

EE382V: System-on-a-Chip (SoC) Design

Lecture 8 – 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

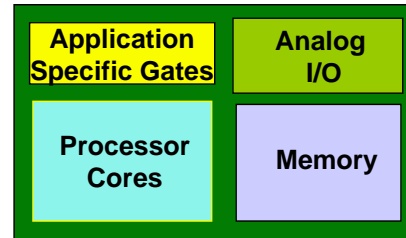


Lecture 8: Outline

- **Introduction**
 - Embedded SoC design
- **HW/SW co-design**
 - Models of Computation
 - 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

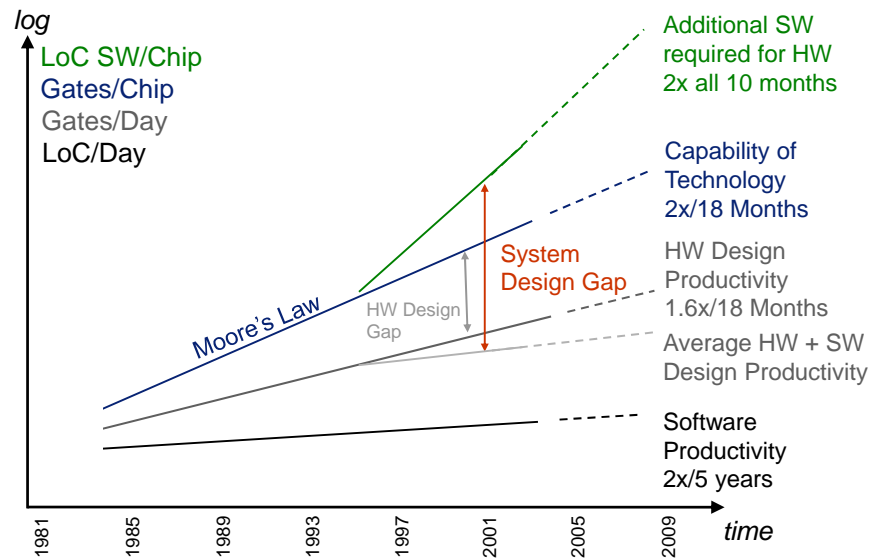


- **Micro-controllers & digital signal processors (DSPs)**
- **ASICs & Field programmable gate arrays (FPGAs)**

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.

EE382V: SoC Design, Lecture 8

© 2014 A. Gerstlauer

5

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

EE382V: SoC Design, Lecture 8

© Margarida Jacome, UT Austin

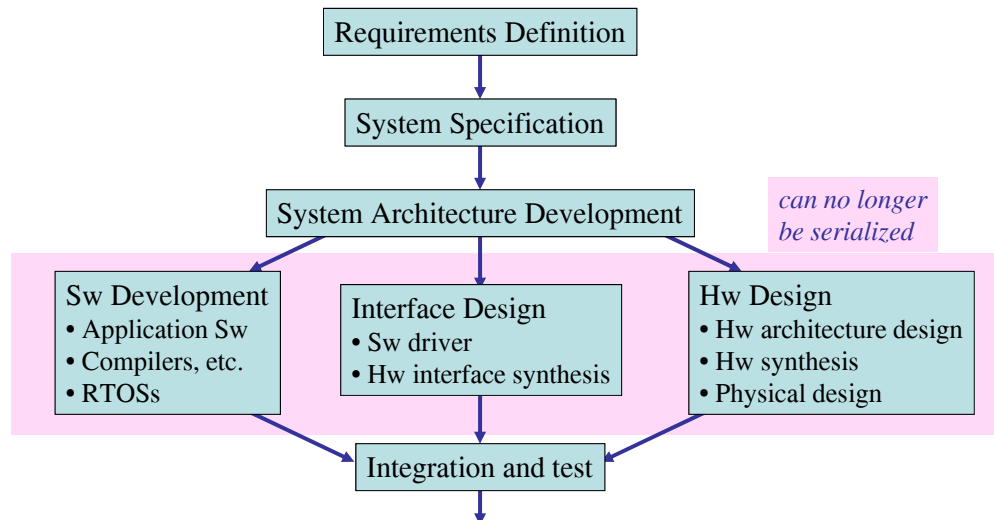
6

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



Desirable Design Methodology

- **Design should be based on the use of one or more *formal models* to describe the *behavior* of the *system* at a high level of abstraction**
 - Such behavior should be captured on an unbiased way, that is, before a decision on its decomposition into hardware and software components is taken
- **The final implementation of the system should be generated as much as possible using *automatic synthesis* from this high level of abstraction**
 - To explore the design space and optimize implementations
 - To ensure implementations that are “correct by construction”
- **Validation (through *simulation* or *verification*) should be done as much as possible at the higher levels of abstraction**

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 8: Outline

✓ Introduction

✓ Embedded SoC design

• HW/SW co-design

- Models of Computation
- Analysis
- Synthesis

Formal Model of a Design

- **Most tools and designers describe the 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...
- **A formal model of a design should consist of the following components:**
 - Functional specification
 - Set of properties
 - Set of performance indices
 - Set of constraints on performance indices

Models of Computation (MoCs)

- **Consider essential aspects of embedded systems**
 - Time/synchronization
 - Concurrency
 - Heterogeneity
- **Classify models based on**
 - How to specify behavior
 - How to specify communication
 - Implementability
 - Composability
 - Availability of tools for validation and synthesis

Main MoCs for Embedded Systems

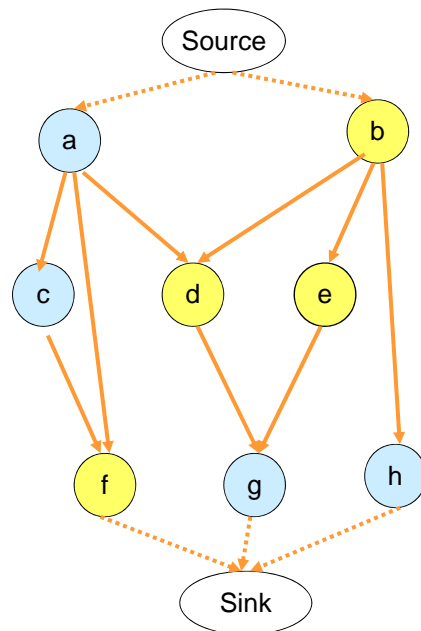
- **Programming models**
 - Imperative & declarative
 - Synchronous/reactive
- **Process-based models**
 - Discrete event
 - Kahn Process Networks (KPNs)
 - (Synchronous) Dataflow models ((S)DF)
- **State-based models**
 - Finite State Machines (FSM)
 - Hierarchical, Concurrent State Machines (HCFSM)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

➤ **EE382V: 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 8: Outline

- ✓ Introduction

 - ✓ Design methodology

- HW/SW co-design

 - ✓ Models of Computation

 - Analysis

 - Execution time analysis (Lecture 5)

 - Scheduling analysis (Lecture 9)

 - Synthesis

Lecture 8: Outline

- ✓ Introduction

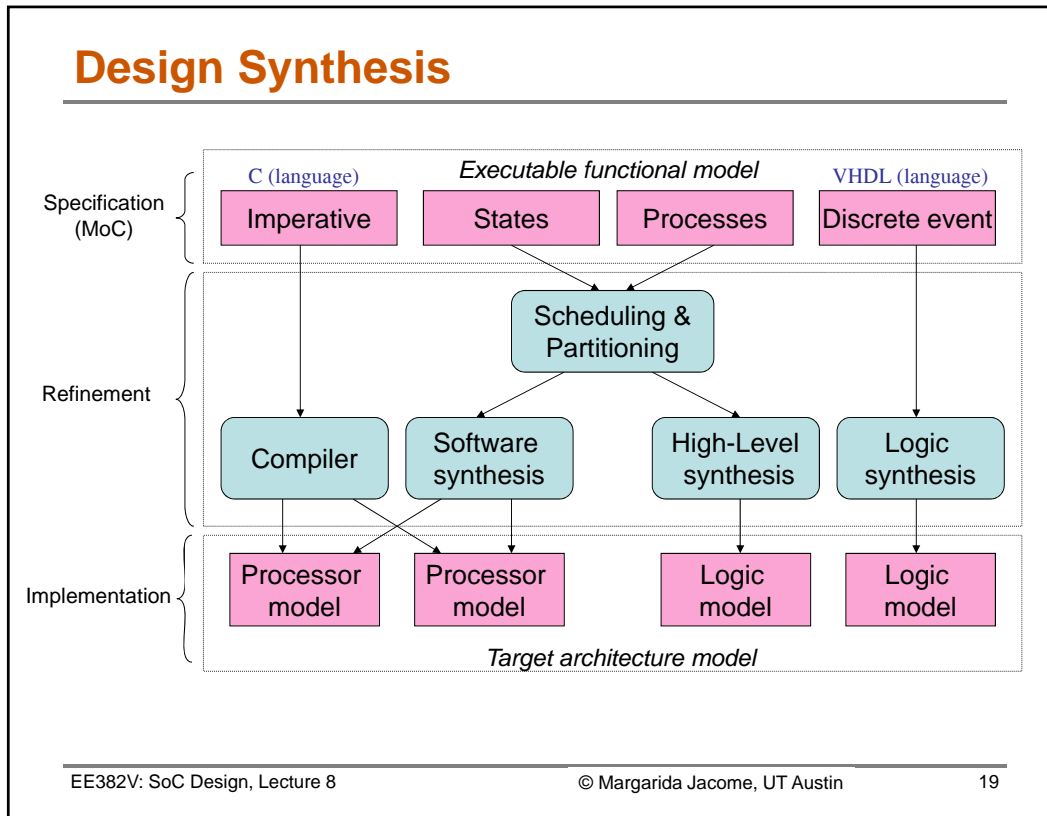
 - ✓ Embedded SoC design

- HW/SW co-design

 - ✓ Models of Computation

 - ✓ Analysis

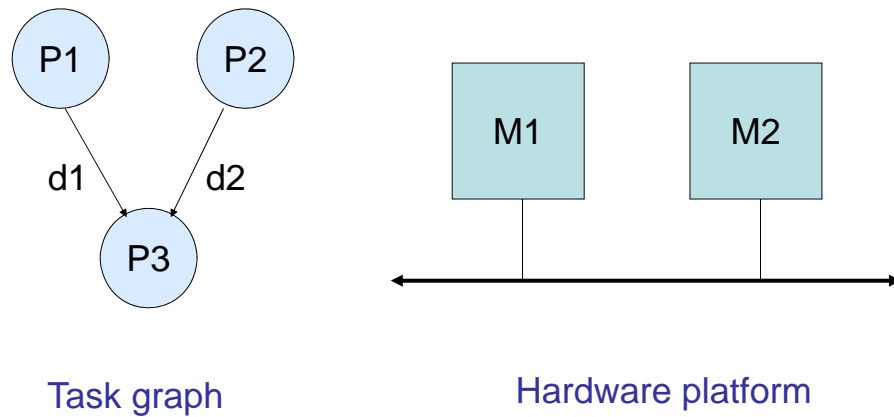
 - Synthesis



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



EE382V: SoC Design, Lecture 8

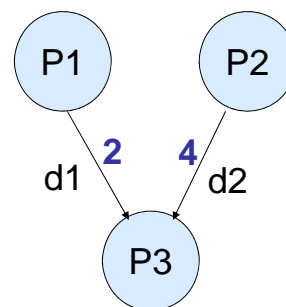
© Margarida Jacome, UT Austin

21

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 $d2 = 4$

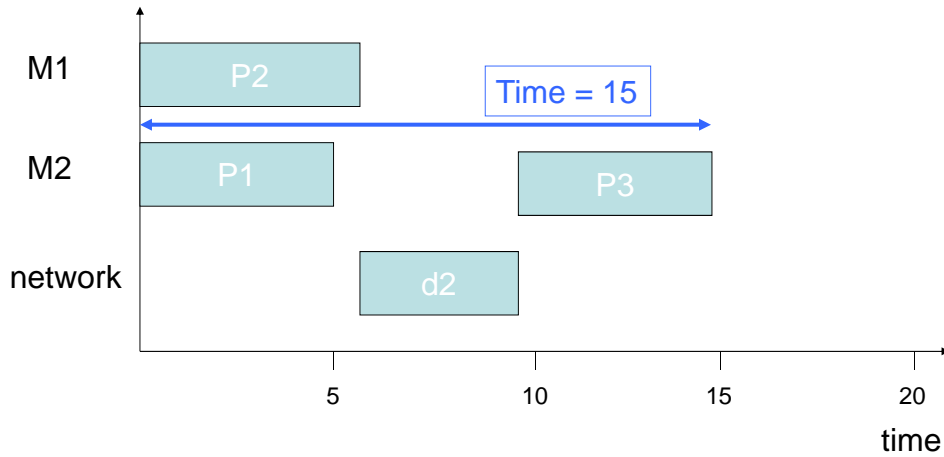
EE382V: SoC Design, Lecture 8

© Margarida Jacome, UT Austin

22

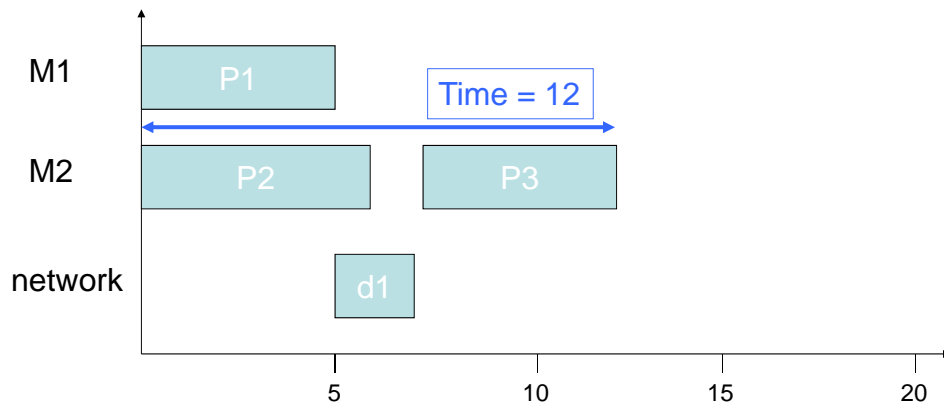
First Design

- 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
 - Real-time scheduling