ECPS 203
Embedded Systems Modeling and Design
Lecture 17

Rainer Dömer

doemer@uci.edu

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

CENTER FOR EMBEDDED AND CYBER-PHYSICAL SYSTEMS

MECPS UCI University of California, Irvine

Lecture 17: Overview

- Course Administration
  - Final course evaluation
  - Final report
- Project Discussion
  - A7: Performance measurement on prototyping board
  - A8: Back-annotation of timing estimates into SystemC model
  - A8: Pipelining and parallelization of the DUT module
  - A9: Throughput optimization by pipeline load balancing
- Unified Modeling Language (UML)
  - Overview
  - Example Diagrams

Course Administration

- Final Course Evaluation
  - Open until end of 10th week (Sunday night)
  - Nov. 26, 2018, through Dec. 9, 2018, 11:45pm
  - Online via EEE Evaluation application
- Mandatory Evaluation of Course and Instructor
  - Voluntary
  - Anonymous
  - Very valuable
- Please spend 5 minutes for this survey!
  - Your feedback is appreciated!

Course Administration

- Final Report (in lieu of Final Exam)
  - Allocated time and room for final exam
    - Monday, December 10, 8:00-10:00am (DBH 1200)
    - ➢ *Not applicable, we use electronic submission instead!*
  - Format: Final Project Report
    - Submission script: **`~ecps203/bin/turnin.sh`**
    - Directory name: **`final`**
    - Deliverables: **`ECPS203_Report.pdf`** **`Canny.cpp`**
  - A9 deadline: Draft report (for early feedback)
    - Wednesday, December 5, 2018, 6pm
  - Hard deadline: Final report (graded!)
    - Monday, December 10, 2018, 6pm

Project Assignment 7

- Task: Performance measurement on prototyping board
  - Measured delays on Raspberry Pi 3 (in **Canny.txt**):

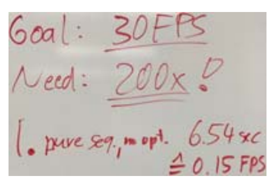
```
Gaussian_Smooth                              3.53 sec
|------ Gaussian_Kernel  0.00 sec
|------ BlurX            1.71 sec
\------ BlurY            1.82 sec
Derivative_X_Y                               0.48 sec
Magnitude_X_Y                                1.03 sec
Non_Max_Supp                                 0.83 sec
Apply_Hysteresis                             0.67 sec
TOTAL                                        6.54 sec
```

  - ➢ This performance is far too slow for real-time video!
  - ➢ Discussion: What options exist to speed this up?

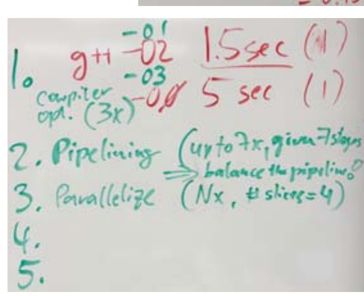
ECPS203: Embedded Systems Modeling and Design, Lecture 17 (c) 2018 R. Doemer 5

Project Assignment 7

- Discussion: Measured delays on Raspberry Pi 3
  - **TOTAL** **6.54 seconds**
  - ➢ This performance is far too slow for real-time video!



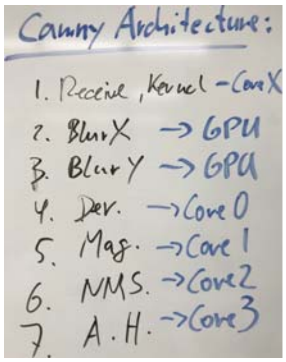
  - ➢ Discussion: What options exist to speed this up?

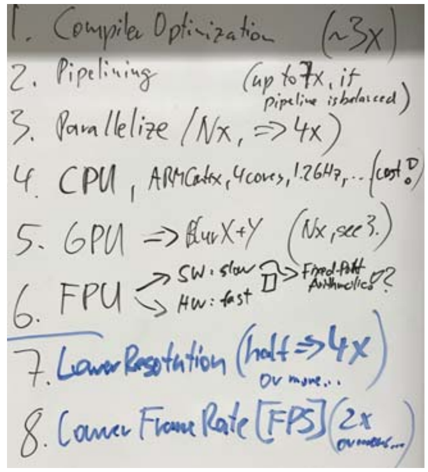


ECPS203: Embedded Systems Modeling and Design, Lecture 17 (c) 2018 R. Doemer 6

Project Assignment 7

- Discussion: Measured delays on Raspberry Pi 3
  - `TOTAL                    6.54 seconds`
  - ➢ Discussion: What options exist to speed this up?



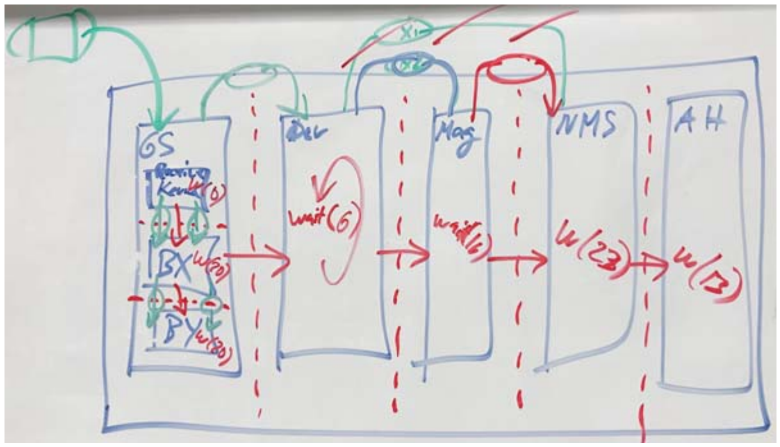


Project Assignment 8

- 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 simulated time to observe frame delay
  2. Back-annotate estimated timing into DUT components
  3. Improve test bench to observe frame throughput
  4. Pipeline the DUT into a sequence of 7 stages with buffer size 1
  5. Slice the BlurX and BlurY modules into 4 parallel threads
- Deliverables
  - **`Canny.cpp:`** pipelined and parallelized SystemC model
  - **`Canny.txt:`** table of observed frame delays and throughput
- Due: Wednesday, November 28, 2018, 6pm

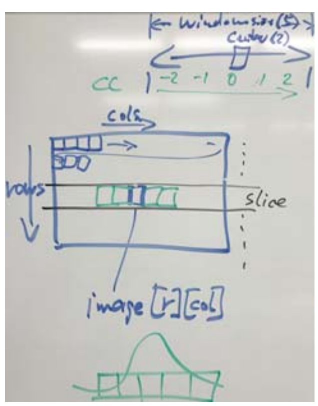
Project Assignment 8

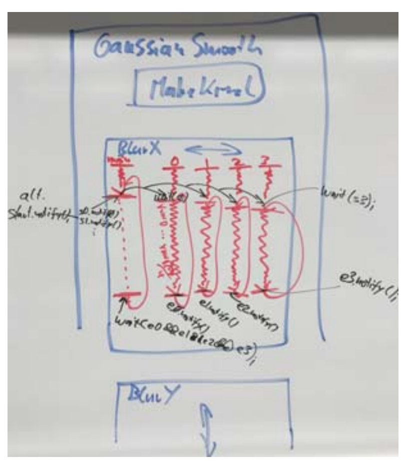
- Pipelined and parallel model of the Canny Edge Detector
  - Discussion on whiteboard: Chart of pipelined DUT structure



Project Assignment 8

- Pipelined and parallel model of the Canny Edge Detector
  - Discussion on whiteboard: Parallel BlurX, BlurY functions (step 5)





Project Assignment 8

- Pipelined and parallel model of the Canny Edge Detector
  - Back-annotation of measured timing delays
  - ➢ 4-way parallelization of BlurX and BlurY modules (step 5)

```
Receive, Make_Kernel     0 ms        0 ms
BlurX                 1710 ms      427 ms
BlurY                 1820 ms      455 ms
Derivative_X_Y         480 ms      480 ms
Magnitude_X_Y         1030 ms     1030 ms
Non_Max_Supp           830 ms      830 ms
Apply_Hysteresis       670 ms      670 ms
                      =======     =======
TOTAL:                6540 ms     3892 ms
                      =======     =======

Throughput:           1/1820ms    1/1030ms
                      0.549 FPS   0.971 FPS
```


Project Assignment 8

- Pipelined and parallel model of the Canny Edge Detector
  - Expected execution log with timing (after step 5)

```
      0 s: Stimulus sent frame  1.
      0 s: Stimulus sent frame  2.
      0 s: Stimulus sent frame  3.
  [...]
  3422 ms: Stimulus sent frame 16.
  3892 ms: Monitor received frame  1 with  3892 ms delay.
  [...]
 30672 ms: Monitor received frame 27 with 15920 ms delay.
 30672 ms: 1.030 seconds after previous frame, 0.971 FPS.
 31702 ms: Monitor received frame 28 with 15920 ms delay.
 31702 ms: 1.030 seconds after previous frame, 0.971 FPS.
 32732 ms: Monitor received frame 29 with 15920 ms delay.
 32732 ms: 1.030 seconds after previous frame, 0.971 FPS.
 33762 ms: Monitor received frame 30 with 15920 ms delay.
 33762 ms: Monitor exits simulation.
```

Project Assignment 8

- Task: Pipelining and parallelization of the DUT module
  - Expected simulated performance values (in **`Canny.txt`**):

Model	Frame Delay	Throughput	Total
CannyA8_step1	... ms		... ms
CannyA8_step2	... ms		... ms
CannyA8_step3	... ms	... FPS	... ms
CannyA8_step4	... ms	... FPS	... ms
CannyA8_step5	... ms	... FPS	... ms

Project Assignment 9

- Task: Throughput optimization by pipeline load balancing
  - Optimize the bottleneck stages to improve throughput
  - Prepare final report
- Steps
  1. Apply compiler optimizations for maximum execution speed
  2. Consider fixed-point instead of floating-point arithmetic
  3. Prepare draft of project report
- Deliverables
  - **`Canny.cpp`** (final SystemC model, graded)
  - **`Canny.txt`** (extended performance table, graded)
  - **`Canny.pdf`** (draft report, reviewed but not graded)
- Due
  - Wednesday, December 5, 2018, 6pm

Project Assignment 9

- Step 1: Apply compiler optimizations to reduce execution time
  - Experiment with various compiler options, including:

```
-O2
-O3
-mfloat-abi=hard
-fmpu=neon-fp-armv8
-mneon-for-64bits
```

  - Refer to documentation on
    - GNU compiler
    - ARMv8 Cortex-A53

ECPS203: Embedded Systems Modeling and Design, Lecture 17 (c) 2018 R. Doemer 15

Project Assignment 9

- Step 2: Consider fixed-point calculations instead of floating-point arithmetic
  - Focus on **Non_Max_Supp** module only
  - Convert **float** type variables to **int** types
  - Replace these lines of code…

```
xperp = -(gx = *gxptr)/((float)m00);
yperp = (gy = *gyptr)/((float)m00);
```

  - … with this code

```
gx = *gxptr;
gy = *gyptr;
xperp = -(gx<<16)/m00;
yperp = (gy<<16)/m00
```

  - Measure the timing difference on the prototyping board
  - Measure and evaluate the image quality (**ImageDiff**)

ECPS203: Embedded Systems Modeling and Design, Lecture 17 (c) 2018 R. Doemer 16

Project Optimization Chart

- Optimizations and their Effect on Throughput
  - Chart to visualize optimizations applied in assignments

ECPS203: Embedded Systems Modeling and Design, Lecture 17 (c) 2018 R. Doemer 17

Final Project Report

- Technical Report about the Course Project
  - Title
    - *Modeling of a Canny Edge Detector for Embedded Systems Design*
  - Contents
    - "Story" of the Canny Edge Detector project
      - From downloading the initial C reference code
      - Via modeling and simulating in SystemC
      - To performance optimization for real-time video
    - Describe relevant project assignments 1 through 9
    - Focus on the *reasoning* behind the optimizations
  - Length
    - About 12 pages (including title page, figures, and bibliography)

ECPS203: Embedded Systems Modeling and Design, Lecture 17 (c) 2018 R. Doemer 18

Final Project Report

1. Title page
   - Project title, author, date, course number and title
   - Abstract
2. Introduction
   - Embedded system modeling and design concepts
   - The IEEE SystemC language
3. Case Study of a Canny Edge Detector for Real-time Video
   - Structure of the Canny edge detection algorithm
   - Modeling and simulation in IEEE SystemC
   - Model refinement for pipelining and parallelization
   - Performance estimation and throughput optimization
   - Real-time video performance results
4. Summary and Conclusion
   - Lessons learned
   - Future work
5. References

Unified Modeling Language (UML)

- Goals
  - Raising the level of abstraction
  - Modeling of software applications
    - ➢ before coding!
  - Specification of software architecture
  - Enabling
    - scalability
    - security
    - robustness
    - maintenance
    - extendability
    - code reuse
  - Model Driven Architecture (MDA)
- Status
  - UML 2.0: Modeling Language in Software Engineering
  - standardized by OMG (Object Management Group) in 1997
  - standardized by ISO (Intl. Org. for Standardization) in 2005

Unified Modeling Language (UML)

- What is UML?
  - Graphical representation of …
    - Software architecture
    - Software structure
    - Software behavior
    - Object relations
    - ...
  - 13 standard diagrams
    - Specification
    - Design
    - Documentation
  - ➢ Not executable!
  - Commercial tools available for …
    - Graphical capture
    - Editing
    - Code generation (template code)

Unified Modeling Language (UML)

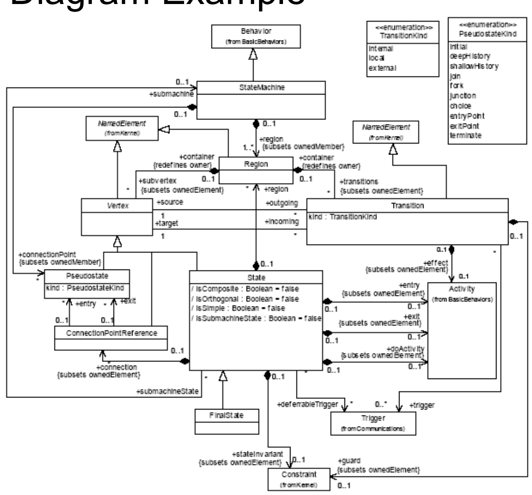
- UML Standard Diagrams
  - Structure Diagrams
    - Class Diagram
    - Object Diagram
    - Component Diagram
    - Composite Structure Diagram
    - Package Diagram
    - Deployment Diagram
  - Behavior Diagrams
    - Use Case Diagram
    - Activity Diagram
    - State Machine Diagram
  - Interaction 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 Tutorials
      - `https://www.tutorialspoint.com/uml/`
      - `http://www.sparxsystems.com/uml-tutorial.html`
  - Invited Talk at UCI in 2004
    - Dr. Wolfgang Mueller, C-LAB, Paderborn, Germany
    - Source of the following UML diagram examples

Unified Modeling Language (UML)

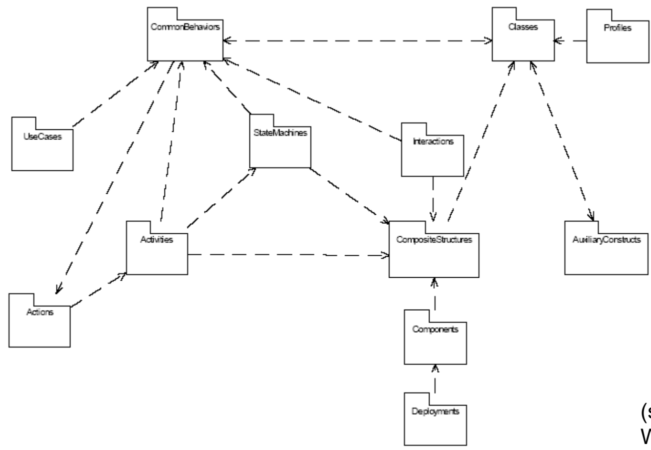
- Class Diagram Example



(source: W. Mueller)

Unified Modeling Language (UML)

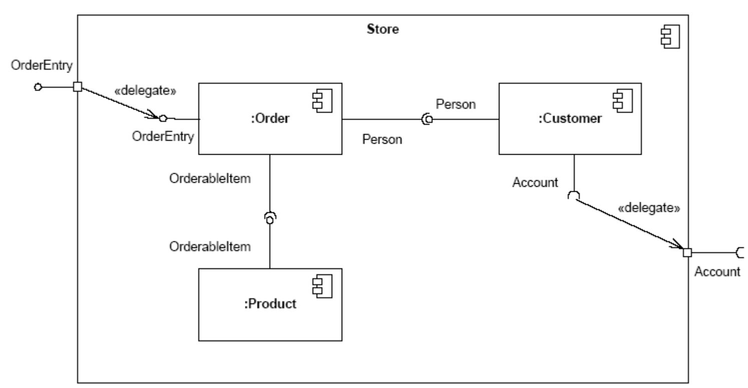
- Package Diagram Example



(source:
W. Mueller)

Unified Modeling Language (UML)

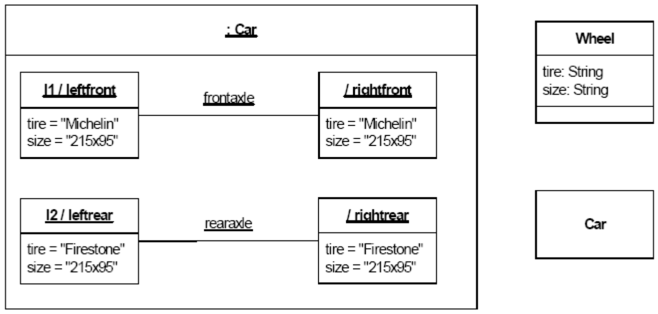
- Component Diagram Example



(source:
W. Mueller)

Unified Modeling Language (UML)

- Composite Structure Diagram Example

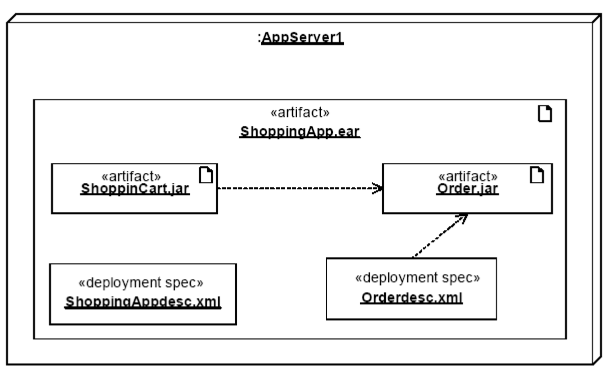


(source:
W. Mueller)

ECPS203: Embedded Systems Modeling and Design, Lecture 17 (c) 2018 R. Doemer 27

Unified Modeling Language (UML)

- Deployment Diagram Example

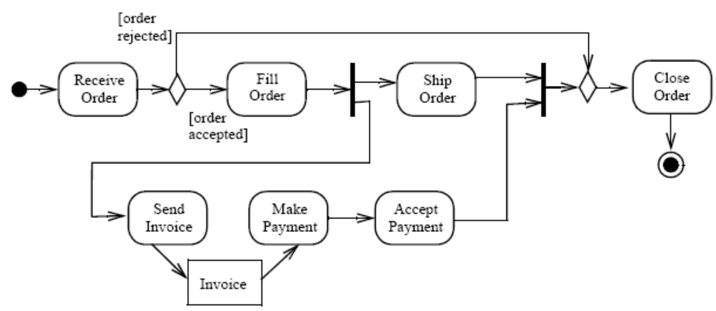


(source:
W. Mueller)

ECPS203: Embedded Systems Modeling and Design, Lecture 17 (c) 2018 R. Doemer 28

Unified Modeling Language (UML)

- Activity Diagram Example

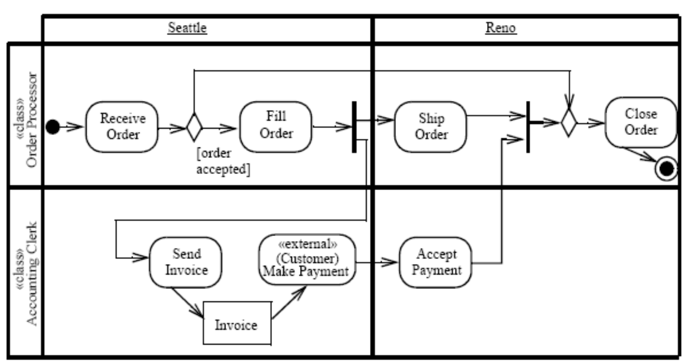


(source:
W. Mueller)

ECPS203: Embedded Systems Modeling and Design, Lecture 17 (c) 2018 R. Doemer 29

Unified Modeling Language (UML)

- Activity Diagram Example with “swim lanes”

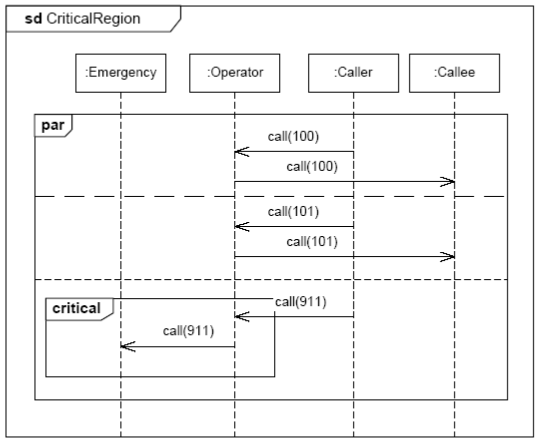


(source:
W. Mueller)

ECPS203: Embedded Systems Modeling and Design, Lecture 17 (c) 2018 R. Doemer 30

Unified Modeling Language (UML)

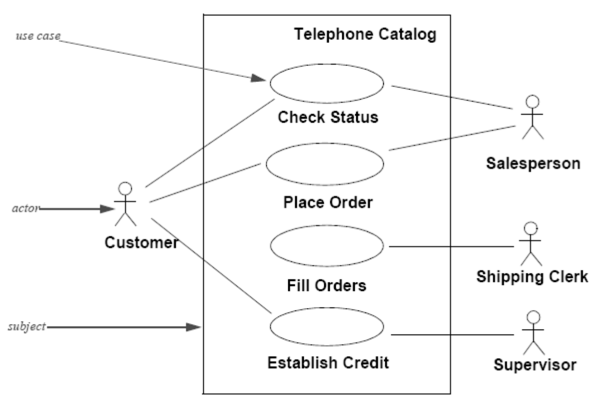
- Sequence Diagram Example



(source: W. Mueller)

Unified Modeling Language (UML)

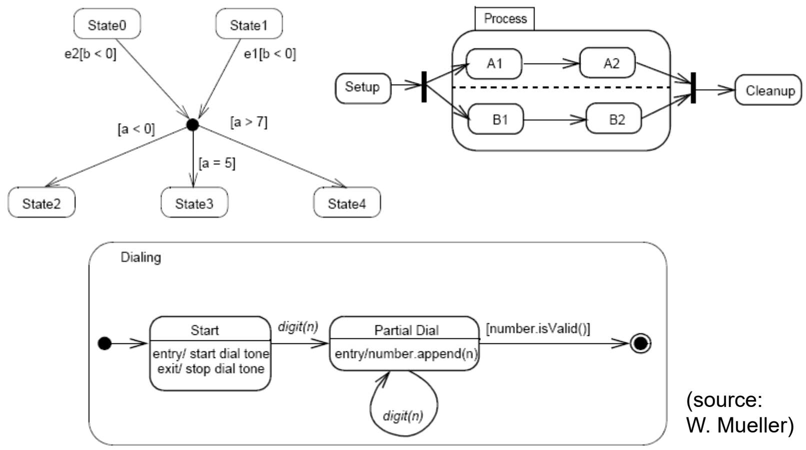
- Use Case Diagram Examples



(source: W. Mueller)

Unified Modeling Language (UML)

- State Machine Diagram Examples



(source: W. Mueller)