ECPS 203
Embedded Systems Modeling and Design
Lecture 13

Rainer Dömer

doemer@uci.edu

Center for Embedded and Cyber-physical Systems
University of California, Irvine

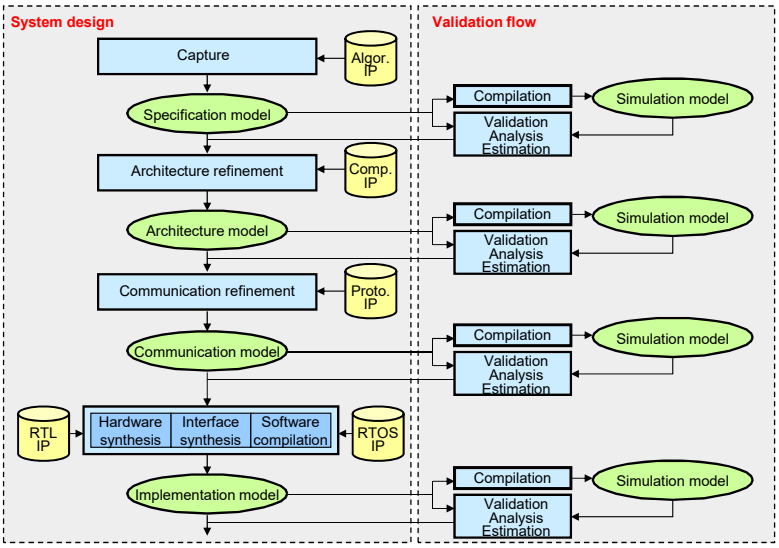
Lecture 13: Overview

- Assignment 7
  - Performance measurement on prototyping board
- Embedded System Design Flow
  - Refinement-based design flow
    - Specify
    - Explore
    - Refine
- System-on-Chip Environment (SCE)
  - Application example: GSM Vocoder
  - Interactive demonstration (part 2)

Project Assignment 7

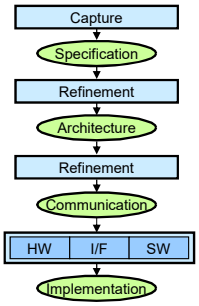
- Task: Performance measurement on prototyping board
  - Run C++ model of Canny Edge Detector on Raspberry Pi
  - Obtain absolute timing measurements of Canny functions
- Steps
  1. Prepare the prototyping board with Raspbian operating system
  2. Upload **`Canny.cpp`** from Assignment 4 and compile it
  3. Instrument the source code with real-time measurements
  4. Note the computation delays of the major Canny functions
- Deliverables
  - **`Canny.cpp`** (model instrumented with timing measurements)
  - **`Canny.txt`** (table of measured delays)
- Due
  - Wednesday, November 20, 2019, 6pm

Specify, Explore, Refine - Design Flow



Specify, Explore, Refine - Design Flow

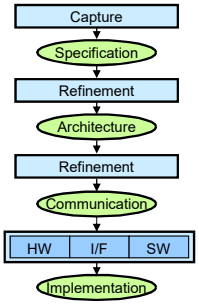
- Refinement Step 1: System Architecture
  - Allocation of Processing Elements (PE)
    - Type and number of processors
    - Type and number of custom hardware blocks
    - Type and number of system memories
  - Mapping to PEs
    - Map each behavior to a PE
    - Map each channel to a PE
    - Map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



ECPS203: Embedded Systems Modeling and Design, Lecture 13 (c) 2019 R. Doemer 5

Specify, Explore, Refine - Design Flow

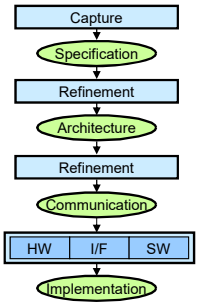
- Refinement Step 2: PE Scheduling
  - For each PE, serialize the execution of behaviors to a single thread of control
  - Option (a): Static scheduling
    - For each set of concurrent behaviors, determine fixed order of execution
  - Option (b): Dynamic RTOS scheduling
    - Choose scheduling policy, 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



ECPS203: Embedded Systems Modeling and Design, Lecture 13 (c) 2019 R. Doemer 6

Refinement-based System Design Flow

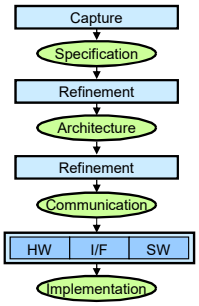
- Step 3: Network / Communication Refinement
  - Allocation of system busses
    - Type and number of system busses
    - Type of bus protocol for each bus (if applicable)
    - Number of transducers (if applicable)
    - System connectivity
  - Mapping of channels to busses
    - Map each channel to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



ECPS203: Embedded Systems Modeling and Design, Lecture 13 (c) 2019 R. Doemer 7

Refinement-based System Design Flow

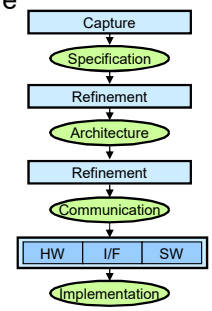
- Step 4: Hardware Refinement (for HW PE)
  - Allocation of Register Transfer Level (RTL) components
    - Type and number of functional units (e.g. adder, multiplier, ALU)
    - Type and number of storage units (e.g. registers, register files)
    - Type and number of interconnecting busses (drivers, 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



ECPS203: Embedded Systems Modeling and Design, Lecture 13 (c) 2019 R. Doemer 8

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



ECPS203: Embedded Systems Modeling and Design, Lecture 13 (c) 2019 R. Doemer 9

SCE Demonstration

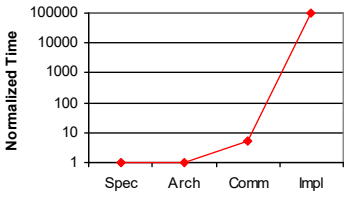
- Application Example: GSM Vocoder
  - Enhanced full-rate voice codec
  - GSM standard for mobile telephony (GSM 06.10)
  - Lossy voice encoding/decoding
    - Incoming speech samples @ 104 kbit/s
    - Encoded bit stream @ 12.2 kbit/s
    - Frames of 4 x 40 = 160 samples (4 x 5ms = 20ms of speech)
  - 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)

ECPS203: Embedded Systems Modeling and Design, Lecture 13 Copyright © 2003 CECS 10

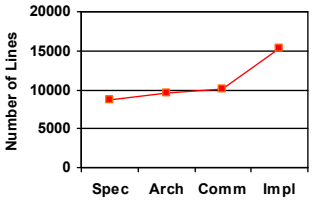
SCE Experimental Results

- Design Example 1: GSM Vocoder

Simulation Speed



Code Size



Refinement Effort

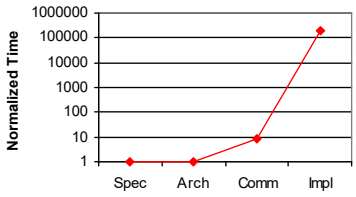
	Modified lines	Manual effort	Automated User / Refine
Spec → Arch	3,275	3~4 month	15 min / < 1 min
Arch → Comm	914	1~2 month	5 min / < 0.5 min
Comm → Impl	6,146	5~6 month	30 min / < 2 min
Total	10,355	9~12 month	50 min / < 4 min

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

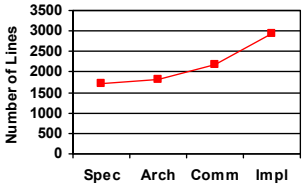
SCE Experimental Results

- 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 !

