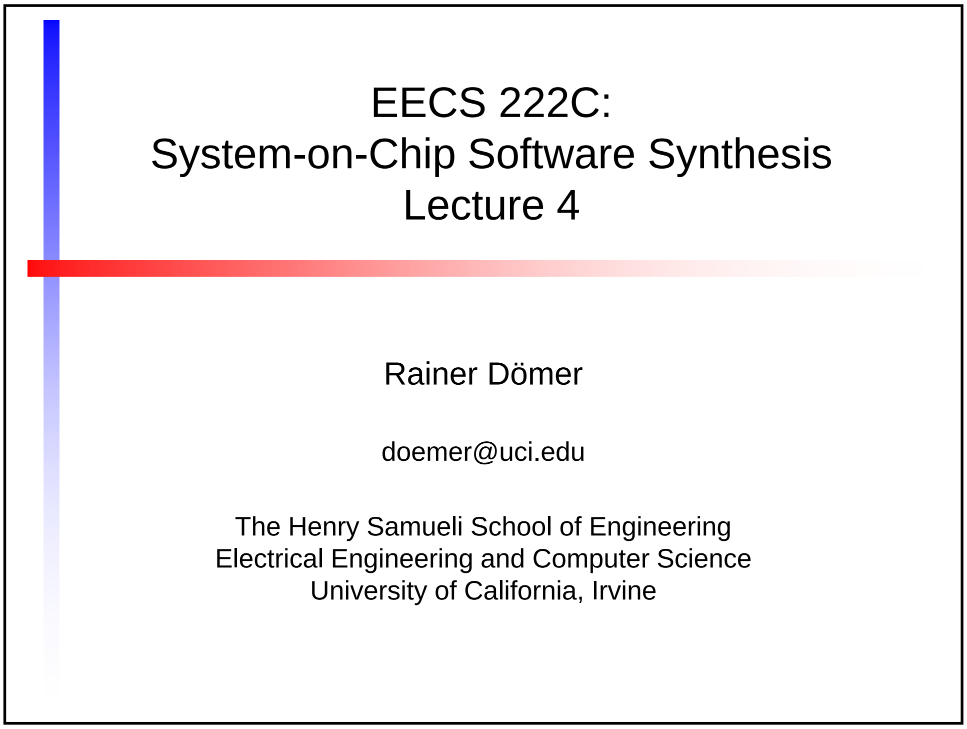
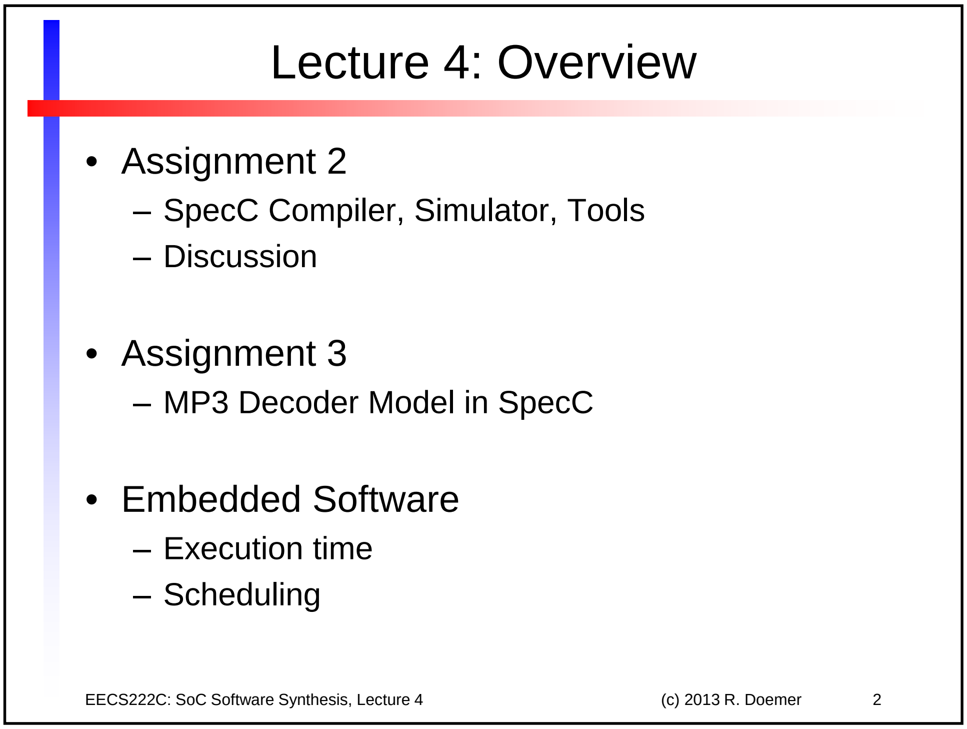


EECS 222C:
System-on-Chip Software Synthesis
Lecture 4

Rainer Dömer

doemer@uci.edu

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



Lecture 4: Overview

- Assignment 2
  - SpecC Compiler, Simulator, Tools
  - Discussion
- Assignment 3
  - MP3 Decoder Model in SpecC
- Embedded Software
  - Execution time
  - Scheduling

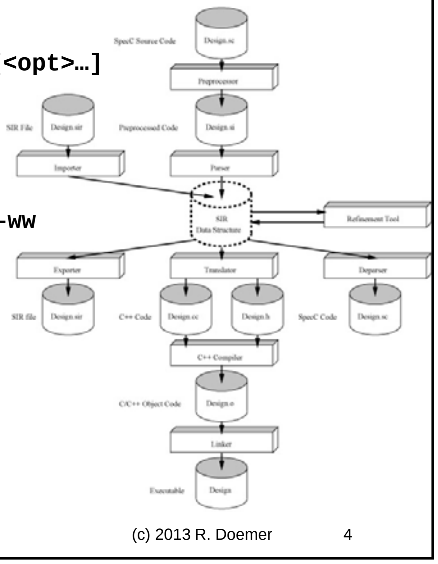
Assignment 2

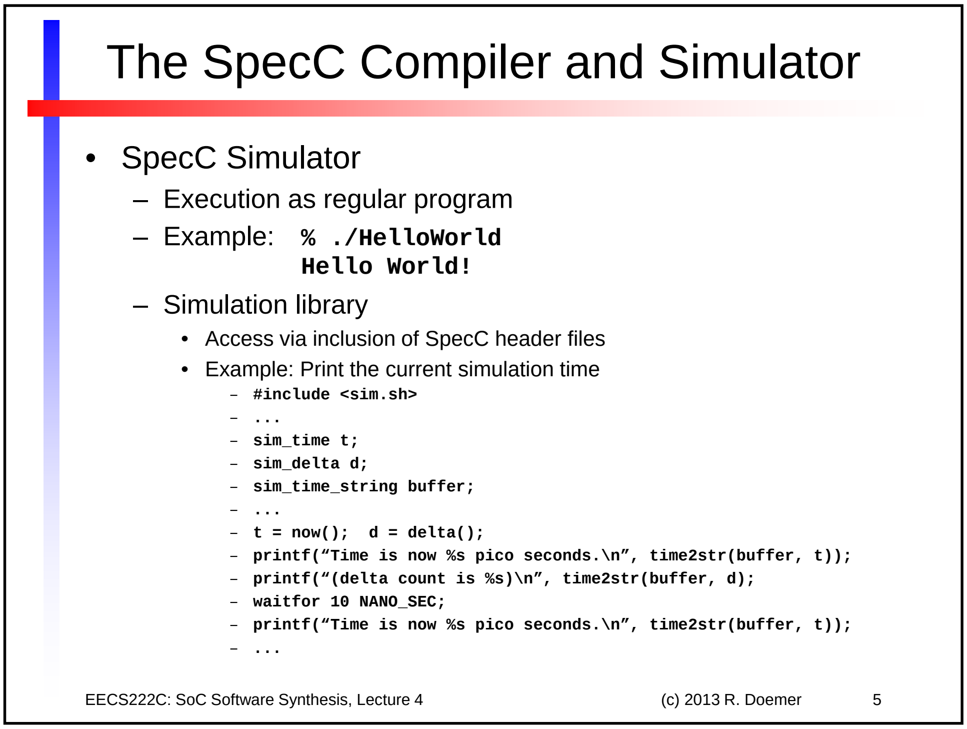
1. Practice the use of SpecC Command Line Tools
   - Setup
     - `source /opt/sce-20100908/bin/setup.csh`
   - Examine simple examples
     - `mkdir simple_tests`
     - `cd simple_tests`
     - `cp $SPECC/examples/simple/* .`
     - `ls`
     - `vi HelloWorld.sc`
   - Practice the compiler
     - `man scc`
     - `scc HelloWorld –sc2out –vv -ww`
   - Practice the simulator
     - `./HelloWorld`
   - Practice the tools
     - `man sir_tree`
     - `scc Adder -sc2sir -o Adder.sir`
     - `sir_tree -bt Adder.sir FA`

The SpecC Compiler and Simulator

- SpecC Compiler
  - Command line interface
  - Usage: `scc <design> [<cmd>] [<opt>…]`
  - Help: `scc –h`
    `man scc`
  - Example:

```
% scc HelloWorld –sc2out –v –ww
scc: SpecC Compiler V 2.2.1
(c)2010 CECS, UC Irvine
Preprocessing...
Parsing...
Translating...
Compiling...
Linking…
Done.
```

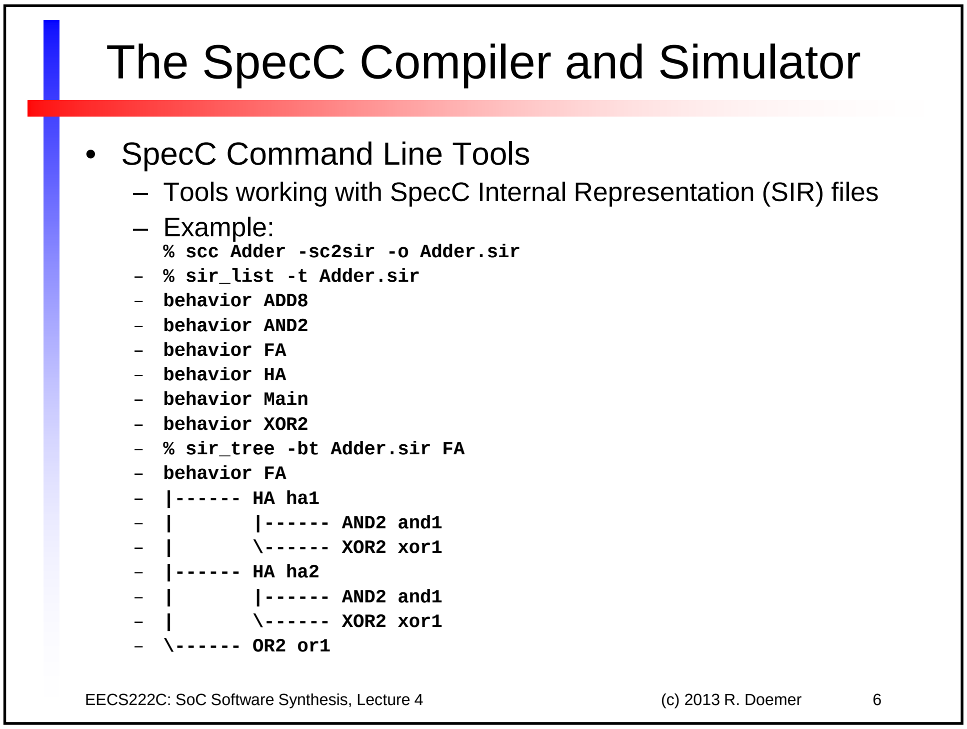




The SpecC Compiler and Simulator

- SpecC Simulator
  - Execution as regular program
  - Example:
    ```
    % ./HelloWorld
    Hello World!
    ```
  - Simulation library
    - Access via inclusion of SpecC header files
    - Example: Print the current simulation time

```
#include <sim.sh>
...
sim_time t;
sim_delta d;
sim_time_string buffer;
...
t = now();  d = delta();
printf(“Time is now %s pico seconds.\n”, time2str(buffer, t));
printf(“(delta count is %s)\n”, time2str(buffer, d);
waitfor 10 NANO_SEC;
printf(“Time is now %s pico seconds.\n”, time2str(buffer, t));
...
```



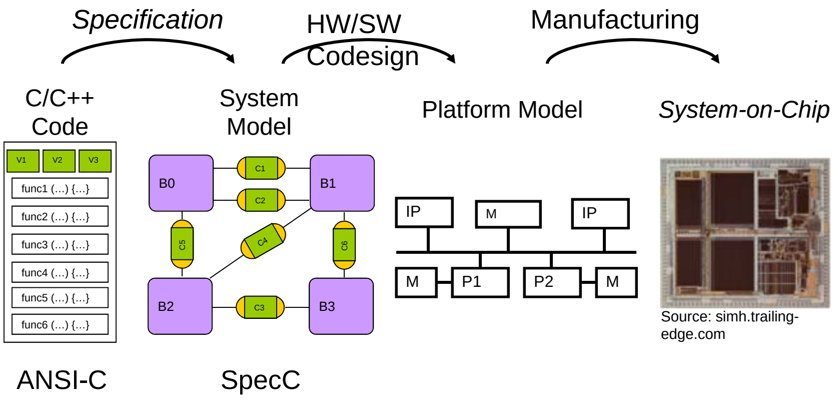
The SpecC Compiler and Simulator

- SpecC Command Line Tools
  - Tools working with SpecC Internal Representation (SIR) files
  - Example:

```
% scc Adder -sc2sir -o Adder.sir
% sir_list -t Adder.sir
behavior ADD8
behavior AND2
behavior FA
behavior HA
behavior Main
behavior XOR2
% sir_tree -bt Adder.sir FA
behavior FA
|------ HA ha1
|       |------ AND2 and1
|       \------ XOR2 xor1
|------ HA ha2
|       |------ AND2 and1
|       \------ XOR2 xor1
\------ OR2 or1
```

System-on-Chip Co-Design Flow

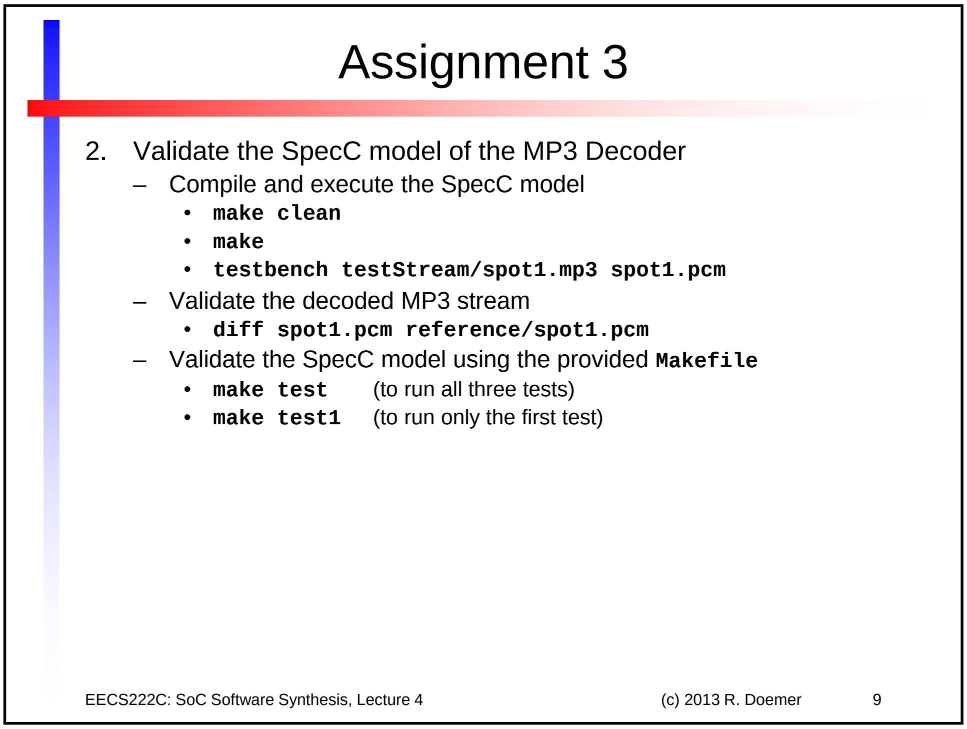
- Application Case Study, Project Status:
  - Given: Reference source code (**`mad_C.tar.gz`**)
  - Next: Specification of System Model (**`mad_SpecC.tar.gz`**)



Source: simh.trailing-edge.com

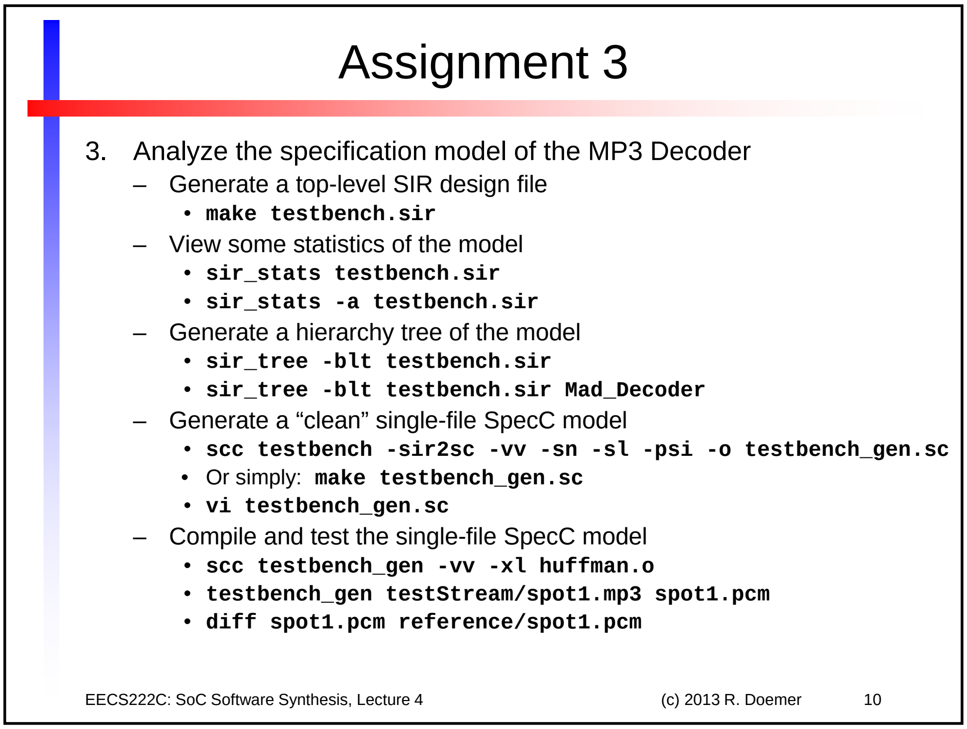
Assignment 3

1. Install a SpecC model of the MP3 Decoder
   - Setup and unpack source code
     - **`source /opt/sce-20100908/bin/setup.csh`**
     - **`cd hw3`**
     - **`gtar xvzf ~eecs222/EECS222C_S13/mad_SpecC.tar.gz`**
     - **`ls`**
   - Reuse test streams from original C code as “golden” reference streams
     - **`ln -s ../hw1/mad_C/testStream`**
     - **`mkdir reference`**
     - **`cp ../hw1/mad_C/spot1.pcm reference/`**
     - **`cp ../hw1/mad_C/spot1_3K.pcm reference/`**
     - **`cp ../hw1/mad_C/classic1.pcm reference/`**
     - **`vi Makefile`**
       - ➢ **`TESTSTREAMS = spot1_3K classic1 spot1`**



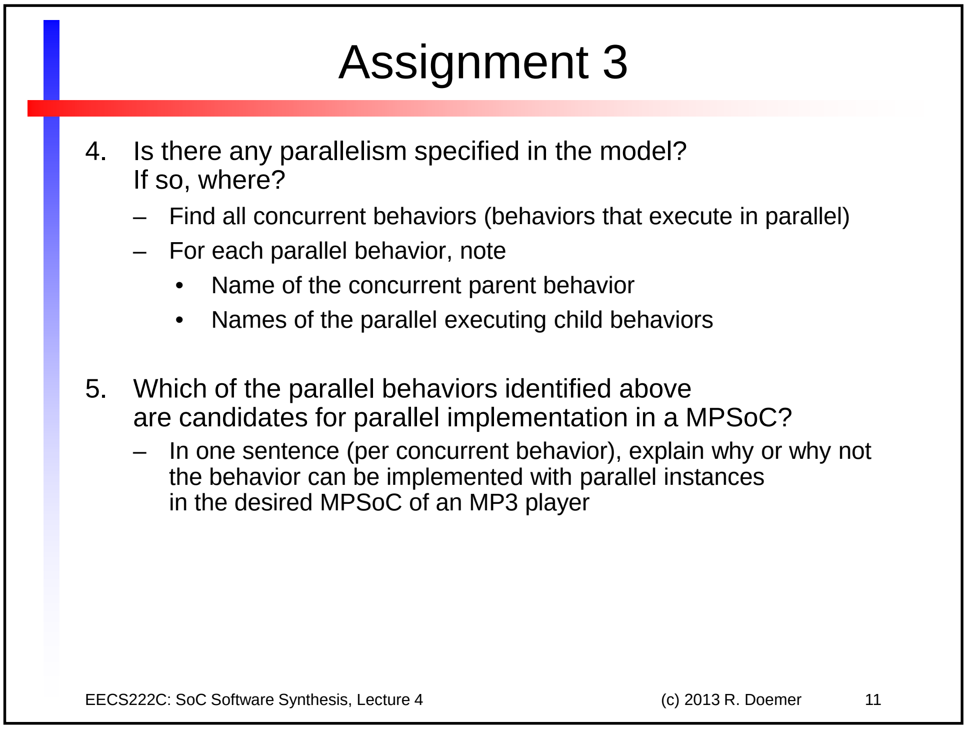
Assignment 3

2. Validate the SpecC model of the MP3 Decoder
   - Compile and execute the SpecC model
     - **`make clean`**
     - **`make`**
     - **`testbench testStream/spot1.mp3 spot1.pcm`**
   - Validate the decoded MP3 stream
     - **`diff spot1.pcm reference/spot1.pcm`**
   - Validate the SpecC model using the provided **`Makefile`**
     - **`make test`** (to run all three tests)
     - **`make test1`** (to run only the first test)



Assignment 3

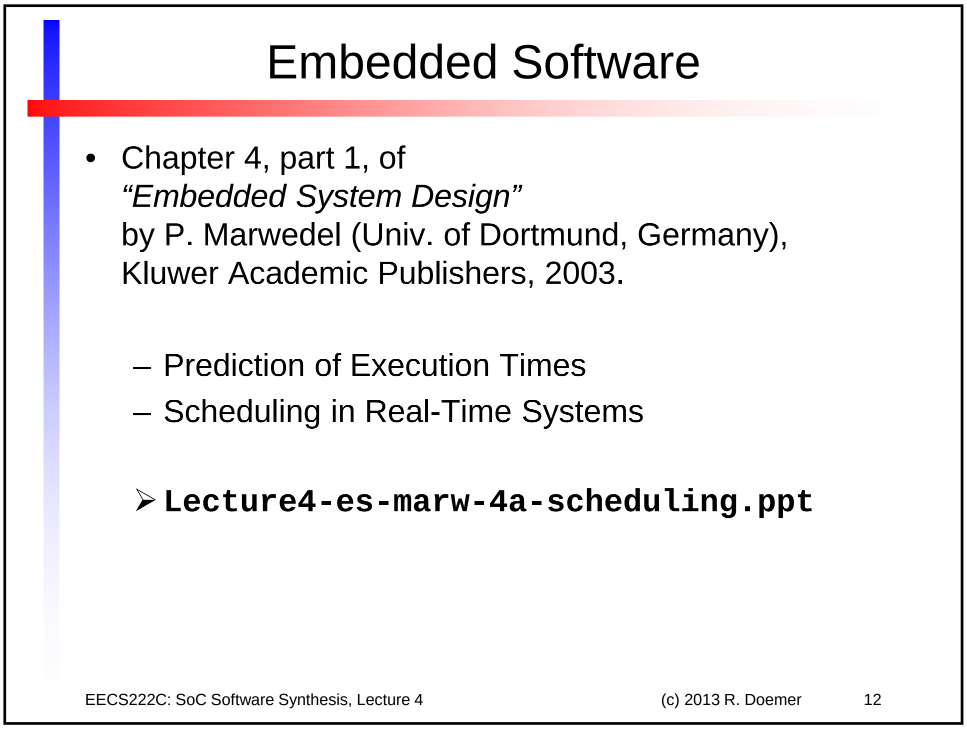
3. Analyze the specification model of the MP3 Decoder
   - Generate a top-level SIR design file
     - **`make testbench.sir`**
   - View some statistics of the model
     - **`sir_stats testbench.sir`**
     - **`sir_stats -a testbench.sir`**
   - Generate a hierarchy tree of the model
     - **`sir_tree -blt testbench.sir`**
     - **`sir_tree -blt testbench.sir Mad_Decoder`**
   - Generate a “clean” single-file SpecC model
     - **`scc testbench -sir2sc -vv -sn -sl -psi -o testbench_gen.sc`**
     - Or simply: **`make testbench_gen.sc`**
     - **`vi testbench_gen.sc`**
   - Compile and test the single-file SpecC model
     - **`scc testbench_gen -vv -xl huffman.o`**
     - **`testbench_gen testStream/spot1.mp3 spot1.pcm`**
     - **`diff spot1.pcm reference/spot1.pcm`**



Assignment 3

4. Is there any parallelism specified in the model?
   If so, where?
   - Find all concurrent behaviors (behaviors that execute in parallel)
   - For each parallel behavior, note
     - Name of the concurrent parent behavior
     - Names of the parallel executing child behaviors

5. Which of the parallel behaviors identified above
   are candidates for parallel implementation in a MPSoC?
   - In one sentence (per concurrent behavior), explain why or why not the behavior can be implemented with parallel instances in the desired MPSoC of an MP3 player

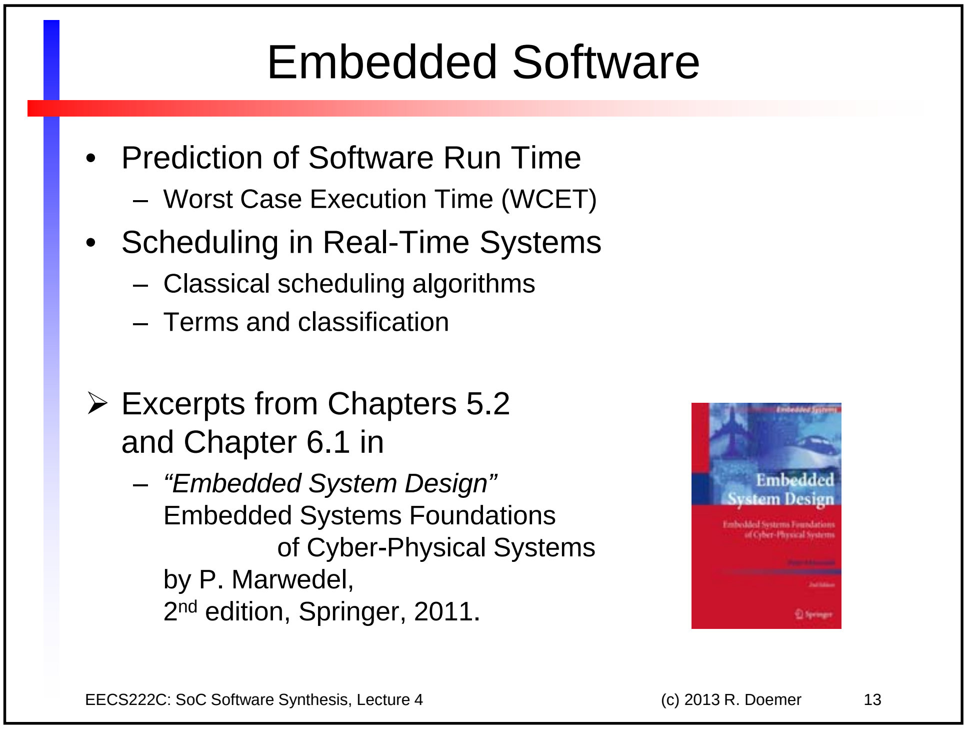


Embedded Software

- Chapter 4, part 1, of
  "Embedded System Design"
  by P. Marwedel (Univ. of Dortmund, Germany),
  Kluwer Academic Publishers, 2003.

  - Prediction of Execution Times
  - Scheduling in Real-Time Systems

  ➢ **`Lecture4-es-marw-4a-scheduling.ppt`**



Embedded Software

- Prediction of Software Run Time
  - Worst Case Execution Time (WCET)
- Scheduling in Real-Time Systems
  - Classical scheduling algorithms
  - Terms and classification

➢ Excerpts from Chapters 5.2 and Chapter 6.1 in

- *“Embedded System Design”*
  Embedded Systems Foundations
  of Cyber-Physical Systems
  by P. Marwedel,
  2nd edition, Springer, 2011.