EECS 222A: System-on-Chip Description and Modeling Lecture 9

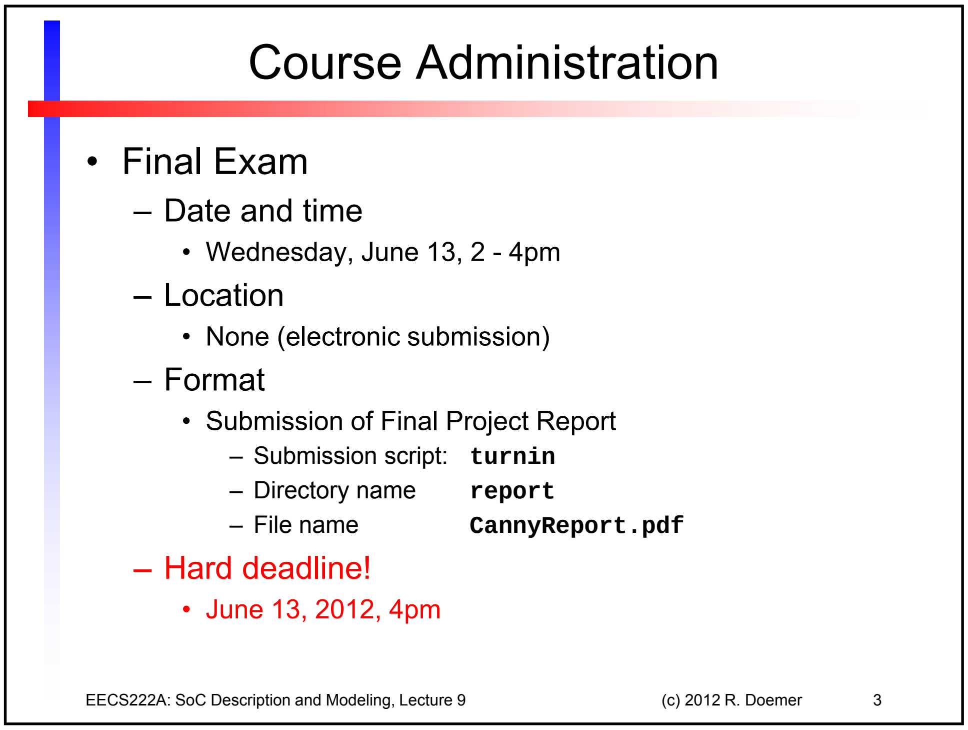
Rainer Dömer

doemer@uci.edu

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

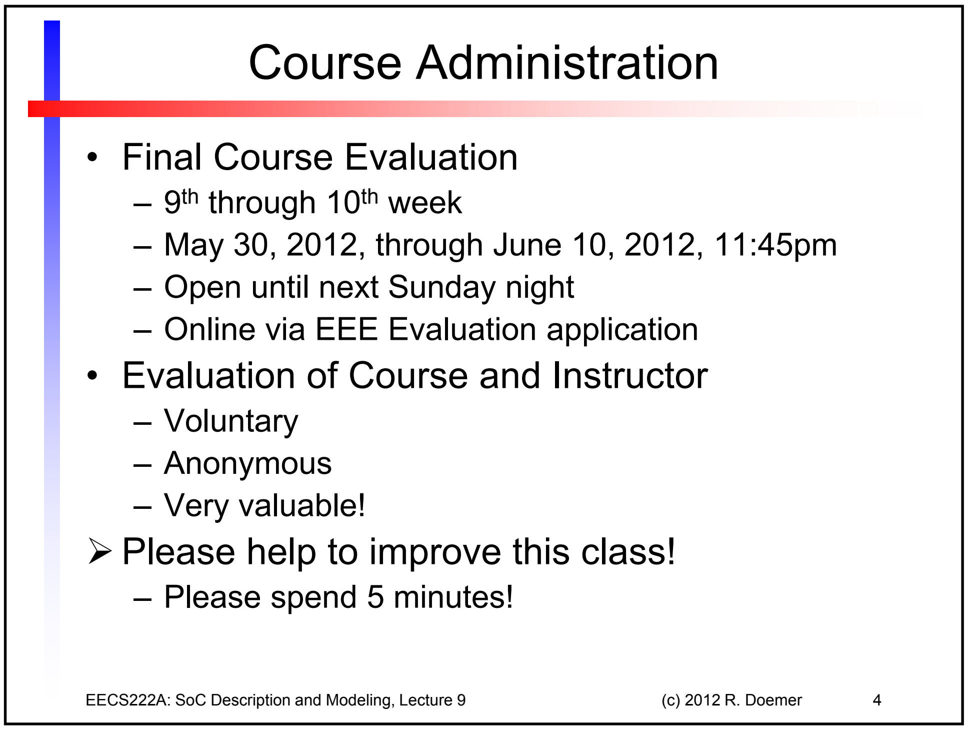
Lecture 9: Overview

- Course Administration
- Modeling with SystemC SLDL
  - Assignment 6: Producer/Consumer Example in SystemC
  - SystemC 2.0 Tutorial
    - Presentation by Thorsten Groetker, Synopsys
- *"Testbenches for Electronic System Level Design"*
  - Invited Presentation by
    Dr. Wolfgang Mueller, Paderborn University, Germany
- Project Discussion: Canny Edge Detector
  - Assignment 5
  - 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 next Sunday night
  - Online via EEE Evaluation application
- Evaluation of Course and Instructor
  - Voluntary
  - Anonymous
  - Very valuable!

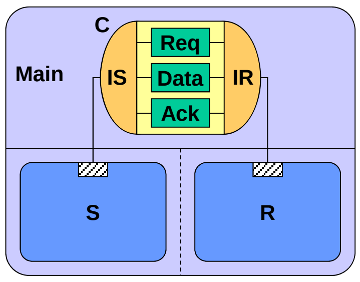
➢ Please help to improve this class!
  - Please spend 5 minutes!

Homework Assignment 6

- Producer/Consumer Example in SystemC
  - Review the FIFO example by Stuart Swan
    - See **`Lecture8_SystemC_Intro.pdf`**
  - Compile and simulate the example using SystemC
    - **`mkdir hw6; cd hw6`**
    - **`cp /opt/pkg/systemc-2.2.0/examples/sysc/simple_fifo/simple_fifo.cpp .`**
    - **`g++ simple_fifo.cpp -I/opt/pkg/systemc-2.2.0/include -L/opt/pkg/systemc-2.2.0/lib-linux –lsystemc -o simple_fifo`**
    - **`./simple_fifo`**
  - Translate the producer/consumer example from Assignment 1 to SystemC and simulate it
    - Reference: **`/home/eecs222/EECS222A_S12/ProdCons.sc`**
- Deliverables
  - Source file: **`SystemC_ProdCons.cpp`**
  - Simulation log: **`SystemC_ProdCons.log`**
- Due
  - By next week: June 1, 2012, 12pm (noon!)

Review: Producer/Consumer in SpecC

- Add behavior **`Main`**
- Add loop to **`S`**, name **`Prod`**
- Add loop to **`R`**, name **`Cons`**
- Compile and Simulate
- Create log file



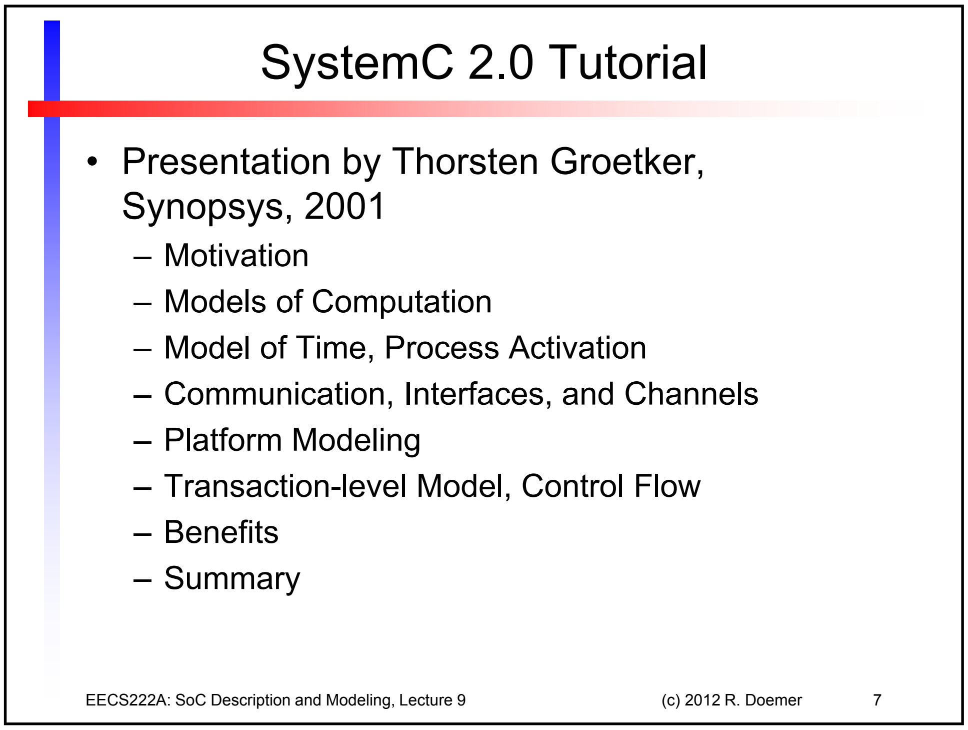
```
behavior S(IS Port)
{
 float X;
 void main(void)
 { ...
   Port.Send(X);
   ...
 }
};
```

```
behavior R(IR Port)
{
 float Y;
 void main(void)
 {...
  Y=Port.Receive();
  ...
 }
};
```

```
interface IS
{
 void Send(float);
};
interface IR
{
 float Receive(void);
};
```

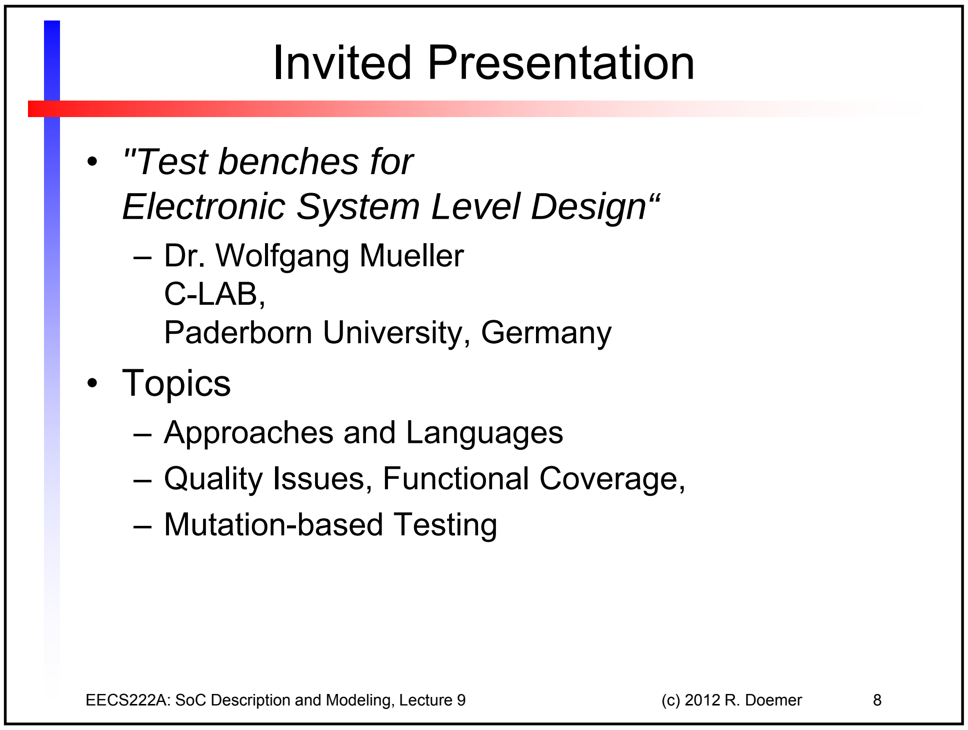
```
channel C
    implements IS, IR
{
 event Req;
 float Data;
 event Ack;

 void Send(float X)
 { Data = X;
   notify Req;
   wait Ack;
 }
 float Receive(void)
 { float Y;
   wait Req;
   Y = Data;
   notify Ack;
   return Y;
 }
};
```



SystemC 2.0 Tutorial

- Presentation by Thorsten Groetker, Synopsys, 2001
  - Motivation
  - Models of Computation
  - Model of Time, Process Activation
  - Communication, Interfaces, and Channels
  - Platform Modeling
  - Transaction-level Model, Control Flow
  - Benefits
  - Summary



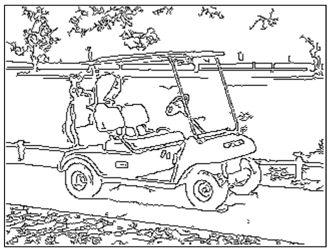
Invited Presentation

- *"Test benches for*
  Electronic System Level Design“
  - Dr. Wolfgang Mueller
    C-LAB,
    Paderborn University, Germany
- Topics
  - Approaches and Languages
  - Quality Issues, Functional Coverage,
  - Mutation-based Testing

Project Discussion

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

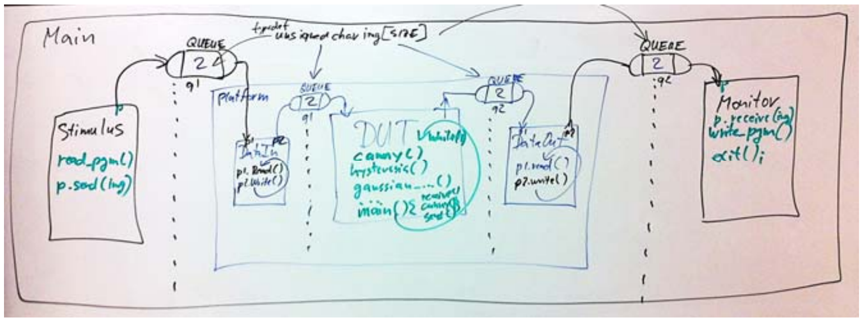
golfcart.pgm


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

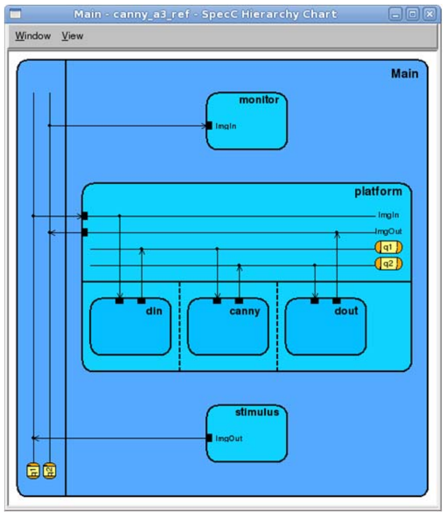
Project Discussion

- Structural Hierarchy for Canny Edge Detector
  - Test bench Structure



Project Discussion

- SCE Chart of Test bench Structure

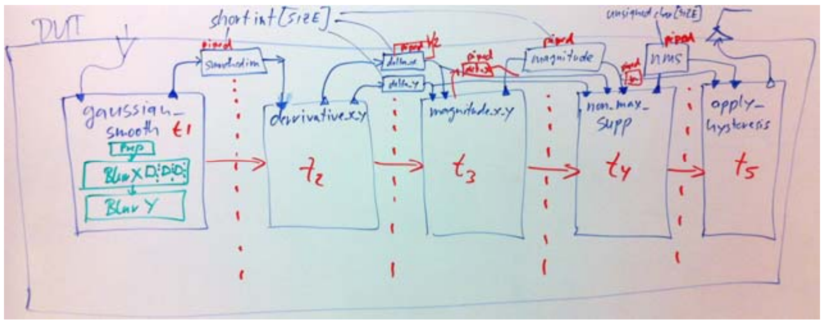


Project Discussion

- 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



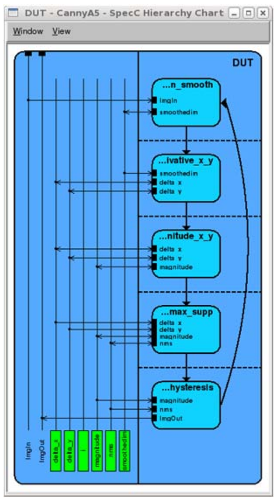
  - Potential for parallelism
    - 5 pipeline stages in DUT (red color)
    - Parallel decomposition of BlurX and BlurY blocks in Gaussian Smooth behavior (green color)

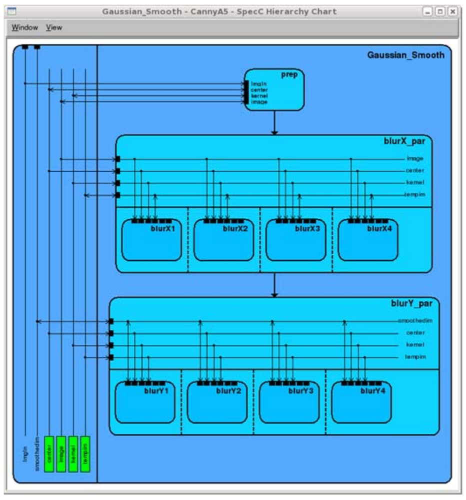
Project Discussion

- 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: Assignment 5

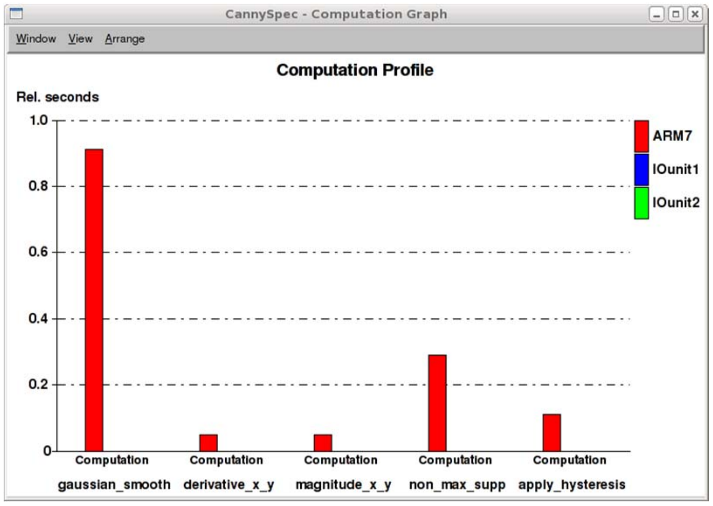
- Task: Refine the Canny Edge Detector using SCE
  - Setup: use latest **`sce`** to evaluate and refine
    - **`source /opt/sce/bin/setup.csh`**
  - Provided Files
    - canny_a5_start.sc, Makefile
    - golfcart.pgm, ref_golfcart.pgm_s_0.60_l_0.30_h_0.80.pgm
  - Tasks
    - Architecture and Scheduling Refinement (ARM7TDMI plus HW accelerators)
    - Network and Communication Refinement (AMBA_AHB plus DblHsk busses)
    - Instruction Level Model (cycle-accurate instruction-set simulation, optional)
- Deliverables
  - Source file: **`canny.tree`**
  - Description: **`canny.txt`**
- Due
  - By next week: May 25, 2012, 12pm (noon!)

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

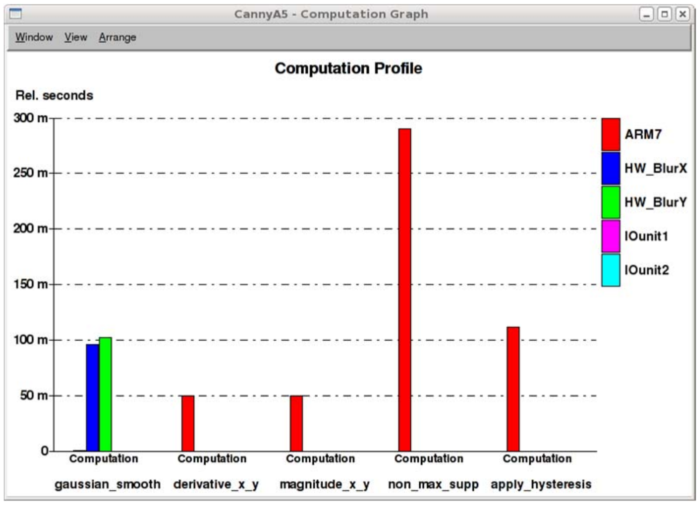
- Hardware Blocks for Gaussian Smooth
  - Allocate additional HW components
    - HW_BlurX, HW_BlurY of type HW_Standard for blurX_par, blurY_par
  - Estimation Results
    - 501.9 ms for Canny on ARM7
    - 4 x 23.9 ms for BlurX_par on BlurX
    - 4 x 26.5 ms for BlurY_par on BlurY
- Refinement using 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…

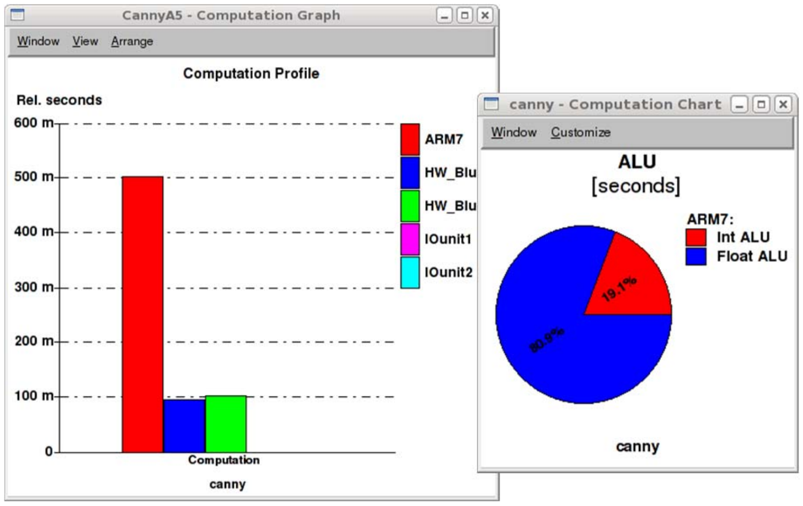
Project Discussion: Performance Optimization

- Computation Profile of DUT Pipeline



Project Discussion: Performance Optimization

- Computation Profile of DUT Operations



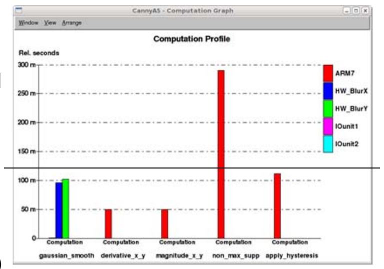
Project Discussion: Algorithm Recoding

- 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)

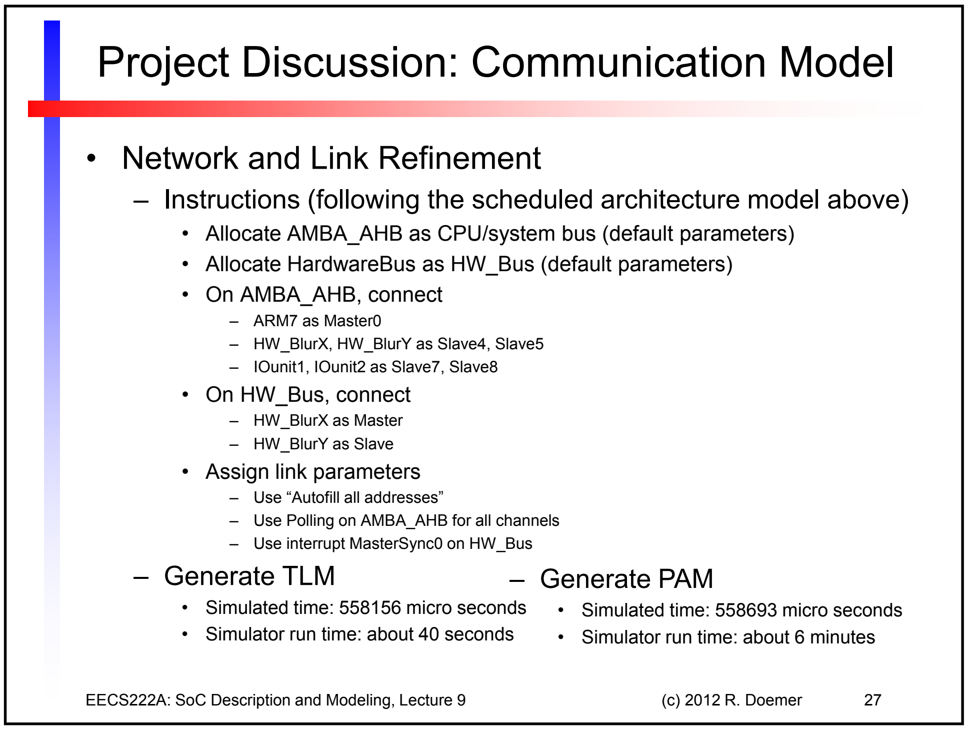


EECS222A: SoC Description and Modeling, Lecture 9 (c) 2012 R. Doemer 25

Project Discussion: Open Issues

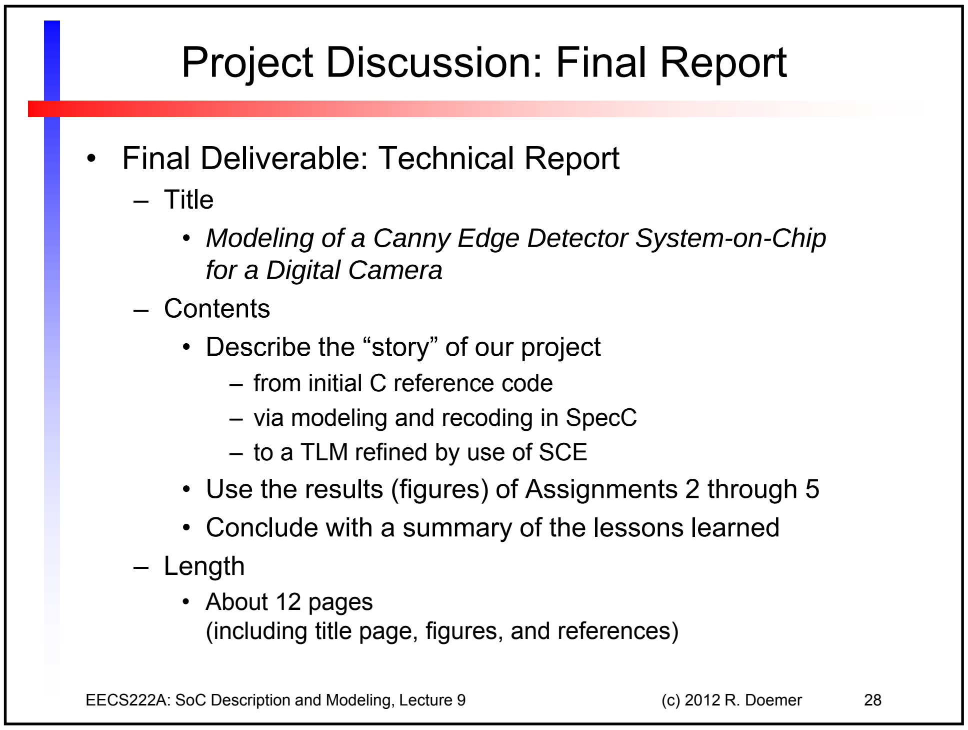
- Entire discussion so far is based on *estimated* values!
  - Estimation by table-based retargetable profiling
  - Typically no absolute accuracy, only *fidelity*!
- Communication delays are not considered
  - Scheduled architecture model assumes 0 communication delay
  - Network and Link Refinement 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!

EECS222A: SoC Description and Modeling, Lecture 9 (c) 2012 R. Doemer 26



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)