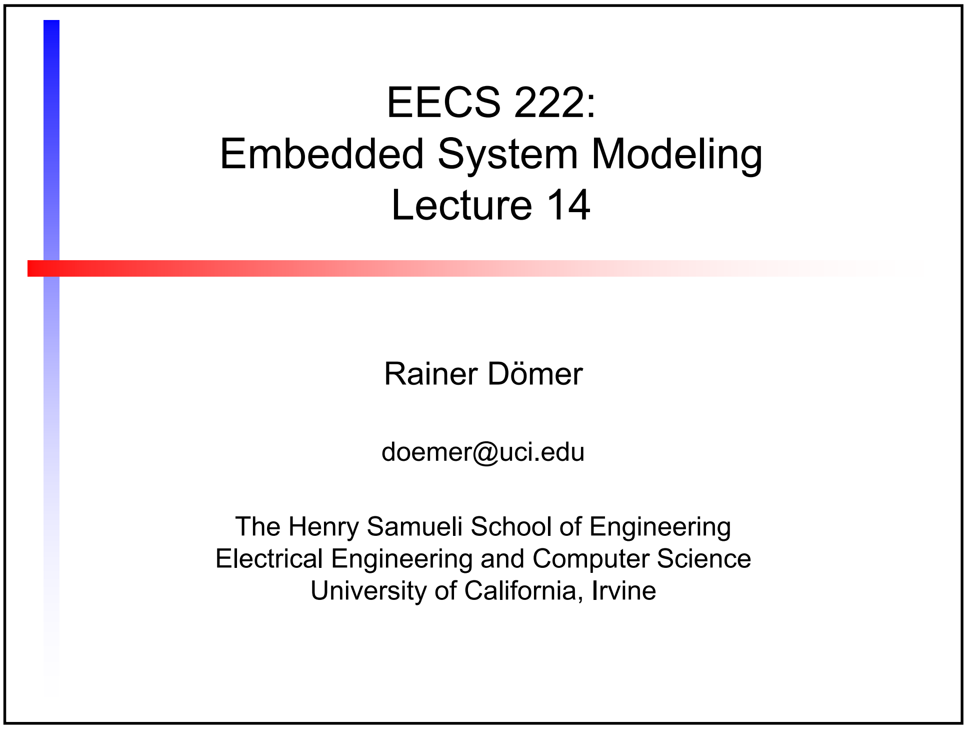
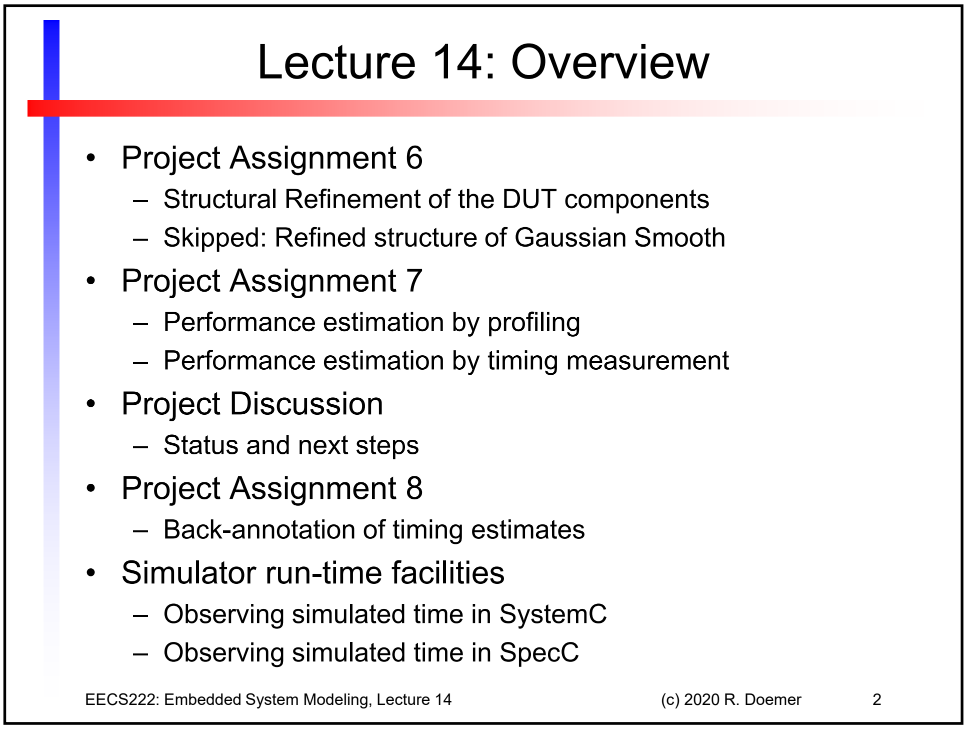


EECS 222: Embedded System Modeling Lecture 14

Rainer Dömer

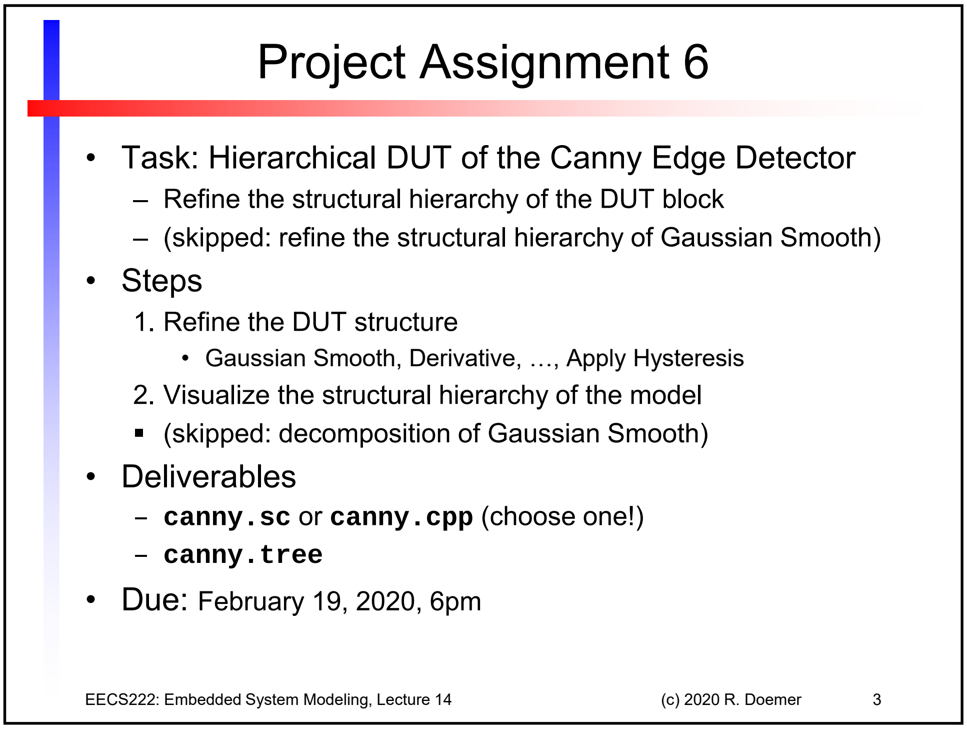
doemer@uci.edu

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



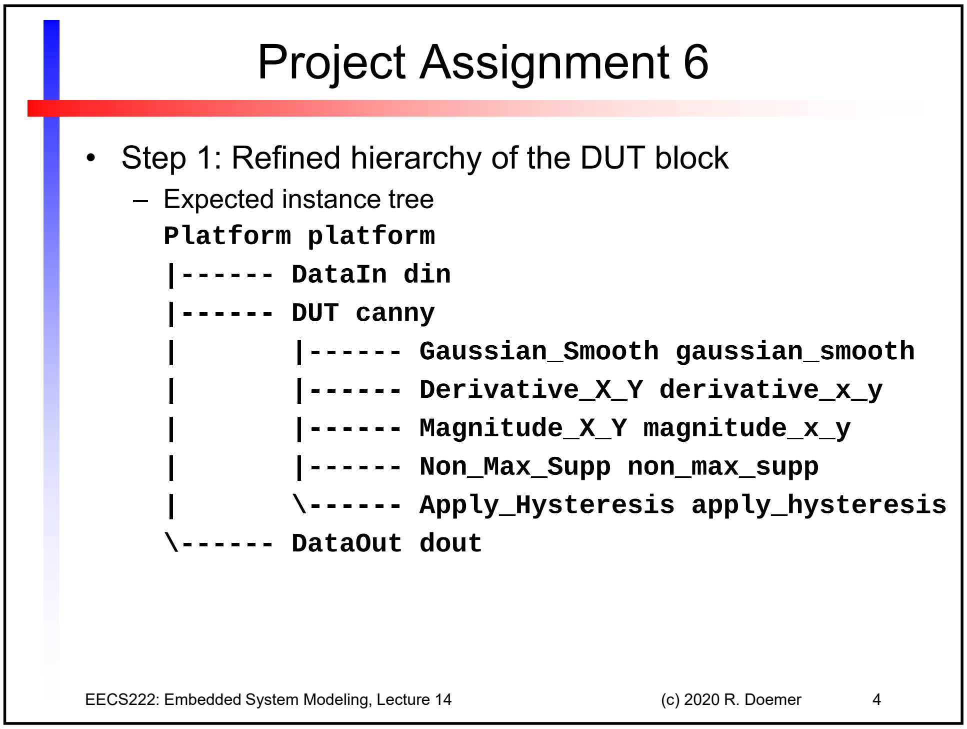
Lecture 14: Overview

- Project Assignment 6
  - Structural Refinement of the DUT components
  - Skipped: Refined structure of Gaussian Smooth
- Project Assignment 7
  - Performance estimation by profiling
  - Performance estimation by timing measurement
- Project Discussion
  - Status and next steps
- Project Assignment 8
  - Back-annotation of timing estimates
- Simulator run-time facilities
  - Observing simulated time in SystemC
  - Observing simulated time in SpecC



Project Assignment 6

- Task: Hierarchical DUT of the Canny Edge Detector
  - Refine the structural hierarchy of the DUT block
  - (skipped: refine the structural hierarchy of Gaussian Smooth)
- Steps
  1. Refine the DUT structure
     - Gaussian Smooth, Derivative, …, Apply Hysteresis
  2. Visualize the structural hierarchy of the model
  - (skipped: decomposition of Gaussian Smooth)
- Deliverables
  - **`canny.sc`** or **`canny.cpp`** (choose one!)
  - **`canny.tree`**
- Due: February 19, 2020, 6pm



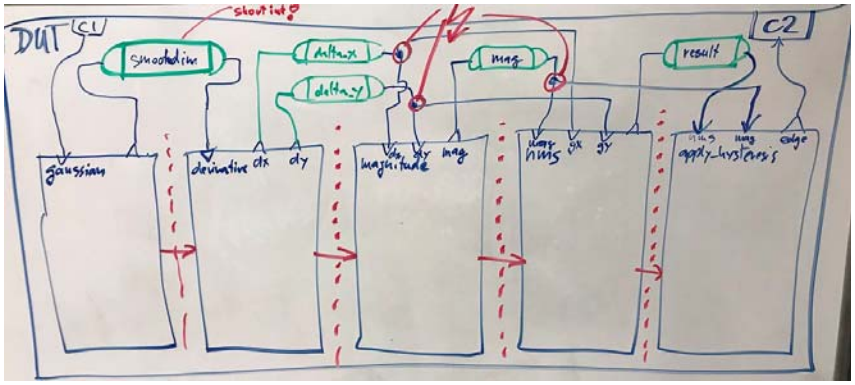
Project Assignment 6

- Step 1: Refined hierarchy of the DUT block
  - Expected instance tree

```
Platform platform
|------ DataIn din
|------ DUT canny
|       |------ Gaussian_Smooth gaussian_smooth
|       |------ Derivative_X_Y derivative_x_y
|       |------ Magnitude_X_Y magnitude_x_y
|       |------ Non_Max_Supp non_max_supp
|       \------ Apply_Hysteresis apply_hysteresis
\------ DataOut dout
```

Project Assignment 6

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



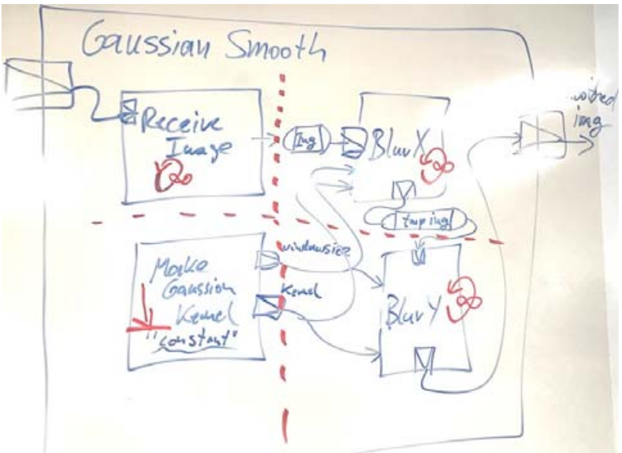
Project Assignment 6

- Skipped: Refined Hierarchy of Gaussian Smooth block
  - 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
```

Project Assignment 6

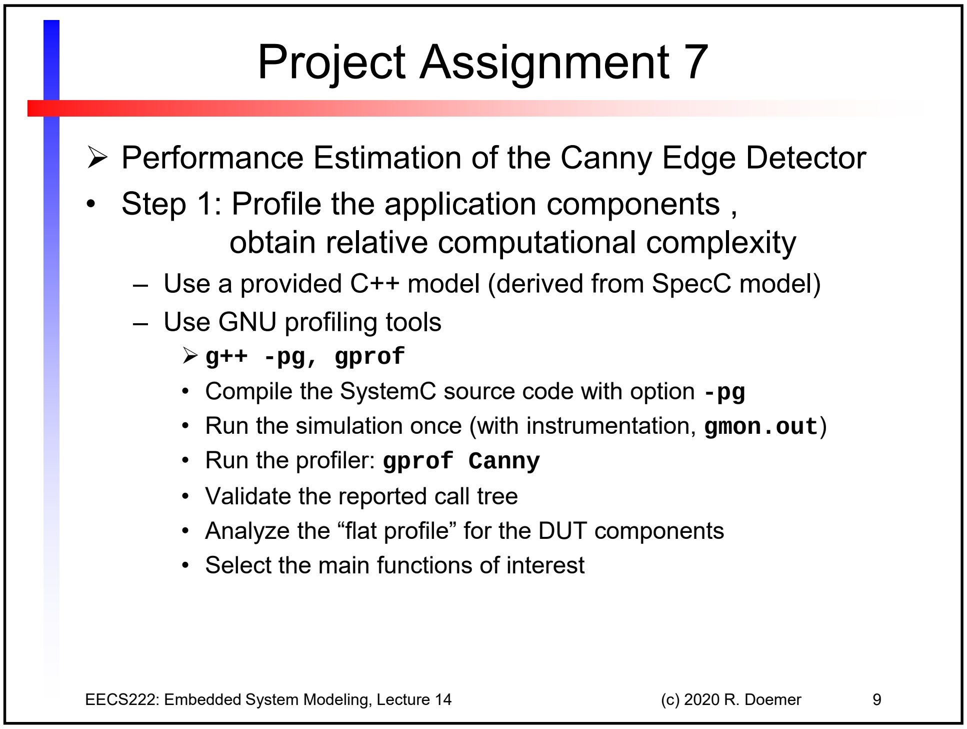
- Skipped: Refined Hierarchy of Gaussian Smooth block
  - Separate components for Kernel, BlurX, and BlurY



(whiteboard image from a prior course)

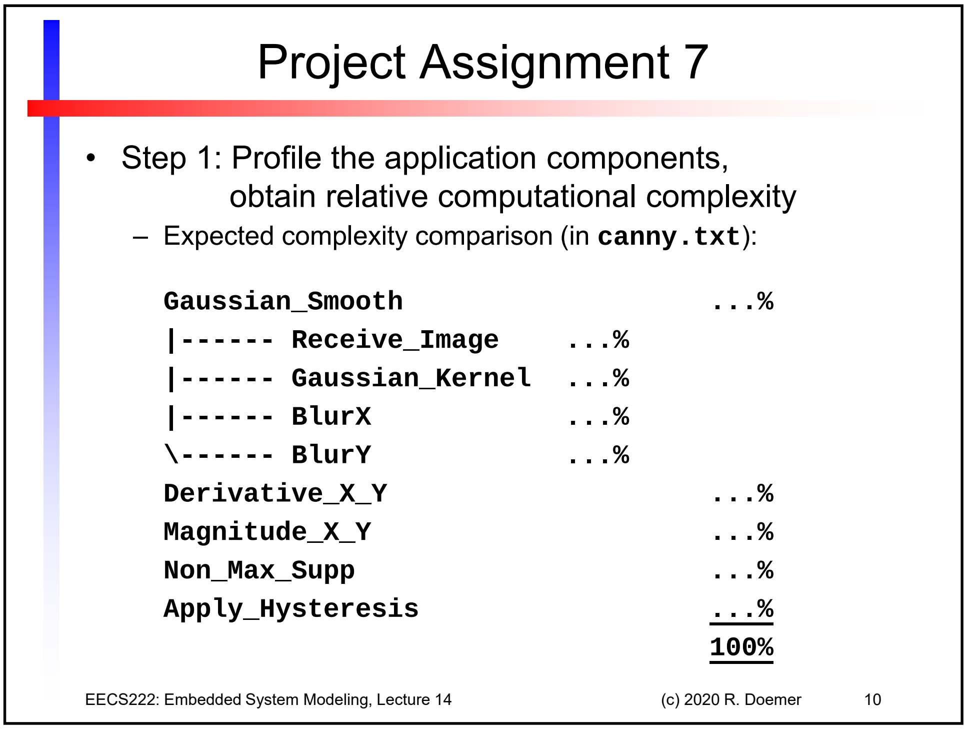
Project Assignment 7

- 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%
```

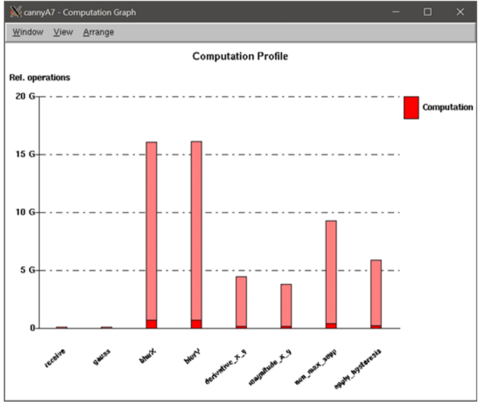
Project Assignment 7

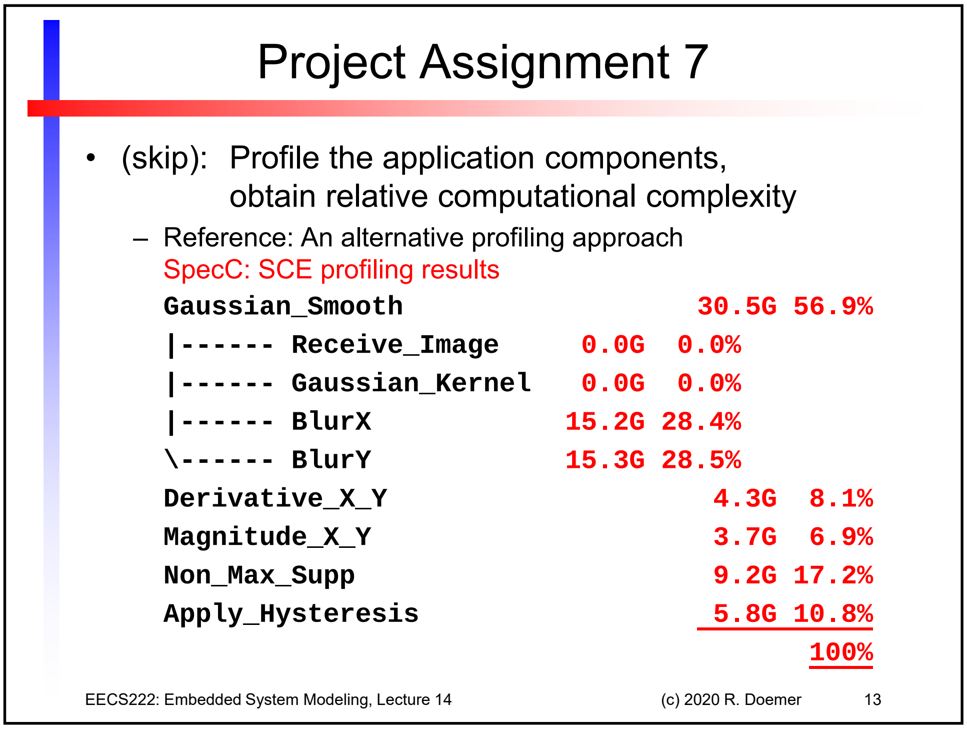
- Step 1: Profile the application components,
  obtain relative computational complexity
  - Expected complexity comparison (in **canny.txt**):

```
Gaussian_Smooth                          9.15s 61.7%
|------ Receive_Image    0.00s  0.0%
|------ Gaussian_Kernel  0.00s  0.0%
|------ BlurX            4.34s 29.2%
\------ BlurY            4.81s 32.4%
Derivative_X_Y                           0.95s  6.4%
Magnitude_X_Y                            0.66s  4.4%
Non_Max_Supp                             2.10s 14.2%
Apply_Hysteresis                         1.98s 13.3%
                                               100%
```

Project Assignment 7

- (skip): Profile the application components,
  obtain relative computational complexity
  - Reference: An alternative profiling approach
    SpecC: SCE profiling results

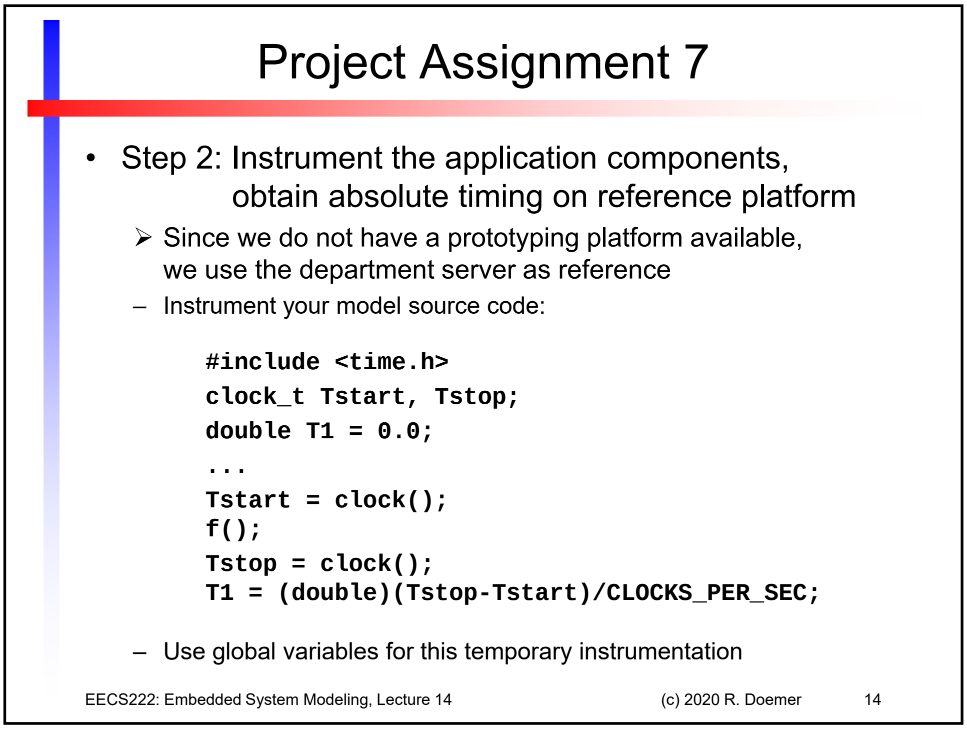




Project Assignment 7

- (skip): Profile the application components, obtain relative computational complexity
  - Reference: An alternative profiling approach
    SpecC: SCE profiling results

```
Gaussian_Smooth                              30.5G 56.9%
|------ Receive_Image       0.0G   0.0%
|------ Gaussian_Kernel     0.0G   0.0%
|------ BlurX              15.2G  28.4%
\------ BlurY              15.3G  28.5%
Derivative_X_Y                                4.3G  8.1%
Magnitude_X_Y                                 3.7G  6.9%
Non_Max_Supp                                  9.2G 17.2%
Apply_Hysteresis                              5.8G 10.8%
                                                    100%
```

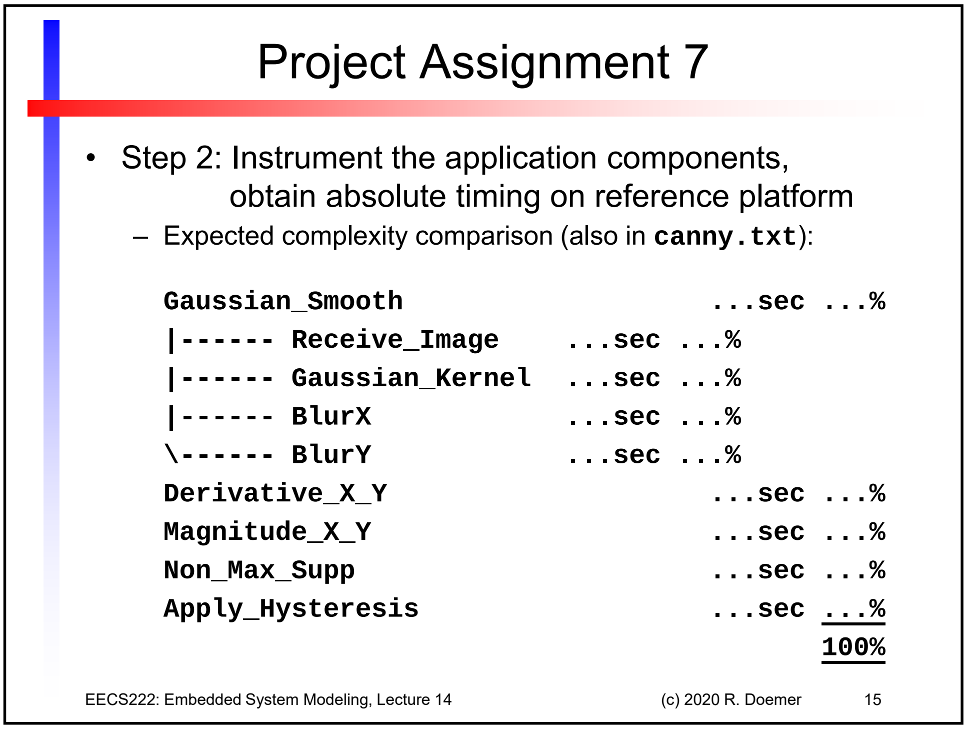


Project Assignment 7

- 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;
```

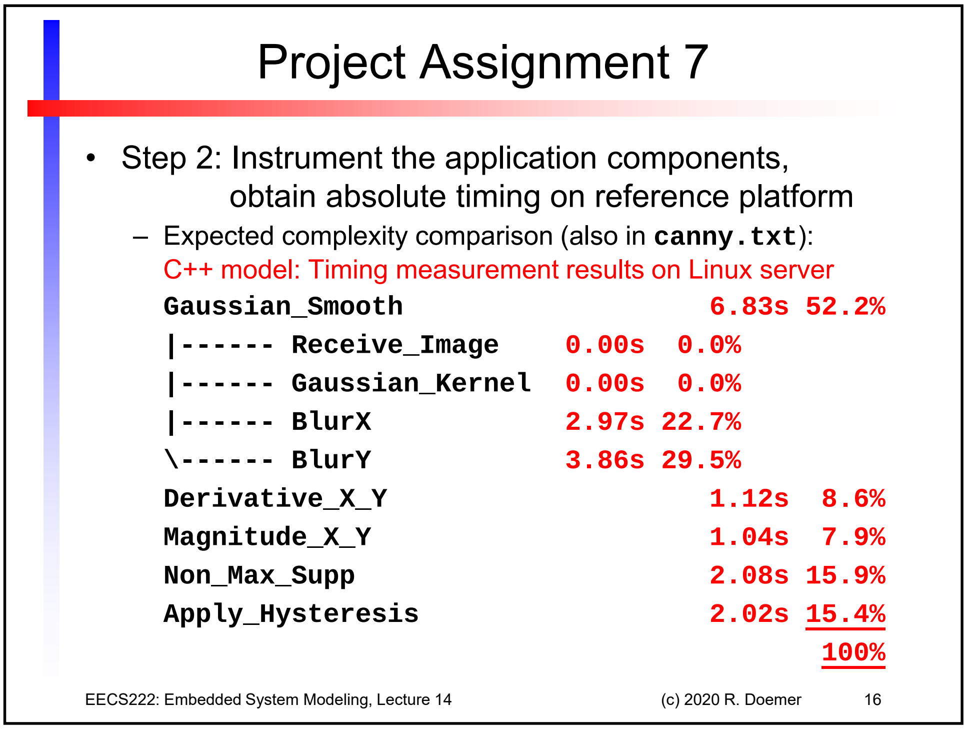
  - Use global variables for this temporary instrumentation



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%
```



Project Assignment 7

- Step 2: Instrument the application components,
  obtain absolute timing on reference platform
  - Expected complexity comparison (also in **canny.txt**):
    C++ model: Timing measurement results on Linux server

```
Gaussian_Smooth                           6.83s 52.2%
|------ Receive_Image     0.00s  0.0%
|------ Gaussian_Kernel   0.00s  0.0%
|------ BlurX             2.97s 22.7%
\------ BlurY             3.86s 29.5%
Derivative_X_Y                            1.12s  8.6%
Magnitude_X_Y                             1.04s  7.9%
Non_Max_Supp                              2.08s 15.9%
Apply_Hysteresis                          2.02s 15.4%
                                                 100%
```

Project Discussion

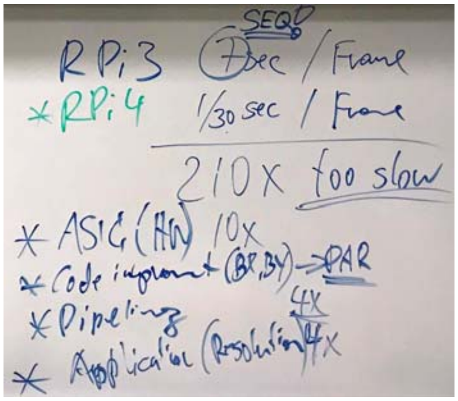
- Reference: Instrument the application components, obtain absolute timing on **prototyping** platform
  - Measured timing on Raspberry Pi 3 board: ARM-based quad-core processor (1.2GHz)

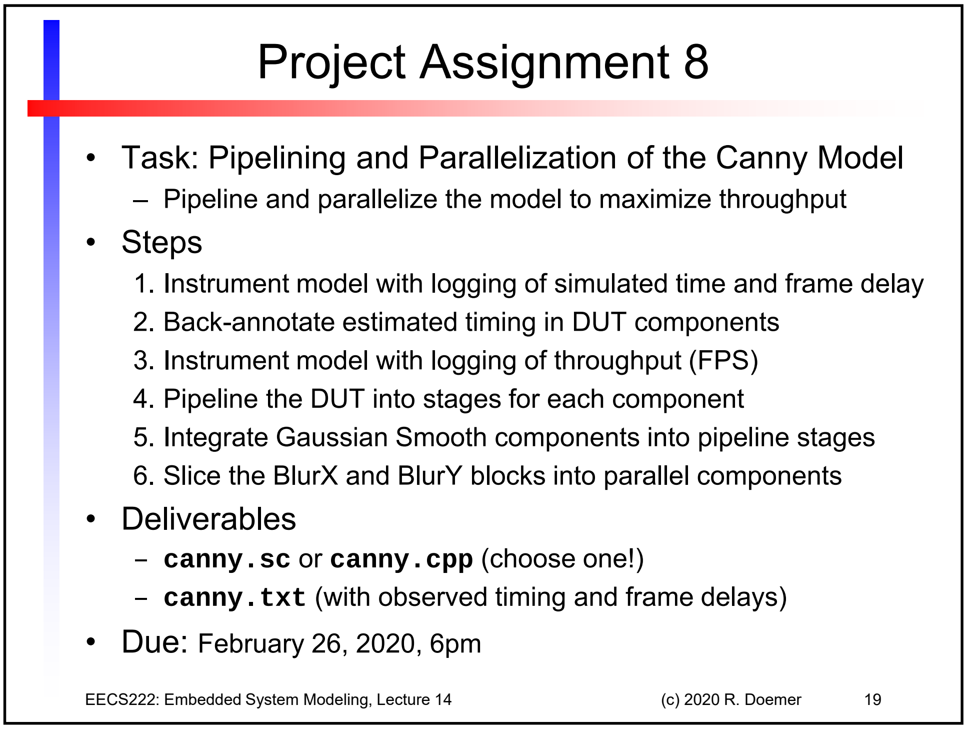
Receive_Image	**0 ms per frame**
Make_Kernel	**0 ms per frame**
BlurX	**1880 ms per frame**
BlurY	**2010 ms per frame**
Derivative_X_Y	**530 ms per frame**
Magnitude_X_Y	**910 ms per frame**
Non_Max_Supp	**960 ms per frame**
Apply_Hysteresis	**740 ms per frame**

Project Discussion

- Discussion Questions
  - Does the timing meet our real-time goals?
  - What can be done to improve the speed?

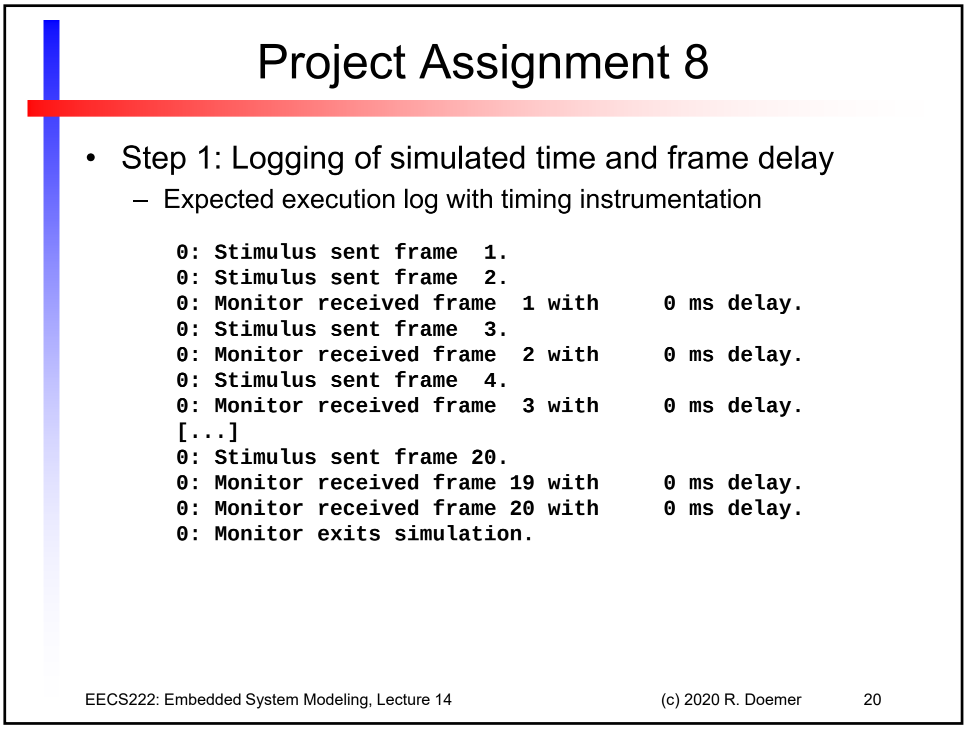
➢ Pipelining
➢ Parallelization
➢ Hardware optimizations
➢ Software optimizations
➢ Application adjustments





Project Assignment 8

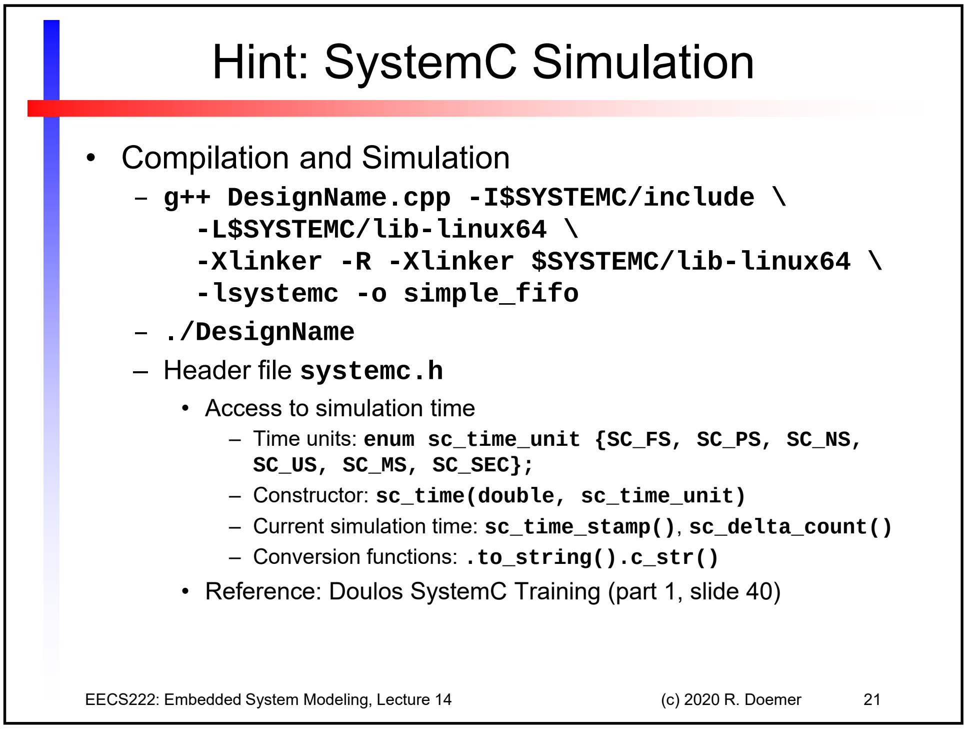
- Task: Pipelining and Parallelization of the Canny Model
  - Pipeline and parallelize the model to maximize throughput
- Steps
  1. Instrument model with logging of simulated time and frame delay
  2. Back-annotate estimated timing in DUT components
  3. Instrument model with logging of throughput (FPS)
  4. Pipeline the DUT into stages for each component
  5. Integrate Gaussian Smooth components into pipeline stages
  6. Slice the BlurX and BlurY blocks into parallel components
- Deliverables
  - **`canny.sc`** or **`canny.cpp`** (choose one!)
  - **`canny.txt`** (with observed timing and frame delays)
- Due: February 26, 2020, 6pm



Project Assignment 8

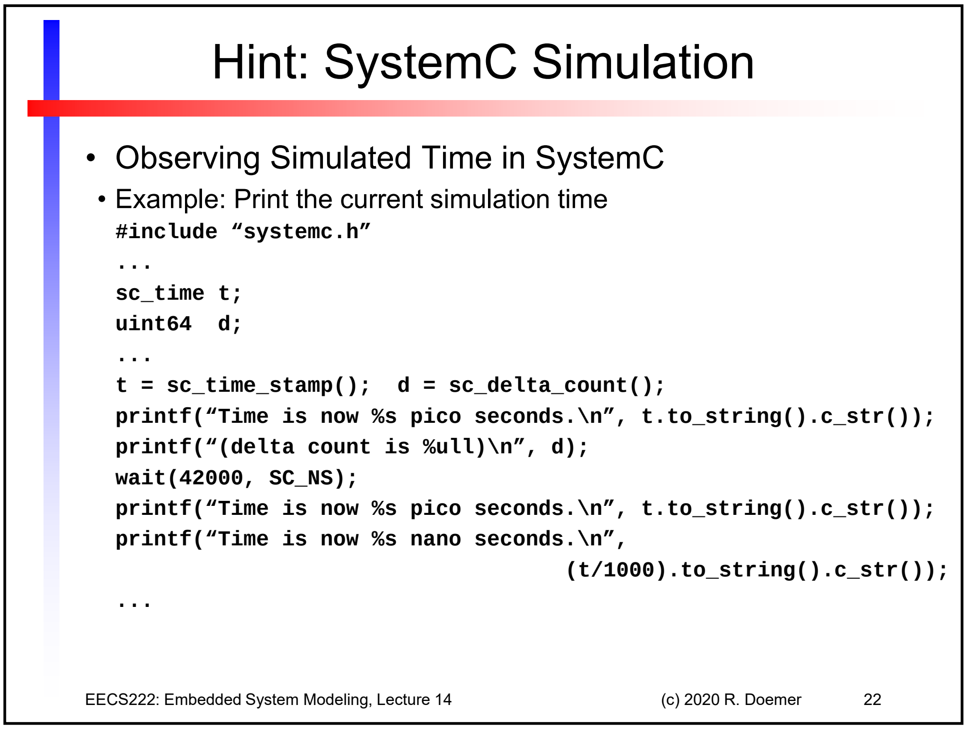
- Step 1: Logging of simulated time and frame delay
  - Expected execution log with timing instrumentation

```
0: Stimulus sent frame  1.
0: Stimulus sent frame  2.
0: Monitor received frame  1 with     0 ms delay.
0: Stimulus sent frame  3.
0: Monitor received frame  2 with     0 ms delay.
0: Stimulus sent frame  4.
0: Monitor received frame  3 with     0 ms delay.
[...]
0: Stimulus sent frame 20.
0: Monitor received frame 19 with     0 ms delay.
0: Monitor received frame 20 with     0 ms delay.
0: Monitor exits simulation.
```



Hint: SystemC Simulation

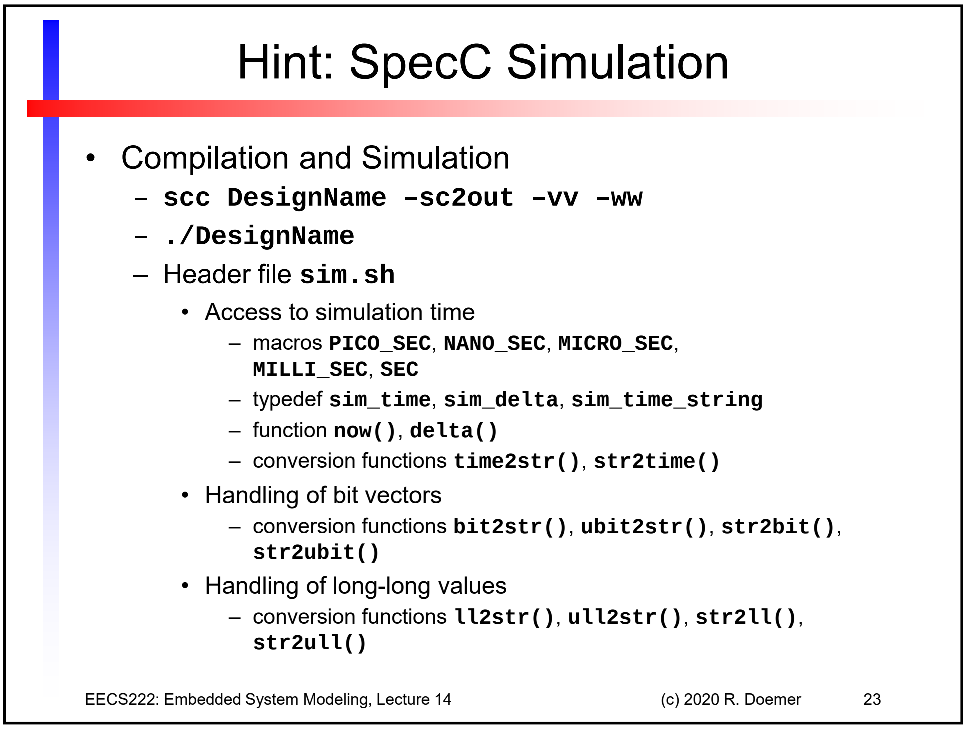
- Compilation and Simulation
  - **`g++ DesignName.cpp -I$SYSTEMC/include \`**
    `-L$SYSTEMC/lib-linux64 \`
    `-Xlinker -R -Xlinker $SYSTEMC/lib-linux64 \`
    `-lsystemc -o simple_fifo`
  - **`./DesignName`**
  - Header file **`systemc.h`**
    - Access to simulation time
      - Time units: **`enum sc_time_unit {SC_FS, SC_PS, SC_NS, SC_US, SC_MS, SC_SEC};`**
      - Constructor: **`sc_time(double, sc_time_unit)`**
      - Current simulation time: **`sc_time_stamp()`**, **`sc_delta_count()`**
      - Conversion functions: **`.to_string().c_str()`**
    - Reference: Doulos SystemC Training (part 1, slide 40)



Hint: SystemC Simulation

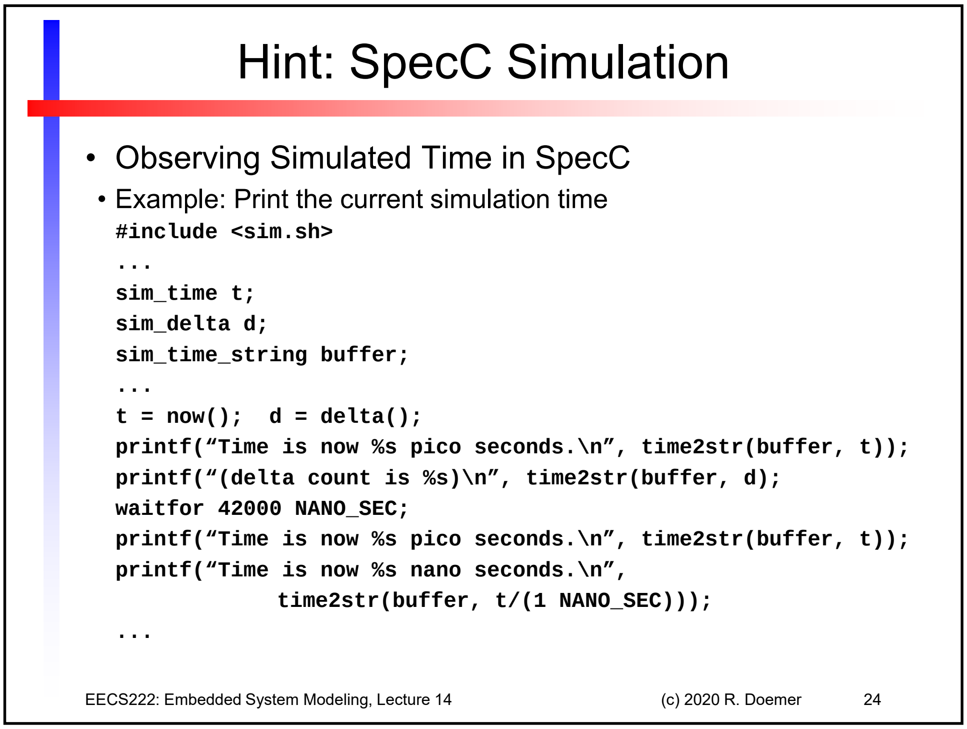
- Observing Simulated Time in SystemC
- Example: Print the current simulation time

```
#include “systemc.h”
...
sc_time t;
uint64  d;
...
t = sc_time_stamp();  d = sc_delta_count();
printf(“Time is now %s pico seconds.\n”, t.to_string().c_str());
printf(“(delta count is %ull)\n”, d);
wait(42000, SC_NS);
printf(“Time is now %s pico seconds.\n”, t.to_string().c_str());
printf(“Time is now %s nano seconds.\n”,
                                   (t/1000).to_string().c_str());
...
```



Hint: SpecC Simulation

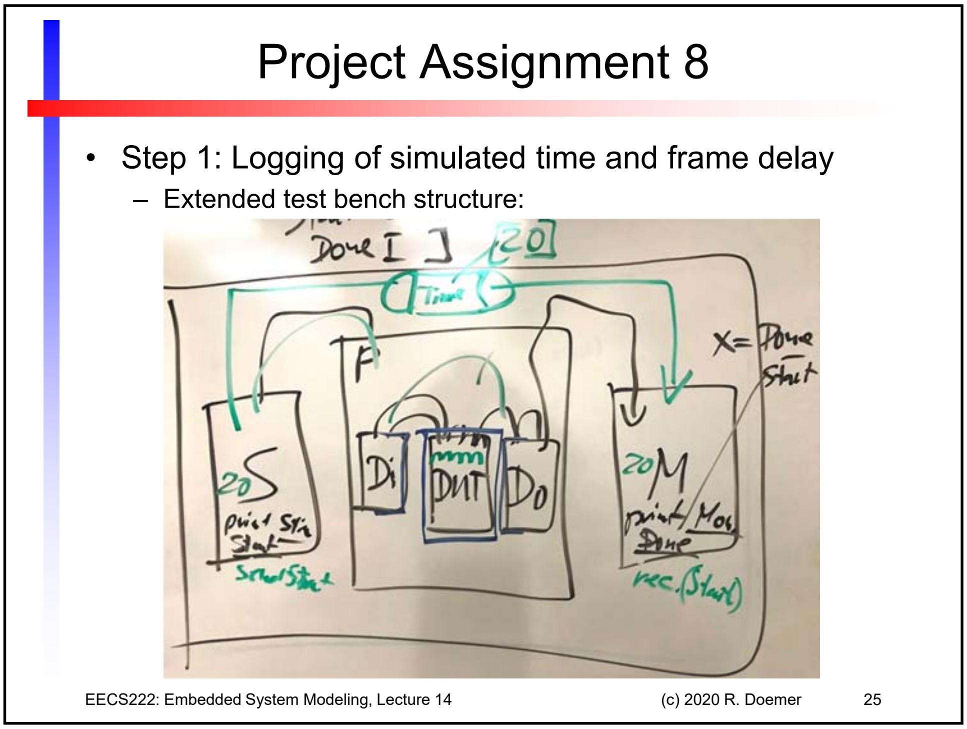
- Compilation and Simulation
  - **`scc DesignName -sc2out -vv -ww`**
  - **`./DesignName`**
  - Header file **`sim.sh`**
    - Access to simulation time
      - macros **`PICO_SEC`**, **`NANO_SEC`**, **`MICRO_SEC`**, **`MILLI_SEC`**, **`SEC`**
      - typedef **`sim_time`**, **`sim_delta`**, **`sim_time_string`**
      - function **`now()`**, **`delta()`**
      - conversion functions **`time2str()`**, **`str2time()`**
    - Handling of bit vectors
      - conversion functions **`bit2str()`**, **`ubit2str()`**, **`str2bit()`**, **`str2ubit()`**
    - Handling of long-long values
      - conversion functions **`ll2str()`**, **`ull2str()`**, **`str2ll()`**, **`str2ull()`**



Hint: SpecC Simulation

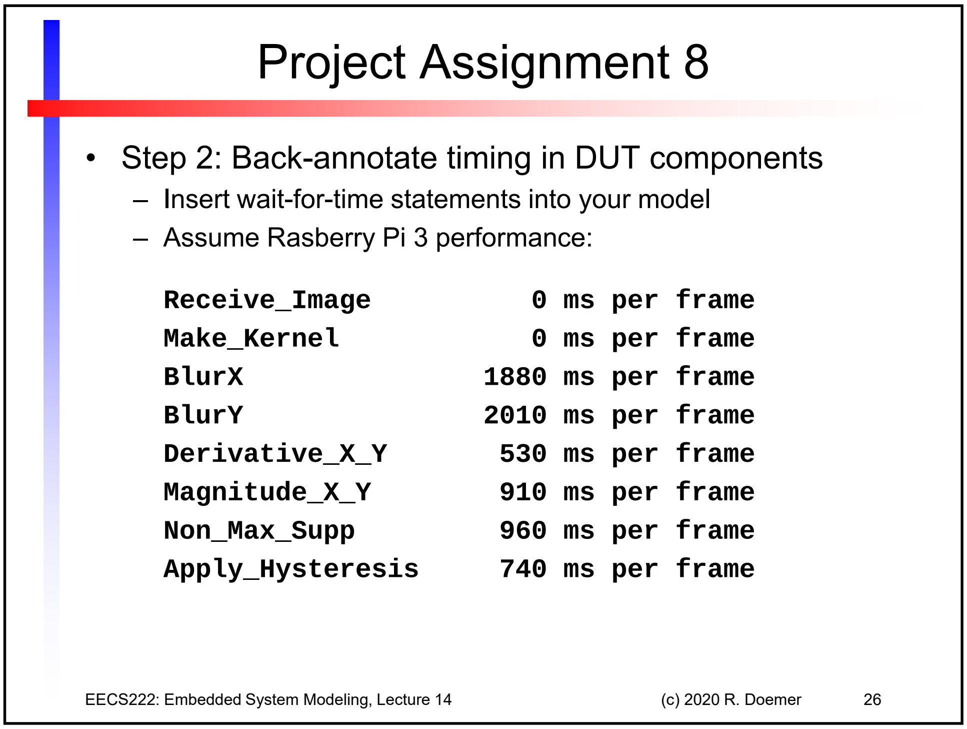
- Observing Simulated Time in SpecC
- Example: Print the current simulation time

```
#include <sim.sh>
...
sim_time t;
sim_delta d;
sim_time_string buffer;
...
t = now();  d = delta();
printf(“Time is now %s pico seconds.\n”, time2str(buffer, t));
printf(“(delta count is %s)\n”, time2str(buffer, d);
waitfor 42000 NANO_SEC;
printf(“Time is now %s pico seconds.\n”, time2str(buffer, t));
printf(“Time is now %s nano seconds.\n”,
             time2str(buffer, t/(1 NANO_SEC)));
...
```



Project Assignment 8

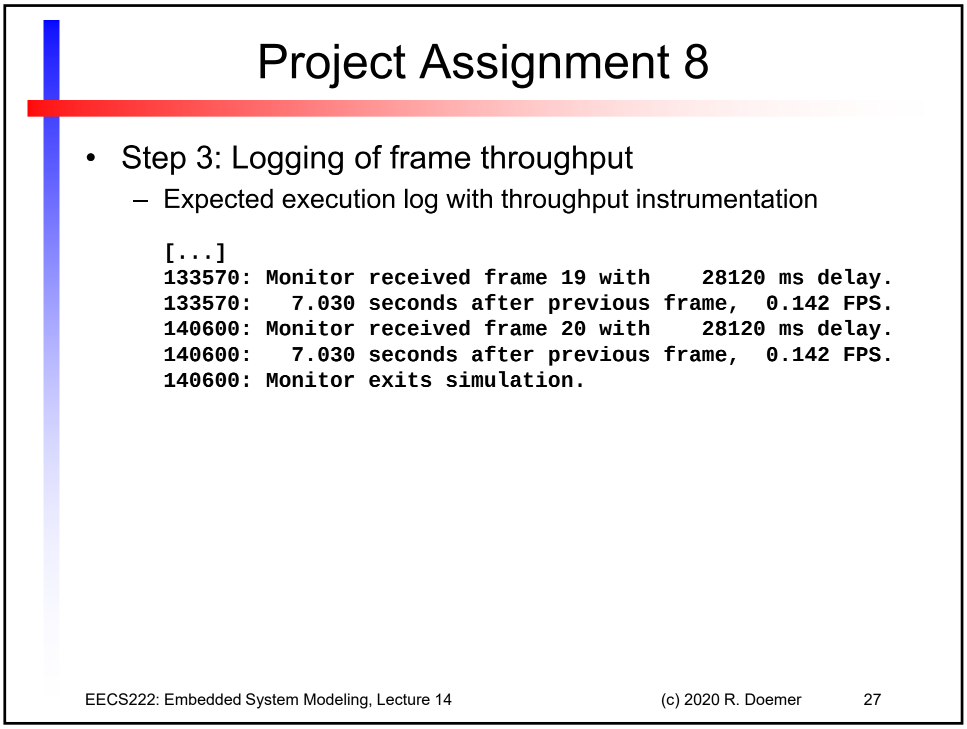
- Step 1: Logging of simulated time and frame delay
  - Extended test bench structure:



Project Assignment 8

- Step 2: Back-annotate timing in DUT components
  - Insert wait-for-time statements into your model
  - Assume Rasberry Pi 3 performance:

Receive_Image	0 ms per frame
Make_Kernel	0 ms per frame
BlurX	1880 ms per frame
BlurY	2010 ms per frame
Derivative_X_Y	530 ms per frame
Magnitude_X_Y	910 ms per frame
Non_Max_Supp	960 ms per frame
Apply_Hysteresis	740 ms per frame



Project Assignment 8

- Step 3: Logging of frame throughput
  - Expected execution log with throughput instrumentation

```
[...]
133570: Monitor received frame 19 with    28120 ms delay.
133570:   7.030 seconds after previous frame,  0.142 FPS.
140600: Monitor received frame 20 with    28120 ms delay.
140600:   7.030 seconds after previous frame,  0.142 FPS.
140600: Monitor exits simulation.
```