# EECS 222: Embedded System Modeling Lecture 18

Rainer Dömer

doemer@uci.edu

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

## Lecture 18: Overview

- Course Administration
  - Instructor evaluation
  - Final exam
- System-on-Chip Environment (SCE)
  - Interactive demonstration (part 3)
  - Hardware synthesis
  - Software synthesis
- EECS 222 Project Discussion
  - Review of Assignments 1, 4 through 7
  - Discussion
  - Assignment 8
  - Final technical report

## Course Administration

- Final Course Evaluation
  - 9th through 10th week
  - May 22, 2017, through June 11, 2017, 11:45pm
  - Open until next Sunday night
  - Online via EEE evaluation application
- Evaluation of Course and Instructor
  - Voluntary
  - Anonymous
  - Very valuable!

➢ Please help to improve this class!

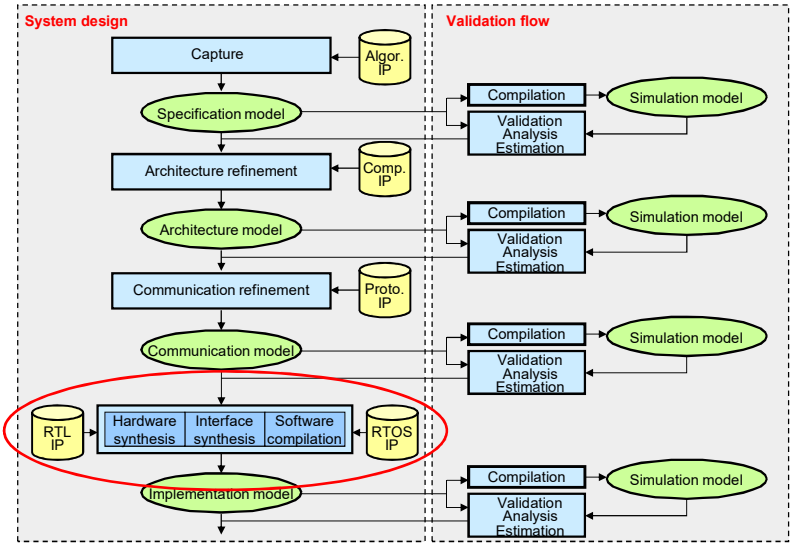
  - Please spend 5 minutes!



## Course Administration

- Final Exam
  - Allocated time
    - Wednesday, June 14, 4:00-6:00pm
  - Location
    - Not applicable, we use electronic submission!
  - Format: Final Project Report
    - Submission script: `~eecs222/bin/turnin.sh`
    - Directory name: `hw8`
    - File names: `EECS222_Report.pdf` `Canny.sc` *or* `Canny.cpp`
  - Hard deadline!
    - June 14, 2017, 6pm

## SCE Design Flow: HW and SW Synthesis






## SCE Design Flow: HW and SW Synthesis

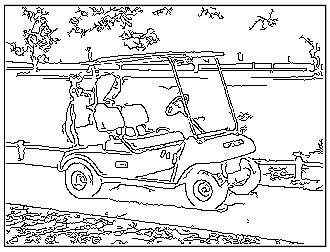
- Application Example: GSM Vocoder
  - Real-time constraint
    - max. 20ms per speech frame (total of 3.26s for speech file)
- Interactive Demonstration (Part 3)
  - From pin-accurate communication model
  - Down to pin- and cycle-accurate implementation model
- Hardware Synthesis
  - RTL refinement of Build_Code block
  - Allocation, scheduling, and binding
  - Accellera RTL modeling
- Software Synthesis
  - Software code generation
  - Cycle-accurate Instruction Set Simulation (ISS)

# EECS 222 Project

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

golfcart.pgm



golfcart.pgm_s_0.60_l_0.30_h_0.80.pgm

  - Application Source and Documentation:
    - http://marathon.csee.usf.edu/edge/edge_detection.html
    - http://en.wikipedia.org/wiki/Canny_edge_detector



# Project: Homework Assignment 1

- Task: Introduction to Application Example
  - Canny Edge Detector
  - Algorithm for edge detection in digital images
- Steps
  1. Setup your Linux programming environment
  2. Download, adjust, and compile the application C code with the GNU C compiler (gcc)
  3. Study the application
- Deliverables
  - None at this time (preparation for following assignments)
- Due
  - By next week: April 10, 2017, 12pm (noon!)

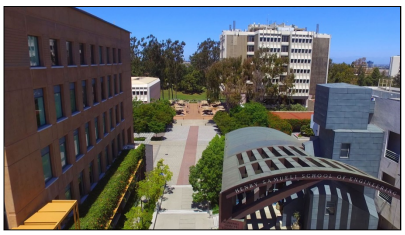
# Project: Homework Assignment 4

- Task: SLDL Model of the Canny Edge Detector
  - Convert ANSI-C source code into SLDL model
  - Choose either SpecC or SystemC for simulation
- Steps
  1. Fix the off-by-one bug in the **`non_max_supp`** function
  2. Clean-up the code for compilation without warnings
  3. Fix configuration parameters to compile-time constants
  4. Remove or replace dynamic memory allocation
- Deliverables
  - **`Canny.sc`** or **`Canny.cpp`** (choose one!)
  - **`Canny.txt`**
- Due
  - By next week: May 8, 2017, 12pm (noon!)

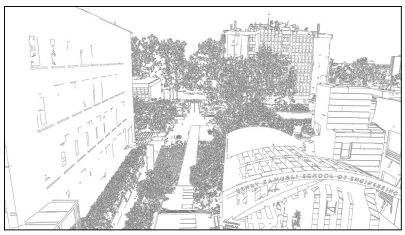


# Project: Video Stream Processing

- 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

## Project: 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!)



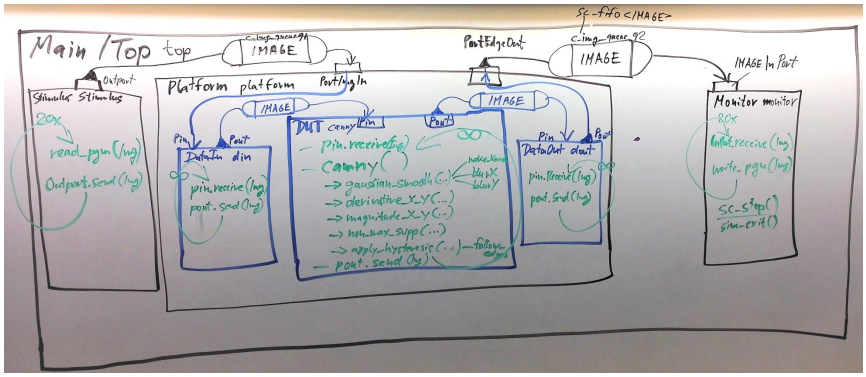
## Project: Homework Assignment 5

- 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
```

# Project: Homework Assignment 5

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






# Project: 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!)

# Project: 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
```



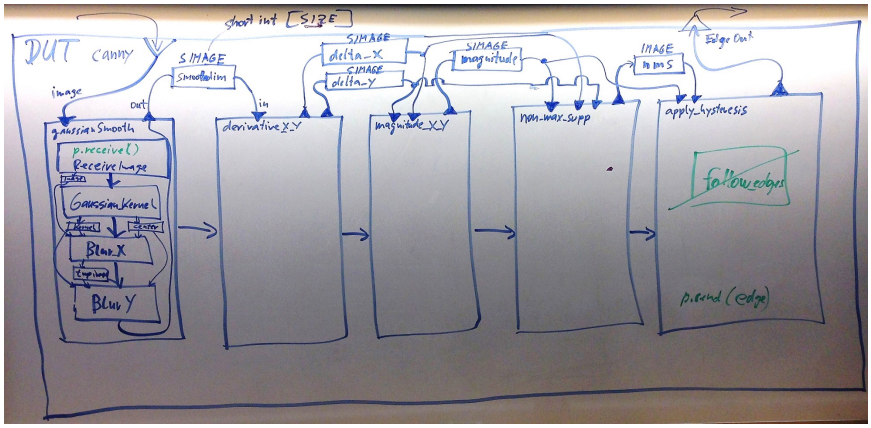
# Project: Homework Assignment 6

- 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
```

# Project: Homework Assignment 6

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






# Project: 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`**)

# Project: Homework Assignment 6

- Step 3: Profile the application components, obtain relative computational complexity
  - Expected complexity comparison (in **`Canny.txt`**):
    Using GNU profiler on SystemC model:

```
Gaussian_Smooth                  53%
|------ Gaussian_Kernel    0%
|------ BlurX              43%
\------ BlurY              57%
Derivative_X_Y                   10%
Magnitude_X_Y                     5%
Non_Max_Supp                     19%
Apply_Hysteresis                 14%
                                100%
```



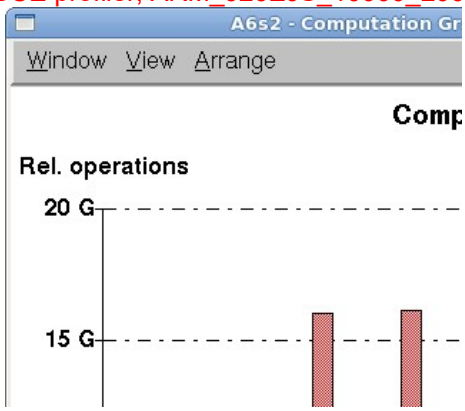
# Project: Homework Assignment 6

- Step 3: Profile the application components, obtain relative computational complexity
  - Expected complexity comparison (in **`Canny.txt`**):
    Using SCE profiler, ARM_926EJS_10000_20000_0 CPU:

```
Gaussian_Smooth                813.5/1268.9=64%
|------ Receive_Image      0.0/854.2=  0%
|------ Gaussian_Kernel    0.0/854.2=  0%
|------ BlurX            416.8/854.2= 49%
\------ BlurY            437.4/854.2= 51%
Derivative_X_Y                53.4/1268.9= 4%
Magnitude_X_Y                 53.4/1268.9= 4%
Non_Max_Supp                272.0/1268.9=21%
Apply_Hysteresis             76.5/1268.9= 6%
                          1268.9/1268.9=100%
```

# Project: Homework Assignment 6

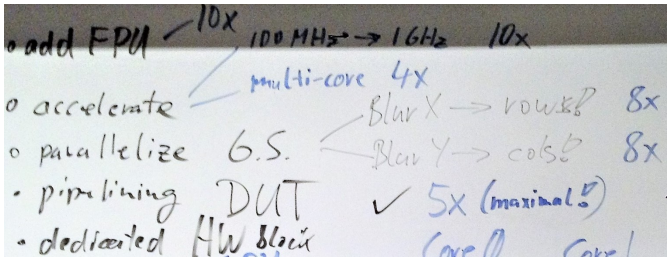
- Step 3: Profile the application components, obtain relative computational complexity
  - Expected complexity comparison (in **`Canny.txt`**):
    Using SCE profiler, ARM_926EJS_10000_20000_0 CPU






# Project: Optimization Options

- Step 3: Profile the application components, obtain relative computational complexity
- Step 4: (discussion in class)
  - What about absolute timing? How long does it take?
  - Does it meet our real-time goals?
  - What can be done to improve the speed?



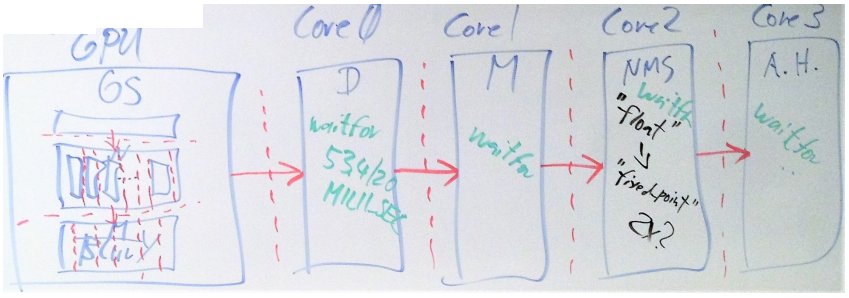
# Project: Homework Assignment 7

- Task: Pipelining and Parallelization of the Canny Model
  - Back-annotate estimated delays to observe timing in the model
  - Pipeline and parallelize the model to maximize throughput
- Steps
  1. Instrument model with logging of simulation time and frame delay
  2. Back-annotate estimated timing in DUT components
  3. Pipeline the DUT into stages for each component
  4. Integrate Gaussian Smooth components into pipeline stages
  5. 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: By next week: May 31, 2017, 12pm (Wednesday, noon!)



# Project: Homework Assignment 7

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



# Project: Homework Assignment 7

- Task: Pipelining and Parallelization of the Canny Model
  - Expected instance tree

```
DUT canny
|------ Gaussian_Smooth gaussian_smooth
|       |------ Receive_Image receive
|       \------ Gaussian_Kernel gauss
|------ BlurX blurX
|       |------ BlurX_Slice sliceX1
|       |------ BlurX_Slice sliceX2
|       |       [...]
|       \------ BlurX_Slice sliceX8
|------ BlurY blurY
|       |------ BlurY_Slice sliceY1
|       |       [...]
|       \------ BlurY_Slice sliceY8
|------ Derivative_X_Y derivative_x_y
|------ Magnitude_X_Y magnitude_x_y
|------ Non_Max_Supp non_max_supp
\------ Apply_Hysteresis apply_hysteresis
```



# Project: Homework Assignment 7

- Task: Pipelining and Parallelization of the Canny Model
  - Expected execution log with timing (after step 2)

```
          0: Stimulus sent frame  1.
          0: Stimulus sent frame  2.
          0: Stimulus sent frame  3.
          0: Stimulus sent frame  4.
          0: Stimulus sent frame  5.
          0: Stimulus sent frame  6.
    6547500: Monitor received frame  1 with  6547500 us delay.
    6547500: Stimulus sent frame  7.
   13095000: Monitor received frame  1 with 13095000 us delay.
   13095000: Stimulus sent frame  8.
   19642500: Monitor received frame  2 with 19642500 us delay.
   19642500: Stimulus sent frame  9.
[...]
  111307500: Monitor received frame 16 with 39285000 us delay.
  117855000: Monitor received frame 17 with 39285000 us delay.
  124402500: Monitor received frame 18 with 39285000 us delay.
  130950000: Monitor received frame 19 with 39285000 us delay.
  130950000: Monitor exits simulation.
```

# Project: Homework Assignment 7

- Task: Pipelining and Parallelization of the Canny Model
  - Timing observed after each step: SpecC models

| Model | Frame Delay | Total simulated time |
|---|---|---|
| CannyA7_step1 | 0 us | 0 us |
| CannyA7_step2 | 39285000 us | 130950000 us |
| CannyA7_step3 | 38821500 us | 98615500 us |
| CannyA7_step4 | 24057000 us | 52767500 us |
| CannyA7_step5 | 14960000 us | 28663125 us |

  - Timing observed after each step: SystemC models

| Model | Frame Delay | Total simulated time |
|---|---|---|
| CannyA7_step1 | 0 s | 0 s |
| CannyA7_step2 | 6561 ms | 48100 ms |
| CannyA7_step3 | 6561 ms | 48100 ms |
| CannyA7_step4 | 6561 ms | 48100 ms |
| CannyA7_step5 | 4080 ms | 28649 ms |



# Project: Discussion of Performance

- Performance metrics observed (in Assignment 7)
  - Total simulated time
    - Total processing time for our stream of 20 frames
  - Frame delay
    - Processing time for each frame from pipeline input to pipeline output
    - Incurs "extra stages" due to testbench queues of size 2
- Performance metrics wanted (in Assignment 8)
  - Stage delay
    - Delay incurred in each pipeline stage; *maximum* matters!
  - Pipeline latency
    - N*max(StageDelay), where N is the number of stages
  - Pipeline throughput
    - Number of frames coming out of the pipeline per second (FPS)
- Discussion in class: **`Lecture18_Canny.xlsx`**

# Project: Homework Assignment 8

- Task: Throughput optimization of Canny Edge Decoder
  - Back-annotate more realistic delays and observe throughput
  - Maximize the pipeline throughput by using fixed-point arithmetic
- Steps
  1. Improve test bench with logging of frames per second (FPS)
  2. Estimate timing based on allocated CPU cores and ASICs
  3. Replace floating-point with fixed-point arithmetic in NMS block
  4. Back-annotate the improved NMS stage delay
  5. Final technical project report
- Deliverables
  - **`Canny.sc`** or **`Canny.cpp`** (choose one!)
  - **`EECS222_Report.pdf`** (in lieu of final exam)
- Due: By next week: June 14, 2017, 6pm (Thursday evening)



# Project: Final Technical Report

- Final Technical Project Report
  - Title
    - *Specification and Modeling of a Canny Edge Detector for System-on-Chip Design*
  - Contents
    - "Story" of the course project
      - From downloading initial C reference code
      - Via describing and simulating in SpecC or SystemC SLDL
      - To modeling and optimization for SoC design
    - Use the results of Assignments 1, and 4 through 8
    - Conclude with a summary of the lessons learned
  - Length
    - About 12 pages (including title page, figures, and references)

## Project: Outline of Final Technical Report

1. Title page
   - Project title, author, date, course number and title
   - Abstract
2. Introduction
   - System-level modeling and design
   - Essential concepts and coverage in SpecC/SystemC SLDL
3. Case study using the Canny Edge Detector application
   - Obtaining and studying the Canny application
   - Creating a simulatable model in SpecC/SystemC SLDL
   - Creating structural hierarchy with test bench
   - Pipelining and parallelization
   - Performance estimation and throughput optimization
4. Summary and Conclusion
   - Lessons Learned
   - Future work
5. References