# EECS 222A: System-on-Chip Description and Modeling Lecture 10

Rainer Dömer

doemer@uci.edu

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

## Lecture 10: Overview

- Course Administration
- Unified Modeling Language (UML)
  - Overview
  - Example Diagrams
- Project Discussion: Canny Edge Detector
  - Survey Results on Eclipse Usage
  - Review
  - Final Report

## Course Administration

- Final Exam
  - Date and time
    - Wednesday, June 13, 2 - 4pm
  - Location
    - None (electronic submission)
  - Format
    - Submission of Final Project Report
      - Submission script: `turnin`
      - Directory name `report`
      - File name `CannyReport.pdf`
  - Hard deadline!
    - June 13, 2012, 4pm



## Course Administration

- Final Course Evaluation
  - 9th through 10th week
  - May 30, 2012, through June 10, 2012, 11:45pm
  - Open until 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!

## Unified Modeling Language (UML)

- Status
  - UML 2.0 Superstructure
  - developed and maintained by OMG (Object Management Group)
- Goals
  - Raising the Level of Abstraction
  - Modeling of software applications
    - before coding
  - Specification of software architecture
  - High-level description of software architecture to enable
    - scalability
    - security
    - robustness
    - maintenance
    - extendability
    - code reuse
  - Model Driven Architecture (MDA)



## Unified Modeling Language (UML)

- What is UML?
  - 13 Standard Diagrams
    - Specification
    - Design
    - Documentation
  - Graphical Representation of
    - Software architecture
    - Software structure
    - Software behavior
    - Object relations
    - ...
  - Not executable!
  - Tools available
    - Graphical capture
    - Editing
    - Code generation (template code)

## Unified Modeling Language (UML)

- UML Standard Diagrams
  - Structure Diagrams
    - Class Diagram
    - Object Diagram
    - Component Diagram
    - Composite Structure Diagram
    - Package Diagram
    - Deployment Diagram
  - Behavior Diagrams
    - Use Case Diagram
    - Activity Diagram
    - State Machine Diagram
  - Interaction Diagrams
    - Sequence Diagram
    - Communication Diagram
    - Timing Diagram
    - Interaction Overview Diagram



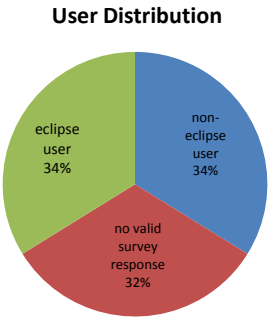
## Unified Modeling Language (UML)

- UML Resources
  - Online Documents
    - Object Management Group (OMG)
      - **`www.uml.org`**
  - Online Tutorial
    - Borland's UML Tutorial
      - **`bdn.borland.com`**
  - Talk at UCI in 2004
    - Dr. Wolfgang Mueller, C-LAB, Paderborn, Germany
      - **`Lecture10_UML.pdf`**

# Survey Results on Eclipse Usage

- Usage Distribution for Assignment 4

| Eclipse | Non-Eclipse | Not in Survey | Total |
|---|---|---|---|
| 23 | 23 | 22 | 68 |




- Reasons not to use Eclipse
  - 39% say their network connection is too slow or unstable
  - 9% say they are unfamiliar with IDE or do not like GUI
  - 52% did not state any reasons



# Survey Results on Eclipse Usage

- Usage Distribution for Assignments 2, 3, and 4
  - Based on log data

| | A2 | A3 | A4 |
|---|---|---|---|
| # of users | 38 | 32 | 34 |

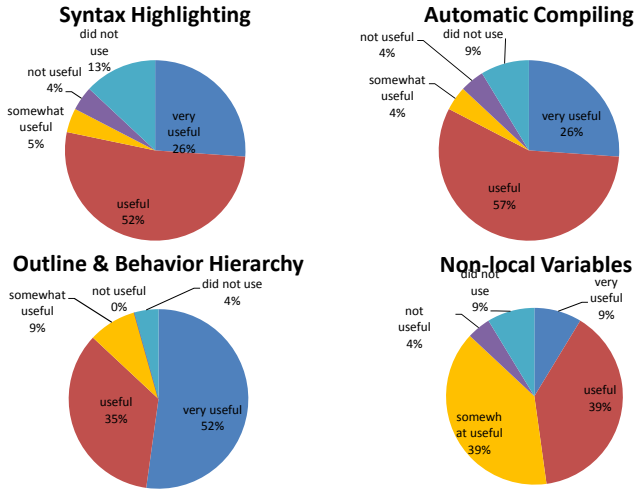
Table 1. User counts in each assignment

| Usage | # of users |
|---|---|
| Never used | 16 |
| Used in all assignments | 25 |
| Used in A2 or A3 only | 16 |
| Used in A3 or A4 only | 9 |

Table 2. Usage trends

# Survey Results on Eclipse Usage

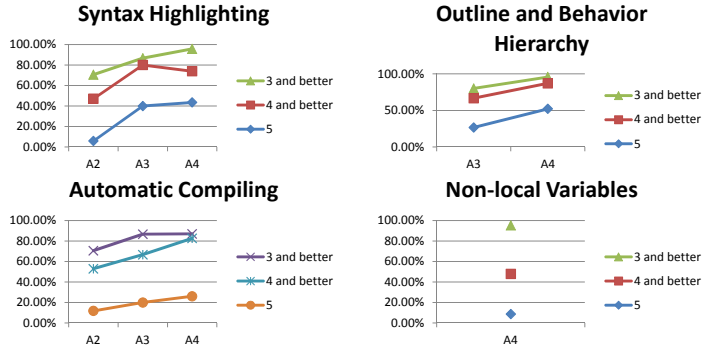
- User Rating for Assignment 4






# Survey Results on Eclipse Usage

- User Rating for Assignment 2, 3, and 4
  - Values 'very useful' - 5, 'useful' - 4, 'somewhat useful' - 3, 'not useful' - 2, 'did not use' - 1
  - Percentages shown exclude non-eclipse users

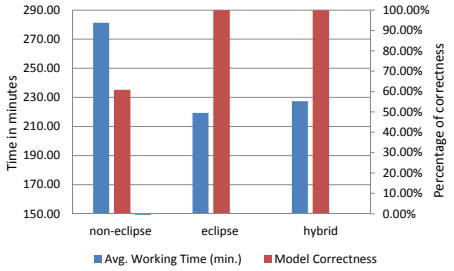



- ➢ Results show user satisfaction grows with later assignments
  - ➢ after continued usage, or with more features!?

# Survey Results on Eclipse Usage

- Assignment 4 Reported Time and Correctness

| | non-eclipse | eclipse | hybrid |
|---|---|---|---|
| # of users | 23 | 18 | 5 |
| Avg. time spent (minutes) | 281.30 | 219.39 | 227.40 |
| Successful model simulation | 60.87% | 100.00% | 100.00% |



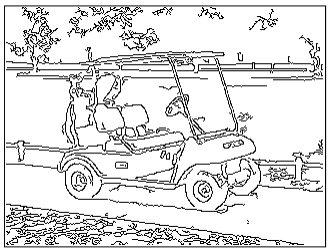

- Eclipse users spent less time than non-users!
- 100% of Eclipse users were successful!



# Project Discussion

- Application Example: Canny Edge Detector
  - Model a SoC for Edge Detection in a Digital Camera

golfcart.pgm

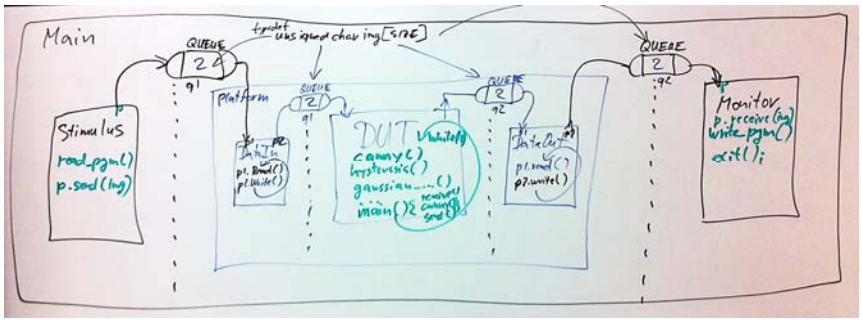


golfcart.pgm_s_0.60_l_0.30_h_0.80.pgm

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

# Project Discussion

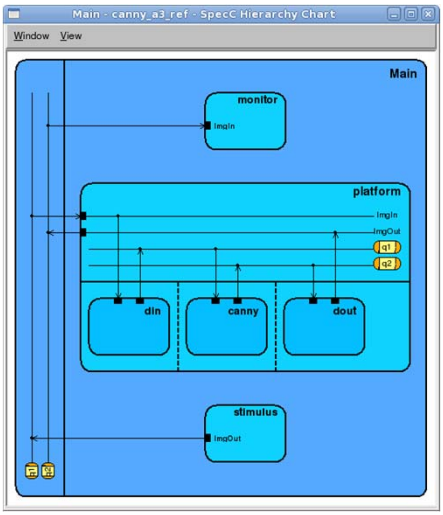
- Structural Hierarchy for Canny Edge Detector
  - Test bench Structure






# Project Discussion

- SCE Chart of Test bench Structure



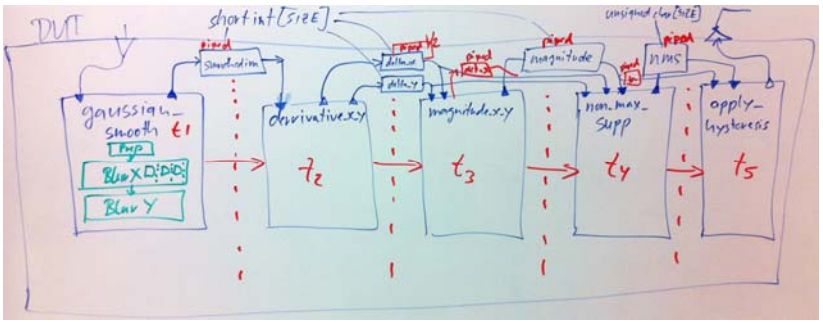
# Project Discussion

- Structural Hierarchy for Canny Edge Detector
  - Test bench Structure
    - `B i o behavior Main`
    - `B i l |------ Monitor monitor`
    - `B i c |------ Platform platform`
    - `B i l | |------ DUT canny`
    - `B i l | |------ DataIn din`
    - `B i l | |------ DataOut dout`
    - `C i l | |------ c_img_queue q1`
    - `C i l | \------ c_img_queue q2`
    - `B i l |------ Stimulus stimulus`
    - `C i l |------ c_img_queue q1`
    - `C i l \------ c_img_queue q2`



# Project Discussion

- Additional Level of Hierarchy inside DUT




  - Potential for parallelism
    - 5 pipeline stages in DUT (red color)
    - Parallel decomposition of BlurX and BlurY blocks in Gaussian Smooth behavior (green color)

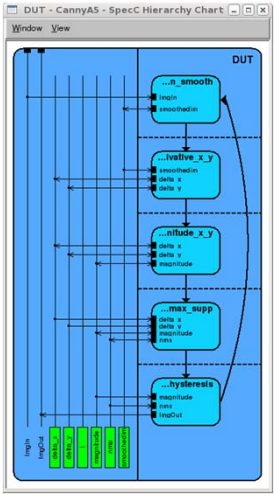
# Project Discussion

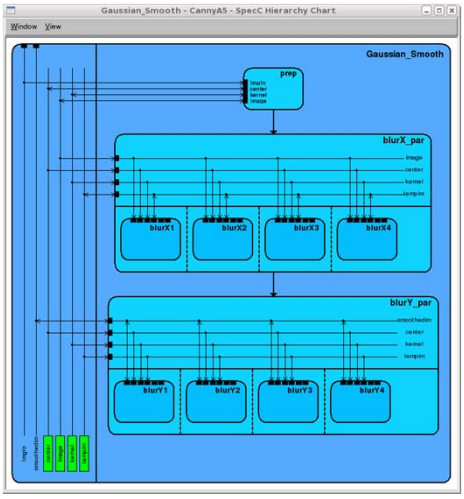
- Recoding the DUT
  - Additional level of hierarchy inside DUT
    - Behavioral Composition
      - Parallel execution desirable
      - Sequential execution as needed
    - Structural Composition
      - Standard channels, or
      - Variables shared through port maps
  - Canny Edge Detector: **`canny( )`**
    - **`gaussian_smooth( )`**
      - » **`make_gaussian_kernel( )`**
    - **`derrivative_x_y( )`**
    - **`magnitude_x_y( )`**
    - **`non_max_supp( )`**
    - **`apply_hysteresis( )`**
      - » **`follow_edges( )`**



# Project Discussion

- Pipelined DUT with parallelized Gaussian_Smooth



# Project Discussion

- Pipelined DUT with parallelized Gaussian_Smooth

```
– B i p   behavior DUT
– B i l   |------ Apply_Hysteresis apply_hysteresis
– B i l   |------ Derivative_X_Y derivative_x_y
– B i s   |------ Gaussian_Smooth gaussian_smooth
– B i c   |       |------ BlurX_par blurX_par
– B i l   |       |       |------ BlurX blurX1
– B i l   |       |       |------ BlurX blurX2
– B i l   |       |       |------ BlurX blurX3
– B i l   |       |       \------ BlurX blurX4
– B i c   |       |------ BlurY_par blurY_par
– B i l   |       |       |------ BlurY blurY1
– B i l   |       |       |------ BlurY blurY2
– B i l   |       |       |------ BlurY blurY3
– B i l   |       |       \------ BlurY blurY4
– B i l   |       \------ Prep prep
– B i l   |------ Magnitude_X_Y magnitude_x_y
– B i l   \------ Non_Max_Supp non_max_supp
```

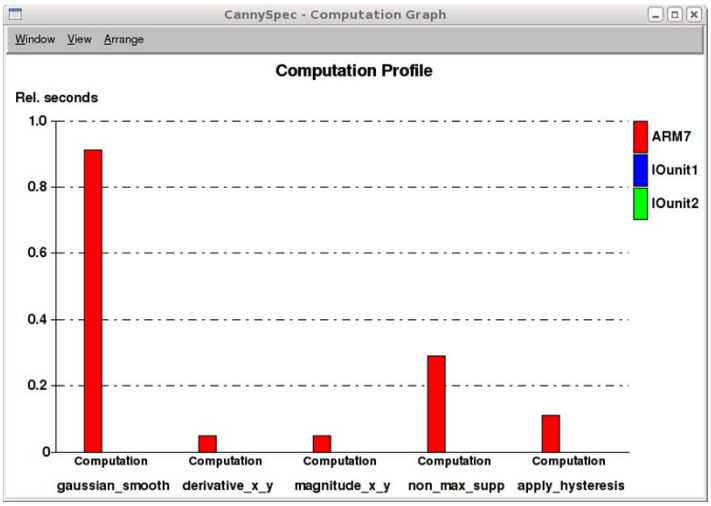


# Project Discussion: Initial Estimation

- Specification Model
  - Import Canny_a5_start.sc
  - Compile and simulate
    - Untimed, 0 ms
- Estimation of Software-only Architecture
  - Top-level Platform
  - Allocate
    - ARM7 of type ARM_7TDMI_10000_20000_0 for DUT
    - IOunit1, IOunit2 of type HW_Virtual for DataIn, DataOut
  - Estimate
    - 1358.8 ms for DUT on ARM7
  - Architecture Model
    - Unscheduled timing: 716086 micro seconds
  - Scheduled Model
    - Scheduled timing: 1358694 micro seconds

## Project Discussion: Initial Profile

- Computation Profile of Canny Algorithm






## Project Discussion: Hardware Acceleration

- Hardware Blocks for Gaussian Smooth
  - Allocate additional HW components
    - HW_BlurX, HW_BlurY of type HW_Standard for blurX_par, blurY_par
  - Estimation Results
    - 501.9 ms for Canny on ARM7
    - 4 x 23.9 ms for BlurX_par on BlurX
    - 4 x 26.5 ms for BlurY_par on BlurY
- Refinement using SCE
  - Architecture Model
    - Unscheduled timing: 551471 micro seconds
  - Scheduled Model
    - ARM7 statically scheduled
    - HW units *not* scheduled!
      - We assume HW PEs internally process data in parallel
    - Scheduled timing: 551471 micro seconds

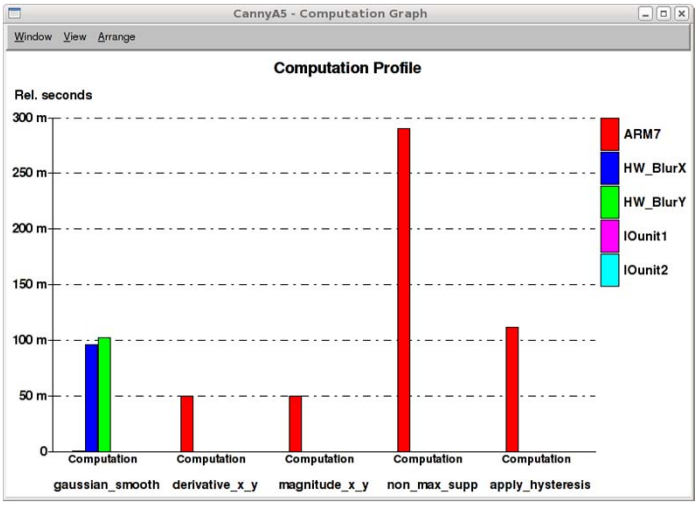
## Project Discussion: Real-Time Constraints

- Digital Still Image Camera
  - Weak timing constraint
  - 0.55 seconds delay for edge detection may be acceptable
- Digital Video Camera
  - Hard real-time constraint
  - 30 frames per second (FPS) are needed
    - 30 FPS equals maximum delay of 33.3 ms
    - 551 ms is 16.5 times too high
  - Considering higher clock speed
    - SCE assumptions:
      - Default 100 MHz, max. 500 MHz for ARM_7TDMI
      - Default 100 MHz, max. 500 MHz for HW_Standard
    - ➢ This would result in 5x speedup!
      - Architecture model with 500 MHz PEs: 110294 micro seconds
      - ➢ Still 3.3x too slow for real-time video…



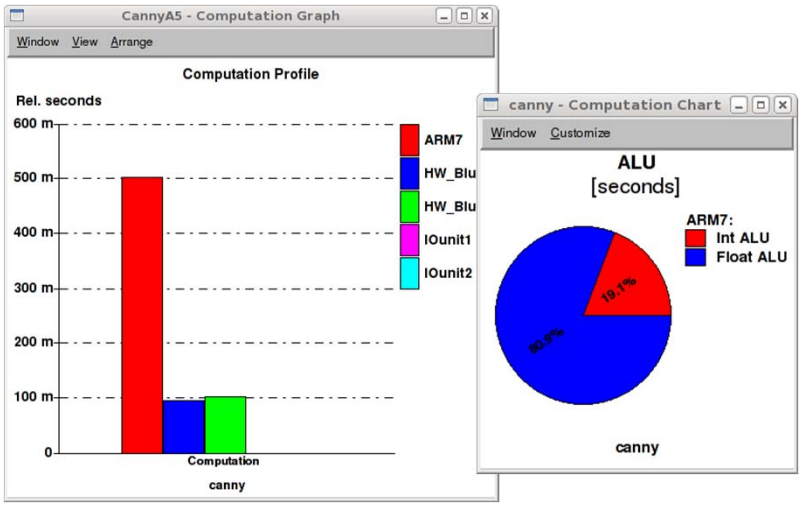
## Project Discussion: Performance Optimization

- Computation Profile of DUT Pipeline



## Project Discussion: Performance Optimization

- Computation Profile of DUT Operations



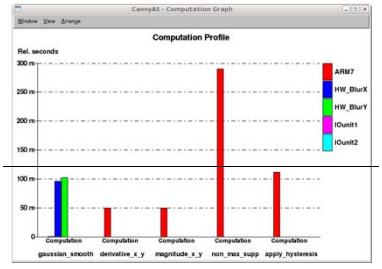



## Project Discussion: Algorithm Recoding

- Observation: Majority of computation is floating-point
  - In magnitude_x_y, 69.5% of all operations are floating-point
  - In non_max_supp, 95.6% of all operations are floating-point
- ➢ Replace Floating- with Fixed-Point Computation!
- Benefits Estimation:
  - ARM_7TDMI profiling tables provide very rough estimates
    - Addition: 4 cycles for float, 1 cycle for int
    - Multiplication: 8 cycles for float, 2 cycles for int
    - Division: 40 cycles for float, 10 cycles for int
    - ➢ Can get 4x speedup for using fixed-point computation!
  - Overall profile for Canny behavior
    - 80.9% floating-point usage (against 19.1% integer)
    - Assumption: 80.9% of CPU time can be sped up by 4x
    - Potential gain: 80.9% / 4 + 19.1% = 39.325% (about 2.5x)

## Project Discussion: Pipelining for Video

- For real-time streaming video, we need 30 FPS
  - Throughput must be < 33.3 ms!
  - Latency can be longer!
- ➢ Pipelining!
  - ➢ Assuming fixed-point implementation for non_max_supp
    - ➢ Estimated delay reduced to 40% of 290.4 ms = 116.16ms
  - ➢ Assuming 5 pipeline stages
    - ➢ Max. stage delay becomes 116.16 ms
    - ➢ Stages 2 and 3 can even be combined
  - ➢ Balancing by raising CPU speed
    - ➢ Need to increase CPU frequency by 116.16 / 33.3 = 3.5
    - ➢ Change ARM7 to ≥ 350 MHz
    - ➢ No need to increase HW frequency, already below 33.3 ms (4x parallelism)






## Project Discussion: Open Issues

- Entire discussion so far is based on *estimated* values!
  - Estimation by table-based retargetable profiling
  - Typically no absolute accuracy, only *fidelity*!
- Communication delays are not considered
  - Scheduled architecture model assumes 0 communication delay
  - Network and Link Refinement needed
    - Result: Communication adds about 7 ms delay (about 1.3%)
    - See example instructions on next slide!
- Cycle-Accurate Model
  - To obtain absolute accuracy, we would need
    - Cycle-accurate SW timing: Instruction Set Simulation
    - Cycle-accurate HW timing: RTL Synthesis
  - ➢ Both are beyond the objectives of this course, so we will base our project only on the estimated times!

## Project Discussion: Communication Model

- Network and Link Refinement
  - Instructions (following the scheduled architecture model above)
    - Allocate AMBA_AHB as CPU/system bus (default parameters)
    - Allocate HardwareBus as HW_Bus (default parameters)
    - On AMBA_AHB, connect
      - ARM7 as Master0
      - HW_BlurX, HW_BlurY as Slave4, Slave5
      - IOunit1, IOunit2 as Slave7, Slave8
    - On HW_Bus, connect
      - HW_BlurX as Master
      - HW_BlurY as Slave
    - Assign link parameters
      - Use "Autofill all addresses"
      - Use Polling on AMBA_AHB for all channels
      - Use interrupt MasterSync0 on HW_Bus
  - Generate TLM
    - Simulated time: 558156 micro seconds
    - Simulator run time: about 40 seconds
  - Generate PAM
    - Simulated time: 558693 micro seconds
    - Simulator run time: about 6 minutes



## Project Discussion: Final Report

- Final Deliverable: Technical Report
  - Title
    - *Modeling of a Canny Edge Detector System-on-Chip for a Digital Camera*
  - Contents
    - Describe the "story" of our project
      - from initial C reference code
      - via modeling and recoding in SpecC
      - to a TLM refined by use of SCE
    - Use the results (figures) of Assignments 2 through 5
    - Conclude with a summary of the lessons learned
  - Length
    - About 12 pages
      (including title page, figures, and references)

## Project Discussion: Outline of Final Report

- Title page
  - Project title, author, date, and course
  - Abstract

1. Introduction
   a. System-Level Modeling
   b. System-Level Description Languages
2. Case Study on a Canny Edge Detector SoC
   a. Canny Application Reference C Code
   b. System Level Model in SpecC
      - Test bench structure
      - Algorithm structure and parallelization
   c. Estimation, Optimization and Refinement using SCE
   d. Transaction Level Model
3. Conclusion
   a. Summary
   b. Lessons Learned
4. References