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

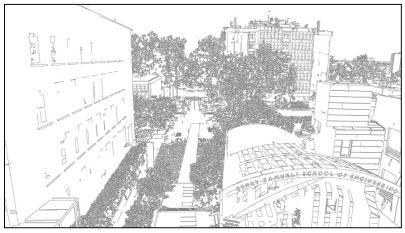
Lecture 14: Overview

- Project Discussion
  - Status and next steps
  - Structural model of the DUT
    - Model development on the whiteboard
    - Discussion
- Homework Assignment 6
  - Performance Estimation of the Canny Edge Detector

EECS 222 Project

- Application Example: Canny Edge Detector
  - Embedded system model for image processing: Automatic Edge Detection in a Digital Video Camera

EngPlaza001.bmp



EngPlaza001_edges.pgm

  - Video taken by a drone hovering over UCI Engineering Plaza
    - Available on the server: **`~eecs222/public/video/`**
    - High resolution, 2704 by 1520 pixes
    - Video length 9 seconds, using 20 extracted frames for test bench model

Homework Assignment 5

- Task: Structural Model of the Canny Edge Detector
  - Convert the application to process a stream of video frames
  - Add test bench structure to the SLDL model from Assignment 4
  - Choose either SpecC or SystemC for simulation
- Steps
  1. Create test bench structure: Stimulus, Platform, Monitor
  2. Create platform model: DataIn, DUT, DataOut
  3. Localize functions and add loops for stream processing
- Deliverables
  - **`Canny.sc`** or **`Canny.cpp`** (choose one!)
  - **`Canny.txt`**
- Due
  - By next week: May 15, 2017, 12pm (noon!)

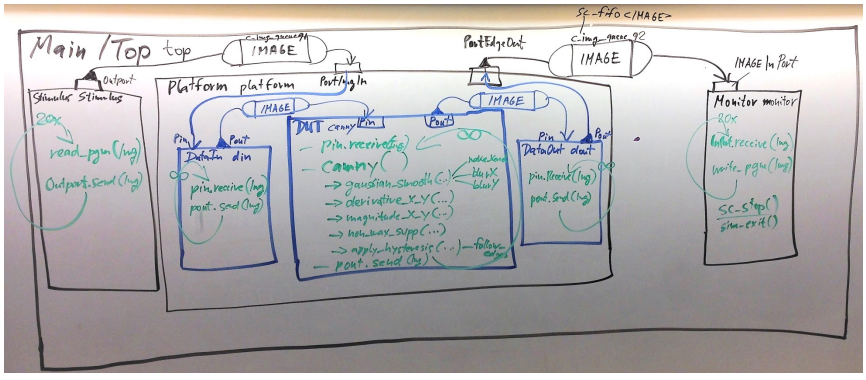
Homework Assignment 5

- Task: Structural Model of the Canny Edge Detector
  - Expected instance tree

```
Main / Top
|------ Monitor monitor
|------ Platform platform
|       |------ DUT canny
|       |------ DataIn din
|       |------ DataOut dout
|       |------ c_img_queue q1
|       \------ c_img_queue q2
|------ Stimulus stimulus
|------ c_img_queue q1
\------ c_img_queue q2
```


Homework Assignment 5

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



Homework Assignment 6

- Task: Performance Estimation of the Canny Edge Detector
  - Refine the structural hierarchy of the DUT block
  - Refine the structural hierarchy of the Gaussian Smooth block
  - Estimate the computational complexity of the DUT components
- Steps
  1. Create DUT structure: Gaussian Smooth, …, Apply Hysteresis
  2. Create Gaussian Smooth structure: Receive, Gauss, BlurX, BlurY
  3. Profile the application, obtain relative computational complexity
- Deliverables
  - **`Canny.sc`** or **`Canny.cpp`** (choose one!)
  - **`Canny.txt`** (with numerical values for block complexity!)
- Due
  - By next week: May 22, 2017, 12pm (noon!)

Homework 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
|------ c_img_queue q1
\------ c_img_queue q2
```

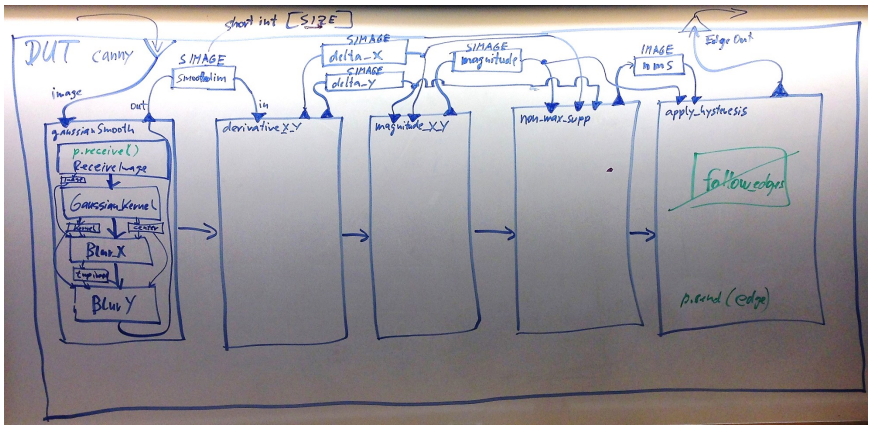
Homework Assignment 6

- Step 2: Refined hierarchy of the Gaussian Smooth block
  - 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
```


Homework Assignment 6

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



Homework Assignment 6

- Step 3: Profile the application components
  - ➢ Performance Estimation of the Canny Edge Detector
  - – SpecC model profiling: Use SCE profiler
    - ➢ **`/opt/sce/bin/sce`**
    - Create a new project, import SpecC source code
    - Compile and simulate in SCE (with instrumentation)
    - Run the profiler, view raw computation graph for DUT components
  - – SystemC model profiling: Use GNU profiler
    - ➢ **`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 (**`self`**)

Homework Assignment 6

- Step 3: Profile the application components, 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%
```