

# *Codesign Framework*

Partsofthislectureareborrowedfrom  
lecturesofJohan LiliusofTUCSandASV/LL  
ofUCBerkeleyavailableintheirweb.

# *System Design*

What is System Design?

“Is the process of implementing a desired  
functionality using a set of components”

## *System Design contd.*

### Step 1

Design must begin with specifying the desired functionality

## *Specification*

For precise specification, we need to

- think the system to be a collection of simpler subsystems
- a method (rules) to compose these pieces

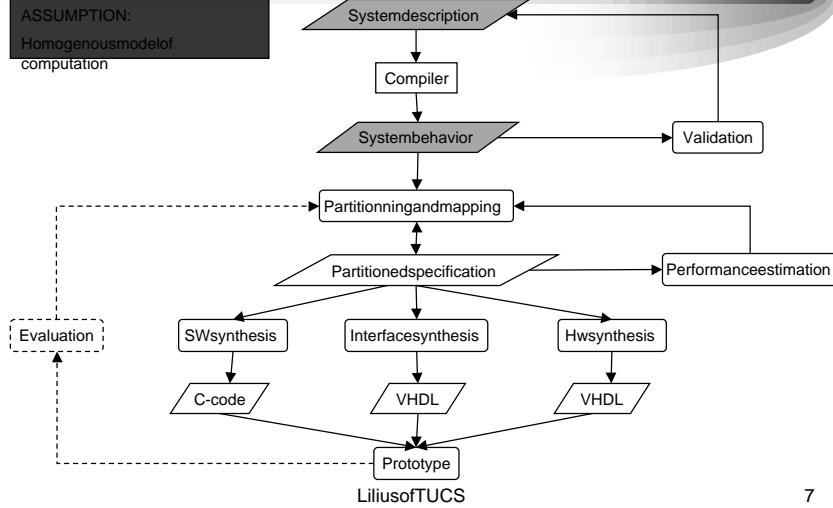
## *Functions vs. Computation*

- Functions specify only a relation between two sets of variables (input and output)
- Computation describes how the output variables can be derived from the value of input variable

## *Essential Issues*

- Modeling
  - System description
  - System behavior
- Validation (verification)
- Performance estimation
- Partitioning and mapping
- Synthesis:
  - Software synthesis
  - Hardware synthesis
  - Interface synthesis

# ACo-designframework



7

## Model

The method or rule to compose the pieces of subsystems to create a system functionality is usually called a Model.

## Model

Model should be...

- *formal* : no ambiguity
- *complete*: with set of properties, performance indices and constraints
- *comprehensive* and easy to modify
- *natural* to understand

## Model

What is a *model*?

Model is a formal system consisting of objects and composition rules, and is used for describing a system's characteristics.

## *Modeling*

- Modeling digital systems is often complicated by the heterogeneity of its components.
- Distinguish between:
  - models of computation
  - hardware/software (specification) languages.
- A language may imply many models:
  - UML State Machines:
    - synchronous behavior within a state machine
    - asynchronous behavior between state machines

## *Model of Computation*

- A MoC is a framework in which to express what sequence of actions must be taken to complete a computation
- An instance of a model of computation is a representation of a function under a particular interpretation of its constituents.
- Not necessarily a bijection (almost never!)

## *Models of Computation*

- Often existing models belong to several categories
  - Finite state machines:
    - totally ordered discrete events
  - Petri nets:
    - partially ordered events
  - Synchronous data-flow:
    - multirate discrete time
    - partially-ordered events

LiliasTUCS

13

## *Why different models?*

- Various models have certain strong properties that might be useful for some applications
- Some problems might be undecidable

14

## *Specification languages*

- Finite-statebased
  - synchronouscommunication:
    - Statecharts,Esterel
  - asynchronouscommunication:
    - SDL
- Partialordersoftasks:
  - VHDLatbehaviorallevel
- Discretetime,cyclebased:
  - VHDLatRTLlevel
- Synchronousdata-flow:Silage,DFL,Lustre

LiliusTUCS

15

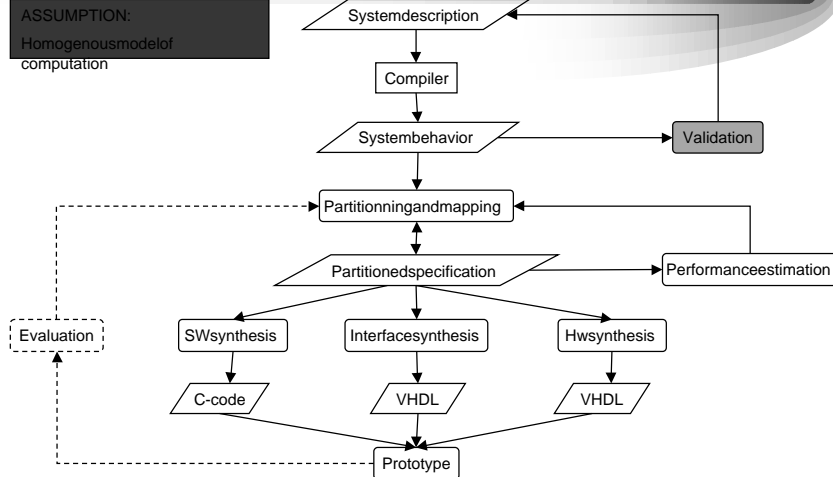
## *Specification languages*

- Noperfectlanguageexists!
  - Control(software):asynchronous(fsm)
  - DSP:data-flow
  - Hardware:synchronous
- ⇒ Forceusertothinkintermsof:
  - onemodelofcomputationsystem:POLIS
  - manymodelsofcomputation:PTOLEMY

LiliusTUCS

16

# ACo-designframework



17

## Validation

- System-level validation (co-validation):
  - Methods for gaining reasonable certainty that the design is free from errors.
- Methods:
  - Verification
  - (Co-)Simulation
  - Emulation

## Verification

- Specification verification
  - Is the specification consistent?
  - Does it have the required properties?
- Implementation verification
  - Have we implemented the specification?
- Checking of:
  - *safety*: nothing *bad* ever happens
  - *liveness*: something *good* eventually happens

## Weakly heterogeneous systems

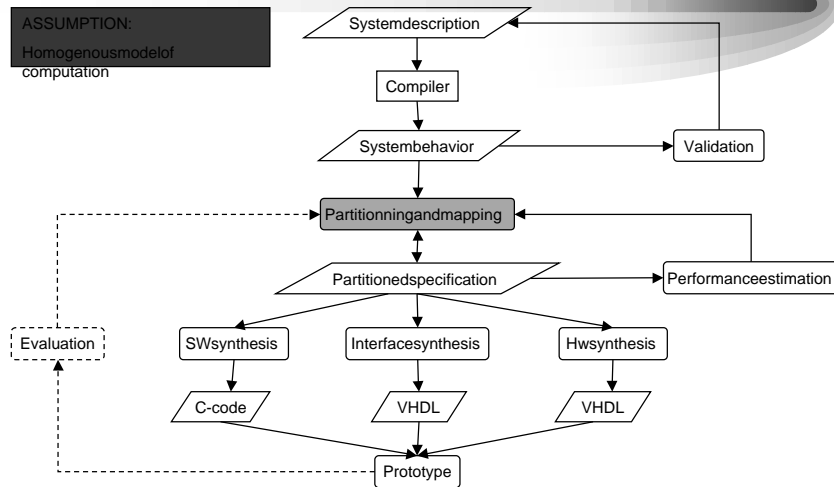
- One or more processors, some dedicated hardware, and several software layers.
- Desired features of simulator:
  - adequate timing accuracy
  - fast execution
  - visibility of internal registers for debugging purposes
- Problems:
  - One step in program is equivalent to many steps in hardware
    - ⇒ long running times
    - ⇒ Availability of hardware models with the required abstraction level.

# Highly heterogeneous systems

- Specialized simulators:
  - PTOLEMY (U.C. Berkeley)

21

# ACo-design framework



22

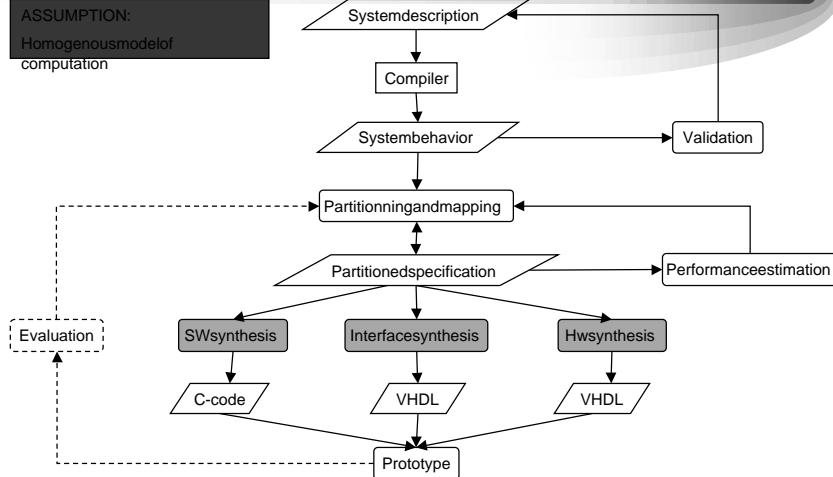
## Partitioning

- Input: functional specification
- Output:
  - an architecture composed of
    - hardware blackboxes,
    - software blackboxes and
    - interconnection media and mechanisms
  - a **mapping function** that assigns functional units to architectural units

## Partitioning

- Construction of mapping is an optimization problem:
  - mapping function optimizes a cost
    - time, area, communication
- What is the architecture?
  - Automated synthesis of custom architectures difficult, common restrictions:
    - limited to a library of predefined choices
    - communication mechanisms are standardized

# ACo-designframework



25

# Hardware synthesis

- Well established research field
- Several commercial tools exist
- Levels of abstraction:
  - Behavioral synthesis: algorithmic synthesis
  - Register-Transfer level synthesis: VHDL, Verilog
  - Logic level synthesis: netlist
- Research issue: reuse of hardware

LiliusTUCS

26

## *Softwaresynthesis*

- Difficult problem for general purpose computing
- Forembedded system much more constrained:
  - no swapping devices
  - no stacks
  - only polling and static variables
- Simple algorithms
  - Translating FSM: stop programs especially simple
- Specification consists of concurrent tasks
  - Problem: How do we find a linear execution order that satisfies the timing constraints?
    - Uses scheduling theory.

27

## *Interfacesynthesis*

- Interface between processor and ASIC
  - synthesis of software
  - synthesis of "glue logic"
- Automatic generation of bus interfaces
  - PCI, VME
- Interfacing of sensors and actuators

LiliusTUCS

28

## Existing Tools

- Academic:
  - POLIS:U.C. Berkeley
  - PTOLEMY:U.C. Berkeley
  - VULCAN:Stanford U.(HardwareC)
  - CHINOOK:U.of Washington(VHDL)
  - COSYMA:U.of Braunschweig(C\*)
  - MELIE:INRIAand
- Commercial:
  - Arexys:SDL,VHDL, C
  - CoWare:C/C++
  - LavalLogic:Javato Verilog
  - Cynlib:C++to Verilog
  - Art,AlgorithmttoRT: C++toRTL
  - SUPERLOG:System<sub>29</sub> leveledescription

## POLIS

- Specification:FSM-basedlanguages(Esterel,...)
- InternalRepresentation:CFSMnetwork
- Validation:
  - high-levelco-simulation
  - FSM-basedformalverification
- Partitioning:byhand,basedonco-simulation estimates
- Scheduling:classicalreal-timealgorithms
- Synthesis:
  - S-graph-basedcodesynthesisforsoftware
  - logicsynthesisforhardware
- Mainemphasison *unbiasedverifiablespecification*

30

## *PTOLEMY*

- Specification: Dataflow graph
- Internal representation: DFG
- Validation: multi-paradigm co-simulation (DF, discrete events, ...)
- Partitioning: greedy, based on scheduling
- Scheduling: linear, sorting blocks by "criticality"
- Synthesis:
  - DSP code
  - custom DSP synthesis for hardware

31

## *Projects*

- Mini-Esterel
- Cosimulation using Cand Verilog
- Implementation of a DSP algorithm in PTOLEMY
- Implementing a small ES in POLIS
- Code sign using FPGA
- Your own project

32