EECS 222: Embedded System Modeling Lecture 12

Rainer Dömer

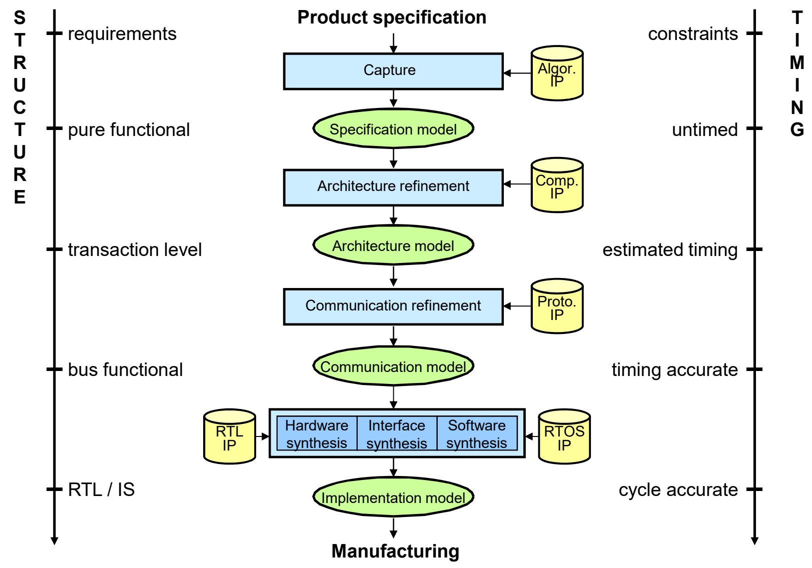
doemer@uci.edu

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

Lecture 12: Overview

- Top-Down Embedded Design Flow
  - Specification Modeling Guidelines
- System-on-Chip Environment (SCE)
  - Design Example: GSM Vocoder
  - Profiling and performance estimation
  - ➢ Interactive demonstration
- Homework Assignment 7
  - Performance estimation of the Canny Edge Detector

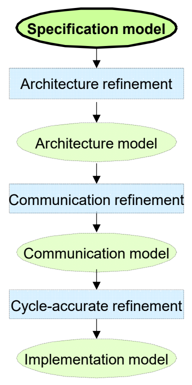
Top-Down Embedded Design Flow



EECS222: Embedded System Modeling, Lecture 12 (c) 2020 R. Doemer 3

Specification Model

- High-level, abstract model
  - Pure system functionality
  - Algorithmic behavior
  - No implementation details
- No implicit structure / architecture
  - Pure behavioral hierarchy
- Untimed
  - Execution in zero (logical) time
  - Causal ordering
  - Synchronization



(Source: A. Gerstlauer)

EECS222: Embedded System Modeling, Lecture 12 (c) 2020 R. Doemer 4

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`**

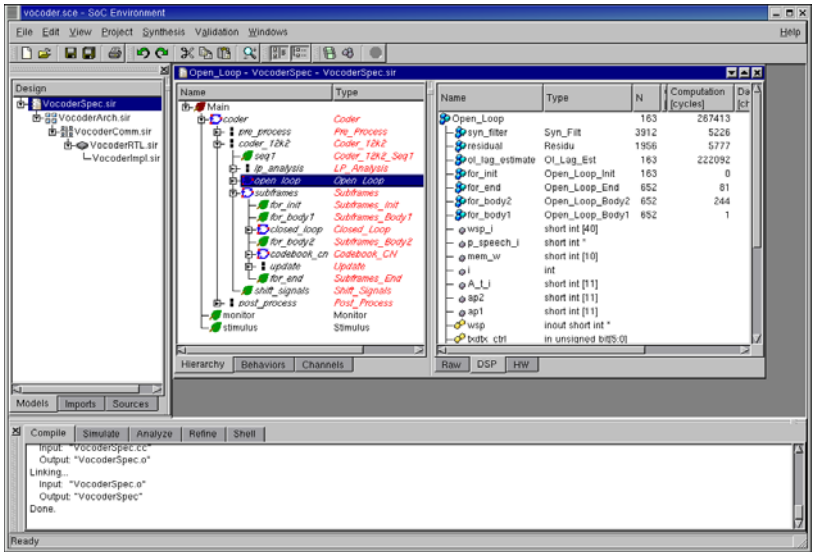
Specification Modeling Guidelines

- 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)

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

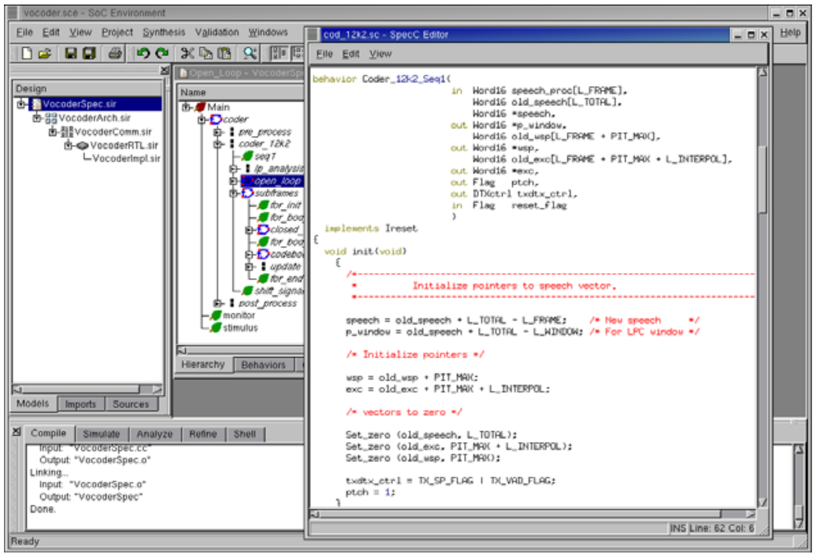

vocoder.sce - SoC Environment
File Edit View Project Synthesis Validation Windows Help
Design
VocoderSpec.sir
VocoderArch.sir
VocoderComm.sir
VocoderRTL.sir
VocoderImpl.sir
Models Imports Sources
Open_Loop - VocoderSpec - VocoderSpec.sir
Hierarchy Behaviors Channels
Raw DSP HW
Compile Simulate Analyze Refine Shell
Ready

Name	Type
Main	
coder	Coder
pre_process	Pre_Process
coder_12k2	Coder_12k2
seq1	Coder_12k2_Seq1
lp_analysis	LP_Analysis
open_loop	Open_Loop
subframes	Subframes
for_init	Subframes_Init
for_body1	Subframes_Body1
closed_loop	Closed_Loop
for_body2	Subframes_Body2
codebook_cn	Codebook_CN
update	Update
for_end	Subframes_End
shift_signals	Shift_Signals
post_process	Post_Process
monitor	Monitor
stimulus	Stimulus

Name	Type	N	Computation [cycles]	Da [ch
Open_Loop		163	267413	
syn_filter	Syn_Filt	3912	5226	
residual	Residu	1956	5777	
ol_lag_estimate	Ol_Lag_Est	163	222092	
for_init	Open_Loop_Init	163	0	
for_end	Open_Loop_End	652	81	
for_body2	Open_Loop_Body2	652	244	
for_body1	Open_Loop_Body1	652	1	
wsp_i	short int [40]			
p_speech_i	short int *			
mem_w	short int [10]			
i	int			
A_t_i	short int [11]			
ap2	short int [11]			
ap1	short int [11]			
wsp	inout short int *			
txdtx_ctrl	in unsigned bit[5:0]			

```
   Input: "VocoderSpec.cc"
   Output: "VocoderSpec.o"
Linking...
   Input: "VocoderSpec.o"
   Output: "VocoderSpec"
Done.
```

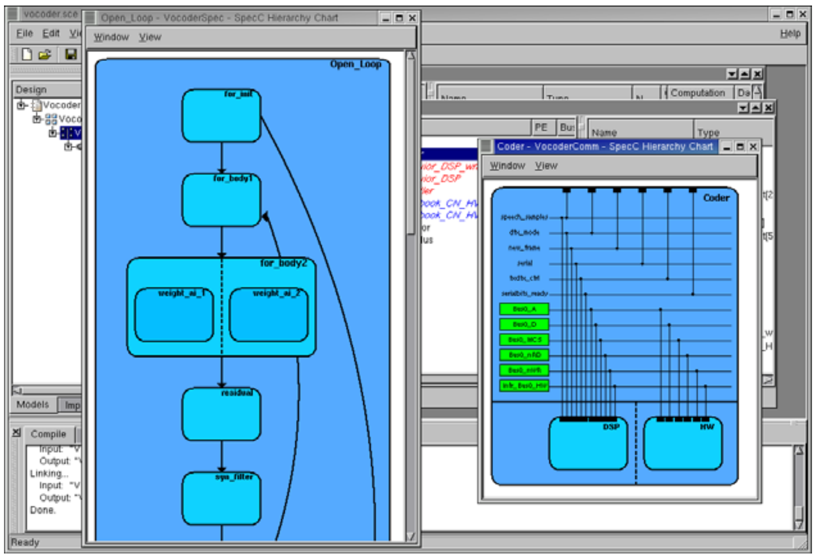
SCE Source Editor



```
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_MAX],
                          out Word16 *wsp,
                              Word16 old_exc[L_FRAME + PIT_MAX + L_INTERPOL],
                          out Word16 *exc,
                          out Flag   ptch,
                          out DTXctrl 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_MAX;
     exc = old_exc + PIT_MAX + L_INTERPOL;

     /* vectors to zero */

     Set_zero (old_speech, L_TOTAL);
     Set_zero (old_exc, PIT_MAX + L_INTERPOL);
     Set_zero (old_wsp, PIT_MAX);

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

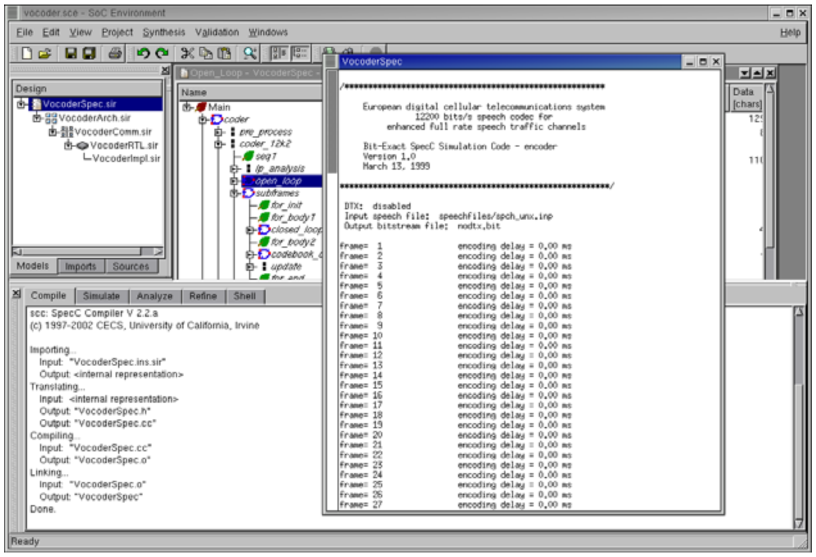
```
  Input: "VocoderSpec.cc"
  Output: "VocoderSpec.o"
Linking...
  Input: "VocoderSpec.o"
  Output: "VocoderSpec"
Done.
```

SCE Hierarchy Displays



```
  Input: "V
  Output: "V
Linking...
  Input: "V
  Output: "V
Done.
```

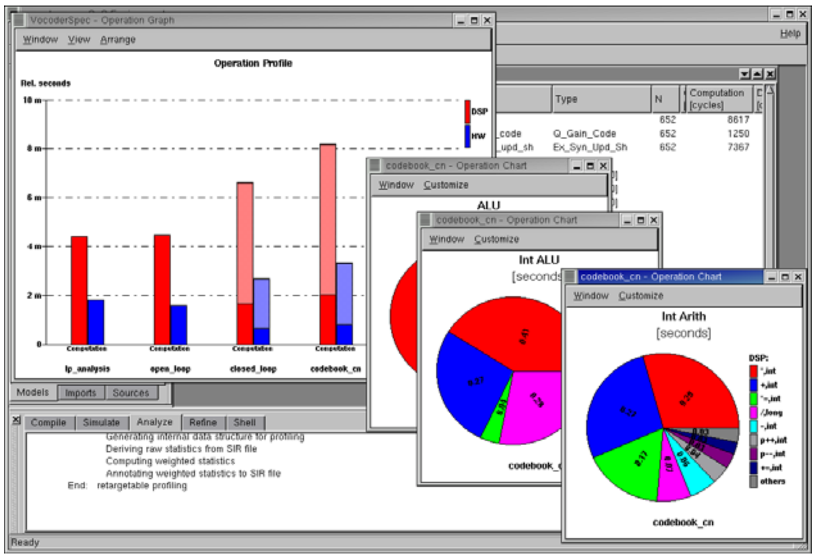
SCE Compiler and Simulator



```
scc: SpecC Compiler V 2.2.a
(c) 1997-2002 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.
```

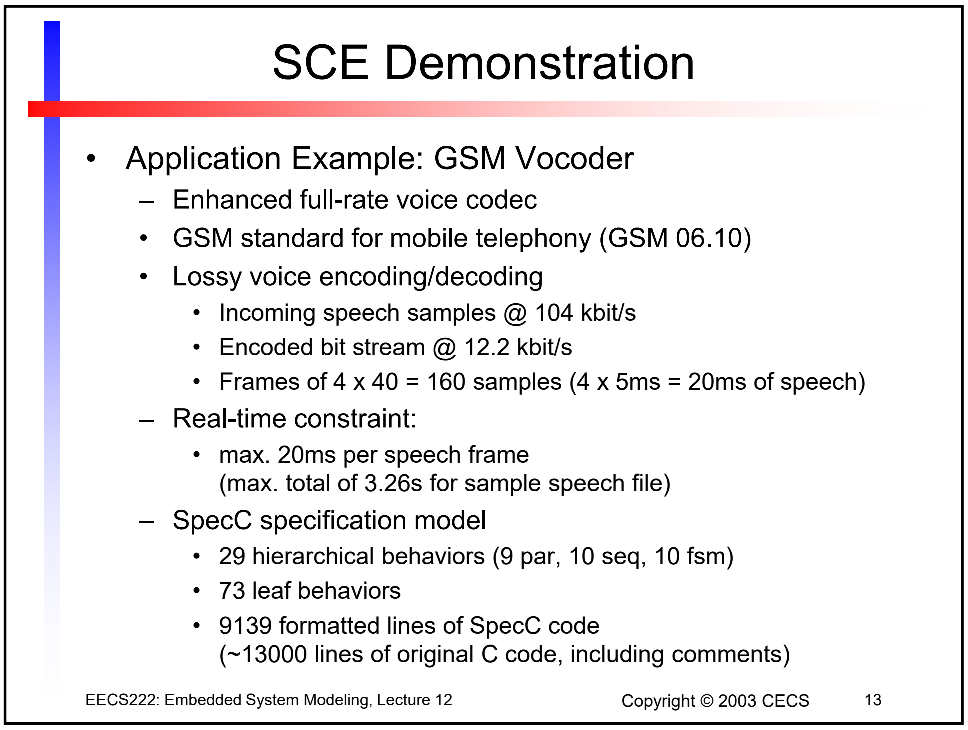
```
/*************************************************************

    European digital cellular telecommunications system
              12200 bits/s speech codec for
         enhanced full rate speech traffic channels

    Bit-Exact SpecC Simulation Code - encoder
    Version 1.0
    March 13, 1999

*************************************************************/

 DTX:  disabled
 Input speech file:  speechfiles/spch_unx.inp
 Output bitstream file:  rodtx.bit

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
```

SCE Profiling and Analysis



	Type	N	Computation [cycles]
		652	8617
_code	Q_Gain_Code	652	1250
_upd_sh	Ex_Syn_Upd_Sh	652	7367

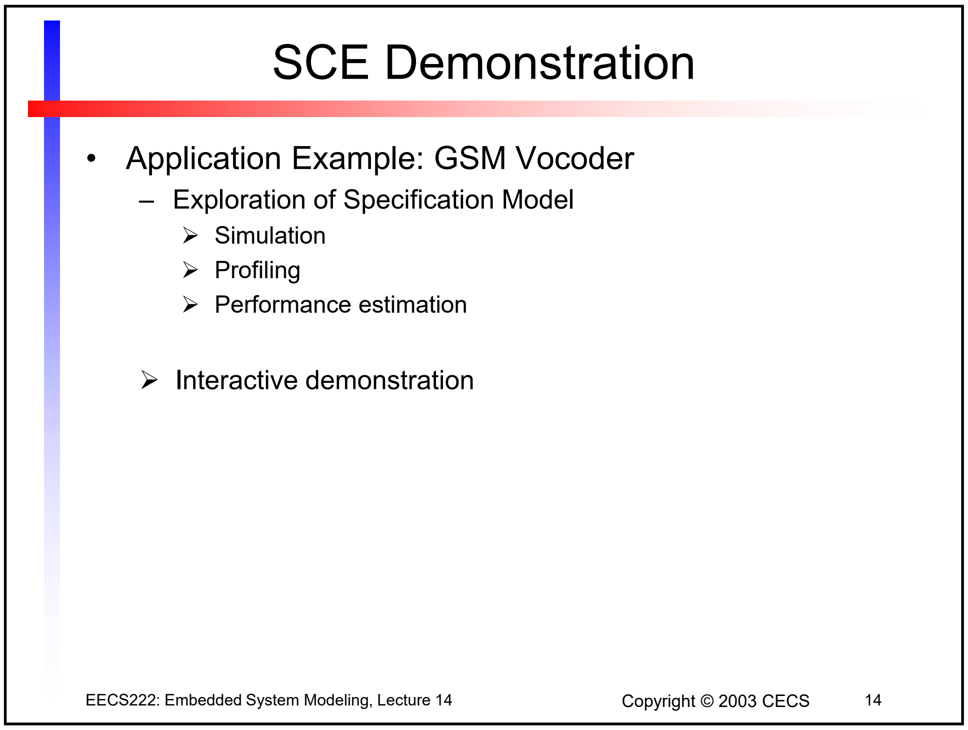
```
        Generating internal data structure for profiling
        Deriving raw statistics from SIR file
        Computing weighted statistics
        Annotating weighted statistics to SIR file
  End:  retargetable profiling
```



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)

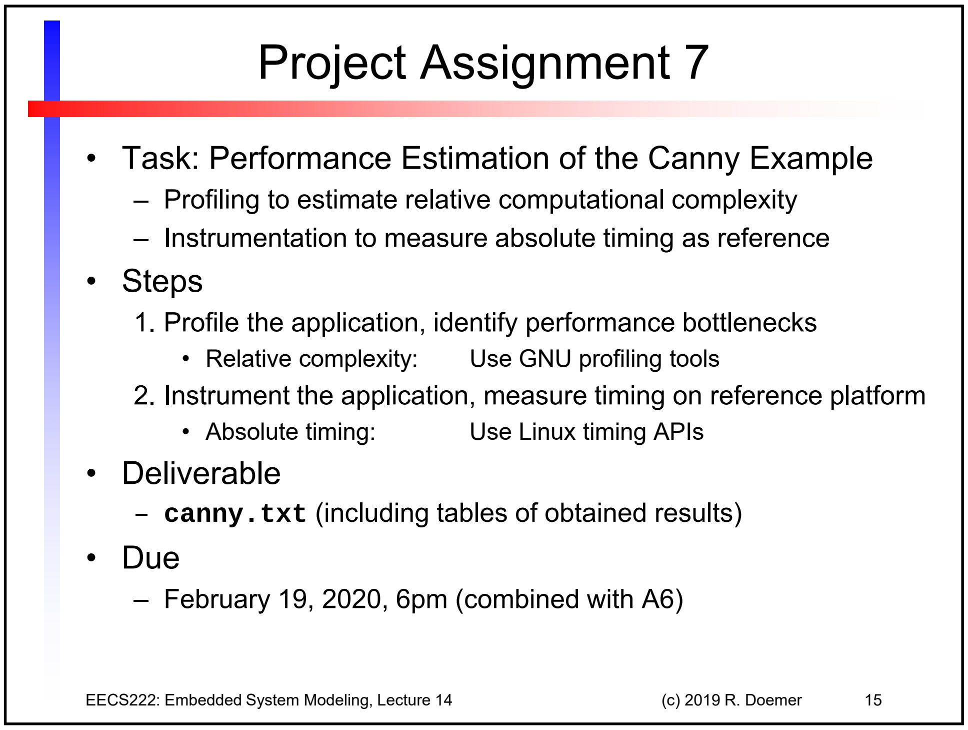
EECS222: Embedded System Modeling, Lecture 12 Copyright © 2003 CECS 13



SCE Demonstration

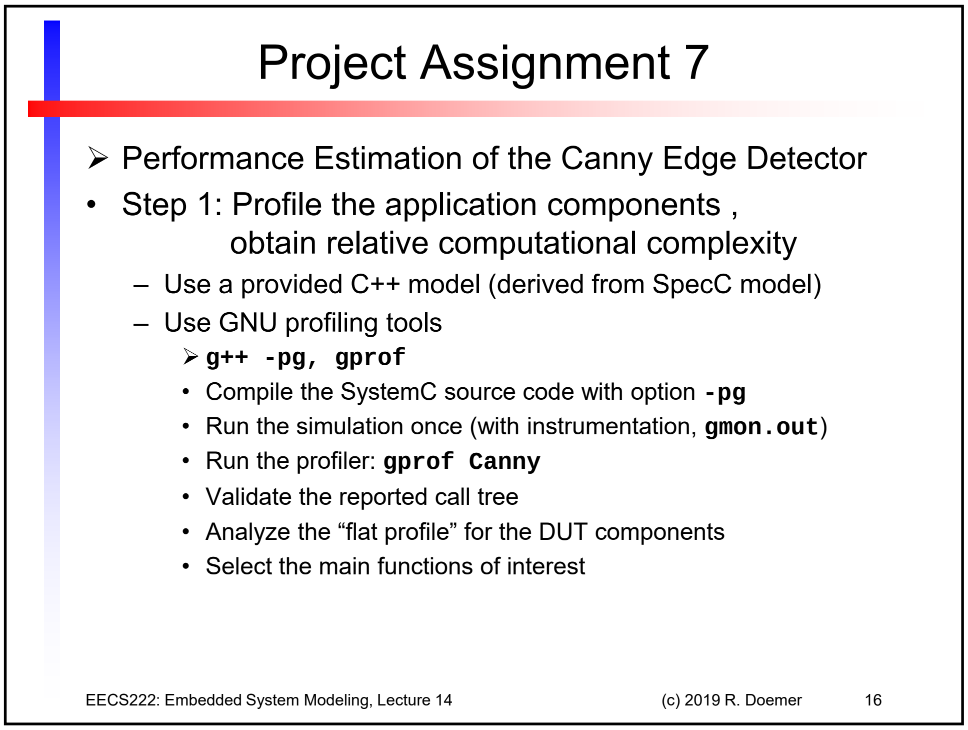
- Application Example: GSM Vocoder
  - Exploration of Specification Model
    - ➢ Simulation
    - ➢ Profiling
    - ➢ Performance estimation
  - ➢ Interactive demonstration

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



Project Assignment 7

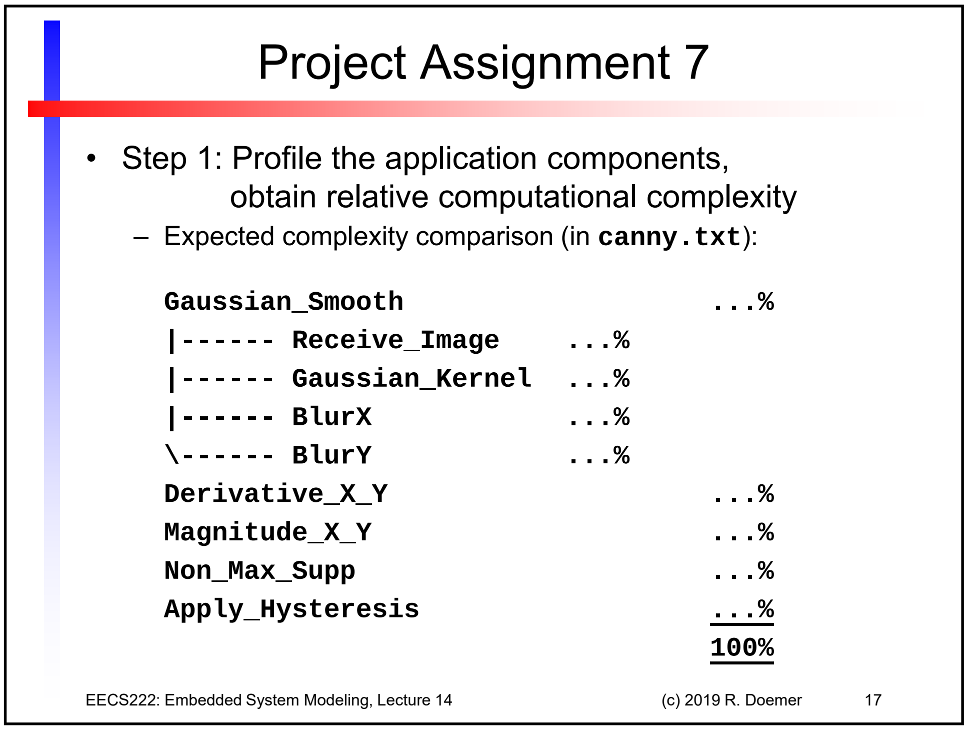
- Task: Performance Estimation of the Canny Example
  - Profiling to estimate relative computational complexity
  - Instrumentation to measure absolute timing as reference
- Steps
  1. Profile the application, identify performance bottlenecks
     - Relative complexity: Use GNU profiling tools
  2. Instrument the application, measure timing on reference platform
     - Absolute timing: Use Linux timing APIs
- Deliverable
  - **`canny.txt`** (including tables of obtained results)
- Due
  - February 19, 2020, 6pm (combined with A6)



Project Assignment 7

➢ Performance Estimation of the Canny Edge Detector

- Step 1: Profile the application components ,
  obtain relative computational complexity
  - Use a provided C++ model (derived from SpecC model)
  - Use GNU profiling tools
    - ➢ **`g++ -pg, gprof`**
    - Compile the SystemC source code with option **`-pg`**
    - Run the simulation once (with instrumentation, **`gmon.out`**)
    - Run the profiler: **`gprof Canny`**
    - Validate the reported call tree
    - Analyze the “flat profile” for the DUT components
    - Select the main functions of interest

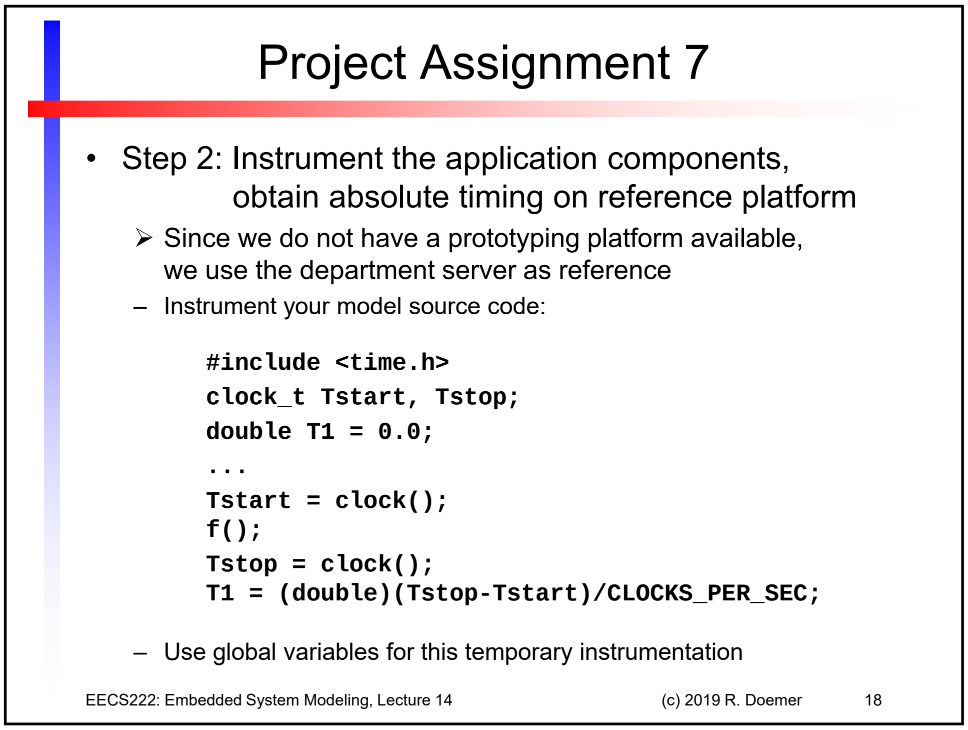


Project Assignment 7

- Step 1: Profile the application components,
  obtain relative computational complexity
  - Expected complexity comparison (in **`canny.txt`**):

```
Gaussian_Smooth                            ...%
|------ Receive_Image     ...%
|------ Gaussian_Kernel   ...%
|------ BlurX             ...%
\------ BlurY             ...%
Derivative_X_Y                             ...%
Magnitude_X_Y                              ...%
Non_Max_Supp                               ...%
Apply_Hysteresis                           ...%
                                           100%
```

EECS222: Embedded System Modeling, Lecture 14 (c) 2019 R. Doemer 17



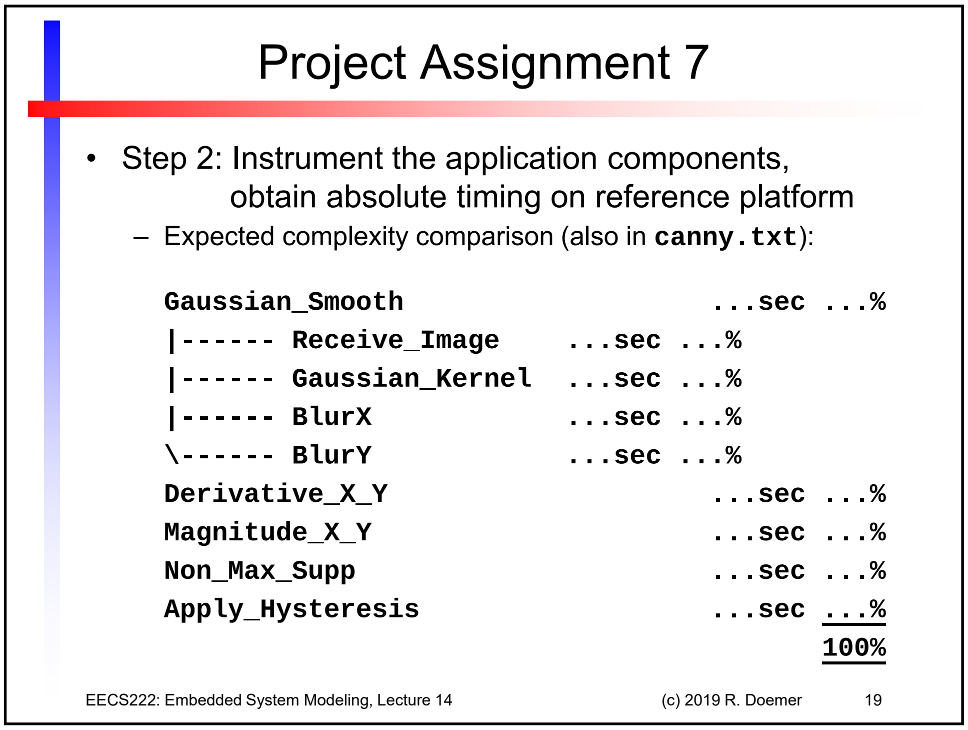
Project Assignment 7

- Step 2: Instrument the application components,
  obtain absolute timing on reference platform
  - ➢ Since we do not have a prototyping platform available, we use the department server as reference
  - Instrument your model source code:

```
#include <time.h>
clock_t Tstart, Tstop;
double T1 = 0.0;
...
Tstart = clock();
f();
Tstop = clock();
T1 = (double)(Tstop-Tstart)/CLOCKS_PER_SEC;
```

  - Use global variables for this temporary instrumentation

EECS222: Embedded System Modeling, Lecture 14 (c) 2019 R. Doemer 18



Project Assignment 7

- Step 2: Instrument the application components, obtain absolute timing on reference platform
  - Expected complexity comparison (also in **canny.txt**):

```
Gaussian_Smooth                             ...sec ...%
|------ Receive_Image    ...sec ...%
|------ Gaussian_Kernel  ...sec ...%
|------ BlurX            ...sec ...%
\------ BlurY            ...sec ...%
Derivative_X_Y                              ...sec ...%
Magnitude_X_Y                               ...sec ...%
Non_Max_Supp                                ...sec ...%
Apply_Hysteresis                            ...sec ...%
                                                   100%
```

EECS222: Embedded System Modeling, Lecture 14 (c) 2019 R. Doemer 19