EECS 222: Embedded System Modeling Lecture 15

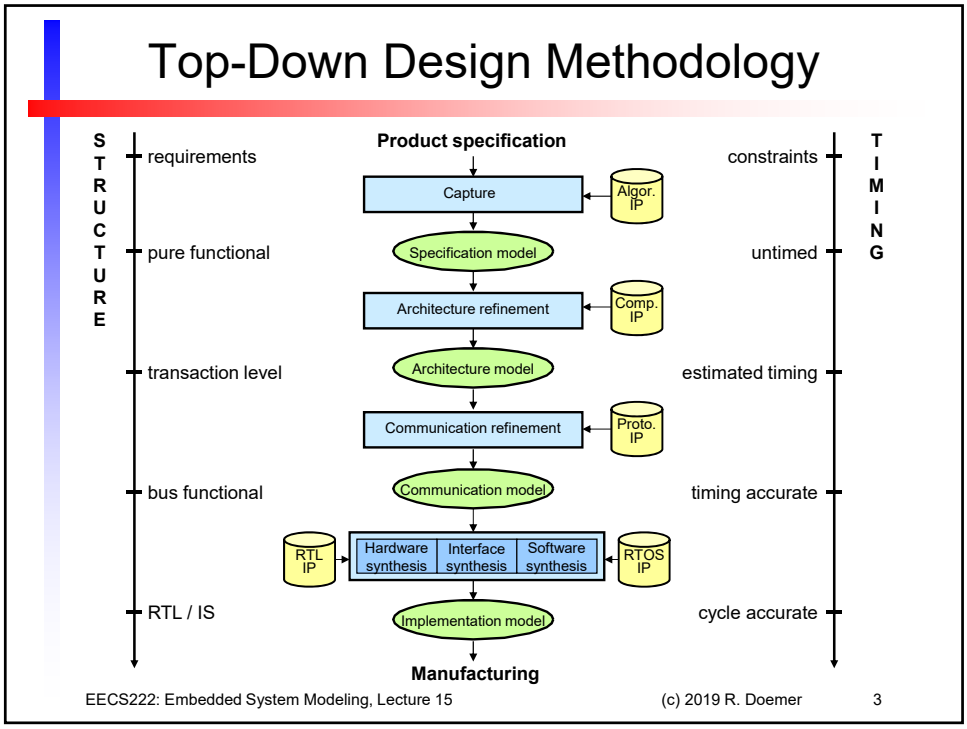
Rainer Dömer

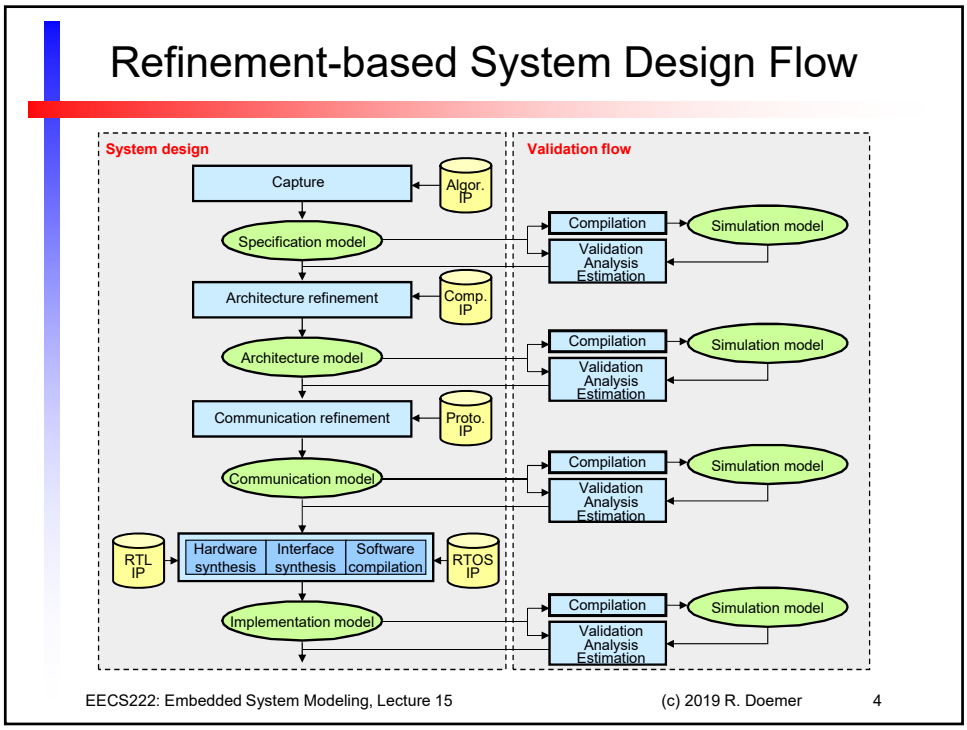
doemer@uci.edu

The Henry Samueli School of Engineering
Electrical Engineering and Computer Science
University of California, Irvine

Lecture 15: Overview

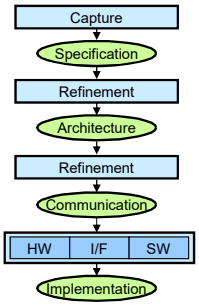
- Embedded System Design Flow
  - Top-down design methodology
  - Refinement-based design flow
    - Specify
    - Explore
    - Refine
- System-on-Chip Environment (SCE)
  - Design example: GSM Vocoder
  - Architecture refinement
  - Communcation refinement
  - Interactive demonstration


Top-Down Design Methodology
STRUCTURE
TIMING
requirements
pure functional
transaction level
bus functional
RTL / IS
constraints
untimed
estimated timing
timing accurate
cycle accurate
Product specification
Capture
Algor. IP
Specification model
Architecture refinement
Comp. IP
Architecture model
Communication refinement
Proto. IP
Communication model
RTL IP
Hardware synthesis
Interface synthesis
Software synthesis
RTOS IP
Implementation model
Manufacturing
EECS222: Embedded System Modeling, Lecture 15
(c) 2019 R. Doemer
3


Refinement-based System Design Flow
System design
Validation flow
Capture
Algor. IP
Specification model
Architecture refinement
Comp. IP
Architecture model
Communication refinement
Proto. IP
Communication model
RTL IP
Hardware synthesis
Interface synthesis
Software compilation
RTOS IP
Implementation model
Compilation
Validation Analysis Estimation
Simulation model
EECS222: Embedded System Modeling, Lecture 15
(c) 2019 R. Doemer
4

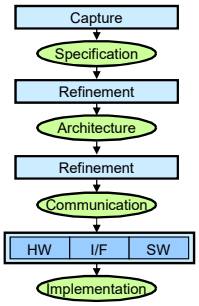
Refinement-based System Design Flow

- Refinement steps
  - Architecture refinement (Specification -> Architecture)
  - Communication refinement (Architecture -> Communication)
  - Cycle-accurate refinement (Communication -> RTL/IS)
    - HW / SW / interface synthesis
- Levels of abstraction
  - Specification model: untimed, functional
  - Architecture model: estimated, structural
  - Communication model: timed, bus-functional
  - Implementation model: cycle-accurate, RTL/IS
- Component data bases
  - Algorithms for specification
  - Components for architecture
  - Busses for communication
  - RTOS for SW
  - RTL components for HW



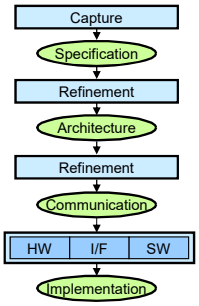
Refinement-based System Design Flow

- Step 1: Architecture Refinement
  - Allocation of Processing Elements (PE)
    - Type and number of processors
    - Type and number of custom hardware blocks
    - Type and number of system memories
  - Mapping to PEs
    - Map each behavior to a PE
    - Map each channel to a PE
    - Map each variable to a PE
  - ➢ Result
    - System architecture of concurrent PEs with abstract communication via channels



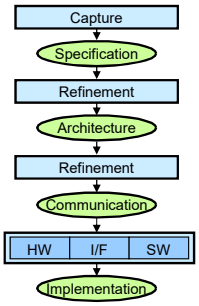
Refinement-based System Design Flow

- Step 2: Scheduling Refinement
  - For each PE, serialize the execution of behaviors to a single thread of control
  - Option (a): Static scheduling
    - For each set of concurrent behaviors, determine fixed order of execution
  - Option (b): Dynamic RTOS scheduling
    - Choose scheduling policy, e.g. round-robin or priority-based
    - For each set of concurrent behaviors, determine scheduling priority
  - ➢ Result
    - System model with abstract scheduler inserted in each PE



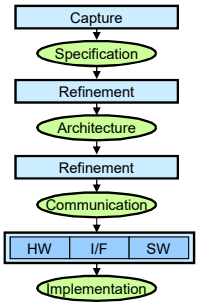
Refinement-based System Design Flow

- Step 3: Network / Communication Refinement
  - Allocation of system busses
    - Type and number of system busses
    - Type of bus protocol for each bus (if applicable)
    - Number of transducers (if applicable)
    - System connectivity
  - Mapping of channels to busses
    - Map each channel to a system bus (or a network of multiple busses)
  - ➢ Result
    - Transaction-Level Model (TLM), or Bus-Functional Model (BFM)



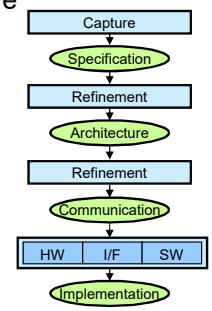
Refinement-based System Design Flow

- Step 4: Hardware Refinement (for HW PE)
  - Allocation of Register Transfer Level (RTL) components
    - Type and number of functional units (e.g. adder, multiplier, ALU)
    - Type and number of storage units (e.g. registers, register files)
    - Type and number of interconnecting busses (drivers, multiplexers)
  - Scheduling
    - Basic blocks assigned to super-states
    - Individual operations assigned to clock cycles
  - Binding
    - Bind functional operations to functional units
    - Bind variables to storage units
    - Bind assignments/transfers to busses
  - ➢ Result
    - Clock-cycle accurate model of each HW PE
    - Output: Synthesizable Verilog description



Refinement-based System Design Flow

- Step 5: Software Refinement (for SW PE)
  - C code generation
    - For selected target processor
  - RTOS targeting
    - Thin adapter layer for selected target RTOS
  - Cross-compilation to Instruction Set Architecture
    - for Instruction Set Simulation (ISS)
    - for target processor embedded in target system
  - Assembly and Linking
  - ➢ Result
    - Clock-cycle accurate, or instruction-accurate model of each SW PE
    - Output: binary image



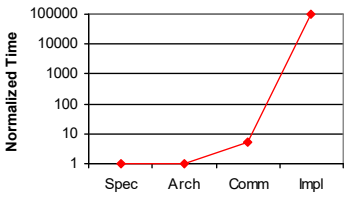
SCE Demonstration

- Application Example: GSM Vocoder
  - Enhanced full-rate voice codec
  - GSM standard for mobile telephony (GSM 06.10)
  - Lossy voice encoding/decoding
    - Incoming speech samples @ 104 kbit/s
    - Encoded bit stream @ 12.2 kbit/s
    - Frames of 4 x 40 = 160 samples (4 x 5ms = 20ms of speech)
  - Real-time constraint:
    - max. 20ms per speech frame
      (max. total of 3.26s for sample speech file)
  - SpecC specification model
    - 29 hierarchical behaviors (9 par, 10 seq, 10 fsm)
    - 73 leaf behaviors
    - 9139 formatted lines of SpecC code
      (~13000 lines of original C code, including comments)

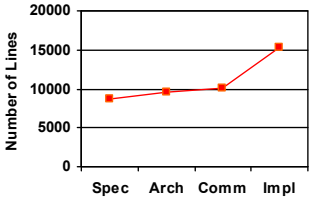
SCE Experimental Results

- Design Example 1: GSM Vocoder

Simulation Speed



Code Size



Refinement Effort

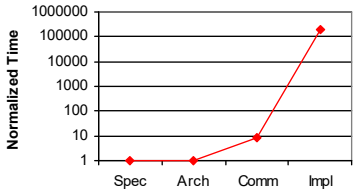
	Modified lines	Manual effort	Automated User / Refine
Spec → Arch	3,275	3~4 month	15 min / < 1 min
Arch → Comm	914	1~2 month	5 min / < 0.5 min
Comm → Impl	6,146	5~6 month	30 min / < 2 min
Total	10,355	9~12 month	50 min / < 4 min

➢ Productivity gain: 12 months vs. 1 hour = 2000x !

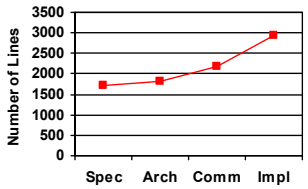
SCE Experimental Results

- Design Example 2: JPEG Encoder

Simulation Speed



Code Size



Refinement Effort

	Modified lines	Manual effort	Automated User / Refine
Spec → Arch	751	1~2 month	5 min / < 0.5 min
Arch → Comm	492	~1 month	3 min / < 0.5 min
Comm → Impl	1,278	3~4 month	20 min / < 1 min
Total	2,521	5~7 month	28 min / < 2 min

➢ Productivity gain: 6 months vs. 1/2 hour = 2000x !