EECS 222C: System-on-Chip Software Synthesis Lecture 10

Rainer Dömer

doemer@uci.edu

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

Lecture 10: Overview

- Course Administration
  - Course evaluation
  - Final exam
- Final Technical Report
  - Contents and Outline
- Project Review
  - Case Study: MP3 Decoder Design
  - Discussion
- Embedded Operating Systems
  - RTOS requirements, examples
- Embedded Software Synthesis
  - Summary and Conclusion

Course Administration

- Final Course Evaluation
  - 8th through 10th week
  - Nov. 15, 2010 through Dec. 5, 2010, 11:45pm
  - Closes Sunday night!
  - Online via EEE Evaluation application
- Evaluation of Course and Instructor
  - Voluntary
  - Anonymous
  - Very valuable
- Help to improve this class!
  - Please spend 5 minutes!

Course Administration

- Final Exam
  - Date and time
    - Friday, December 10, 2010, 2-4pm
  - Location
    - Room ET 201
  - Format
    - Delivery of Final Technical Report
      - Option 1: PDF by email to **doemer@uci.edu**
      - Option 2: Hardcopy
    - Hard deadline!
      - December 10, 2010, 4pm

Final Technical Report

- Title
  - Embedded Software Synthesis of a MP3 Audio Decoder
    - Final Technical Report for EECS 222C, Fall 2010
- Author
  - Your Name
- Contents
  - Describe the overall software synthesis approach
  - Outline the major steps in the design flow
  - Use the MP3 Decoder as case study
    - Tell the story of the project assignments!
  - Conclude with a summary of the lessons learned
- Length
  - about 10 pages

Final Technical Report

- Outline
  - Title page with Abstract
  - Introduction
    - Top-down embedded software design flow
  - Case Study on a MP3 Decoder
    - Application reference code
    - System specification model
    - Validation and estimation
    - Target architecture exploration
    - Transaction Level Model (TLM)
    - Bus Functional Model (BFM)
    - C code generation and cross-compilation
    - Instruction Set Simulation (ISS) Model
  - Conclusion
    - Summary of results, lessons learned
  - References

Project Review: Embedded Systems

- System embedded into another system
  - Constraints from external input (often real-time)
  - Application specific (not general purpose)
- Omnipresent in our environment
  - In many application domains
  - In 2005 [Source Netrino]
    - Only 2% of all processors in workstations
    - Remaining 8.8 billion in embedded systems
  - Pervasive

Source: P. Chou, UCI

Source: Edumicator

Source: Miele

Source: Philips

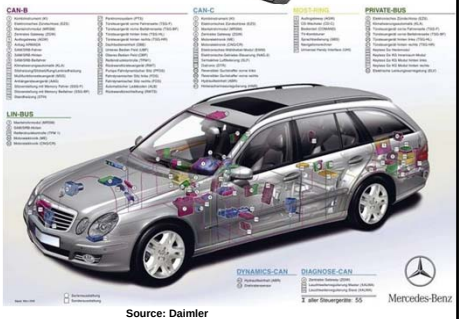
Source: www.trouper.com

Source: www.medicacorp.com/

Project Review: Motivation

- Design challenges
  - Often mobile
    - Battery powered (low power)
  - Often highly reliable
    - Extreme environment (e.g. temperature)
  - High performance constraints
    - Often real-time requirements
  - High complexity
    - E.g. Mercedes Benz E-class
      - 55 electronic control units
      - 5 communication busses
  - Tightly coupled
    - Software
    - Hardware
  - Rapid development for low price…

Source: Motorola Inc

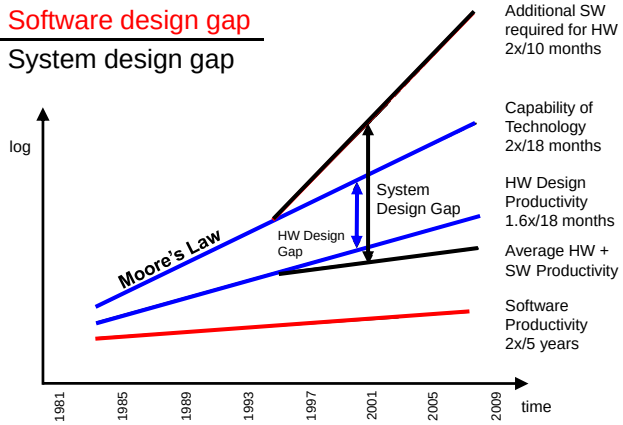


Source: Daimler

Project Review: Complexity Challenge

- Productivity Gap

  Hardware design gap
  + Software design gap
  = System design gap

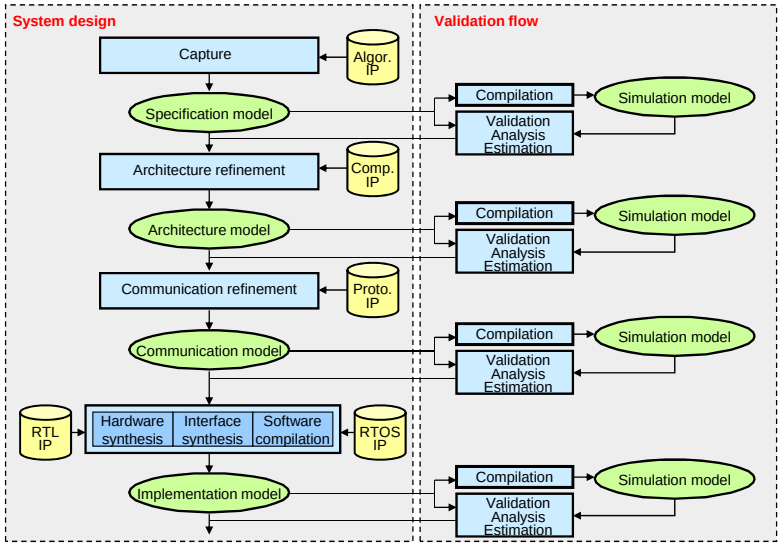


(source: *"Hardware-dependent Software"*, Ecker et al., 2009)

Project Review: HW/SW Codesign

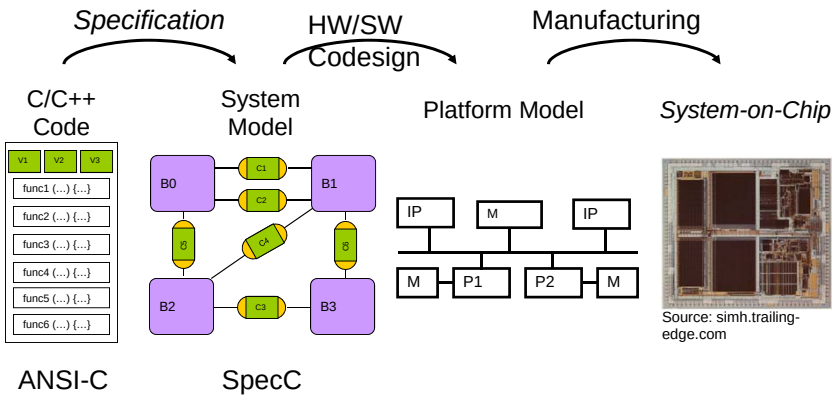
- Traditionally, software development follows hardware
- New: Unified, *concurrent* Design of
  - Hardware and
  - Software
- ➢ Improving Time to Market
  - Faster delivery of new products
  - Higher probability of on time delivery
- ➢ Using a single specification model (System Model)
  - New specification model
  - New specification language
  - ➢ Tight integration of
    - software development
    - hardware development

Project Review: Top-Down Design Flow



Project Review: SoC Design Flow

- Case Study on a MP3 Decoder:
  - Given: Reference source code (**`mad_C.tar.gz`**)
  - Analyzed: Specification Model (**`mad_SpecC.tar.gz`**)
  - Final: Platform Model with integrated ISS



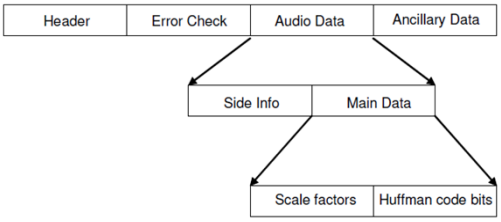
Project Review: Application Case Study

- Project Application: MP3 Audio Decoder
  - Digital compression of audio data reduces
    - Communication bandwidth and
    - Storage requirements
  - MPEG 1 Layer 3 (aka. MP3) compression algorithm
    - most commonly used
    - uses a variety of clever tricks to compress digital music
      - by 90% or more!
    - performs lossy compression
    - applies perceptual science of psycho acoustic models
      - exact input signal does not need to be retained
      - human ear can only distinguish a certain amount of detail
      - sufficient if output signal sounds identical to the human ears

[Source: CECS-TR-05-04.pdf]

Project Review: Application Case Study

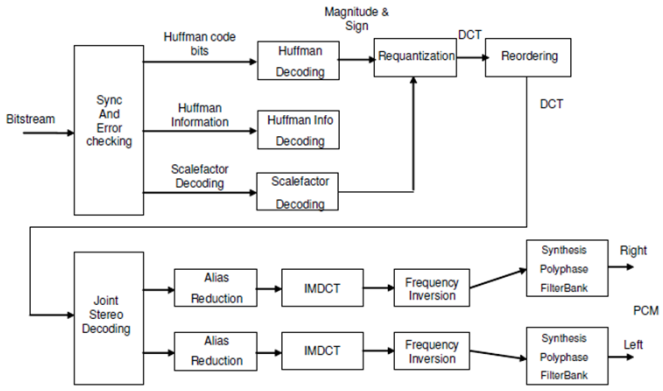
- Project Application: MP3 Audio Decoder
  - MP3 audio bit stream
    - organized in frames of bits
    - each frame contains 1152 encoded PCM samples
    - frame length depends on the bit rate (quality)
    - bit rate may vary in variable rate encoded streams
    - frame header contains information for the frame detection
  - MPEG 1 Layer 3 frame format



[Source: CECS-TR-05-04.pdf]

Project Review: Application Case Study

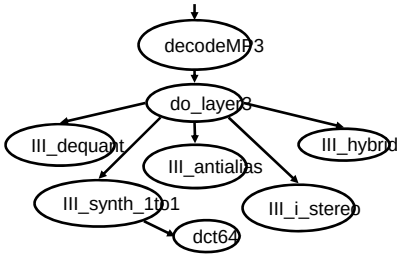
- Project Application: MP3 Audio Decoder
  - MP3 decoder block diagram



[Source: CECS-TR-05-04.pdf]

Project Review: Reference Code

- Project Application: MP3 Audio Decoder
  - MP3 decoder C reference code
    - Underbit Technologies Inc.
    - MAD: MPEG Audio Decoder
    - http://www.underbit.com/products/mad



Partial function hierarchy in MP3 reference code

[Source: P. Chandraiah]

Project Review: Assignment 1

- Administration
  - Linux Servers
    - **`alpha.eecs.uci.edu`** (NSF client)
    - **`gamma.eecs.uci.edu`** (NSF client)
    - **`mu.eecs.uci.edu`** (NSF host)
    - Intel Pentium based PCs
    - RedHat Linux (Fedora Core 12)
    - Access via secure shell protocol (**`ssh`**)
  - Accounts
    - User ID same as your UCI net ID
    - Password as discussed in class
  - SpecC Software (© by CECS, UCI)
    - SpecC Compiler and Simulator
    - System-on-Chip Environment (SCE)

Project Review: Assignment 1

- Login on Server via SSH
  - Account infos will be emailed
- Install MP3 Decoder example
  - **`mkdir eecs222c`**
  - **`cd eecs222c`**
  - **`gtar xvzf /home/doemer/EECS222C/mad_C.tar.gz`**
  - **`cd mad_C`**
  - **`make clean`**
  - **`make`**
  - **`make test`**
- Become familiar with the application and its structure
  - Browse and read the source files
  - Draw a block diagram of the major functions

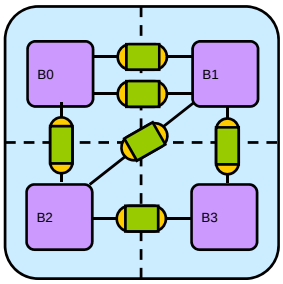
Project Review: Assignment 1

- Analyze the given MP3 Decoder application
- Example questions to investigate:
  - Example MP3 streams
    - Do they play?
    - Length in seconds?
    - Number of samples?
  - Application source code
    - How many source files?
    - How many lines of code?
    - How many functions?
  - What are the major functions?
    - How do they relate?
    - Function call graph?
  - What are the most critical functions?
    - Where is the most time spent?
  - What type of operations are performed?
    - Floating point?
    - Others?
  - Where is any potential for parallel execution?

Project Review: System Model Concepts

- System Level Modeling
  - Abstract description of a complete system
  - Software + Hardware
- Key Concepts in System Modeling
  - Explicit Structure
    - Block diagram structure
    - Connectivity through ports
  - Explicit Hierarchy
    - System composed of components
  - Explicit Concurrency
    - Potential for parallel execution
    - Potential for pipelined execution
  - Explicit Communication and Computation
    - Channels and Interfaces
    - Behaviors / Modules

System Model



Project Review: 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`**

Project Review: Assignment 3

1. Setup and simulate a SpecC model of the MP3 Decoder
   - Setup and unpack source code
     - **`source /opt/sce-20100908/bin/setup.csh`**
     - **`tar xvzpf /home/doemer/EECS222C/mad_SpecC.tar.gz`**
     - **`cd mad_SpecC`**
     - **`ls`**
   - Compile the SpecC model
     - **`make clean`**
     - **`make`**
   - Execute the SpecC model
     - **`testbench testStream/spot1.mp3 spot1.pcm`**
     - **`diff spot1.pcm ../mad_C/spot1.pcm`**
   - Use decoded PCM files from reference C code as “golden” reference
     - **`cp ../mad_C/spot1.pcm reference/`**
     - **`cp ../mad_C/spot1_3K.pcm reference/`**
     - **`cp ../mad_C/classic1.pcm reference/`**
   - Simulate the SpecC model (using the provided **`Makefile`**)
     - **`make test`** (or: **`make test1`** to run only the first test)

Project Review: Assignment 3

2. Analyze the specification model of the MP3 Decoder
   - Setup (as in step 2)
     - **`cd mad_SpecC`**
   - Generate a top-level SIR design file
     - **`make`**
     - **`ls –l 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`**
   - 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`**

Project Review: Assignment 3

3. Is there any parallelism specified in the model?
   If so, where?
   - Find all behaviors that execute in parallel
   - For each parallel behavior, note
     - the name of the parent behavior
     - the names of the parallel child behaviors

- Deliverables
  - Names of concurrent parent behaviors
  - Names of parallel executing child behaviors

- Due
  - by Friday, Oct 22, 2010, at noon
  - by email to **`doemer@uci.edu`**
    with subject “EECS222C HW3”

Project Review: Assignment 4

1. Become familiar with the System-on-Chip Environment (SCE)
   - Setup
     - Note that we will use the 2003 version of SCE for the tutorial:
     - **`source /opt/sce-20030530/bin/setup.csh`**
     - **`rm –rf ~/.sce`**
     - **`mkdir demo`**
     - **`cd demo`**
     - **`setup_demo`**
   - Open the SCE Tutorial document
     - **`acroread SCE_Tutorial/sce-tutorial.pdf &`**
     - To protect the environment and save some trees, please *do not print* the tutorial document! It contains 250 pages and you will likely read it only once… ;-)
   - Follow the SCE Tutorial instructions
     - **`sce &`**
     - **`...`**
   - Cleanup
     - When done (or to start over), clean up your demo directory
     - **`cd ..`**
     - **`rm –rf demo`**

Project Review: Assignment 4

2. Setup your MP3 Decoder model in SCE

   - Setup SCE
     - Note that we will use the 2010 version of SCE for the MP3 decoder:
     - **`source /opt/sce-20100908/bin/setup.csh`**
     - **`rm –rf ~/.sce`**
     - **`cd mad_SpecC`**
     - **`sce &`**

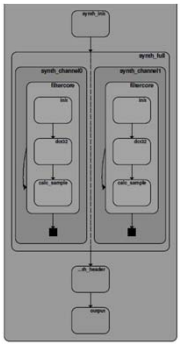
   - Create a new project in SCE
     - **`Project->New`**
     - **`Project->Settings`**
       - Set include path to "**`.`**" (current directory)
       - Set libraries to "**`-xl huffman.o`**"
       - Set both verbosity and warning level to 2
       - In the Simulator tab, set the simulation command as follows (single line!):
         ```
         ./%e testStream/spot1_3K.mp3 spot1_3K.pcm &&
         diff reference/spot1_3K.pcm spot1_3K.pcm
         ```
     - **`Project->SaveAs "mp3.sce"`**

Project Review: Assignment 4

3. Compile and simulate your MP3 Decoder model in SCE
   - … (continued from previous page)
   - Load your design model into SCE
     - **`File->Import "testbench.sc"`**
     - **`Project->AddDesign`**
     - Right-click on **`testbench.sir`** in the project window, and **`Rename`** the model to **`Spec`**
   - Compile and simulate your model in SCE
     - **`Validation->Compile`**
     - **`Validation->Simulate`**

Project Review: Assignment 4

4. Analyze your MP3 decoder model in SCE
   - … (continued from previous page)
   - Browse the structural hierarchy charts
     - Select the **`Main`** behavior in the behavior browser
     - Right-click **`->Chart`**
     - Double-click the chart to add a level of hierarchy
     - **`View->Connectivity`**
     - **`...`**
   - Print the hierarchy chart for the Synthesis Filter
     - Select the **`synth`** behavior in the behavior browser
     - Right-click **`->Chart`**
     - Add several levels of hierarchy
     - **`Window->Print…`** in color (!) to file **`"synth.ps"`**
     - **`ps2pdf synth.ps`**



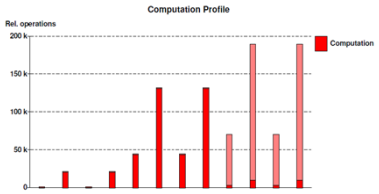
- Deliverable
  - Hierarchy chart **`"synth.pdf"`** (in color!)
  - by Friday, Oct 29, 2010, at noon
  - by email to **`doemer@uci.edu`** with subject "EECS222C HW4"

Project Review: Assignment 5

1. Profile your MP3 Decoder model in SCE
   - (continued from previous assignment)
   - Load your MP3 project in SCE
     - **`Project->Load "mp3.sce"`**
   - Load your design model into SCE
     - **`File->Import "testbench.sc"`**
     - **`Project->AddDesign`**
     - Right-click on **`testbench.sir`** in the project window, and **`Rename`** the model to **`Spec`**
   - Compile and simulate your model in SCE
     - **`Validation->Compile`**
     - **`Validation->Simulate`**
   - Profile your MP3 decoder in SCE
     - **`Validation->Profile`**

Project Review: Assignment 5

2. Analyze your Profiling Results
   - Use the graphical bar charts to compare the complexity of the behaviors in your MP3 decoder
     - In the hierarchy browser, select behaviors of interest (use CTRL-LeftClick to select/deselect)
     - **`RightClick->Graphs->Computation`**
   - Determine the most-critical behaviors that contribute the most computation operations
     - The goal is to find those behavioral blocks that make good choices for hardware accelleration
   - Deliverable 1:
     - Bar chart showing the selected behaviors in comparison to others
       - **`CriticalBlocks.pdf`**
     - Text file briefly (!) explaining your choice
       - **`CriticalBlocks.txt`**



Example Computation Profile (block names omitted)

Project Review: Assignment 5

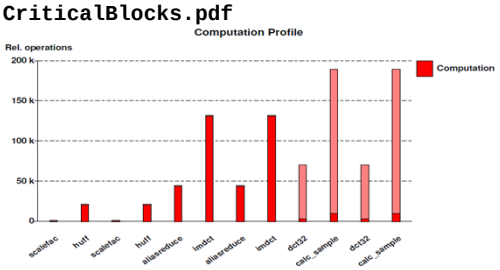
3. Evaluate potential Processors for SW-only Implementation
   - Select DUT as **`Mad_decoder decoder`**
     - RightClick on **`decoder ->SetAsTop-Level`**
   - Consider an ARM7TDMI processor (50MHz)
     - **`Synthesis->Allocate PEs…`**
     - Add Processors, **`ARM_7TDMI`**
     - Choose default port configuration (i.e. 20000ps)
     - Choose 50 MHz (change it from default 100MHz)
     - Name the processor **`ARM7TDMI`**
   - Map the entire decoder on to the ARM7TDMI processor
     - **`Validation->Evaluate`**
     - **`Validation->Show Estimates`**
   - Determine the estimated execution time on the ARM7TDMI!

Project Review: Assignment 5

4. Evaluate alternative Processors for SW-only Implementation
   - Consider as alternative a LEON3 processor (50MHz)
     - **`Synthesis->Allocate PEs…`**
     - Add Processors, **`LEON3`**
     - Choose default port configuration (i.e. 20000ps)
     - Choose default clock frequency (i.e. 50 MHz)
     - Name the processor **`LEON3`**
   - Map the entire decoder on to the LEON3 processor
     - **`Validation->Evaluate`**
   - Determine the estimated execution time on the LEON3!
   - Deliverable 2:
     - Text file with the estimated execution times for the ARM7TDMI and LEON3 processors, and
     - Brief analysis whether or not each processor is expected fast enough for a SW-only implementation of the MP3 decoder
       - **`SWonly.txt`**
   - Due:
     - by Friday, Nov 5, 2010, at noon (email to **doemer@uci.edu,** "EECS222C HW5")

Project Review: Discussion

1. Bar chart showing the most-critical behaviors in terms of computation operations
   - Find blocks suitable for HW acceleration!
     - **`CriticalBlocks.pdf`**



     - **`CriticalBlocks.txt`**
       - Starting from the **`decoder`** DUT, traverse down the hierarchy to find behaviors with most significant computation…
       - **`decode_frame/layer_III/decode/granule/channels`**
         - » **`left/imdct`**
         - » **`right/imdct`**
       - **`synth/synth_full/synth_channel{01}/filtercore`**
         - » **`dct32`**
         - » **`calc_sample`**

Project Review: Discussion

2. Evaluate potential Processors for SW-only Implementation
   - Deliverable 2:
     - Estimated execution times for the ARM7TDMI and LEON3 processors
     - Brief analysis whether or not each processor is expected fast enough
       - **`SWonly.txt`**
   - Candidate 1: ARM7TDMI processor at 50 MHz
     - Profiler-estimated execution time: **`206.0 ms`**
   - Candidate 2: LEON3 processor at 50 MHz
     - Profiler-estimated execution time: **`733.3 ms`**
   - Is this fast enough?
     - Length of test stream **`spot1_3k.pcm:`**
       `36864` bytes **`= 8`** frames **`* 1152`** samples **`* 2`** channels **`* 2`** bytes per sample
     - Monitor source code **`mp3monitor.sc:`**
       `44.1 kHz` stereo stream **`(`**at **`2`** bytes per sample**`)`**
     - Timing constraints:
       Per frame: **`1152 / 44.1 kHz = 26.12 ms`**
       For entire test stream: **`8*1152 / 44.1 kHz = 208.98 ms`**
     - ➢ **`ARM7TDMI`** processor barely meets the timing constraint
     - ➢ **`LEON3`** processor misses the timing constraints (by more than 2x)
   - ➢ Hardware accelleration becomes necessary!

Project Review: Assignment 6

1. Refinement of the MP3 Decoder model in SCE
   - Continue from “Spec” model of previous assignment
   - Select an appropriate system architecture
     - 1 ARM_7TDMI or LEON3 CPU at 50 MHz
     - 0-6 HW_Standard accellerators at 100 MHz
   - Perform the following refinement steps
     - Architecture Refinement
     - Scheduling Refinement
     - Network Refinement
     - Communication Link Refinement
       - Transaction-level model (TLM)
       - Pin-accurate model (PAM)
     - Reference instructions:
       - **`/home/doemer/EECS222C/Assignment6.txt`**
   - Deliverable:
     - Text file “**`refinement.txt`**” with
       - short description of the chosen target architecture
       - simulated execution times of each model
   - Due:
     - by Friday, Nov 12, 2010, at noon
     - by email to **doemer@uci.edu**, with subject “EECS222C HW6”

Project Review: Assignment 7

1. SW Synthesis and ISS Integration of the MP3 Decoder in SCE
   - Continue from “TLM” and “PAM” models of previous assignment
   - Perform C code generation
     - Software Synthesis Refinement, C code generation
   - Perform Instruction Set Simulation
     - Software Synthesis Refinement, ISS integration
   - Reference instructions:
     - **`/home/doemer/EECS222C/Assignment7.txt`**
   - Deliverable:
     - Text file “**`SWgen.txt`**” with
       - short (!) description of “your story”
       - execution time of your ISS model
   - Due:
     - by Friday, Nov 19, 2010, at noon
     - by email to **doemer@uci.edu**, with subject “EECS222C HW7”

Project Review: Problem Solving…

1. Communication models are very slow
   - Symptom:
     - Simulation shows that delay per frame increases drastically in TLM and PAM models
   - Analysis:
     - Communication time is not taken into account before
     - TLM and PAM include accurate communication delay
     - Communication can be a bottleneck for certain architectures
       - Congestion due to single communication bus
       - Indirect communication from slave to slave via master
   - Potential Solutions:
     - Analyze bus traffic by viewing connectivity in Network model
     - Introduce a separate bus between hardware components
       - See **`Lecture7-ASPDAC07-AG-MP3.pdf`**
     - Increase bus frequency

Project Review: Problem Solving…

2. Error message when simulating ISS model
   - Symptom:
     - Simulation of ISS model reports errors such as **`Core: Shifter distance more than 31, i.e. 226`**
   - Analysis:
     - Error message originates from ISS implementation
     - The same (or similar) message shows for other successful examples
   - Solution:
     - Probably a shift-instruction with argument out of range without serious other effects
     - Simply ignore the message!

Project Review: Problem Solving…

3. Priority conflicts in ISS model
   - Symptom:
     - Simulation of ISS model reports errors such as **`TaskCreate Failed prio 2 already exists!`**
   - Analysis:
     - We have been selecting Round-robin scheduling (all priorities 1) or Priority scheduling (with non-exclusive priorities)
     - The ISS model uses micro-C-OS-2 as RTOS which requires fixed non-exclusive priorities
   - Attempted Solutions:
     - Isolate all parallel behaviors so that unique priorities can be assigned
     - Select Priority-based scheduling with different priorities
     - ➢ Error messages persist…
     - ➢ Suspect a bug in the RTOS port that prevents clean task deletion…

Project Review: Problem Solving…

4. Persistent priority conflicts in ISS model
   - Symptom:
     - Simulation of ISS model reports errors such as **`TaskCreate Failed prio 2 already exists!`**
   - Analysis:
     - We have been selecting Round-robin scheduling (all priorities 1) or Priority scheduling (with non-exclusive priorities)
     - The ISS model uses micro-C-OS-2 as RTOS which requires fixed non-exclusive priorities
   - Successful Solution:
     - Revert to static scheduling!
     - In Scheduling refinement, select static scheduling and serialize the entire tree of behaviors mapped to the CPU
     - ➢ Error messages are gone!

Project Review: Problem Solving…

5. Large frame delay in ISS model
   - Symptom:
     - Simulation of TLM/PAM models estimated frame delays such as **`Decode time per frame = 20.010 ms`**
     - Simulation of ISS model reports frame delays such as **`Decode time per frame = 65.979 ms`**
   - Analysis:
     - TLM/PAM computation time is only estimated (by SCE profiler)
     - Profiling produces only fidelity (not absolute accuracy!)
     - Profiler for ARM7 is based on overly optimistic operation cycles
       - Assumes zero cache-misses, pipeline stalls, etc.
   - Possible Solutions:
     - Trust the ISS, not the profiler!
     - ➢ Increase CPU frequency, and/or improve system architecture!
     - ➢ Consider a different CPU, e.g. LEON3

Project Review: Problem Solving…

6. Incorrect output file in ISS model
   - Symptom:
     - Decoded stream by ISS model fails comparison with reference stream
       `Binary files reference/spot1_3K.pcm and spot1_3K.pcm differ`
   - Analysis:
     - Linux **`diff`** reports file difference
     - Linux **`ls -l`** shows difference in number of bytes
     - Linux **`vi -d`** shows that only the end of the file differs
   - Solution:
     - Decoding by ISS model was correct, it just didn't finish!
     - ➢ Extend simulation time before stimulus behavior exits, or
     - ➢ Find a faster architecture, then decoded file will be complete!

Project Review: Problem Solving…

7. Incorrect timing of the ISS model
   - Symptom:
     - ISS model timing is incorrect during simulation
       `Decode time per frame = 65.979 ms`
   - Analysis:
     - ARM7 ISS model assumes 100MHz clock speed for the CPU regardless of what was selected during PE allocation
     - We need to patch the timing delay in the ISS model
   - Solution:
     - **`scc MyISS -sir2sc -sl -psi -o MyISS_patched.sc`**
     - **`vi MyISS_patched.sc`**
       - Search for function call **`cyclebycycle()`**
       - Adjust **`waitfor(result * 10000ull);`**
     - **`scc MyISS_patched -xcx -xlx -vv -w -xl huffman.o`**
     - **`./MyISS_patched testStream/spot1_3K.mp3 spot1_3K.pcm`**
     - **`diff -s reference/spot1_3K.pcm spot1_3K.pcm`**

Project Review: Problem Solving…

8. Slow default bus frequency of the AMBA AHB
   - Symptom:
     - The AMBA AHB connected to the CPU is set to 50MHz by default
       `20000ns (50MHz)`
   - Analysis:
     - Higher clock frequency of the CPU justifies also higher bus speed
     - SCE fixes the bus parameters already at the time of PE Allocation (i.e. in the dialog before Architecture Refinement)
   - Solution:
     - We need to overwrite the default value when allocating the CPU
     - Thus, to change the CPU bus frequency
       we need to go back to the Specification model and
       set the bus speed at same time when the CPU is allocated

Embedded Operating Systems

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

- Embedded Operating Systems
  - General requirements
  - Real-time Operating Systems (RTOS)

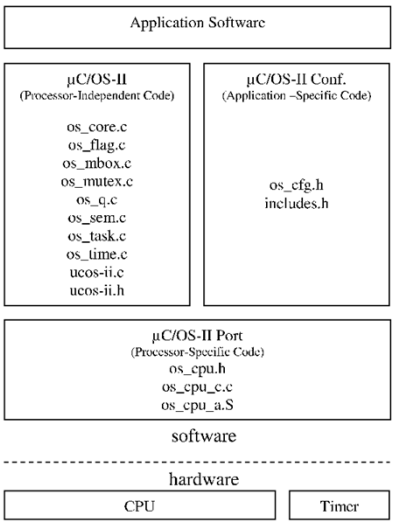
  - **`Lecture10-es-marw-4d-rtos.ppt`**

Embedded Operating Systems

- Example: MicroC/OS-II
  - Ported in SCE for use with ARM_7TDMI CPU
  - Features
    - multi-tasking real-time kernel
    - real-time support (most kernel functions deterministic)
    - task management
    - priority scheduling
    - preemption
    - ROM'able (executable from firmware)
      - memory footprint about 20 KB
    - portable (to over 40 different CPU architectures, 8-64bit)
      - about 5500 lines of ANSI-C source code
      - only small amount of processor-specific assembly code

Embedded Operating Systems

- Example: MicroC/OS-II
  - Software Structure



Embedded Operating Systems

- Example: MicroC/OS-II
  - Kernel Services
    - Task management
      - up to 56 application tasks
      - priority-based scheduling
    - Time management
      - system timer interrupt (10ms – 100ms)
      - 32-bit tick counter
    - Semaphore management
      - inter-task communication through shared memory
      - semaphore API
    - Mutex management
      - binary semaphore
    - Memory management
      - dynamic memory allocation (with fixed block size)

Summary and Conclusion

- Embedded Software Synthesis
  - C/C++ Reference Code
  - SLDL Modeling
    - System specification in SpecC
  - Estimation and Exploration
    - Find a suitable target platform
  - Scheduling and RTOS selection
    - Static vs. dynamic scheduling
  - Target Code Generation
    - ANSI-C code generation for cross-compilation
  - Instruction Set Simulation (ISS)
    - Simulation of execution on the target processor
      - Pin- and cycle-accurate