ECE382M.20: System-on-Chip (SoC) Design

Lecture 5 – HW/SW Co-Design

Sources:
Prof. Margarida Jacome, UT Austin

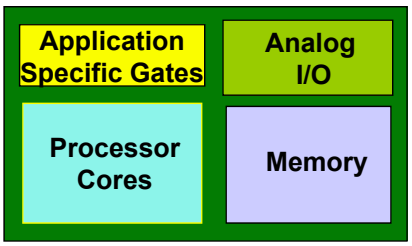
Andreas Gerstlauer
Electrical and Computer Engineering
University of Texas at Austin
gerstl@ece.utexas.edu

Lecture 5: Outline

- **Introduction**
  - Embedded SoC design
- **HW/SW co-design**
  - Specification
  - Analysis
  - Synthesis

Systems-on-Chip (SoCs)

- **Employ a combination of**
  - SW on programmable processors
    - Flexibility, complexity
  - Application-specific, custom HW
    - Performance, low power
  - Transducers, sensors, actuators
  - A/D & D/A converters
    - Interact with analog, continuous-time environment

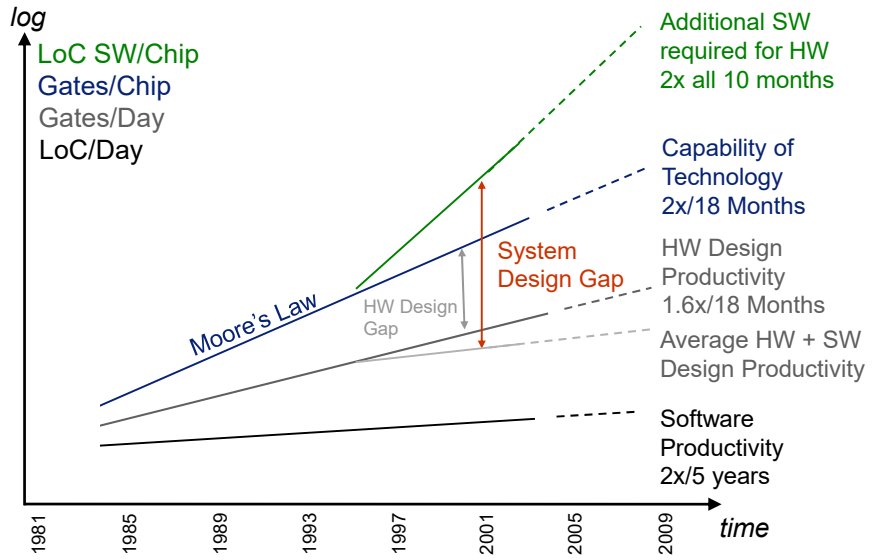


- **Micro-controllers & digital signal processors (DSPs)**
- **ASICs & Field programmable gate arrays (FPGAs)**

Design Problems

- **Design a heterogeneous multiprocessor architecture that satisfies the design requirements.**
  - Use computational unit(s) dedicated to some functions
    - Processing elements (PE): hardwired logic, CPU
  - Program the system
- **A significant part of the design problem is deciding which parts should be in SW on programmable processors, and which in specialized HW**
  - Deciding the HW/SW architecture
- **Ad-hoc approaches today**
  - Based on earlier experience with similar products
  - HW/SW partitioning decided a priori, designed separately

Productivity Gaps



Source: W. Ecker, W. Müller, R. Dömer, *Hardware-dependent Software - Principles and Practice*, Springer 2009.

Design Automation

- **Computer-Aided Design (CAD)**
  Electronic Design Automation (EDA)
  - Tools take care of HW fairly well (at least in relative terms)
  - Productivity gap emerging
- **Situation in SW is worse**
  - HLLs such as C help, but can't cope with exponential increase in complexity and performance constraints

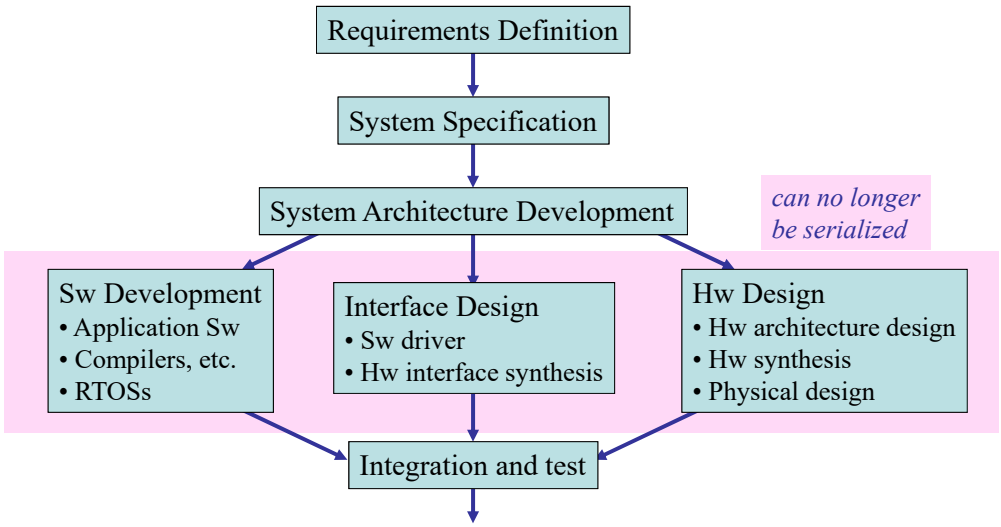
Holy Grail for Tools People: HW-like synthesis & verification from a behavior description of the whole system at a high level of abstraction using formal computation models

Embedded System Design

- **The design of an embedded system consists of correctly implementing a specific set of *functions* while satisfying constraints on**
  - Performance
  - Dollar cost
  - Energy consumption, power dissipation
  - Weight, etc.

The choice of a ***system architecture*** impacts whether designers will implement a ***function*** as custom hardware or as (embedded) software running on a programmable component (processor).

Embedded System Design Process



HW/SW Co-Design

- **Use additional computational unit(s) dedicated to some functions**
  - Hardwired logic, extra CPU
- ➢ **Automated design & optimization of HW/SW systems**
  - Specification
    - Modeling
    - Performance analysis
  - Synthesis
    - HW/SW partitioning (resource allocation & binding)
    - Scheduling
  - HW & SW implementation
    - SW compilation
    - HW synthesis
  - Validation
    - Integration, verification & debugging

Lecture 5: Outline

- ✓ Introduction
  - ✓ Embedded SoC design
- **HW/SW co-design**
  - Specification
  - Analysis
  - Synthesis

System Specification

- **Describe the desired behavior of a design as a relation between a set of inputs and a set of outputs**
  - This relation may be informal, even expressed in natural language
  - Such informal, ambiguous specifications may result in unnecessary redesigns…

➢ **Formal Models of Computation (MoCs)**
- Computation/behavior
- Communication
- Concurrency
- Time/order
- Heterogeneity, composability
- Implementability

Main MoCs for Embedded Systems

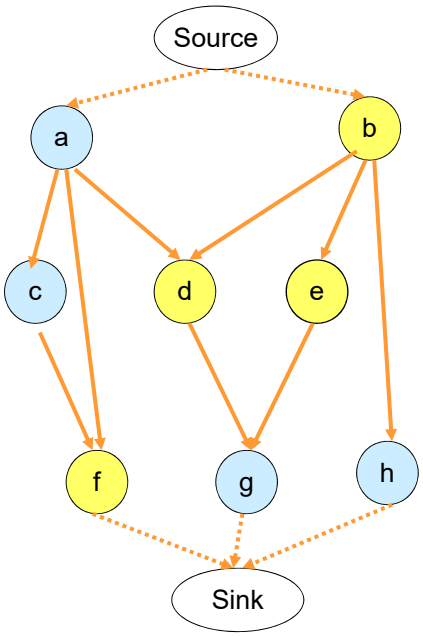
- **Programming models**
  - Imperative & declarative
  - Synchronous/reactive
- **Process-based models**
  - Discrete event
  - Kahn Process Networks (KPNs)
  - (Synchronous) Dataflow models ((S)DF)
- **State-based models**
  - Finite State Machines (FSM)
  - Hierarchical, Concurrent State Machines (HCFSM)
  - Petri Nets

➢ **EE382N.23: Embedded System Design & Modeling**

Task Graph Model

- **A graph representation of the application specification**
  - Derived from data dependency based representation commonly utilized in compilers
- **Application is specified by a graph *G*(*V*,*E*)**
  - *V* is the set of tasks
    - *t*(*v*,*r*) gives the run-time of "*v*" on a processing element "*r*"
  - *E* is the set of directed edges
    - *e*(*u*,*v*) implies data produced by *u* is consumed by *v*
    - *v* cannot begin execution before *u* has finished execution
  - Execution constraints
    - Deadlines, rates, latencies
- **Data-dominated application model**
  - Multimedia and network processing applications can be specified by this model

Task Graph Example



- **Assign weights to nodes and edges**
  - Cost, delays
- **Constraints for nodes or whole graph**
  - Source-to-sink delay
- **Analysis & synthesis**
  - Partitioning
  - Real-Time Scheduling
- **Amdahl's law**

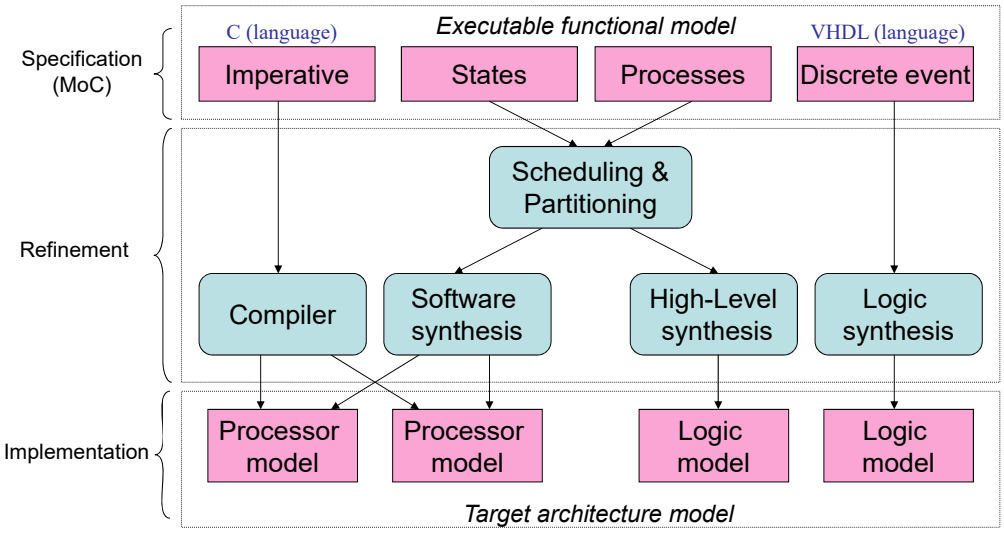
Lecture 5: Outline

- ✓ Introduction
  - ✓ Design methodology
- • **HW/SW co-design**
  - ✓ Models of Computation
  - • Analysis
    - ➢ Execution time analysis (Lecture 4)
    - ➢ Scheduling analysis (Lecture 7)
  - • Synthesis

Lecture 5: Outline

- ✓ Introduction
  - ✓ Embedded SoC design
- • **HW/SW co-design**
  - ✓ Models of Computation
  - ✓ Analysis
  - • Synthesis

Design Synthesis



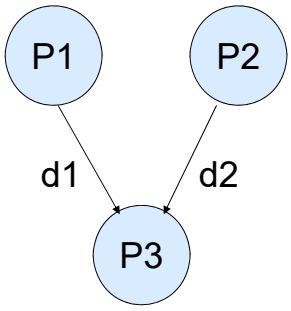
Synthesis Tasks

- **Mapping**
  - Allocate resources (hardware/software processors)
  - Bind computations to resources
  - Schedule operations in time

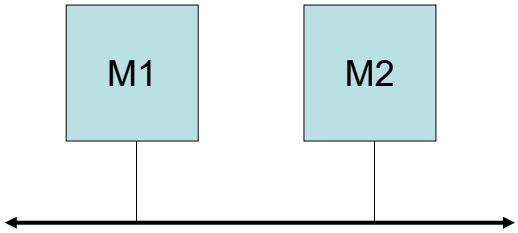
  ➢ Partitioning = (allocation +) binding
  ➢ Mapping = binding + scheduling

- **Allocation, scheduling and binding interact, but separating them helps**
  - Alternatively allocate, bind, then schedule

Mapping Example



Task graph

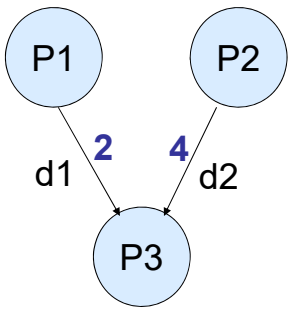


Hardware platform

Example Cost Model

- **Process execution times**

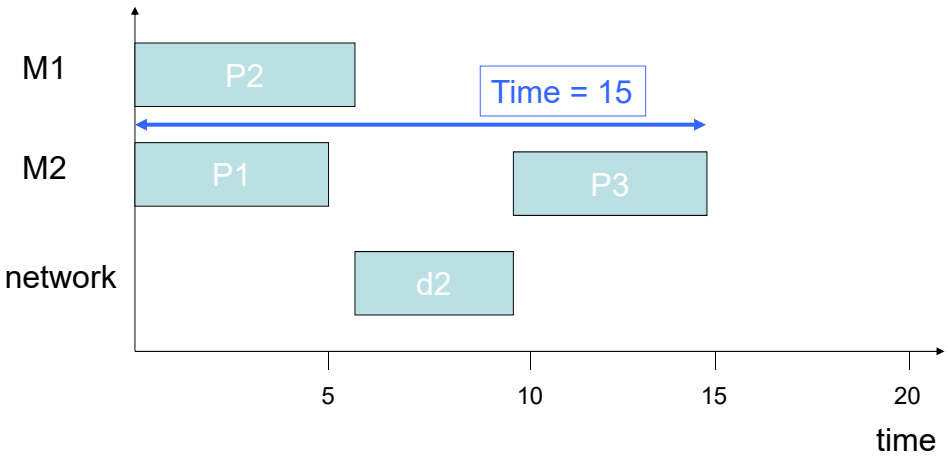
	M1	M2
P1	5	5
P2	5	6
P3	--	5



- **Communication cost**
  - Assume communication within PE is free
  - Cost of communication from P1 to P3 is d1 = 2
  - Cost of P2 to P3 communication is d2 = 4

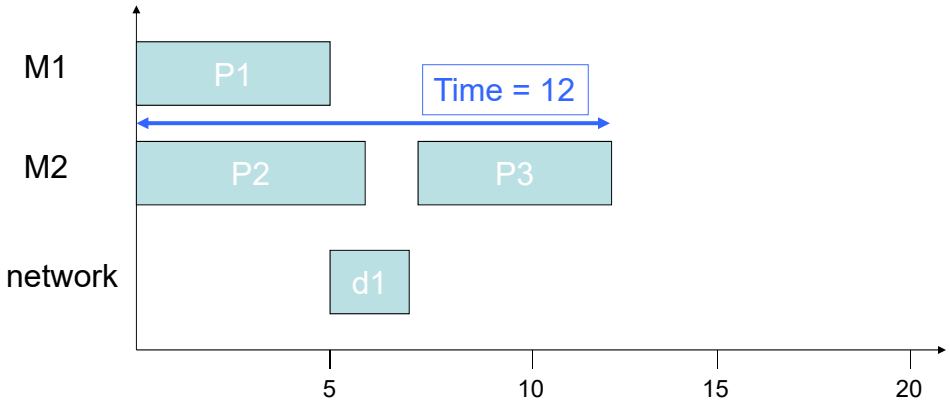
First Design

- **Allocate P2 -> M1; P1, P3 -> M2.**



Second Design

- **Allocate P1 -> M1; P2, P3 -> M2:**



Co-Design Approaches

- **Partitioning**
  - Exact methods
    - Integer linear programming (ILP) formulations
  - Heuristics
    - Constructive: Hierarchical clustering
    - Iterative: Kernighan-Lin
- **Scheduling**
  - Static
    - ILP formulations for combined scheduling & partitioning
    - ➢ Borrowed from high-level synthesis (see later lectures)
  - Dynamic
    - Operating system
    - ➢ Task graph scheduling