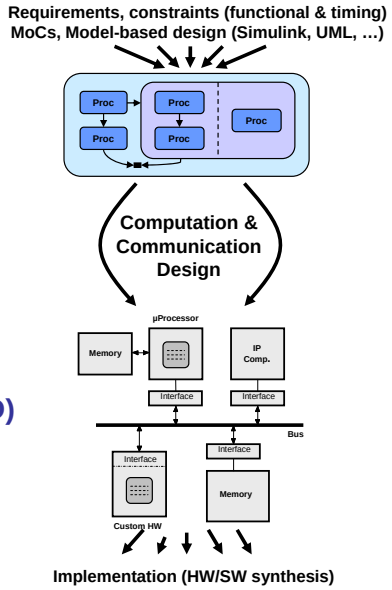
Double Roof Model



System-Level Design (SLD)

- **From system specification**
  - Functionality, behavior
    - Algorithms (process network)
  - Constraints
- **To system architecture**
  - Structure
    - Spatial and temporal order
    - Components and connectivity
    - Across hardware and software
  - Quality metrics

➢ **Design automation (EDA/CAD) at the system level**
  - Modeling and simulation
  - Synthesis and optimization
  - Verification



System Design Languages

- **Netlists**
  - Structure only: components and connectivity
  - ➢ Gate-level [EDIF], system-level [SPIRIT/XML]
- **Hardware description languages (HDLs)**
  - Event-driven behavior: signals/wires, clocks
  - Register-transfer level (RTL): boolean logic
  - ➢ Discrete event [VHDL, Verilog]
- **System-level design languages (SLDLs)**
  - Software behavior: sequential functionality/programs
  - ➢ C-based [SpecC, SystemC, SystemVerilog]

Outline

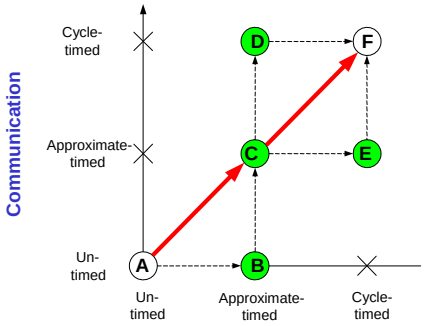
- ✓ Introduction
- ✓ System design flow
- **ESL design**
  - Modeling
  - Synthesis
  - Verification
- ESL landscape
- Summary and conclusions

System Modeling

- **Design models as abstraction of a design instance**
  - Representation for validation and analysis
  - Specification for further implementation
  - ➢ Documentation & specification
- ➢ **Systematic modeling flow and methodology**
  - Set of models and design steps
  - ➢ From specification to implementation
- ➢ **Well-defined, sound abstractions**
  - Data & timing
  - Computation & communication
  - ➢ Exploration, synthesis and verification

System Design Flow

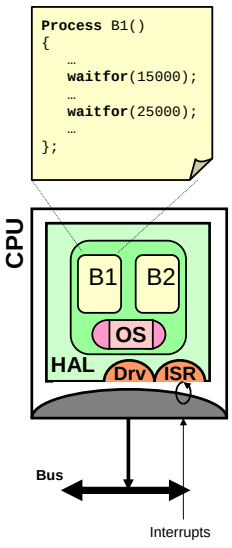
- **Abstraction based on level of detail & granularity**
  - Separation of computation and communication
- **System design flow**
  - Path from model A to model F



A. System specification model
B. Timed functional model
C. Transaction-level model (TLM)
D. Bus cycle-accurate model (BCAM)
E. Computation cycle-accurate model (CCAM)
F. Cycle-accurate model (CAM)

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

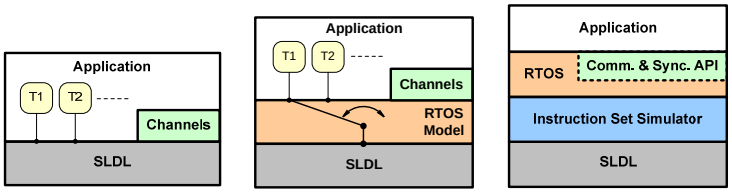
Computation Layers



- **Application model**
  - Model of Computation (MoC)
    - Process-/state-based [KPN, SDF, FSM, …]
  - Back-annotated execution timing
    - Timing granularity (basic block level)
- **Processor model**
  - Operating system
    - Middleware (MoC runtime)
    - Real-time multi-tasking (RTOS)
    - Bus drivers
  - Hardware abstraction layer (HAL)
    - Interrupt handlers
    - Media accesses
  - Processor hardware
    - Bus interfaces (I/O state machines)
    - Interrupt suspension and timing

OS Modeling

- **High-level RTOS abstraction**



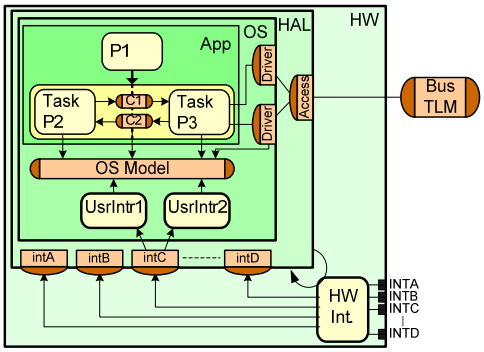
Specification *TLM* Implementation

- Specification is fast but inaccurate
  - Native execution, concurrency model
- Traditional ISS-based validation infeasible
  - Accurate but slow (esp. in multi-processor context), requires full binary
- ➢ Model of operating system
  - ➢ High accuracy but small overhead at early stages
  - ➢ Focus on key effects, abstract unnecessary implementation details
  - ➢ Model all concepts: Multi-tasking, scheduling, preemption, interrupts, IPC

Source: A. Gerstlauer, H. Yu, D. Gajski. "RTOS Modeling for System-Level Design," DATE, 2003.

Processor Model

- **Processor layers**
  - Application
    - Native, host-compiled C
    - Back-annotated timing
  - OS
    - OS model
    - Middleware, drivers
  - HAL
    - Firmware
  - Processor hardware
    - Bus interfaces
    - Interrupts handling & suspension



Features	
Target approx. computation timing	Appl.
Task mapping, dynamic scheduling	OS
Task communication, synchronization	HAL
Interrupt handlers, low level SW drivers	HW-TLM
HW interrupt handling, int. scheduling	HW-BFM
Cycle accurate communication	BFM - ISS
Cycle accurate computation	

Source: G. Schirner, A. Gerstlauer, R. Doemer. "Fast and Accurate Processor Models for Efficient MPSoC Design," TODAES, to appear 2009.

Communication Layers

- **ISO/OSI 7-layer model**

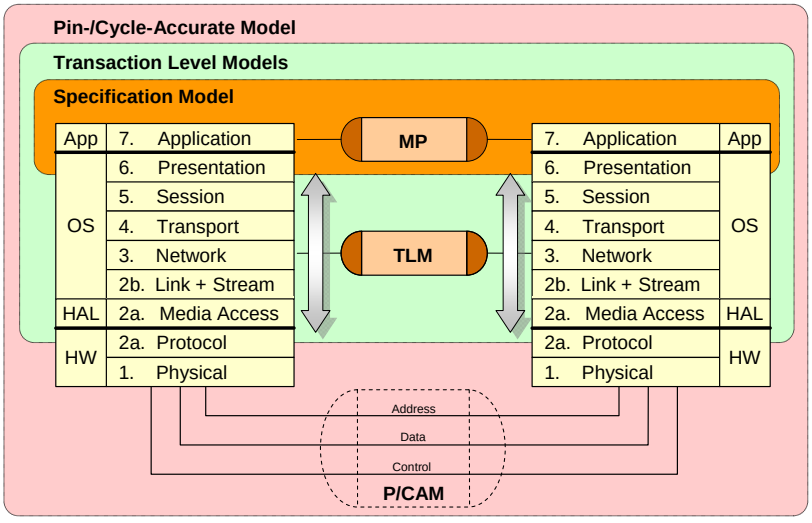
Layer	Semantics	Functionality	Implementation	OSI
Application	Channels, variables	Computation	Application	7
Presentation	End-to-end typed messages	Data formatting	OS	6
Session	End-to-end untyped messages	Synchronization, Multiplexing	OS	5
Transport	End-to-end data streams	Packeting, Flow control	OS	4
Network	End-to-end packets	Subnet bridging, Routing	OS	3
Link	Point-to-point logical links	Station typing, Synchronization	Driver	2b
Stream	Point-to-point control/data streams	Multiplexing, Addressing	Driver	2b
Media Access	Shared medium byte streams	Data slicing, Arbitration	HAL	2a
Protocol	Media (word/frame) transactions	Protocol timing	Hardware	2a
Physical	Pins, wires	Driving, sampling	Interconnect	1

- **A *model*, not an implementation !**

Source: A. Gerstlauer, D. Shin, J. Peng, R. Doemer, D. Gajski. "Automatic, Layer-Based Generation of System-On-Chip Bus Communication Models," TCAD, 2007.

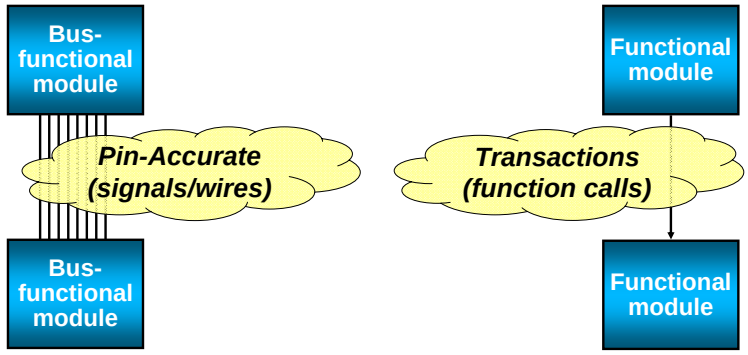
System Models

- **From layers to system models…**



Source: D. Gajski, S. Abdi, A. Gerstlauer, G .Schirner. "Embedded System Design: Modeling, Synthesis, Verification," Springer, 2009.

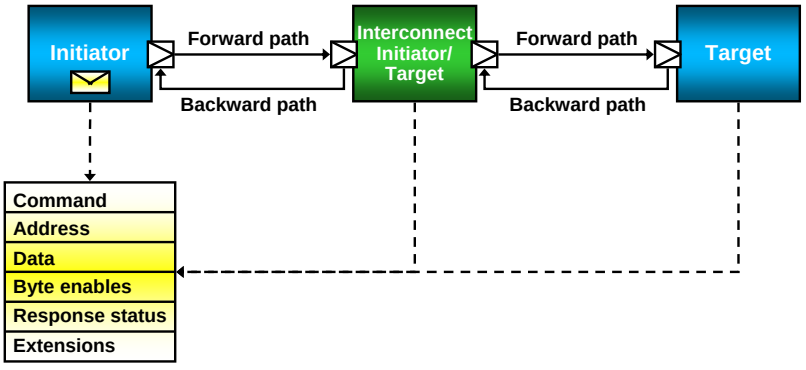
Transaction Level Modeling



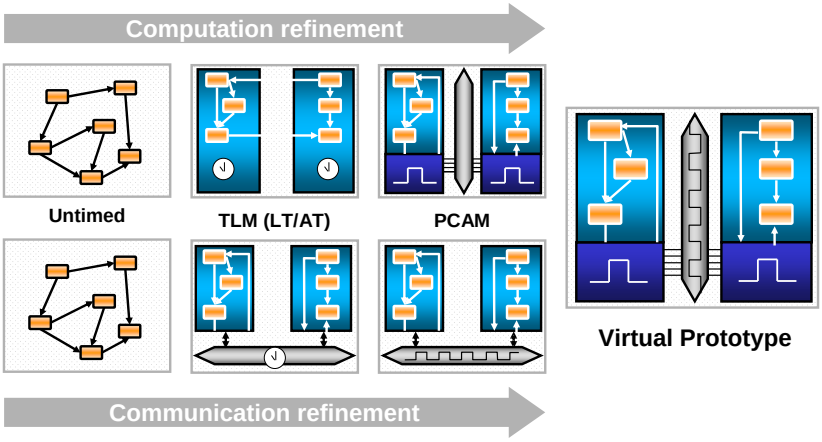
- **Pin-accurate model (PAM)**
  - Simulate every event (protocols)
- **Transaction-level model (TLM)**
  - Communications by transactions (abstract channels)
  - Simulates 100x - 10,000x faster than RTL

SystemC/TLM 2.0

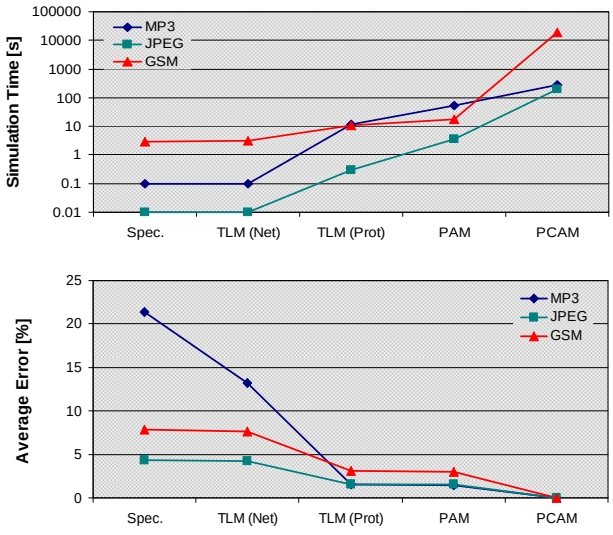
- **Pointer to transaction object is passed from module to module using forward and backward paths**
- **Transactions are of generic payload type**



Virtual Platform Prototyping



Modeling Summary



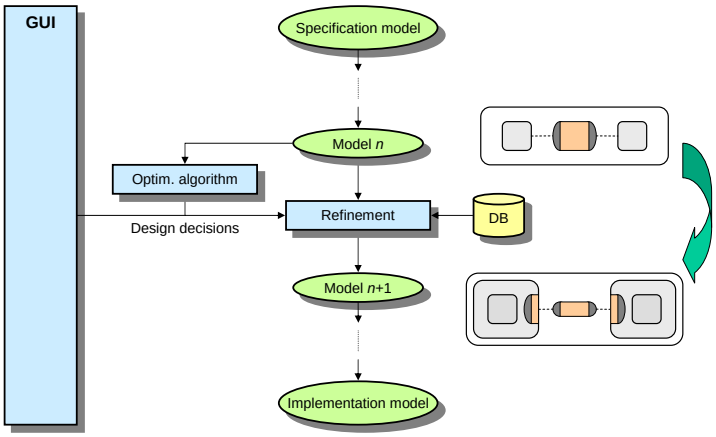
Source: D. Gajski, S. Abdi, A. Gerstlauer, G .Schirner. "Embedded System Design: Modeling, Synthesis, Verification," Springer, 2009.

Outline

- ✓ Introduction
- ✓ System design flow
- **ESL design**
  - ✓ Modeling
  - Synthesis
  - Verification
- ESL landscape
- Summary and conclusions

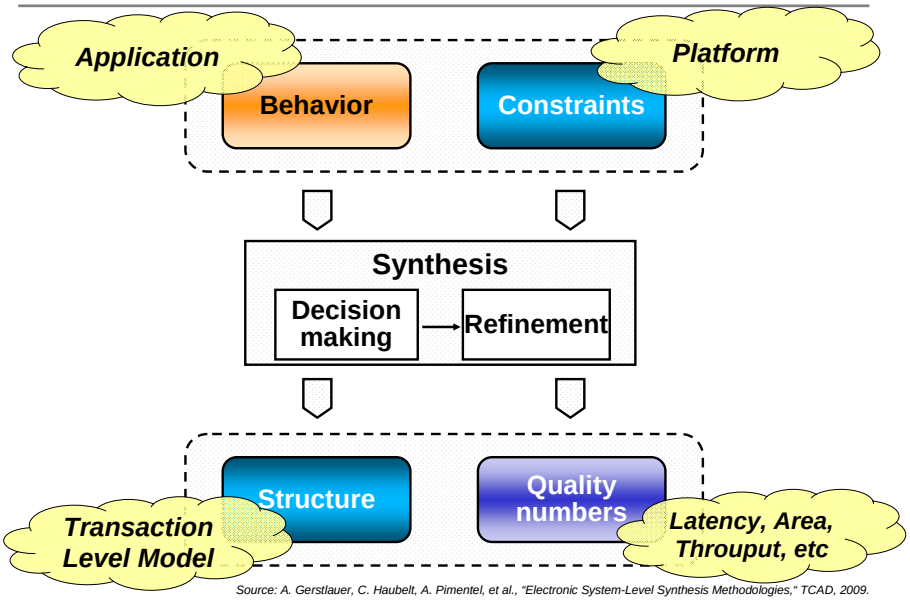
Design Automation

- **Synthesis = Decision making + model refinement**



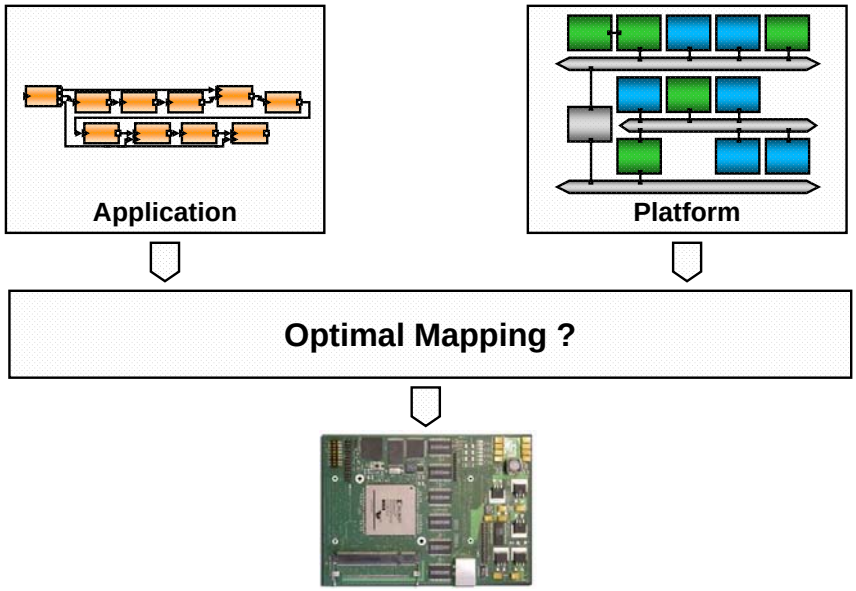
- ➢ **Successive, stepwise model refinement**
- ➢ **Layers of implementation detail**

X-Chart



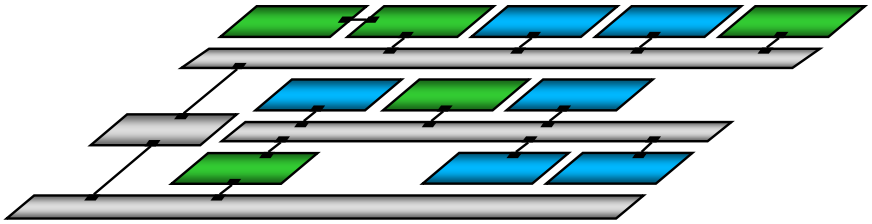
Source: A. Gerstlauer, C. Haubelt, A. Pimentel, et al., "Electronic System-Level Synthesis Methodologies," TCAD, 2009.

Platform-Based System Synthesis



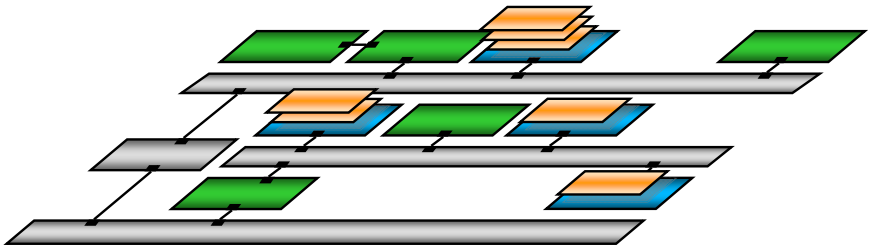
Resource Allocation

- Resource allocation, i.e., select resources from a platform for implementing the application



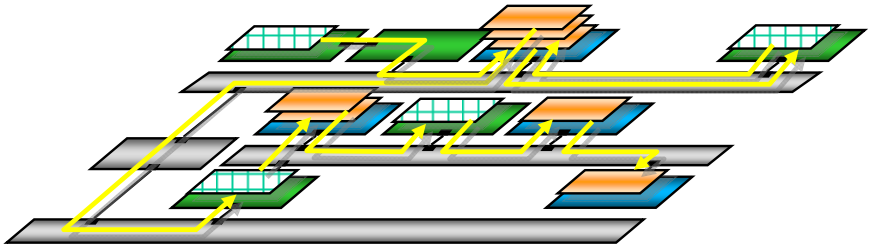
Process Binding

- Process mapping, i.e., bind processes onto allocated computational resources



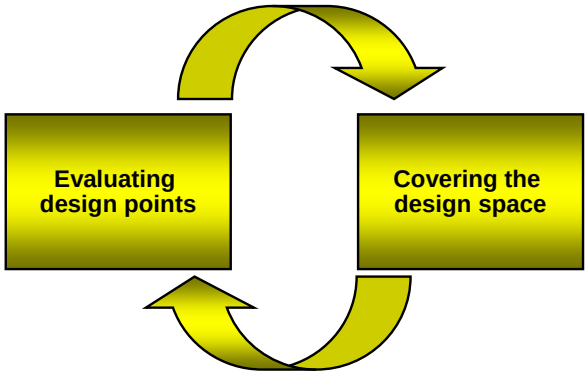
Channel Routing

- **Channel mapping, i.e., assign channels to paths over busses and address spaces**

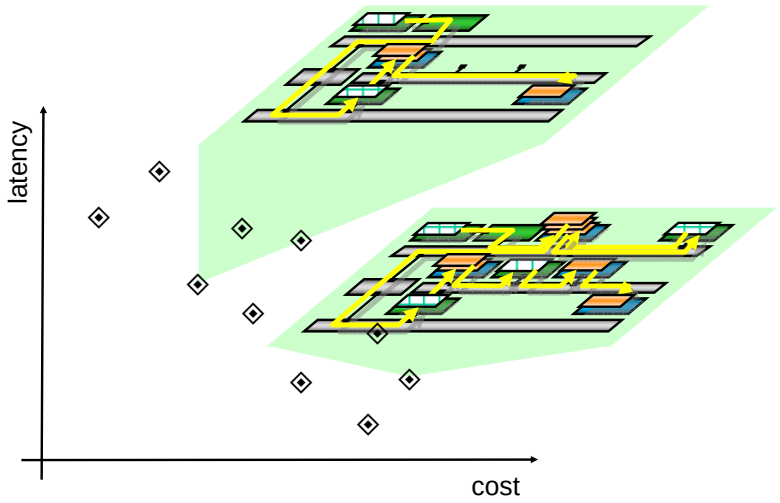


Design Space Exploration

- **Design Space Exploration is an iterative process:**
  - Evaluation through modeling
    - Static analysis or dynamic simulation
  - ➢ How can the design space be covered during the exploration process?

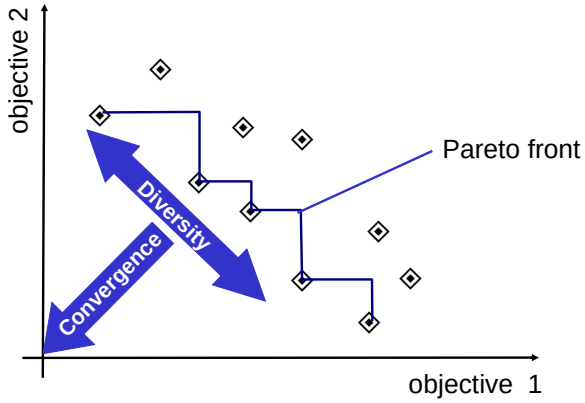


Multi-Objective Space



Optimization Goals

- **Find Pareto-optimal solutions**
  - A decision vector $x \in X$ is not dominated by any y
- **Or a good approximation (convergence, diversity)**
  - With a minimal number of iterations



Optimization Approaches

- **Exact methods**
  - Enumeration, (Integer) Linear Programs
- **Heuristics**
  - Constructive
    - Random mapping, hierarchical clustering
  - Iterative
    - Random search, simulated annealing, min-cut (Kernighan-Lin)
  - Set-based ("intelligent" randomized search)
    - Evolutionary Algorithms (EA),
      Particle Swarm Optimization (PSO),
      Ant Colony Optimization (ACO)

➢ **Exact, constructive & iterative methods are prohibitive**
  ➢ Large design space, multiple objectives, dynamic behavior
➢ **Set-based approaches**
  ➢ Randomized, problem independent (black box), Pareto set

Outline

✓ Introduction

✓ System design flow

- **ESL design**
  - ✓ Modeling
  - ✓ Synthesis
  - Verification
- ESL landscape
- Summary and conclusions

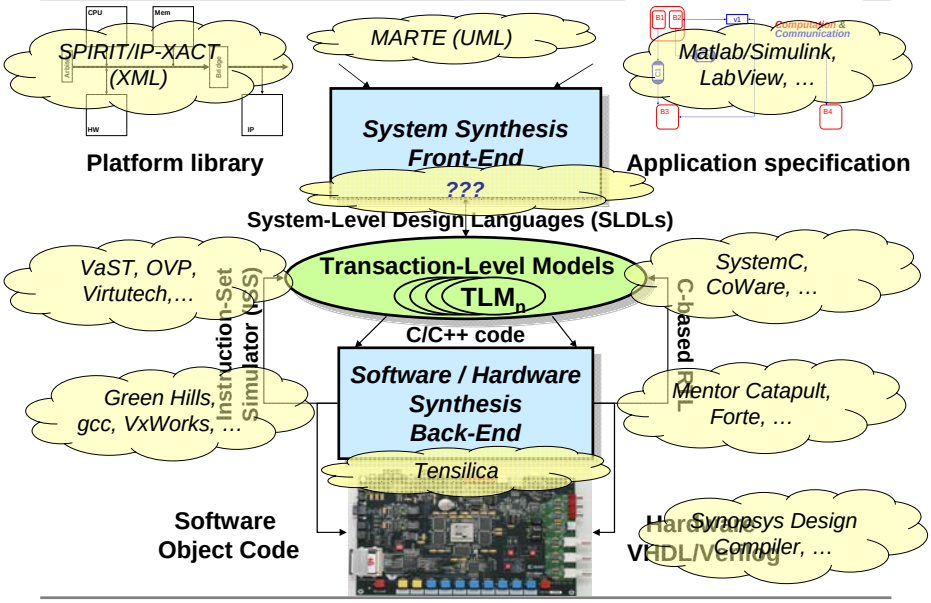
Design Verification Methods

- **Simulation based methods**
  - Specify input test vector, output test vector pair
  - Run simulation and compare output against expected output
- **Formal Methods**
  - Equivalence checking
  - Model checking
- **Semi-formal Methods**
  - Specify inputs and outputs as symbolic expressions
  - Check simulation output against expected expression

Outline

- ✓ Introduction
- ✓ System design flow
- ✓ ESL design
- **ESL landscape**
  - Commercial tools
  - Academic tools
- Summary and conclusions

Electronic System-Level (ESL) Landscape



ESL Tools

- **Electronic System-Level (ESL) terminology**
  - Often single hardware unit only
    - C-to-RTL high-level synthesis (HLS) [Mentor Catapult, Forte Cynthesizer]
- **System-level across hardware and software boundaries**
  - System-level frontend
  - Hardware and software synthesis backend

➢ **Commercial tools for modeling and simulation**
  - Algorithmic modeling (MoC) [UML, Matlab/Simulink, Labview]
  - Virtual system prototyping (TLM) [Coware, VaST, Virtutech]
  - ➢ *Only horizontal integration across models / components*

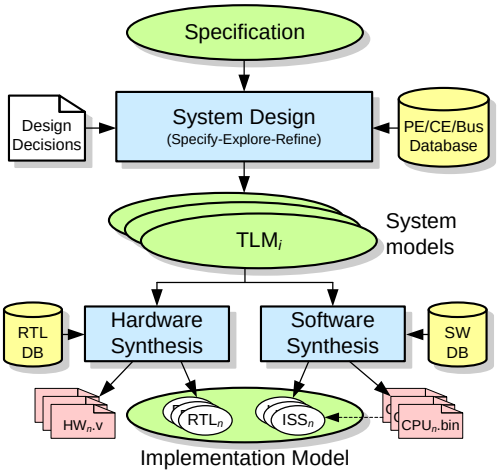
➢ **Academic tools for synthesis and verification**
  - MPSoC synthesis [SCE, Metropolis, SCD, PeaCE, Deadalus]
  - ➢ *Vertical integration for path to implementation*

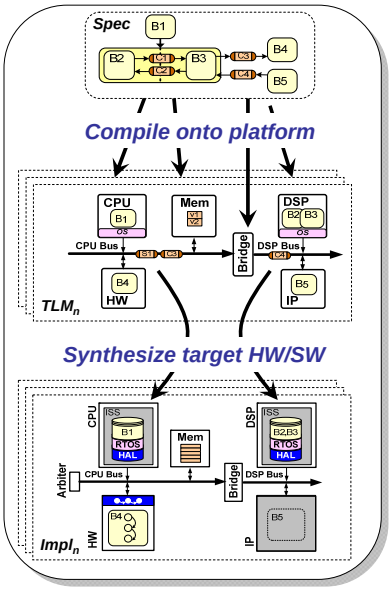
Academic MPSoC Design Tools

Approach	DSE	Comp. decision	Comm. decision	Comp. refine	Comm. refine
Daedalus	●	●	○	●	○
Koski	●	●	○	●	○
Metropolis		○		○	
PeaCE/HoPES	○	○		●	○
SCE				●	●
SystemCoDesigner	●	●	●	○	

Source: A. Gerstlauer, C. Haubelt, A. Pimentel, et al., "Electronic System-Level Synthesis Methodologies," TCAD, 2009.

System-On-Chip Environment (SCE)

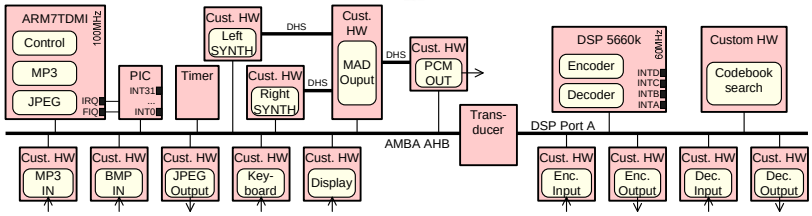




Cellphone Example

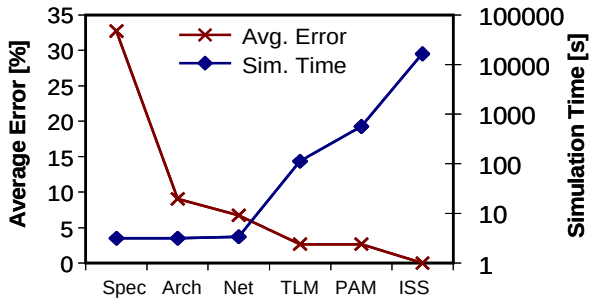
- **2 Subsystems**
  - ARM7TDMI
    - MP3 Decoding
    - Jpeg Encoding
  - Motorola DSP 56600k
    - GSM Transcoding

- **4 Accelerator HW blocks**
- **10 I/O HW blocks**
- **5 Busses**
  - AMBA AHB
  - DSP Port A bus
  - 3 Custom busses



Cellphone Example Modeling Results

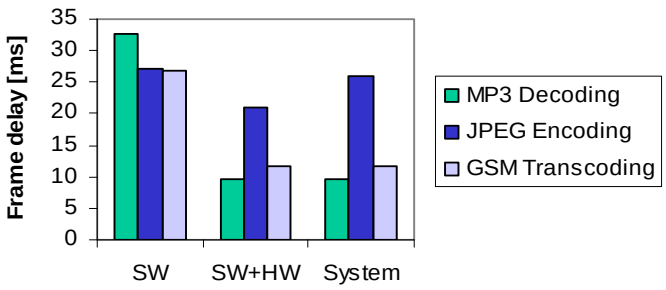
- **Experimental Results**
  - 1.5 second MP3
  - 640x480 picture
  - 1.5 speech GSM
- **Speedup (vs. multi-ISS)**
  - 150x (HW/SW system)
  - 600x (pure SW)
- **Excellent accuracy**
  - <3% error



➢ **Heterogeneous multi-processor platform TLM**

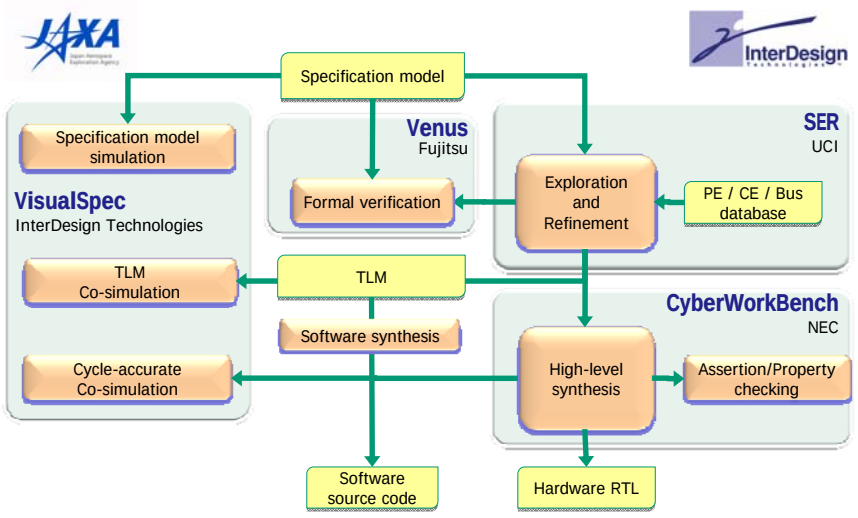
Cellphone Example Exploration Results

- **Single specification of application algorithms**
  - MP3 Decoding + JPEG Encoding + GSM Encoding & Decoding
- **Explored several architectural alternatives**
  - Stand-alone subsystems each, pure SW solution
  - Stand-alone subsystems each, SW + HW acceleration
  - Complete multi-processor SW + HW system



- **Automatically generate code and synthesize final software each**
  - Automatic generation of all models and code within seconds
  - Final binaries for all processors ready for download

ELEGANT Environment



ELEGANT : Electronic Design Guidance Tool for Space Use

Source: InterDesign Technologies, Inc. / Japanese Aerospace Exploration Agency (JAXA)

Summary & Conclusions

- **Seamless electronic system-level (ESL) flow**
  - From top-level specification to RTL implementation
  - Automation of model refinement, decision making and verification
  - ➢ Well-defined semantics of models, transformations

- ➢ **Rapid, early design space exploration**
  - ➢ Fast and accurate models for virtual prototyping
  - ➢ Automatic model generation and backend synthesis

- ➢ **Significant productivity and reliability gains**
  - Concurrent SW, HW and application development
  - Easy capture, upgrade and reuse of code