EECS 222C: System-on-Chip Software Synthesis Lecture 8

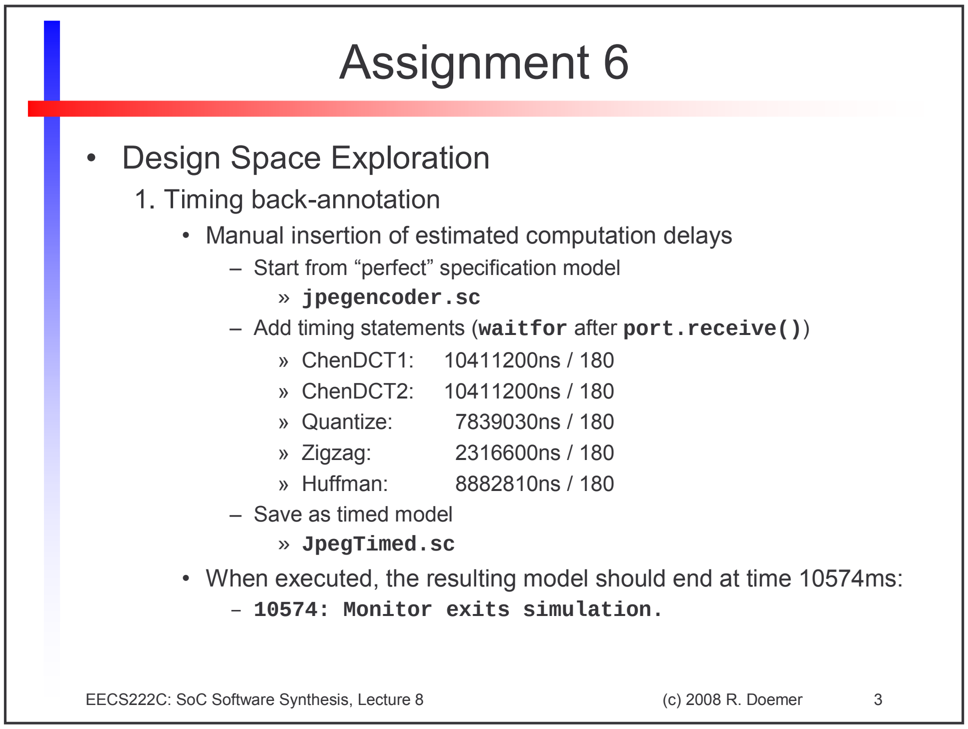
Rainer Dömer

doemer@uci.edu

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

Lecture 8: Overview

- Project Discussion
  - Assignment 6
    - Architecture Exploration
    - Scheduling Exploration
  - Exploration results
    - Trade-offs between Cost and Speed
    - Limitations
- Invited Guest Lecture
  - Systematic Generation of Embedded Software from High-level Models
- Assignment 7



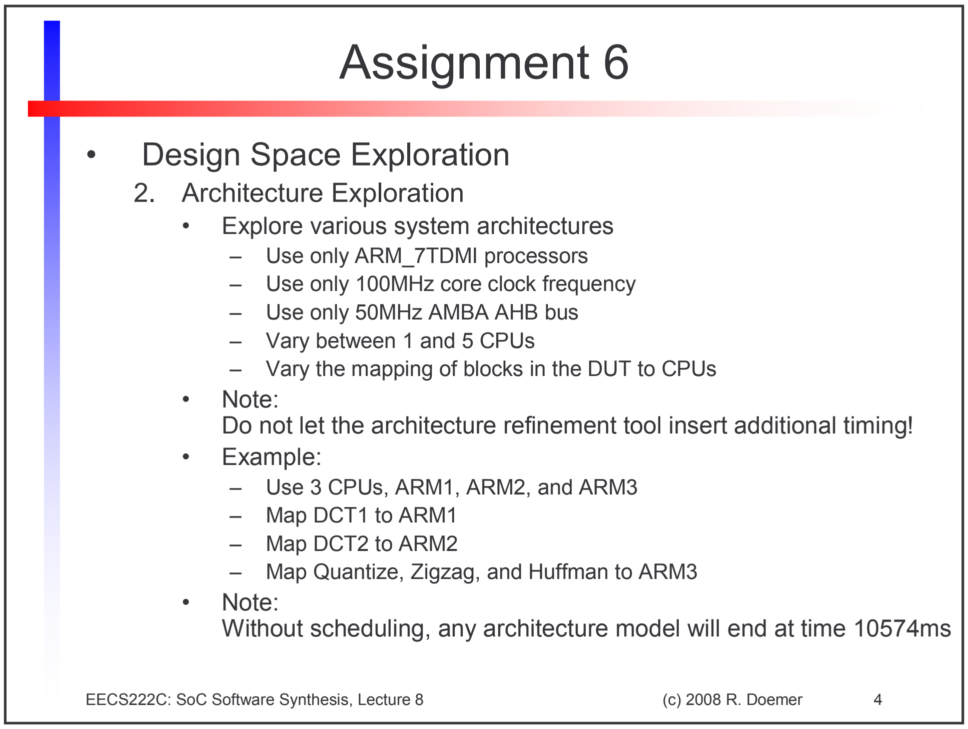
Assignment 6

- Design Space Exploration
  1. Timing back-annotation
     - Manual insertion of estimated computation delays
       - Start from “perfect” specification model
         - » **`jpegencoder.sc`**
       - Add timing statements (**`waitfor`** after **`port.receive()`**)

» ChenDCT1:	10411200ns / 180
» ChenDCT2:	10411200ns / 180
» Quantize:	7839030ns / 180
» Zigzag:	2316600ns / 180
» Huffman:	8882810ns / 180

       - Save as timed model
         - » **`JpegTimed.sc`**
     - When executed, the resulting model should end at time 10574ms:
       - **`10574: Monitor exits simulation.`**

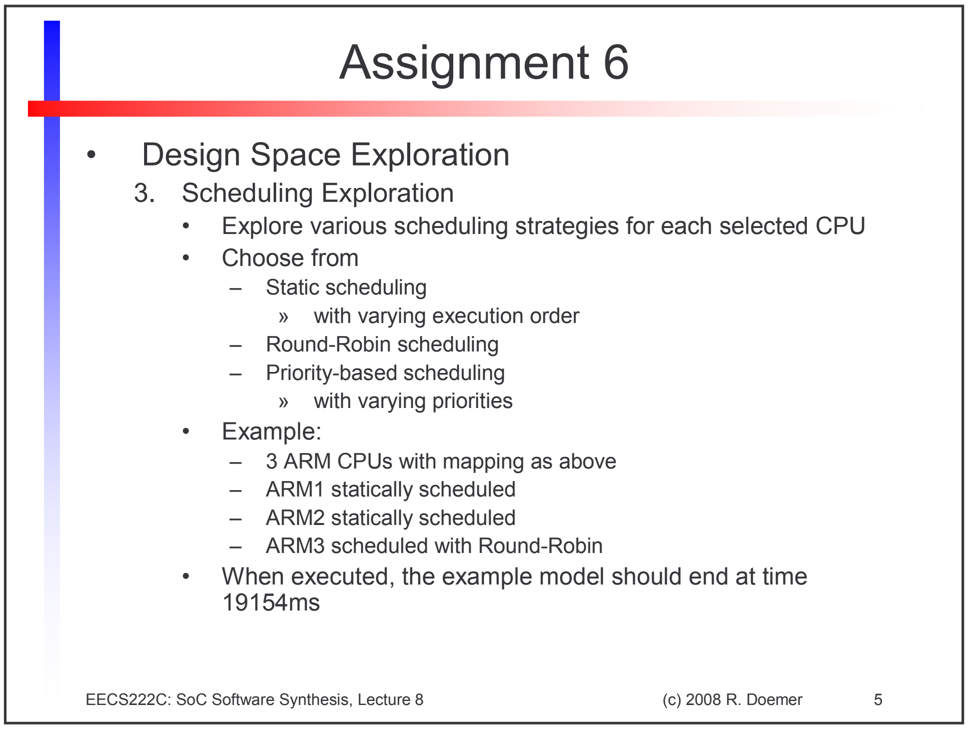
EECS222C: SoC Software Synthesis, Lecture 8 (c) 2008 R. Doemer 3



Assignment 6

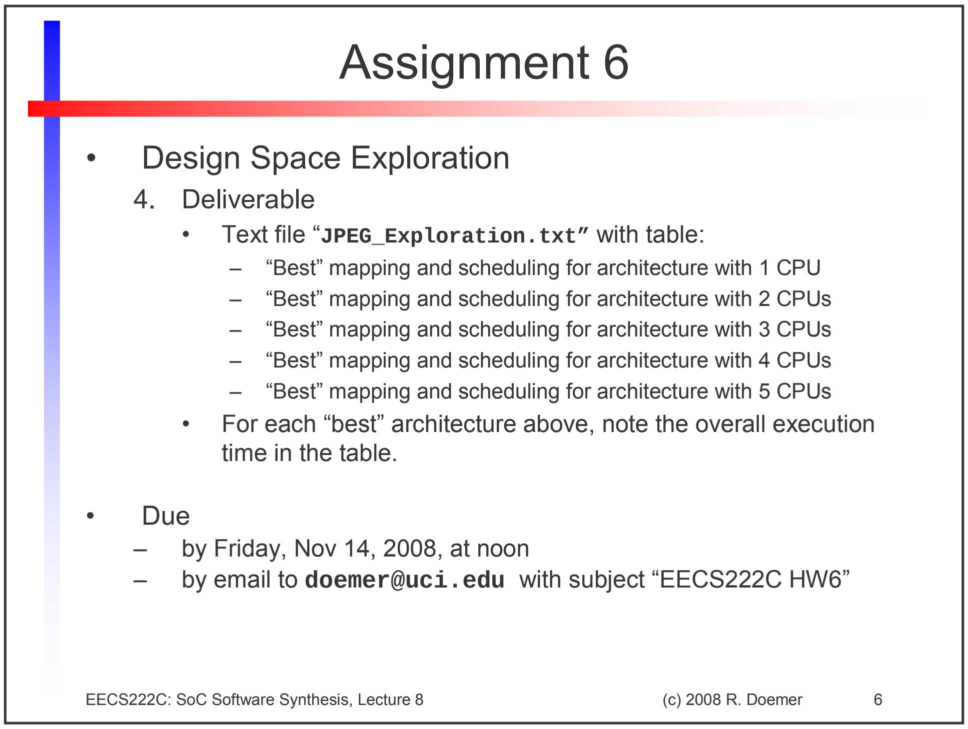
- Design Space Exploration
  2. Architecture Exploration
     - Explore various system architectures
       - Use only ARM_7TDMI processors
       - Use only 100MHz core clock frequency
       - Use only 50MHz AMBA AHB bus
       - Vary between 1 and 5 CPUs
       - Vary the mapping of blocks in the DUT to CPUs
     - Note:
       Do not let the architecture refinement tool insert additional timing!
     - Example:
       - Use 3 CPUs, ARM1, ARM2, and ARM3
       - Map DCT1 to ARM1
       - Map DCT2 to ARM2
       - Map Quantize, Zigzag, and Huffman to ARM3
     - Note:
       Without scheduling, any architecture model will end at time 10574ms

EECS222C: SoC Software Synthesis, Lecture 8 (c) 2008 R. Doemer 4



Assignment 6

- Design Space Exploration
  3. Scheduling Exploration
     - Explore various scheduling strategies for each selected CPU
     - Choose from
       - Static scheduling
         - » with varying execution order
       - Round-Robin scheduling
       - Priority-based scheduling
         - » with varying priorities
     - Example:
       - 3 ARM CPUs with mapping as above
       - ARM1 statically scheduled
       - ARM2 statically scheduled
       - ARM3 scheduled with Round-Robin
     - When executed, the example model should end at time 19154ms

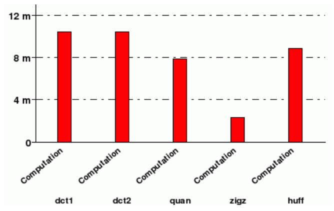


Assignment 6

- Design Space Exploration
  4. Deliverable
     - Text file “**`JPEG_Exploration.txt`**” with table:
       - “Best” mapping and scheduling for architecture with 1 CPU
       - “Best” mapping and scheduling for architecture with 2 CPUs
       - “Best” mapping and scheduling for architecture with 3 CPUs
       - “Best” mapping and scheduling for architecture with 4 CPUs
       - “Best” mapping and scheduling for architecture with 5 CPUs
     - For each “best” architecture above, note the overall execution time in the table.
- Due
  - by Friday, Nov 14, 2008, at noon
  - by email to **`doemer@uci.edu`** with subject “EECS222C HW6”

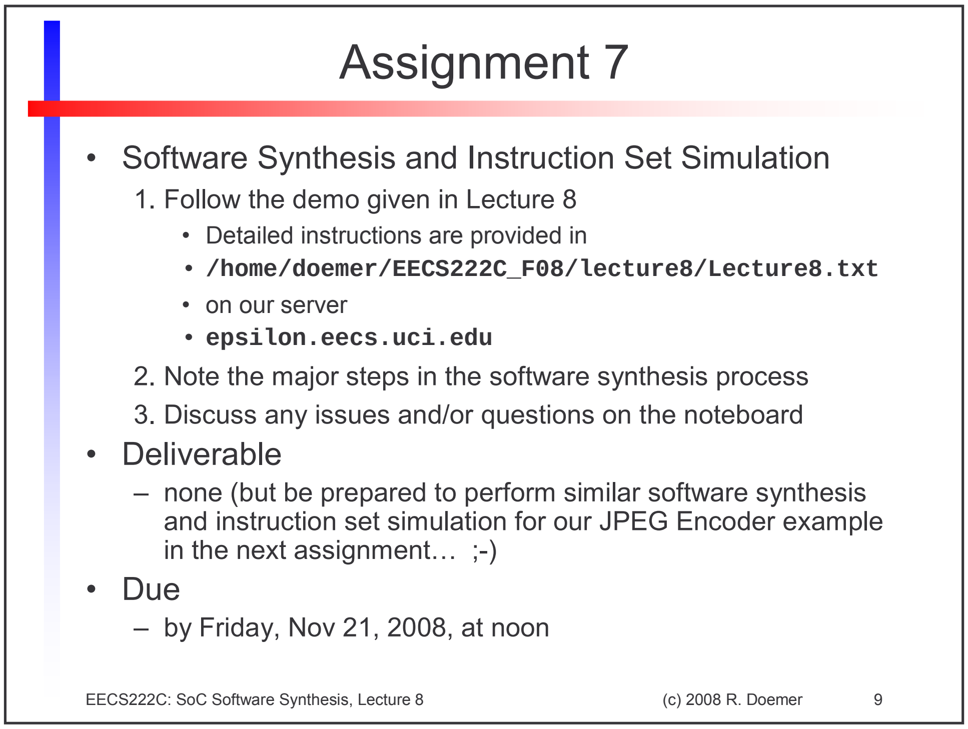
Project Discussion

- Design Space Exploration
  - Estimation results
    - For ARM_7TDMI CPU at 100 MHz
    - For encoding of 180 blocks
      - ChenDCT1: 10.41ms (10411200ns / 180 ≈ 58us)
      - ChenDCT2: 10.41ms (10411200ns / 180 ≈ 58us)
      - Quantize: 7.84ms (7839030ns / 180 ≈ 44us)
      - Zigzag: 2.32ms (2316600ns / 180 ≈ 13us)
      - Huffman: 8.88ms (8882810ns / 180 ≈ 49us)
        - » Sum: 39.86ms (per block ≈ 221us)
  - Exploration results (in-class discussion)
    - Trade-offs between Cost and Speed
    - Limitations



Invited Guest Lecture

- ***"Systematic Generation of Embedded Software from High-level Models"***
- Speaker:
  - **Dr. Gunar Schirner**
    Center of Embedded Computer Systems
    UC Irvine
- Abstract:
  - This talk presents a systematic approach to automatically generate embedded software from an abstract system model. The software generation encompasses RTOS-based multi-tasking, driver generation for external and internal communication, and assembly of the final target binary. The presentation will conclude with a live demonstration that synthesizes embedded software for an example from the automotive domain.



Assignment 7

- Software Synthesis and Instruction Set Simulation
  1. Follow the demo given in Lecture 8
     - Detailed instructions are provided in
     - **`/home/doemer/EECS222C_F08/lecture8/Lecture8.txt`**
     - on our server
     - **`epsilon.eecs.uci.edu`**
  2. Note the major steps in the software synthesis process
  3. Discuss any issues and/or questions on the noteboard
- Deliverable
  – none (but be prepared to perform similar software synthesis and instruction set simulation for our JPEG Encoder example in the next assignment… ;-)
- Due
  – by Friday, Nov 21, 2008, at noon

EECS222C: SoC Software Synthesis, Lecture 8 (c) 2008 R. Doemer 9