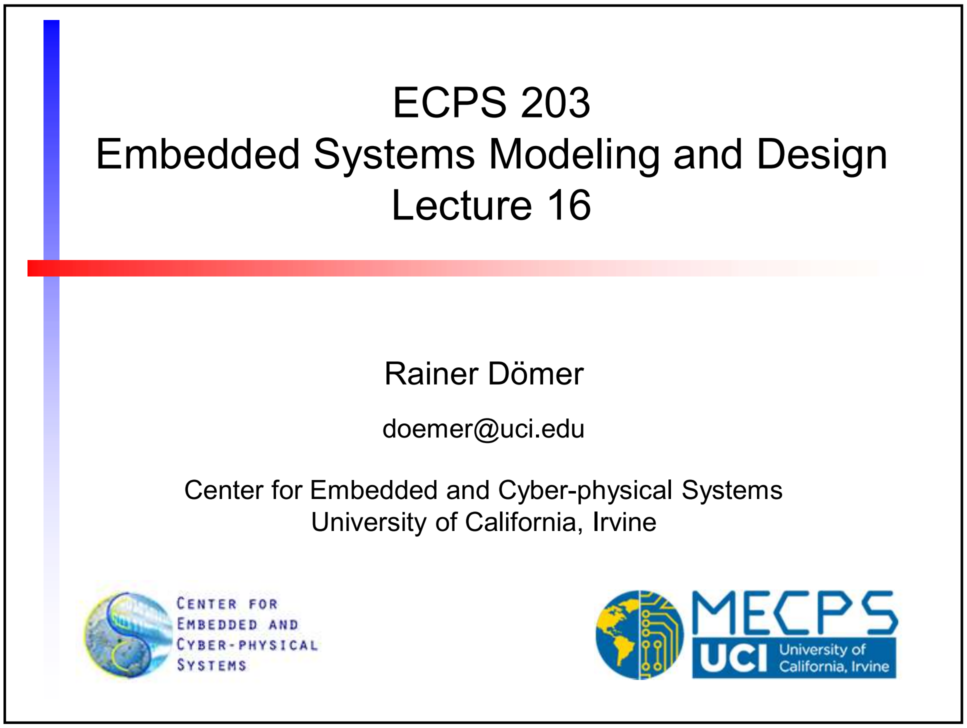
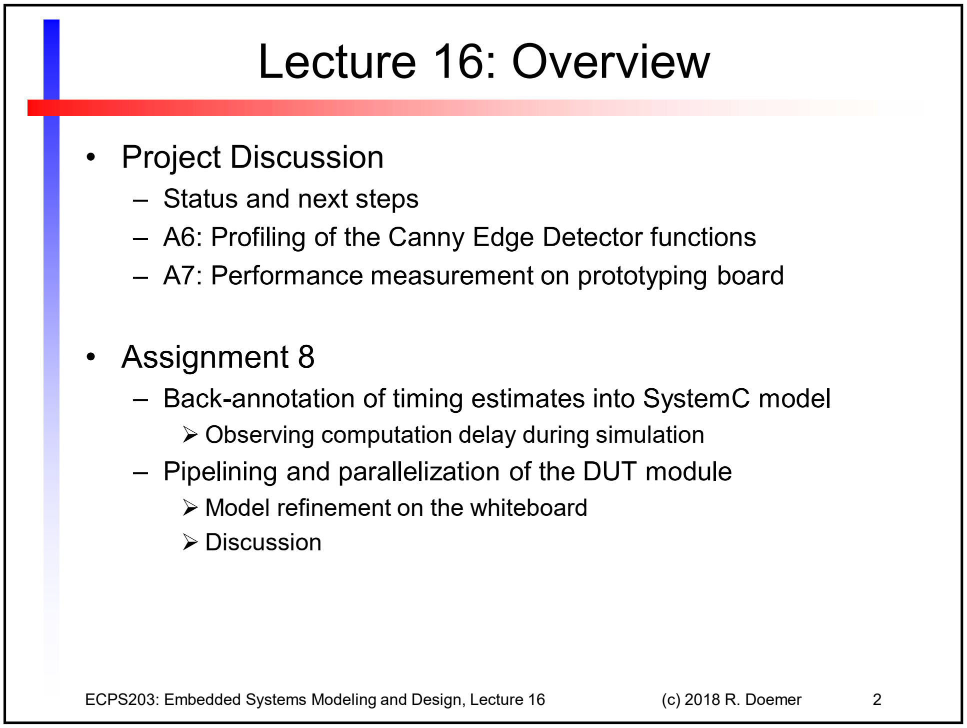


ECPS 203
Embedded Systems Modeling and Design
Lecture 16

Rainer Dömer

doemer@uci.edu

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

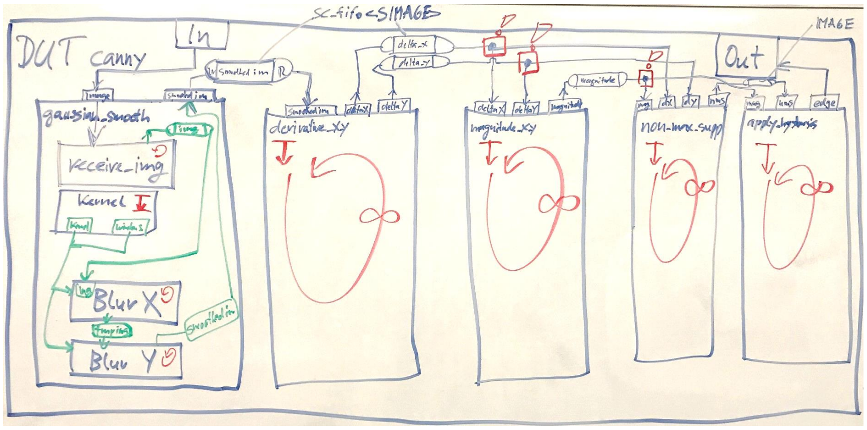


Lecture 16: Overview

- Project Discussion
  - Status and next steps
  - A6: Profiling of the Canny Edge Detector functions
  - A7: Performance measurement on prototyping board
- Assignment 8
  - Back-annotation of timing estimates into SystemC model
    - ➢ Observing computation delay during simulation
  - Pipelining and parallelization of the DUT module
    - ➢ Model refinement on the whiteboard
    - ➢ Discussion

Project Assignment 6

- Task: Structural model of the Canny Edge Detector
  - Discussion on whiteboard: Refined DUT structure

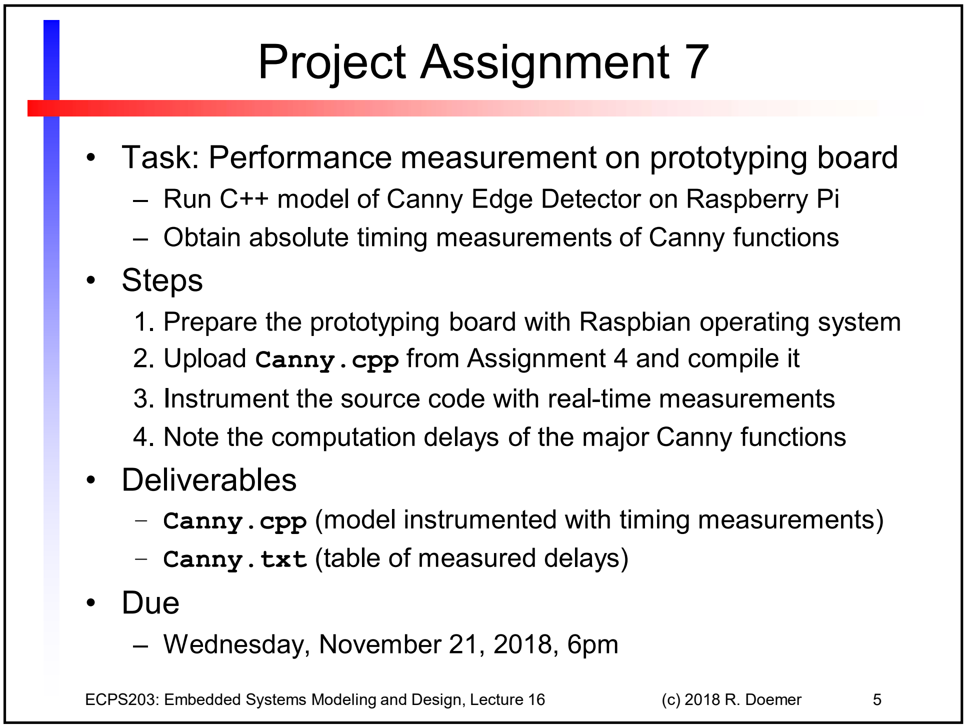


Project Assignment 6

- Step 3: Profile the Canny functions,
  obtain relative computational complexity
  - Profiled complexity comparison (in **Canny.txt**):

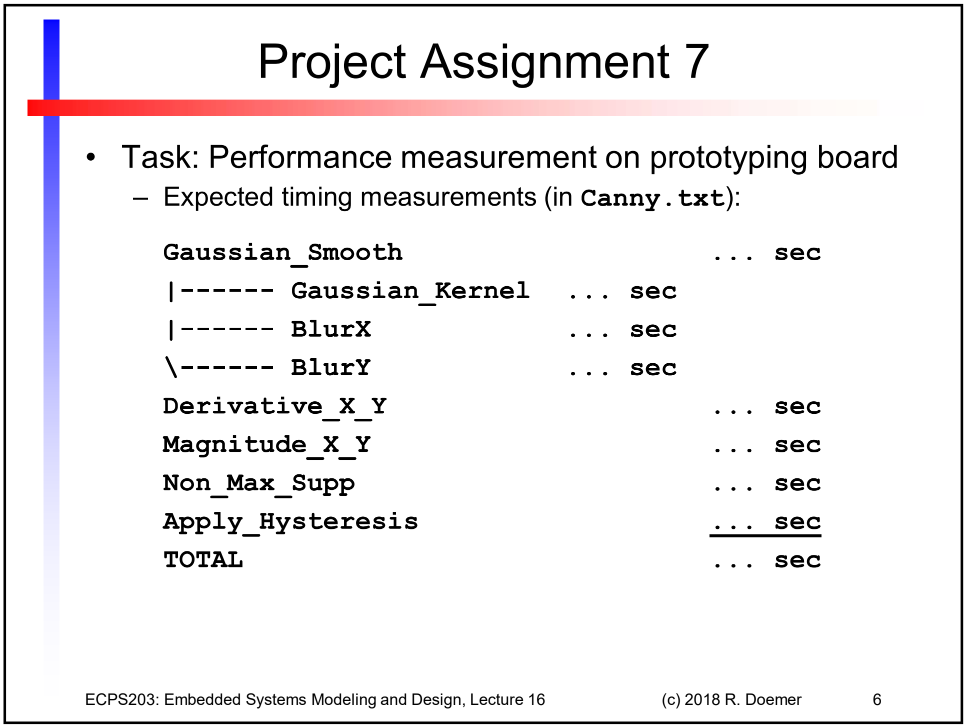
```
Gaussian_Smooth                          40.57%
|------ Gaussian_Kernel    0.00%
|------ BlurX             17.23%
\------ BlurY             23.34%
Derivative_X_Y                            6.26%
Magnitude_X_Y                            15.90%
Non_Max_Supp                             23.98%
Apply_Hysteresis                         12.29%
                                       100%
```

➢ Profiling results vary, but Gaussian Smooth is a bottleneck!



Project Assignment 7

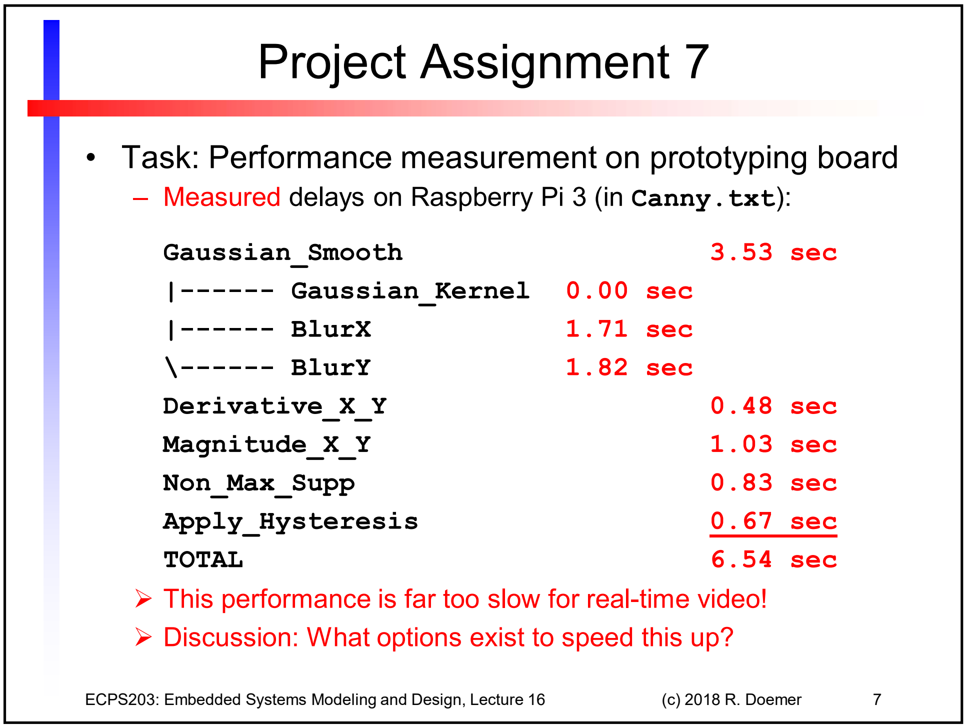
- Task: Performance measurement on prototyping board
  - Run C++ model of Canny Edge Detector on Raspberry Pi
  - Obtain absolute timing measurements of Canny functions
- Steps
  1. Prepare the prototyping board with Raspbian operating system
  2. Upload **`Canny.cpp`** from Assignment 4 and compile it
  3. Instrument the source code with real-time measurements
  4. Note the computation delays of the major Canny functions
- Deliverables
  - **`Canny.cpp`** (model instrumented with timing measurements)
  - **`Canny.txt`** (table of measured delays)
- Due
  - Wednesday, November 21, 2018, 6pm



Project Assignment 7

- Task: Performance measurement on prototyping board
  - Expected timing measurements (in **`Canny.txt`**):

```
Gaussian_Smooth                     ... sec
|------ Gaussian_Kernel   ... sec
|------ BlurX             ... sec
\------ BlurY             ... sec
Derivative_X_Y                      ... sec
Magnitude_X_Y                       ... sec
Non_Max_Supp                        ... sec
Apply_Hysteresis                    ... sec
TOTAL                               ... sec
```

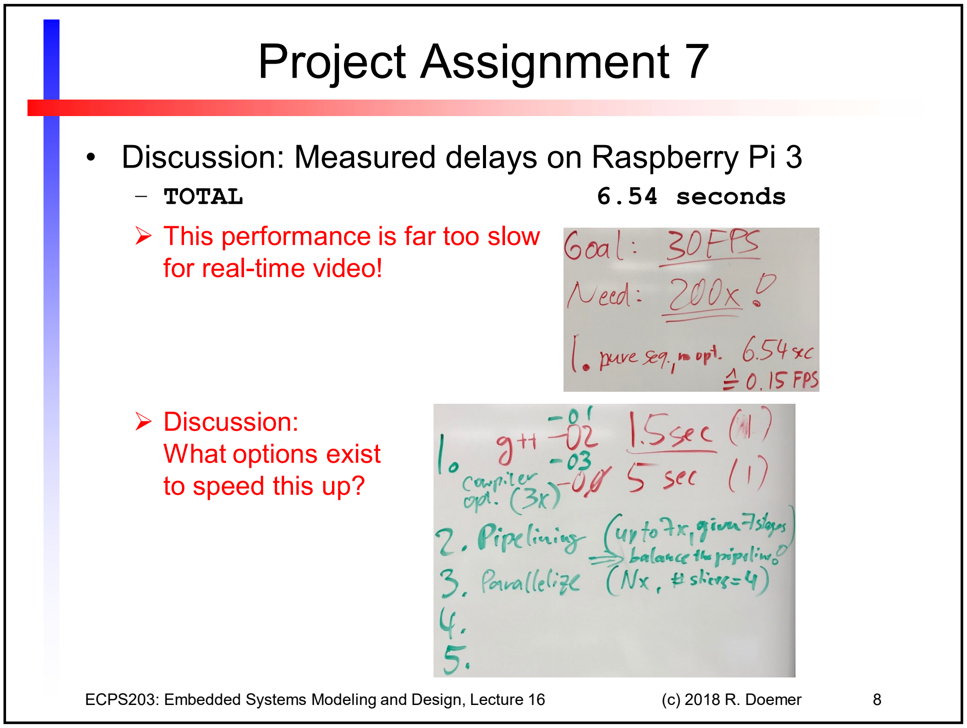


Project Assignment 7

- 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?



Project Assignment 7

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

Goal: 30FPS

Need: 200x !

1. pure seq., no opt. 6.54 sec ≙ 0.15 FPS

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

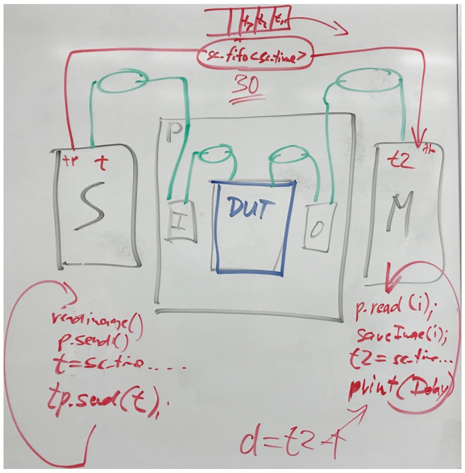
1. g++ -01 -02 -03 -0~~0~~ 1.5sec (1) 5 sec (1)
   compiler opt. (3x)
2. Pipelining (up to 7x, given 7 stages) ⟹ balance the pipeline!
3. Parallelize (Nx, # slices=4)
4.
5.

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

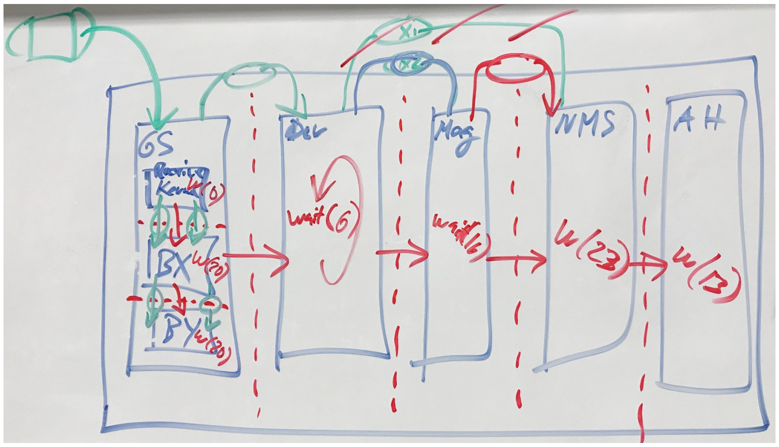
Project Assignment 8

- Timed test bench model for the Canny Edge Detector
  - Discussion on whiteboard: Chart of refined test bench structure



Project Assignment 8

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



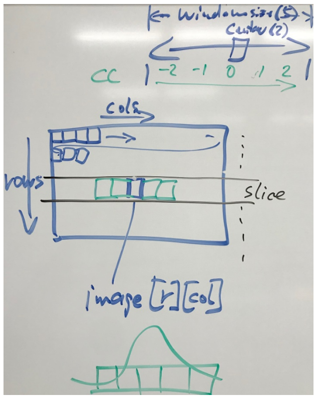
Project Assignment 8

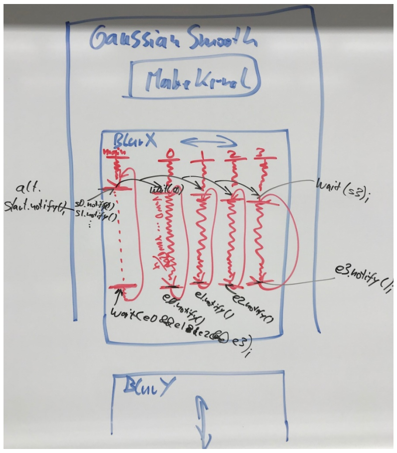
- Pipelined and parallel model of the Canny Edge Detector
  - Back-annotation of measured timing delays (step 2)

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

Throughput:            1/1820ms
                       0.549 FPS
```

Project Assignment 8

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

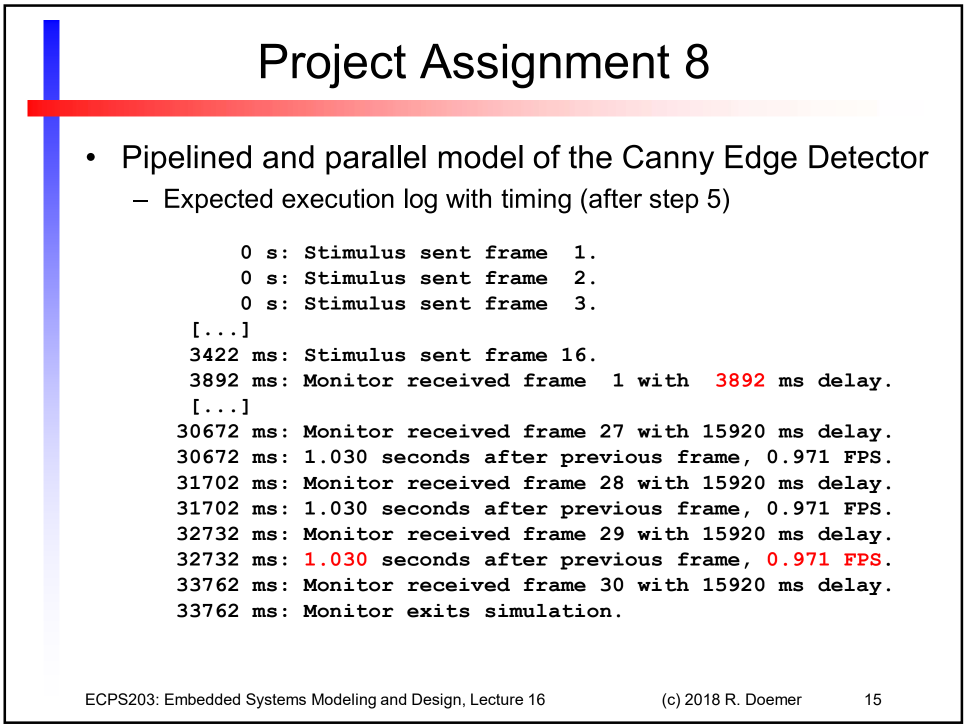




Project Assignment 8

- 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)

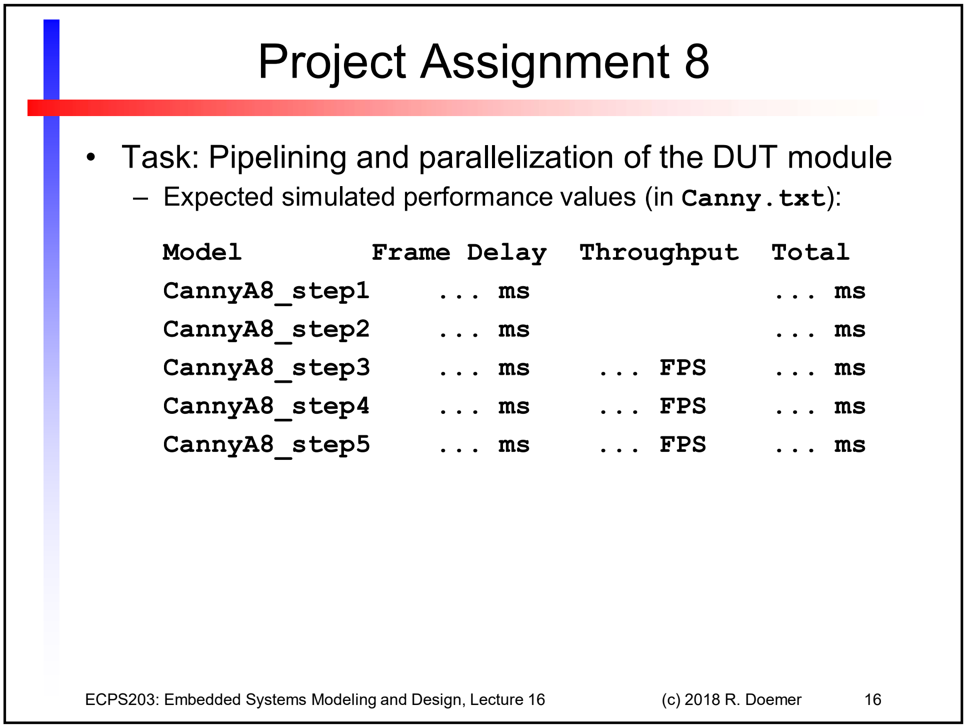
```
Receive, Make_Kernel      0 ms          0 ms
BlurX                  1710 ms        427 ms
BlurY                  1820 ms        455 ms
Derivative_X_Y          480 ms        480 ms
Magnitude_X_Y          1030 ms       1030 ms
Non_Max_Supp            830 ms        830 ms
Apply_Hysteresis        670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