Lecture 4 – System Design Flow and Methodology

Andreas Gerstlauer
Electrical and Computer Engineering
University of Texas at Austin
gerstl@ece.utexas.edu
Lecture 4: Outline

• **SpecC system design methodology**
  • From specification to implementation
    – Specification
    – System design
    – Processor design
  • Models and transformations
    – Specification model
    – Computation model
    – Communication model
    – Implementation model
System Design

- Specification to architecture to implementation
- Behavior to structure
  1. System level: system specification to system architecture
  2. Processor level: component behavior to component microarchitecture
System Design Needs

- **Design models**
  - Representation for validation and analysis
  - Specification for further implementation/synthesis

- **Design languages**
  - Specification down to implementation
    - Models of Computation (MoC)
  - System-level design languages (SLDL)
    - C-based, event-driven SLDLs [SpecC, SystemC]

- **Well-defined, rigorous system-level semantics**
  - Unambiguous, explicit abstractions, models
    - Objects and composition rules
  - Systematic flow from specification to implementation
    - Transformations and refinements

- **Modeling flow**
  - Design automation for synthesis and verification
System Design Process

- Abstraction based on level of detail (structure/timing)
  - Computation and communication

- System design flow
  - Path from model A to model F

A. System specification model
B. Component model
C. Bus-arbitration model
D. Bus-functional model
E. Cycle-accurate computation model
F. RTL/ISS Implementation model


- Design methodology
  - Set of models and transformations between models
Top-Down Design Flow

- Requirements
- Pure functional
- Bus functional
- RTL / IS
- Gates

Structure

- Specification
- System design
- Architecture
- Processor design
- Implementation
- Logic design

Logic design

- Constraints
- Untimed
- Timing accurate
- Cycle accurate
- Gate delays

Timing
Top-Down Design Flow

- requirements
- pure functional
- transaction level
- bus functional
- RTL / IS

Structure

Product planning

Specification model

Computation design

Computation model

Communication design

Communication model

Processor design

Implementation model

Logic design

Timing

Constraints

Untimed

Estimated timing

Timing accurate

Cycle accurate
Top-Down Design Flow

- Requirements
- Pure functional
- Transaction level
- Bus functional
- RTL / IS

Structure

Logic design

- Hardware synthesis
- Interface synthesis
- Software synthesis
- Implementation model

Timing

- RTL IP
- RTOS IP

Product planning

- Capture
- Specification model
- Computation refinement
- Computation model
- Communication refinement

Communication model

- RTL IP
- Proto IP
- Comp IP
- Algor IP

Constraints

- Untimed
- Estimated timing
- Timing accurate
- Cycle accurate
Design Methodology

**System design**

- **Capture**
  - Specification model
  - Computation refinement
  - Communication refinement
  - Communication model
  - Implementation model

**Validation flow**

- **Compilation**
  - Simulation model
  - Validation Analysis Estimation
  - Compilation
  - Simulation model
  - Validation Analysis Estimation
  - Compilation
  - Simulation model
  - Validation Analysis Estimation
  - Compilation
  - Simulation model
  - Validation Analysis Estimation
  - Compilation
  - Simulation model
  - Validation Analysis Estimation

**Backend**

- **RTL IP**
  - Hardware synthesis
  - Interface synthesis
  - Software compilation
  - RTOS IP
  - Implementation model
Specification Model

- High-level, abstract model
  - Pure system functionality
  - Algorithmic behavior
  - No implementation details
- No implicit structure / architecture
  - Behavioral hierarchy
- Untimed
  - Executes in zero (logical) time
  - Causal ordering
  - Events only for synchronization
Specification Model Example

- **Simple, typical specification model**
  - Hierarchical parallel-serial composition
  - Communication through ports and variables, events
Specification Level Communication

- **Message-passing**
  - Abstract communication and synchronization
  - Encapsulate in channel
Clean Specification Model Example

- **Synthesizable specification model**
  - Hierarchical parallel-serial composition
  - Communication through variables and standard channels
Computation Refinement

- PE allocation / selection
- Behavior partitioning
- Variable partitioning
- Scheduling
• Allocate PEs
• Partition behaviors
• Globalize communication

➢ Additional level of hierarchy to model PE structure
Model after Behavior Partitioning

- Synchronization to preserve execution order/semantics
Variable Partitioning

- **Shared memory vs. message passing implementation**
  - Map global variables to local memories
  - Communicate data over message-passing channels

![Diagram](image)
Keep local variable copies in sync
- Communicate updated values at synchronization points
- Transfer control & data over message-passing channel
Timed Computation

- **Execution time of behaviors**
  - Estimated target delay / timing budget
- **Granularity**
  - Behavior / function / basic-block level

- **Annotate behaviors**
  - Simulation feedback
  - Synthesis constraints

```c
behavior B2( in int v1, ISend c2 ) {
    void main(void) {
        ...
        waitfor( B2_DELAY1 );
        c2.send( ... );
        ...
        waitfor( B2_DELAY2 );
    }
};
```
Scheduling

- Serialize behavior execution on components

- Static scheduling
  - Fixed behavior execution order
  - Flattened behavior hierarchy

- Dynamic scheduling
  - Pool of tasks
  - Scheduler, abstracted OS
Computation Model Example
Computation Model

- **Component structure/architecture**
  - Top level of behavior hierarchy

- **Behavioral/functional component view**
  - Behaviors grouped under top-level component behaviors
  - Sequential behavior execution

- **Timed**
  - Estimated execution delays
Communication Refinement

- Network allocation / protocol selection
- Channel partitioning
- Protocol stack insertion
- Inlining
Network Allocation / Channel Partitioning

- Allocate busses
- Partition channels
- Update communication

➢ Additional level of hierarchy to model bus structure
Model after Channel Partitioning
Protocol Insertion

- **Insert protocol layer**
  - Bus protocol channel from database

- **Create network layers**
  - Implement message-passing over bus protocol

- **Replace bus channel**
  - Hierarchical combination of complete protocol stack
Model after Protocol Insertion

**Master**

- PE1
  - B1
  - B13snd
  - B2
  - B34rcv

**Slave**

- PE2
  - B13rcv
  - B3
  - B34snd

**Bus1**

- **BusProtocol**
  - address[15:0]
  - data[31:0]
  - ready
Inlining

- Create bus interfaces and drivers
- Refine communication
Communication Model Example
Communication Model

- **Component & bus structure/architecture**
  - Top level of hierarchy

- **Bus-functional component models**
  - Timing-accurate bus protocols
  - Behavioral component description

- **Timed**
  - Estimated component delays
Processor Refinement

- **Cycle-accurate implementation of PEs**
  - Hardware synthesis down to RTL
  - Software synthesis down to IS
  - Interface synthesis down to RTL/IS
**Schedule operations into clock cycles**

- Define clock boundaries in leaf behavior C code
- Create FSMD model from scheduled C code
Software Synthesis

- Implement behavior on processor instruction-set
  - Code generation
  - Compilation
Interface Synthesis

- Implement communication on components
  - Hardware bus interface logic
  - Software bus drivers
Implementation Model

Software processor

Custom hardware

Instruction Set Simulator (ISS)

OBJ

PORTA

PORTB

PORTC

INTA

OBJ

address[15:0]
data[31:0]
ready
ack

PE1_CLK

PE2_CLK

S0

S1

S2

S3

S4
Implementation Model

- **Cycle-accurate system description**
  - RTL description of hardware
    - Behavioral/structural FSMD view
  - Object code for processors
    - Instruction-set co-simulation
  - Clocked bus communication
    - Bus interface timing based on PE clock
Lecture 4: Summary

• **Design methodology**
  - Four levels of abstraction
    - Specification model: untimed, functional
    - Computation model: estimated, structural
    - Communication model: timed, bus-functional
    - Implementation model: cycle-accurate, RTL/IS
  - Three refinement steps
    - Computation refinement
    - Communication refinement
    - Processor refinement
      » HW / SW / interface synthesis
  - Well-defined, formal models & transformations
    - Automatic, gradual refinement
    - Executable models, test bench re-use
    - Simple verification