ECPS 203
Embedded Systems Modeling and Design
Lecture 15

Rainer Dömer

doemer@uci.edu

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

Lecture 15: Overview

- Project Discussion
  - Status and next steps
  - A5: Test bench model of the Canny Edge Detector
  - A6: Structural refinement of the DUT module
  - A6: Profiling of the Canny Edge Detector functions
  - A7: Performance measurement on prototyping board
- Assignment 8
  - Back-annotation of timing estimates into SystemC model
    - ➢ Observing computation delay during simulation
  - Pipelining and parallelization of the DUT module
    - ➢ Model refinement on the whiteboard
    - ➢ Discussion

ECPS 203 Project

- Application Example: Canny Edge Detector
  - Embedded system model for image processing: Automatic edge detection in a video camera of a drone

Engineering012.png

Engineering012_edges.pgm

  - Video taken by a drone flying over UCI Engineering Plaza
    - Available on the server: **`~ecps203/public/DroneFootage/`**
    - High resolution, 2704 by 1520 pixes
    - Representative sample, using 30 extracted frames for test bench model

ECPS203: Embedded Systems Modeling and Design, Lecture 15 (c) 2019 R. Doemer 3

Project Assignment 5

- Task: Test bench for the Canny Edge Detector
  - Convert C++ model to SystemC model
  - Add a test bench structure around the C++ model
  - Wrap DUT into a platform model with explicit I/O units
- Steps
  1. Create test bench structure: Stimulus, Platform, Monitor
  2. Create platform model: DataIn, DUT, DataOut
  3. Localize functions and use **`sc_fifo`** channels for communication
     - ➢ Pay attention to stack sizes for every thread
- Deliverables
  - SystemC source code and text file: **`Canny.cpp`**, **`Canny.txt`**
- Due
  - Wednesday, November 6, 2019, 6pm

ECPS203: Embedded Systems Modeling and Design, Lecture 15 (c) 2019 R. Doemer 4

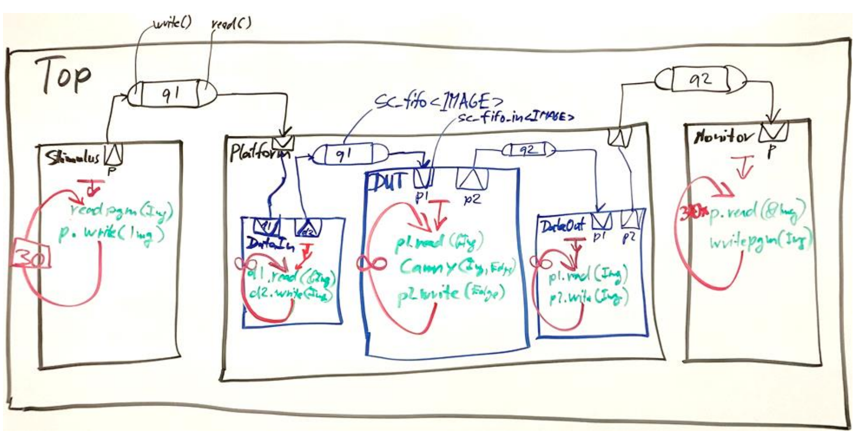
Project Assignment 5

- Task: Test bench for the Canny Edge Detector
  - Expected instance tree

```
Top top
|------ Monitor monitor
|------ Platform platform
|       |------ DUT canny
|       |------ DataIn din
|       |------ DataOut dout
|       |------ sc_fifo<IMAGE> q1
|       \------ sc_fifo<IMAGE> q2
|------ Stimulus stimulus
|------ sc_fifo<IMAGE> q1
\------ sc_fifo<IMAGE> q2
```


Project Assignment 5

- Task: Test bench for the Canny Edge Detector
  - Discussion on whiteboard: Top-level and Platform structure



Project Assignment 6

- Task: Structural refinement of the DUT module
  - Refine the structural hierarchy of the DUT module
  - Refine the structural hierarchy of the Gaussian Smooth module
  - Profile the relative complexity of the Canny functions
- Steps
  1. Create structure in DUT: Gaussian Smooth, …, Apply Hysteresis
  2. Create structure in Gaussian Smooth: Input, Gauss, BlurX, BlurY
  3. Profile the algorithm, obtain relative computational complexity
- Deliverables
  - **`Canny.cpp`** (refined structural model)
  - **`Canny.txt`** (profile of relative complexity of the DUT modules)
- Due
  - Wednesday, November 13, 2019, 6pm

Project Assignment 6

- Step 1: Refined structure of the DUT module
  - Expected module 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

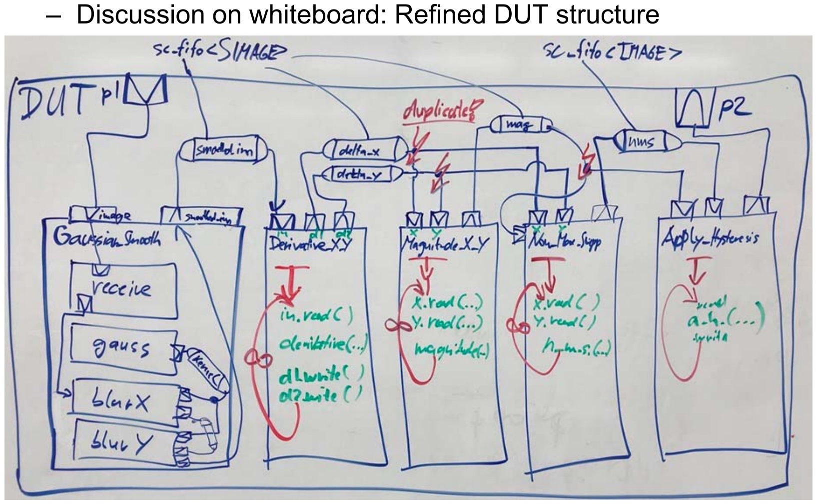
- Step 2: Refined structure of the Gaussian Smooth block
  - Expected module instance tree

```
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
```

ECPS203: Embedded Systems Modeling and Design, Lecture 15 (c) 2019 R. Doemer 9

Project Assignment 6

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



ECPS203: Embedded Systems Modeling and Design, Lecture 15 (c) 2019 R. Doemer 10

Project Assignment 6

- Step 3: Profile the Canny functions
  - ➢ Performance profiling of the Canny Edge Detector
  - ➢ Determine the relative complexity of the Canny functions
    - Is there any performance bottleneck?
    - If so, where?
  - – Use the GNU C/C++ profiling tools
    - ➢ **`g++ -pg`**
    - ➢ **`gprof`**
    1. Compile the SystemC source code with option **`-pg`**
    2. Run the simulation once with instrumentation, obtain **`gmon.out`**
    3. Run the profiler: **`gprof Canny`**
    4. Validate the reported call tree
    5. Analyze the "flat profile" for the DUT components (**`self`**)

Project Assignment 6

- Step 3: Profile the Canny functions, obtain relative computational complexity
  - – Expected complexity comparison (in **`Canny.txt`**):

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

Project Assignment 6

- Step 3: Profile the Canny functions,
  obtain relative computational complexity
  - Profiled complexity comparison (in **`Canny.txt`**):

```
Gaussian_Smooth                                40.57%
|------ Gaussian_Kernel      0.00%
|------ BlurX               17.23%
\------ BlurY               23.34%
Derivative_X_Y                                  6.26%
Magnitude_X_Y                                  15.90%
Non_Max_Supp                                   23.98%
Apply_Hysteresis                               12.29%
                                             100%
```

➢ Profiling results vary, but Gaussian Smooth is a bottleneck!

ECPS203: Embedded Systems Modeling and Design, Lecture 15 (c) 2019 R. Doemer 13

Project Assignment 7

- Task: Performance measurement on prototyping board
  - Run C++ model of Canny Edge Detector on Raspberry Pi
  - Obtain absolute timing measurements of Canny functions
- Steps
  1. Prepare the prototyping board with Raspbian operating system
  2. Upload **`Canny.cpp`** from Assignment 4 and compile it
  3. Instrument the source code with real-time measurements
  4. Note the computation delays of the major Canny functions
- Deliverables
  - **`Canny.cpp`** (model instrumented with timing measurements)
  - **`Canny.txt`** (table of measured delays)
- Due
  - Wednesday, November 20, 2019, 6pm

ECPS203: Embedded Systems Modeling and Design, Lecture 15 (c) 2019 R. Doemer 14

Project Assignment 7

- Task: Performance measurement on prototyping board
  - Expected timing measurements (in **Canny.txt**):

```
Gaussian_Smooth                          ... sec
|------ Gaussian_Kernel  ... sec
|------ BlurX            ... sec
\------ BlurY            ... sec
Derivative_X_Y                           ... sec
Magnitude_X_Y                            ... sec
Non_Max_Supp                             ... sec
Apply_Hysteresis                         ... sec
TOTAL                                    ... sec
```

ECPS203: Embedded Systems Modeling and Design, Lecture 15 (c) 2019 R. Doemer 15

Project Assignment 8

- Task: Pipelining and parallelization of the DUT module
  - Back-annotate estimated delays to observe timing in the model
  - Pipeline and parallelize the model to improve throughput
- Steps
  1. Instrument model with simulated time to observe frame delay
  2. Back-annotate estimated timing into DUT components
  3. Improve test bench to observe frame throughput
  4. Pipeline the DUT into a sequence of 7 stages with buffer size 1
  5. Slice the BlurX and BlurY modules into 4 parallel threads
- Deliverables
  - **Canny.cpp:** pipelined and parallelized SystemC model
  - **Canny.txt:** table of observed frame delays and throughput
- Due: Wednesday, November 27, 2019, 6pm

ECPS203: Embedded Systems Modeling and Design, Lecture 15 (c) 2019 R. Doemer 16

Project Assignment 8

- Task: Pipelining and parallelization of the DUT module
  - Expected simulated performance values (in **`Canny.txt`**):

Model	Frame Delay	Throughput	Total
CannyA8_step1	... ms		... ms
CannyA8_step2	... ms		... ms
CannyA8_step3	... ms	... FPS	... ms
CannyA8_step4	... ms	... FPS	... ms
CannyA8_step5	... ms	... FPS	... ms

Project Assignment 8

- Review: Observing simulated time in SystemC
  - Header file **`systemc.h`**
    - Reference: Doulos SystemC Training (part 1, slide 40)
    - 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()`

Project Assignment 8

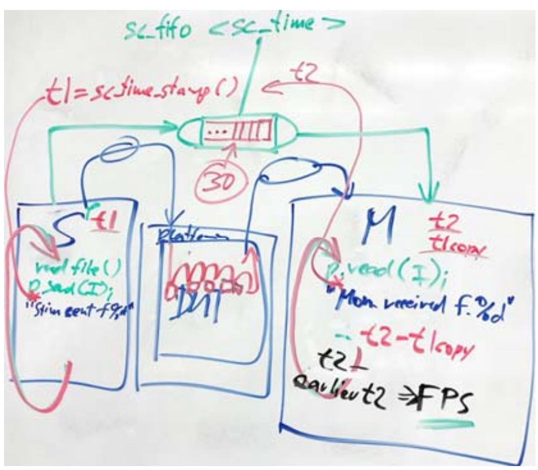
- Review: Observing simulated time in SystemC
  - Example: Print the current simulation time

```
#include "systemc.h"
...
sc_time t;
uint64  d;
...
t = sc_time_stamp();
printf("Time is now %s pico seconds.\n", t.to_string().c_str());
d = sc_delta_count();
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());
...
```

ECPS203: Embedded Systems Modeling and Design, Lecture 15 (c) 2019 R. Doemer 19

Project Assignment 8

- Timed test bench model for the Canny Edge Detector
  - Discussion on whiteboard: Chart of refined test bench structure



ECPS203: Embedded Systems Modeling and Design, Lecture 15 (c) 2019 R. Doemer 20

Project Assignment 8

- Pipelined and parallel model of the Canny Edge Detector
  - Discussion on whiteboard: Chart of pipelined DUT structure