ECPS 203
Embedded Systems Modeling and Design
Lecture 11

Rainer Dömer

doemer@uci.edu

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

Lecture 11: Overview

- Course Administration
  - Midterm course evaluation
  - Discussion of results
- Embedded System Design Flow
  - Top-down design methodology
  - Refinement-based design flow
    - Specify
    - Explore
    - Refine
- System-on-Chip Environment (SCE)
  - Application example: GSM Vocoder
  - Interactive demonstration (part 1)

Course Administration

- Midterm Course Evaluation: Results
  - 13 out of 20 responses: somewhat representative
  - Overall very positive feedback
    - “Prof. Doemer plans well for the courses and is very patient”
    - “It's great to have someone who is involved in the origins of the material teaching a class about it.”
    - “explains the details thoroughly and does encourage us to give suggestions”
    - “he is full of spirit and energy to teach us”
    - “methods are very visual and explanations simple and precise”
    - “organized, knowledgeable”, “very good”
  - Suggestions on lectures
    - “it's a good idea to combine the course with a homework project”
    - “big picture […] hw5. I wish something like this could occur more in our class”
    - “It would help if some real system or scenario examples could be discussed”
    - “PDF file of the lecture […] can be posted before the class”
    - “choices for […] question and […] vote for the answer […] interesting”

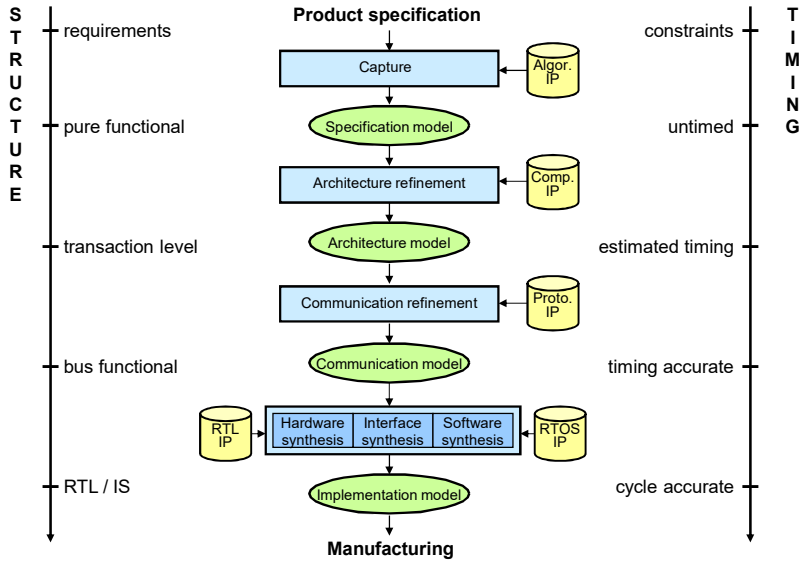
Course Administration

- Midterm Course Evaluation: Results (continued)
  - Suggestions on assignments
    - “step-by-step approach to tackle SystemC implementation in Canny Edge Detection is greatly appreciated”
    - “assignment could be more challenging”
    - “could be tougher and harder”
    - “bigger project in the end”, “use the popular tools or IDEs like vivado of Xilinx”
  - Suggestions on discussions
    - “TA session […] a bit monotonous”
    - “make the discussion more useful”
    - “discussions are...rather unfortunate”
    - “seems like our TA's hands are tied in this course, in that he was instructed to not go through the assignments with too much detail, perhaps in an effort to get the students to explore the assignments themselves”
    - “suggest letting the TA talk about his understanding […], interesting things related to the course”

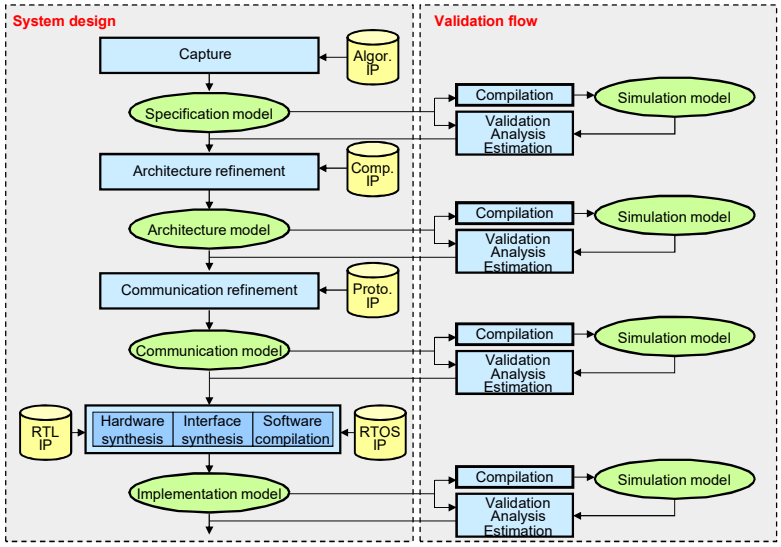
Course Administration

- Midterm Course Evaluation: Results (continued)
  - Numerical scores
    - Instructor comes well-prepared: 85% strongly agree, 15% agree
    - Presentations are well-organized: 77% strongly agree, 23% agree
    - Explanations are clear: 69% strongly agree, 31% agree
    - Helpful examples: 62% strongly agree, 38% agree
    - Opportunities for questions: 77% strongly agree, 23% agree
    - Handwriting is legible: 54% strongly agree, 46% agree
    - Speaks clearly: 75% strongly agree, 25% agree
    - Speaks loud enough: 92% strongly agree, 8% agree
    - Pace is appropriate: 77% strongly agree, 23% agree
  - Overall grade
    - Instructor: 69% A, 31% A-

Top-Down Design Methodology



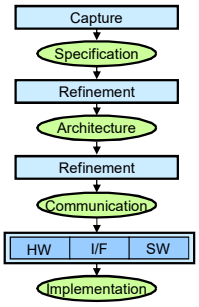
Specify, Explore, Refine - Methodology



ECPS203: Embedded Systems Modeling and Design, Lecture 11 (c) 2018 R. Doemer 7

Specify, Explore, Refine - Design Flow

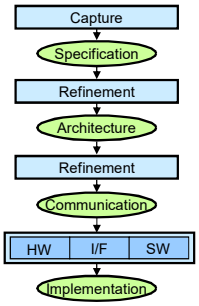
- Refinement steps
  - Architecture refinement (Specification -> Architecture)
  - Communication refinement (Architecture -> Communication)
  - Cycle-accurate refinement (Communication -> RTL/IS)
    - HW / SW / interface synthesis
- Levels of abstraction
  - Specification model: untimed, functional
  - 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



ECPS203: Embedded Systems Modeling and Design, Lecture 11 (c) 2018 R. Doemer 8

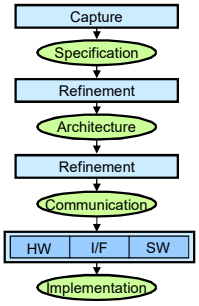
Specify, Explore, Refine - Design Flow

- 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



Specify, Explore, Refine - Design Flow

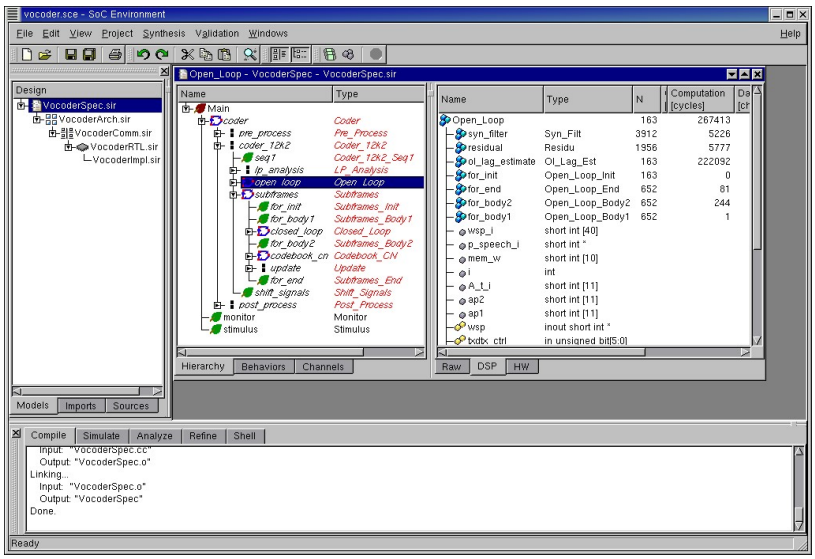
- Refinement Step 2: PE Scheduling
  - For each PE, serialize the execution of behaviors to a single thread of control
  - Option (a): Static scheduling
    - For each set of concurrent behaviors, determine fixed order of execution
  - Option (b): Dynamic RTOS scheduling
    - Choose scheduling policy,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



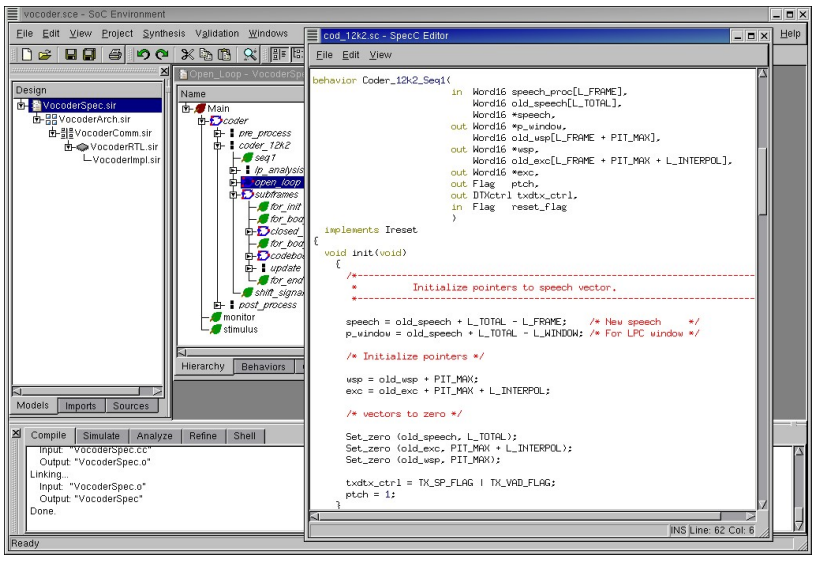
System-on-Chip Environment (SCE)

- Integrated Development Environment (IDE) with support of:
  - Graphical frontend (**`sce, scchart`**)
  - SLDL-aware editor (**`sced`**)
  - Compiler and simulator (**`scc`**)
  - Profiling and analysis (**`scprof`**)
  - Architecture refinement (**`scar`**)
  - RTOS refinement (**`scos`**)
  - Communication refinement (**`sccr`**)
  - RTL refinement (**`scrtl`**)
  - Software refinement (**`sc2c`**)
  - Scripting interface (**`scsh`**)
  - Tools and utilities (**`sir_list, sir_tree,`** …)

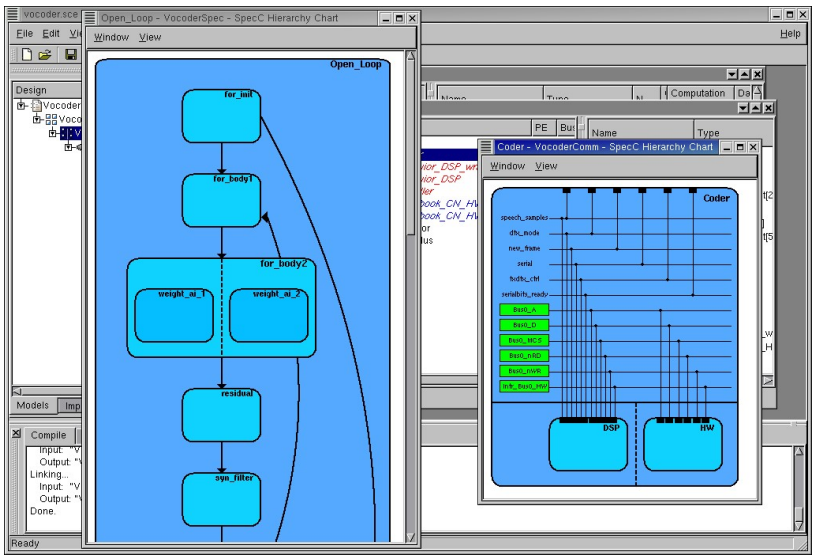
SCE Main Window



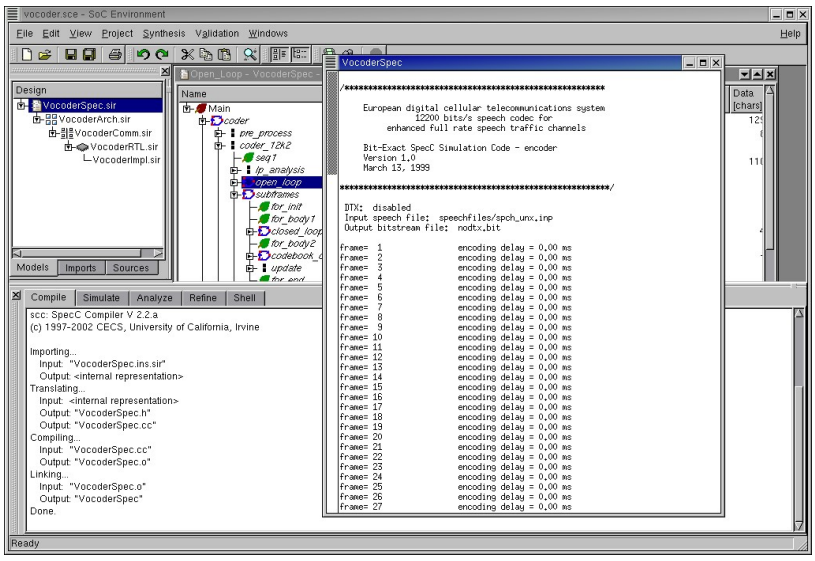
SCE Source Editor



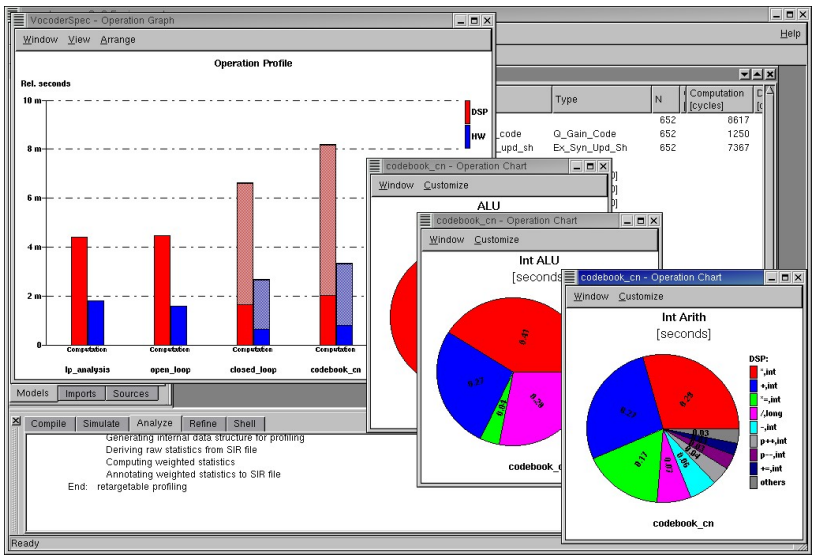
SCE Hierarchy Displays



SCE Compiler and Simulator



SCE Profiling and Analysis



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)

