

# EECS 222: Embedded System Modeling Lecture 13

Rainer Dömer

doemer@uci.edu

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

## Lecture 13: Overview

- Project Assignment 6
  - Refined structure of Gaussian Smooth
- System-on-Chip Environment (SCE)
  - Specification Modeling Guidelines
  - Design Example: GSM Vocoder
  - Interactive Demonstration

## Project Assignment 6

- Task: Hierarchical DUT of the Canny Edge Detector
  - Refine the structural hierarchy of the DUT block
  - Refine the structural hierarchy of the Gaussian Smooth block
- Steps
  1. Refine the DUT structure
    - Gaussian Smooth, Derivative, ..., Apply Hysteresis
  2. Refine the Gaussian Smooth structure
    - Receive Image, Gaussian Kernel, BlurX, BlurY
  3. Visualize the structural hierarchy of the model
- Deliverables
  - **Canny.sc** or **Canny.cpp** (choose one!)
  - **Canny.txt**
- Due: February 20, 2019, 6pm

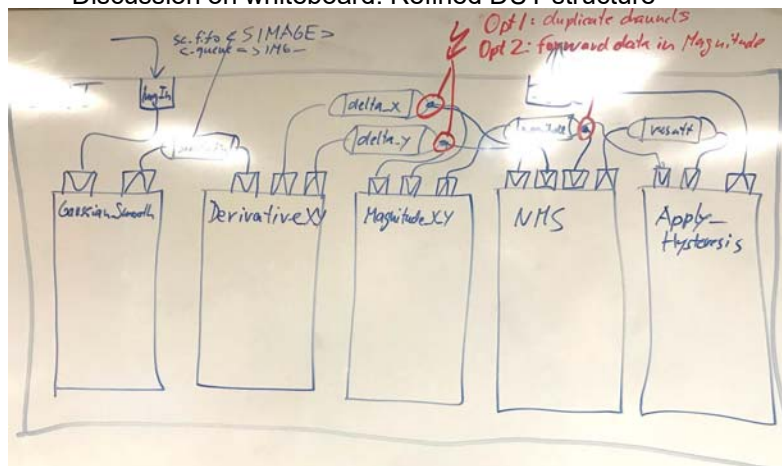
EECS222: Embedded System Modeling, Lecture 13

(c) 2019 R. Doemer

3

## Project Assignment 6

- Structural model of the DUT of the Canny Edge Detector
  - Discussion on whiteboard: Refined DUT structure



EECS222: Embedded System Modeling, Lecture 13

(c) 2019 R. Doemer

4

## Project Assignment 6

- Step 2: Refined hierarchy of the Gaussian Smooth

- Expected instance tree

DUT canny

```

|----- Gaussian_Smooth gaussian_smooth
|         |----- Receive_Image receive
|         |----- Gaussian_Kernel gauss
|         |----- BlurX blurX
|         \----- BlurY blurY
|----- Derivative_X_Y derivative_x_y
|----- Magnitude_X_Y magnitude_x_y
|----- Non_Max_Supp non_max_supp
\----- Apply_Hysteresis apply_hysteresis
  
```

EECS222: Embedded System Modeling, Lecture 13

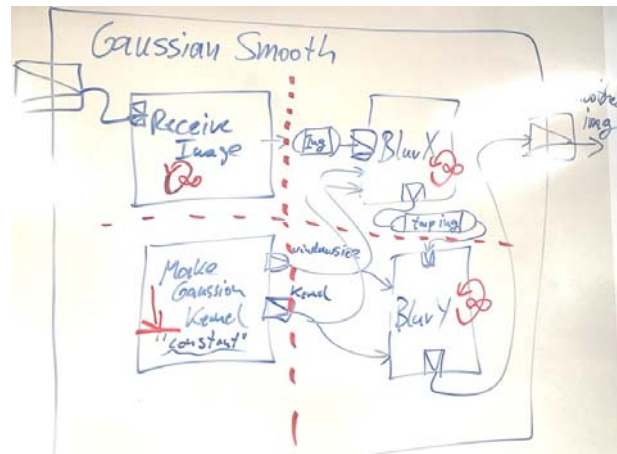
(c) 2019 R. Doemer

5

## Project Assignment 6

- Structural model of the DUT of the Canny Edge Detector

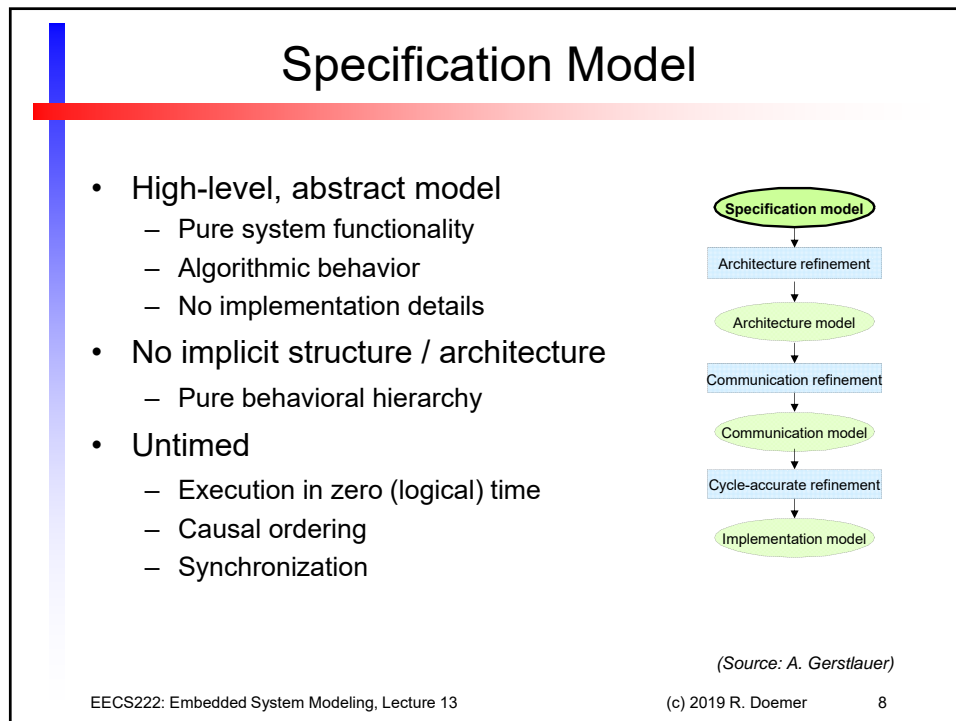
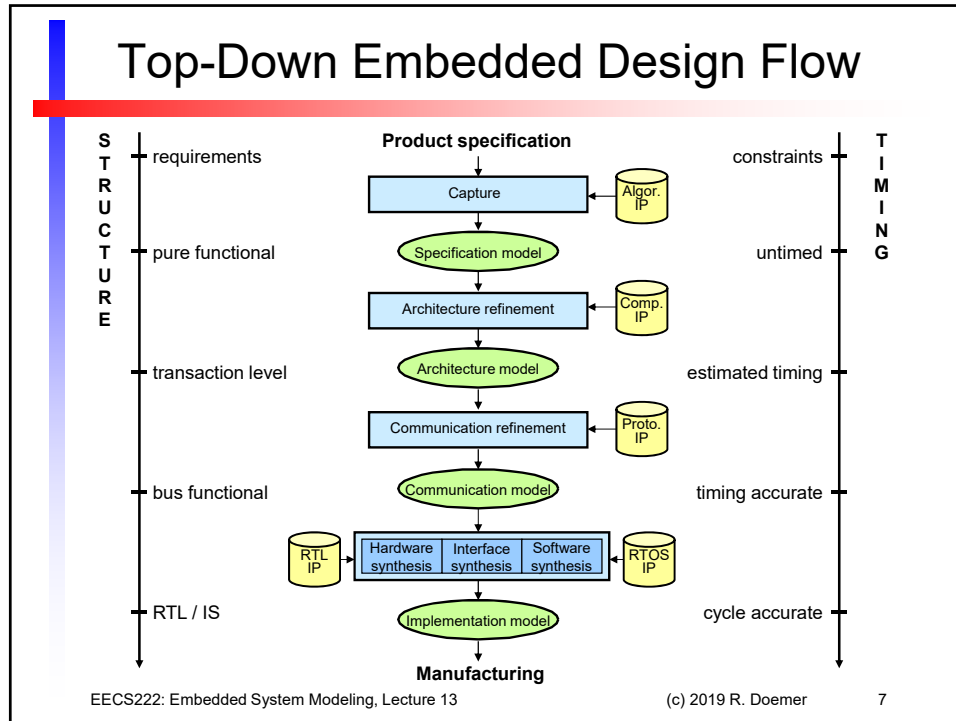
- Discussion on whiteboard: Refined Gaussian Smooth structure



EECS222: Embedded System Modeling, Lecture 13

(c) 2019 R. Doemer

6



## Specification Modeling Guidelines

- Example: Guidelines for SoC Environment (SCE)
  - Clean behavioral hierarchy
    - hierarchical behaviors:  
no code other than par, pipe, seq, fsm, and try-trap statements
    - leaf behaviors:  
Pure ANSI-C code (no SpecC constructs)
  - Clean communication
    - point-to-point communication via standard channels
    - ports of plain type or interface type, no pointers!
    - port maps to local variables or ports only
- Detailed rules for SoC Environment
  - CECS Technical Report:  
“*SCE Specification Model Reference Manual*”  
by A. Gerstlauer, R. Dömer, et al.
    - `/opt/sce-20100908/doc/SpecRM.pdf`

EECS222: Embedded System Modeling, Lecture 13

(c) 2019 R. Doemer

9

## Specification Modeling Guidelines

- Converting C reference code to SpecC
  - Major functions become behaviors
  - Function call tree becomes behavioral hierarchy
    - Function call becomes behavior instance call
    - Sequential statements become leaf behaviors
    - Control flow becomes FSM
      - Conditional statements: `if`, `if-else`, `switch`
      - Loops: `while`, `for`, `do-while`
  - Explicitly specify potential parallelism!
  - Explicitly specify communication!
    - Use standard channels, avoid shared variables
    - No global variables
    - Only local variables in behaviors and functions/methods
  - Data types
    - Avoid dynamic memory allocation
    - Avoid pointers (arrays are preferred)
    - Use explicit data types if suitable (e.g. bit vectors)

EECS222: Embedded System Modeling, Lecture 13

(c) 2019 R. Doemer

10

## System-on-Chip Environment (SCE)

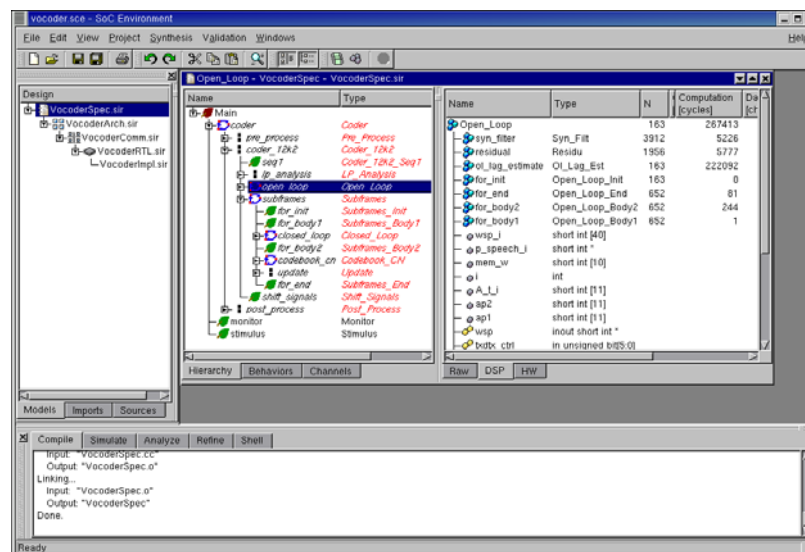
- Integrated Development Environment (IDE) with support of:
  - Graphical frontend (*sce*, *scchart*)
  - SLDL-aware editor (*sced*)
  - Compiler and simulator (*scc*)
  - Profiling and analysis (*scprof*)
  - Architecture refinement (*scar*)
  - RTOS refinement (*scos*)
  - Communication refinement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

EECS222: Embedded System Modeling, Lecture 13

(c) 2019 R. Doemer

11

## SCE Main Window



EECS222: Embedded System Modeling, Lecture 13

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 window shows the source code for 'behavior Coder\_12k2\_Seq1'. The code includes initialization for pointers to speech and LPC window vectors, and sets up control signals like txctrl and ptch.

```

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_exe[L_FRAME + PIT_MAK + L_INTERPOL],
    out  Word16 *exc,
    out  Flag ptch,
    out  DTCtrl txctrl,
    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_exe + PIT_MAK + L_INTERPOL;

    /* vectors to zero */

    Set_zero(old_speech, L_TOTAL);
    Set_zero(old_exe, PIT_MAK + L_INTERPOL);
    Set_zero(old_wsp, PIT_MAK);

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

EECS222: Embedded System Modeling, Lecture 13 Copyright © 2003 CECS 13

## SCE Hierarchy Displays

The screenshot displays two hierarchy charts. The left chart, titled 'Open\_Loop - VocoderSpec - SpecC Hierarchy Chart', shows a vertical flow of components: for\_init, for\_body, for\_body2, residual, and spn\_filter. The right chart, titled 'Coder - VocoderComm - SpecC Hierarchy Chart', shows a more complex signal flow involving DSP and HW blocks, with various control and data signals connecting them.

EECS222: Embedded System Modeling, Lecture 13 Copyright © 2003 CECS 14

## SCE Compiler and Simulator

The screenshot shows the SCE Environment interface. On the left is a Design tree with components like `VocoderSpec.sir`, `VocoderAnch.sir`, `VocoderComm.sir`, `VocoderRTL.sir`, and `VocoderImpl.sir`. The central console window displays the compilation process for `scc: SpecC Compiler V.2.2.a`, showing the input file `"VocoderSpec.ins.sir"` and the output `"VocoderSpec.o"`. The rightmost window shows simulation output for a 12000 bit/s speech code, listing frames 1 through 27 with an encoding delay of 0.00 ms for each.

EECS222: Embedded System Modeling, Lecture 13

Copyright © 2003 CECS

15

## SCE Profiling and Analysis

The screenshot displays the SCE Profiling and Analysis interface. The main window shows an "Operation Profile" bar chart with categories `lp_analysis`, `open_loop`, `closed_loop`, and `codebook_cn`. A table in the top right corner provides computation statistics:

Type	N	Computation [cycles]
codebook_cn	652	8617
codebook_cn	652	1250
codebook_cn	652	7367

Below the table are several pie charts for detailed analysis, including "Int ALU", "Int Arith", and "codebook\_cn". A legend for DSP operations is provided: `int` (red), `int` (green), `int` (blue), `int` (yellow), `int` (purple), `int` (orange), and `others` (grey).

EECS222: Embedded System Modeling, Lecture 13

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)