

ECPS 203

Embedded Systems Modeling and Design

Lecture 12

Rainer Dömer

doemer@uci.edu

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



Lecture 12: Overview

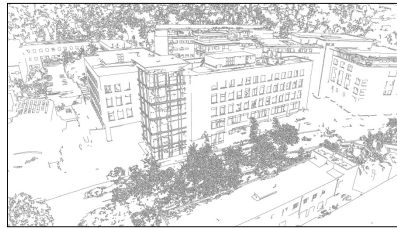
- Project Discussion
 - Status and next steps
- Assignment 5
 - Test bench model of the Canny Edge Detector
- Assignment 6
 - Structural refinement of the DUT module
 - Model development on the whiteboard
 - Profiling of the Canny Edge Detector functions
 - Discussion

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
 - Convert C++ model to SystemC model
 - Add a test bench structure around the C++ model
 - Wrap DUT into a platform model with dedicated 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 thread stack sizes
- Deliverables
 - SystemC source code and text file: `Canny.cpp`, `Canny.txt`
- Due
 - Wednesday, November 8, 2017, 6pm

Project Assignment 5

- Task: Test Bench for the Canny Edge Detector

– Expected instance tree

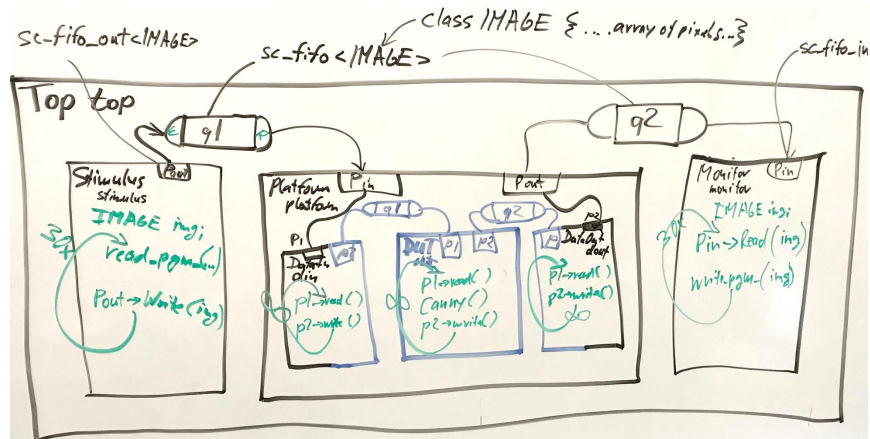
```

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

- 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 15, 2017, 6pm

ECPS203: Embedded Systems Modeling and Design, Lecture 12

(c) 2017 R. Doemer

7

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

```

ECPS203: Embedded Systems Modeling and Design, Lecture 12

(c) 2017 R. Doemer

8

Project Assignment 6

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

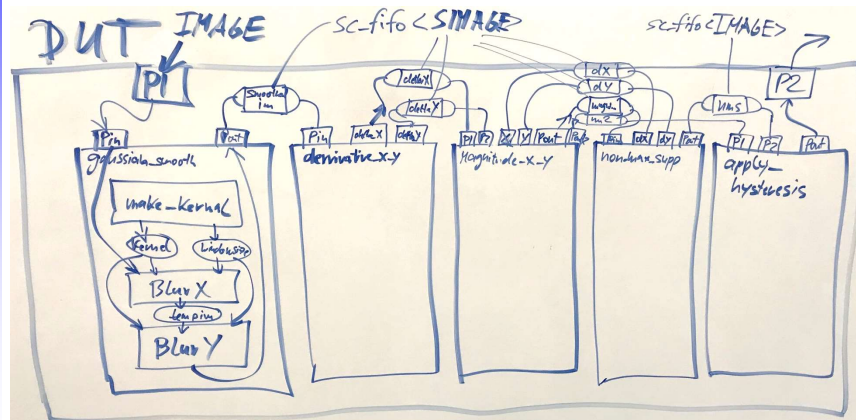
DUT canny

```

|----- 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`)

ECPS203: Embedded Systems Modeling and Design, Lecture 12

(c) 2017 R. Doemer

11

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%

```

ECPS203: Embedded Systems Modeling and Design, Lecture 12

(c) 2017 R. Doemer

12