ECPS 203
Embedded Systems Modeling and Design
Lecture 16

Rainer Dömer

doemer@uci.edu

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

Lecture 16: 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
  - 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 digital video camera

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

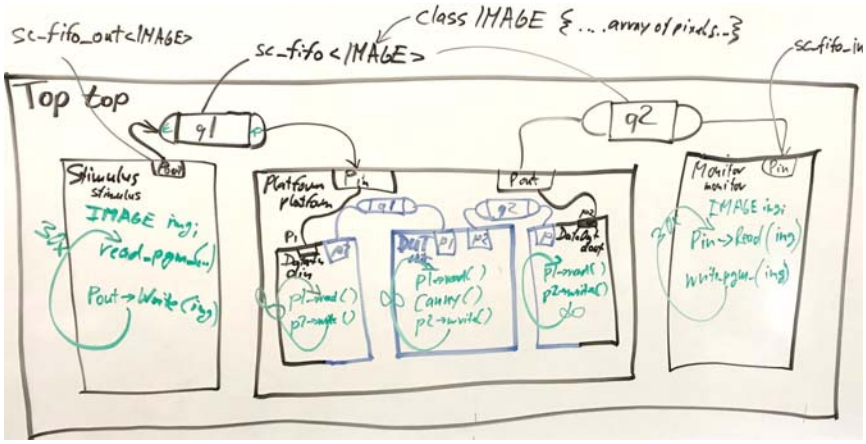
Project Assignment 5

- 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: Structural Model of the Canny Edge Detector
  - Discussion on whiteboard: Chart of top-level structure



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
```

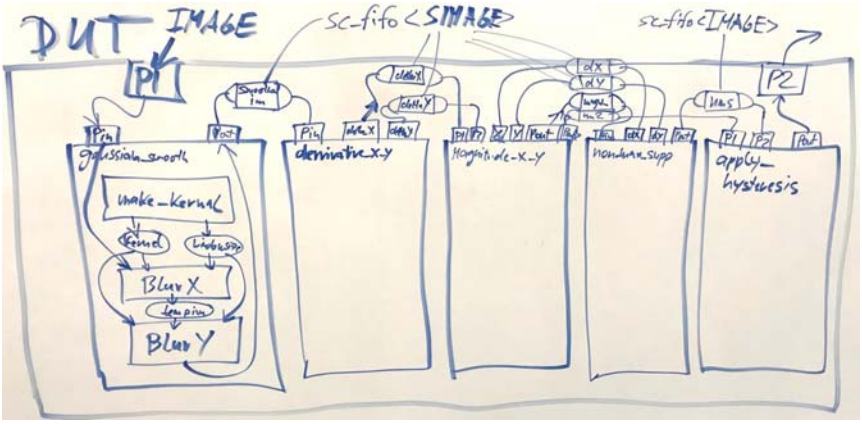
Project Assignment 6

- 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
```


Project Assignment 6

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



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
  - – Profiled complexity comparison (in **`Canny.txt`**):

```
Gaussian_Smooth                  42.64%
|------ Gaussian_Kernel   0%
|------ BlurX            22.73%
\------ BlurY            19.91%
Derivative_X_Y            6.12%
Magnitude_X_Y            16.09%
Non_Max_Supp             25.16%
Apply_Hysteresis          9.80%
                         100%
```

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

Project Assignment 7

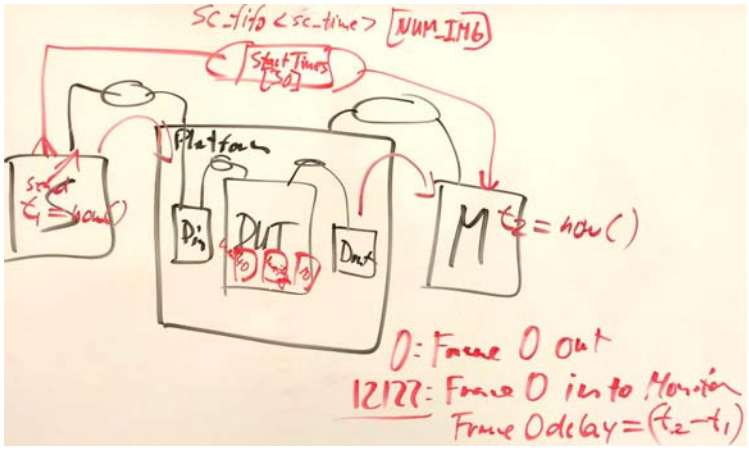
- Task: Performance measurement on prototyping board
  - Run C++ model of Canny Edge Detector on Raspberry Pi
  - Obtain absolute timing measurements of Canny functions
- Steps
  1. Prepare the prototyping board with Raspbian operating system
  2. Upload **`Canny.cpp`** from A4 and compile it
  3. Instrument 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 22, 2017, 6pm

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 simulation time to observe frame delay
  2. Back-annotate estimated timing in DUT components
  3. Pipeline the DUT into a sequence of 7 stages with buffer size 1
  4. Slice the BlurX and BlurY modules into parallel threads
- Deliverables
  - **`Canny.cpp`** (pipelined and parallelized SystemC model)
  - **`Canny.txt`** (table of observed frame delays)
- Due
  - Wednesday, November 29, 2017, 6pm

Project Assignment 8

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



Project Assignment 8

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