

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”

ECPS203: Embedded Systems Modeling and Design, Lecture 11

(c) 2018 R. Doemer

3

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”

ECPS203: Embedded Systems Modeling and Design, Lecture 11

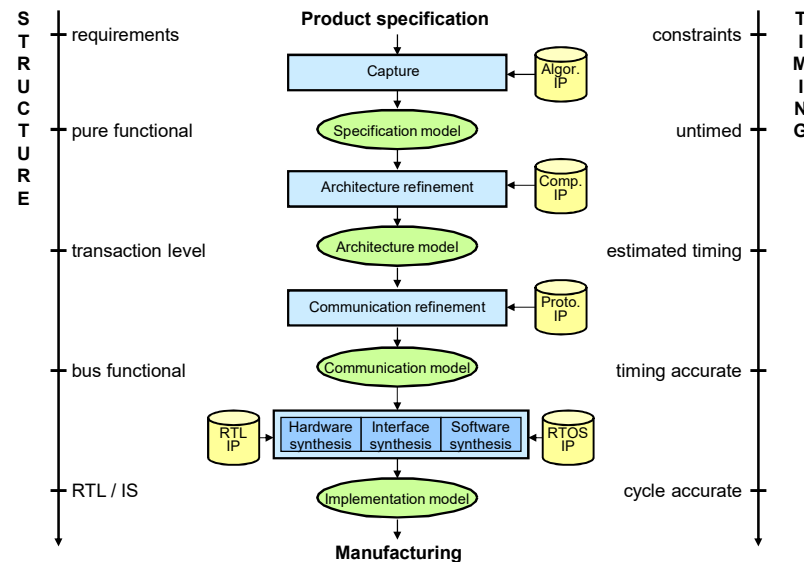
(c) 2018 R. Doemer

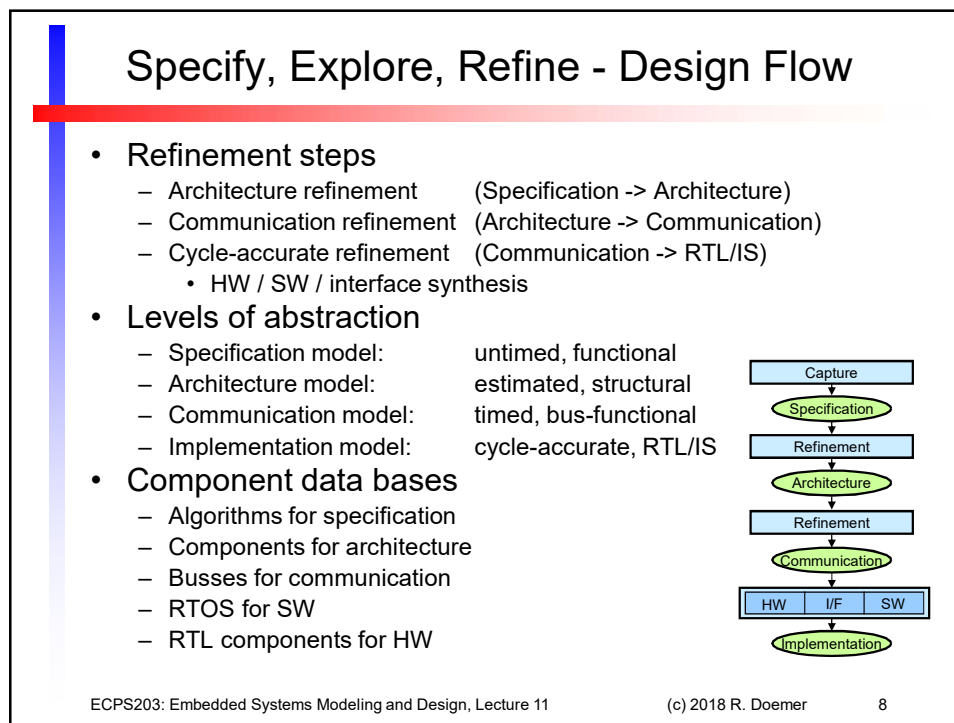
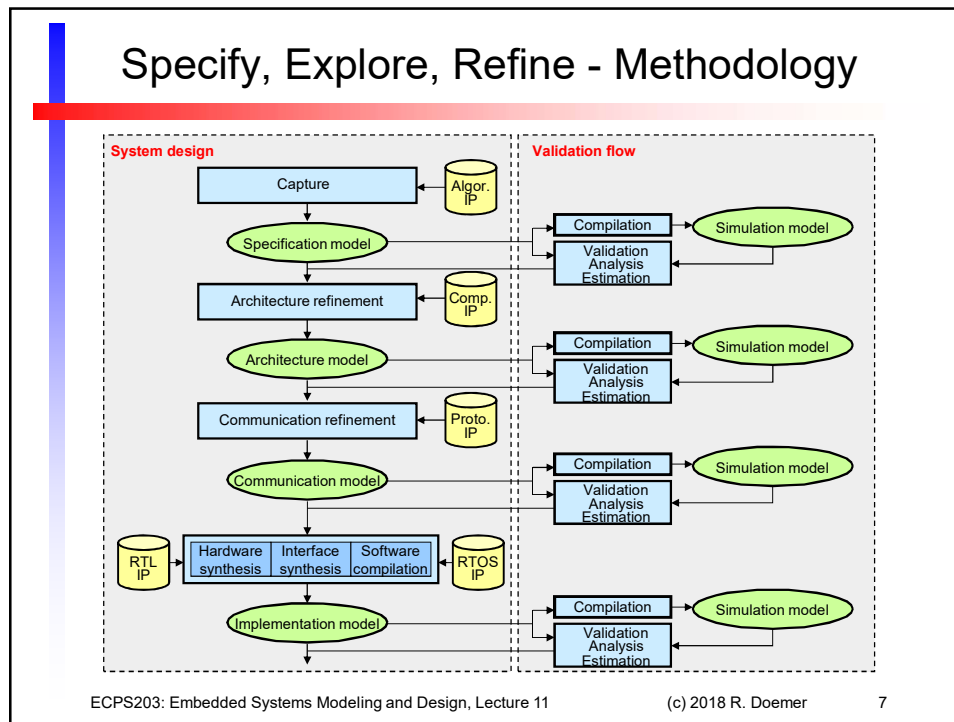
4

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





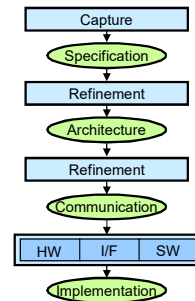
Specify, Explore, Refine - Design Flow

- 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



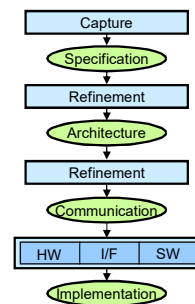
Specify, Explore, Refine - Design Flow

- 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)

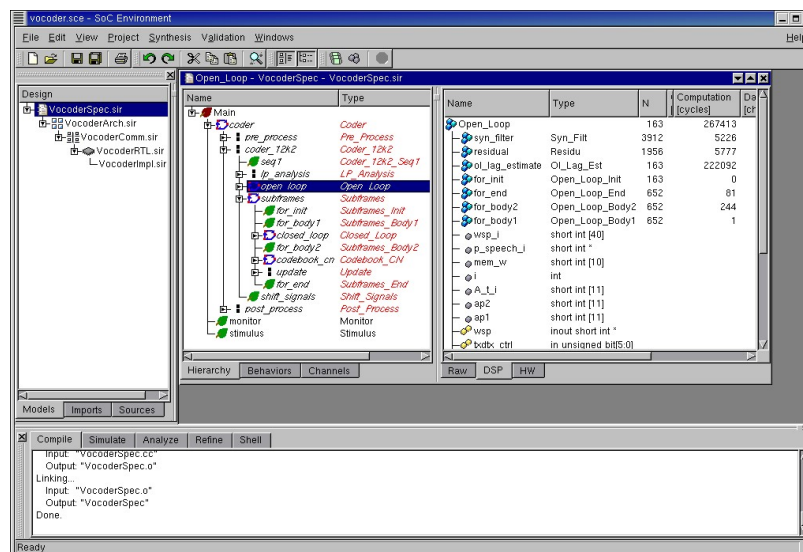
- Integrated Development Environment (IDE) with support of:
 - Graphical frontend (*sce*, *scchart*)
 - SLDL-aware editor (*sced*)
 - Compiler and simulator (*scc*)
 - Profiling and analysis (*scprof*)
 - Architecture refinement (*scar*)
 - RTOS refinement (*scos*)
 - Communication refinement (*sccr*)
 - RTL refinement (*scrtl1*)
 - Software refinement (*sc2c*)
 - Scripting interface (*scsh*)
 - Tools and utilities (*sir_list*, *sir_tree*, ...)

ECPS203: Embedded Systems Modeling and Design, Lecture 11

(c) 2018 R. Doemer

11

SCE Main Window



ECPS203: Embedded Systems Modeling and Design, Lecture 11

Copyright © 2003 CECS

12

SCE Source Editor

The screenshot shows the SCE Source Editor interface. On the left, a project tree displays the hierarchy: VocoderSpec.sir, VocoderArch.sir, VocoderComm.sir, VocoderRTL.sir, and VocoderImpl.sir. The main editor window shows the source code for 'behavior Coder_12k2_Seq1'. The code includes input/output declarations, an 'implements Ireset' block, and an 'init' function that initializes pointers and variables like 'speech', 'p_window', 'wsp', and 'exc'.

```

behavior Coder_12k2_Seq1
in Word16 speech_proc[L_FRAME],
Word16 old_speech[L_TOTAL],
Word16 *speech,
out Word16 *p_window,
Word16 old_wsp(L_FRAME + PIT_MAK),
out Word16 *wsp,
Word16 old_exc(L_FRAME + PIT_MAK + L_INTERPOL),
out Word16 *exc,
out Flag ptch,
out BitCtrl txdtx_ctrl,
in Flag reset_flag
)

implements Ireset
{
void init(void)
{
/* Initialize pointers to speech vector.
speech = old_speech + L_TOTAL - L_FRAME; /* New speech */
p_window = old_speech + L_TOTAL - L_WINDOW; /* For LPC window */

/* Initialize pointers */
wsp = old_wsp + PIT_MAK;
exc = old_exc + PIT_MAK + L_INTERPOL;

/* vectors to zero */
Set_zero(old_speech, L_TOTAL);
Set_zero(old_exc, PIT_MAK + L_INTERPOL);
Set_zero(old_wsp, PIT_MAK);

txdtx_ctrl = TX_SP_FLAG | TX_VAD_FLAG;
ptch = 1;
}
}
    
```

ECPS203: Embedded Systems Modeling and Design, Lecture 11 Copyright © 2003 CECS 13

SCE Hierarchy Displays

The screenshot displays two views of the SCE Hierarchy. The left window, titled 'Open_Loop - VocoderSpec - SpecC Hierarchy Chart', shows a flowchart with nodes: 'for_init', 'for_body1', 'for_body2', 'weight_at_1', 'weight_at_2', 'residual', and 'syn_filter'. The right window, titled 'Coder - VocoderComm - SpecC Hierarchy Chart', shows a block diagram with a 'Coder' block containing 'speech_sampler', 'dft_block', 'res_block', 'serial', 'bitCtrl', 'weight_ready', and 'DSP' blocks, and an 'HW' block.

ECPS203: Embedded Systems Modeling and Design, Lecture 11 Copyright © 2003 CECS 14

SCE Compiler and Simulator

The screenshot shows the VocoderSpec environment with a project tree on the left and a console window displaying the compilation output. The console output includes the following information:

```

scc: SpecC Compiler V 2.2 a
(c) 1997-2000 CECS, University of California, Irvine

Importing...
Input: "VocoderSpec.ins.sir"
Output: <internal representation>
Translating...
Input: <internal representation>
Output: "VocoderSpec.h"
Output: "VocoderSpec.cc"
Compiling...
Input: "VocoderSpec.cc"
Output: "VocoderSpec.o"
Linking...
Input: "VocoderSpec.o"
Output: "VocoderSpec"
Done.
    
```

The main window displays the source code for a Bit-Exact SpecC Simulation Code - encoder, Version 1.0, dated March 15, 1999. The code includes a list of frames with their respective encoding delays:

```

Frame= 1 encoding delay = 0,00 ms
Frame= 2 encoding delay = 0,00 ms
Frame= 3 encoding delay = 0,00 ms
Frame= 4 encoding delay = 0,00 ms
Frame= 5 encoding delay = 0,00 ms
Frame= 6 encoding delay = 0,00 ms
Frame= 7 encoding delay = 0,00 ms
Frame= 8 encoding delay = 0,00 ms
Frame= 9 encoding delay = 0,00 ms
Frame=10 encoding delay = 0,00 ms
Frame=11 encoding delay = 0,00 ms
Frame=12 encoding delay = 0,00 ms
Frame=13 encoding delay = 0,00 ms
Frame=14 encoding delay = 0,00 ms
Frame=15 encoding delay = 0,00 ms
Frame=16 encoding delay = 0,00 ms
Frame=17 encoding delay = 0,00 ms
Frame=18 encoding delay = 0,00 ms
Frame=19 encoding delay = 0,00 ms
Frame=20 encoding delay = 0,00 ms
Frame=21 encoding delay = 0,00 ms
Frame=22 encoding delay = 0,00 ms
Frame=23 encoding delay = 0,00 ms
Frame=24 encoding delay = 0,00 ms
Frame=25 encoding delay = 0,00 ms
Frame=26 encoding delay = 0,00 ms
Frame=27 encoding delay = 0,00 ms
    
```

ECPS203: Embedded Systems Modeling and Design, Lecture 11 Copyright © 2003 CECS 15

SCE Profiling and Analysis

The screenshot displays the 'Operation Graph' window, which provides a detailed analysis of the code's execution. It includes a bar chart titled 'Operation Profile' showing relative seconds for different operations, and several pie charts for 'codebook_cn' showing the distribution of operations like 'Int ALU' and 'Int Arith'.

The 'Operation Profile' bar chart shows the following data (approximate values):

Operation	Relative Seconds
codebook	~4.5
open_loop	~4.5
closed_loop	~6.5
codebook_cn	~8.0

The 'codebook_cn' pie charts show the following data (approximate values):

Operation	Percentage
Int ALU	~40%
Int Arith	~30%
others	~30%

ECPS203: Embedded Systems Modeling and Design, Lecture 11 Copyright © 2003 CECS 16

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 $4 \times 40 = 160$ samples ($4 \times 5\text{ms} = 20\text{ms}$ 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)