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
  - SoC Software Synthesis for a MP3 Audio Decoder
  - Discussion
- Embedded Operating Systems
  - RTOS requirements, examples
- Embedded Software Synthesis
  - Course Summary

Course Administration

- Final Course Evaluation
  - 9th through 10th week
  - May 28, 2013 through June 9, 2013, 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
    - **Wednesday, June 12, 2013, until 12pm (noon)**
  - Format
    - Delivery of Final Technical Report
    - Electronic submission (**`turnin`** on server)
      - **`final/ISS.sir`**
      - **`final/Report.pdf`**

➢ Hard deadline!

- Wednesday, June 12, 2013, 12pm (noon)

Final Technical Report

- Title
  - SoC Software Synthesis for a MP3 Audio Decoder
    - Final Technical Report for EECS 222C, Spring 2013
- Author
  - Your Name and ID
- Contents
  - Describe the overall SoC design 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 12 pages (including title page and references)

Final Technical Report

- Suggested Outline
  1. Title page with abstract
  2. Introduction
     1. Top-down embedded software design flow
  3. Case study on a MP3 decoder
     1. Application reference code
     2. System specification model
     3. Validation and profiling
     4. Target architecture selection
     5. Transaction Level Model (TLM)
     6. C code generation and cross-compilation
     7. Pin-Accurate Model (PAM)
     8. Instruction Set Simulation (ISS) Model
  4. Conclusion
     1. Lessons learned
  5. 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

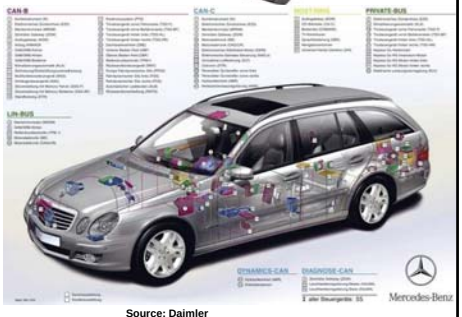

Source: www.trouper.com


Source: www.medicacorp.com/

Project Review: Motivation

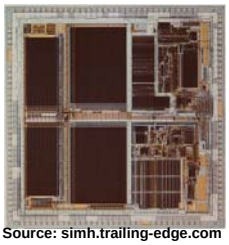
- Design challenges
  - Often mobile
    - Battery powered (low power)
  - Often highly reliable
    - Extreme environment (e.g. temperature)
  - High performance constraints
    - Often real-time requirements
  - High complexity
    - E.g. Mercedes Benz E-class
      - 55 electronic control units
 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


Source: Motorola Inc


Source: Daimler

Project Review: Embedded System Design

- Design Advantages
  - *Application known at design time*
  - *Environment known at design time*
  - Allows for customized / optimized solution
    - Improved performance
    - More functionality
    - At lower power



Source: simh.trailing-edge.com

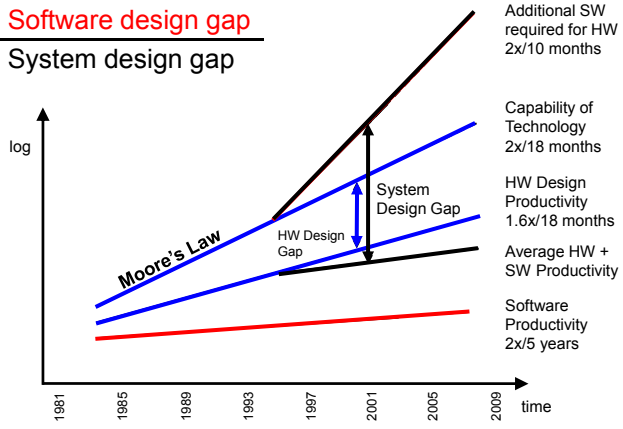
- Custom Platform, SW and HW components
  - Multi-Processor System-on-Chip (MPSoC),
    - Complete embedded system integrated on a chip
  - General-purpose and application-specific processors
  - Application Specific Integrated Circuit (ASIC)
  - Field Programmable Gate Array (FPGA)
  - Circuit board with off-the-shelf-components

Source: Xilinx

Project Review: Design Complexity Challenge

- Productivity Gap

  Hardware design gap
  + Software design gap
  = System design gap



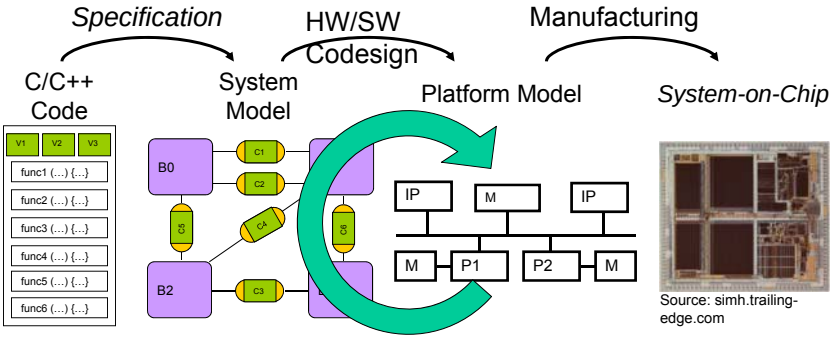
(source: *"Hardware-dependent Software"*, Ecker et al., 2009)

Project Review: HW/SW Codesign

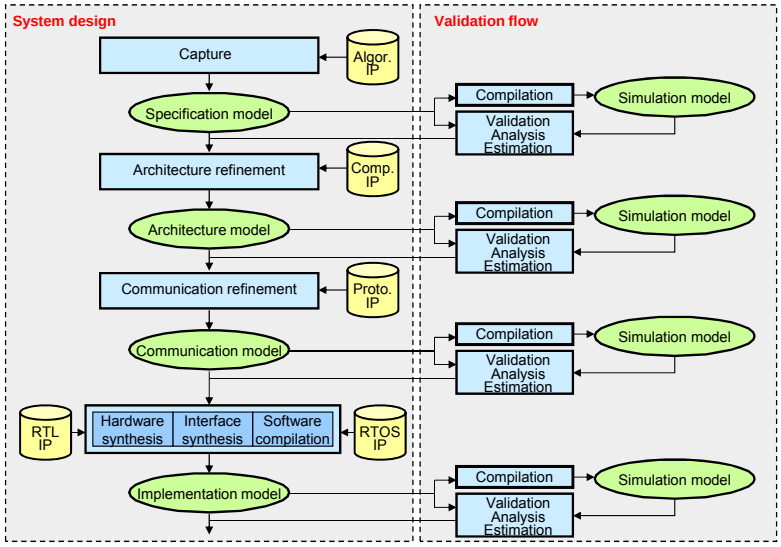
- Traditionally, software development follows hardware
- New: Unified, *concurrent* Design of
  - Hardware and
  - Software
- ➢ Improving Time to Market
  - Faster delivery of new products
  - Higher probability of on time delivery
- ➢ Using a single specification model (System Model)
  - New specification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

Project Review: SoC Co-Design Flow

- Application Case Study on a MP3 Audio Decoder:
  - Given: Reference source code (**`mad_C.tar.gz`**)
  - Analyzed: System Model (**`mad_SpecC.tar.gz`**)
  - Refined: Platform Model with integrated ISS
  - Next: System design cycle(s) to meet timing
  - Finally: Hand-off to manufacturing…



Project Review: SCE Design Flow



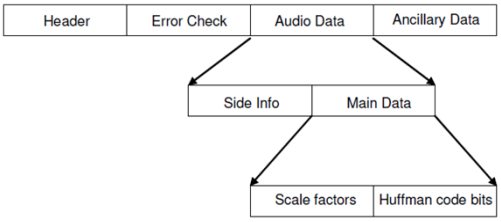
Project Review: Application Case Study

- 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

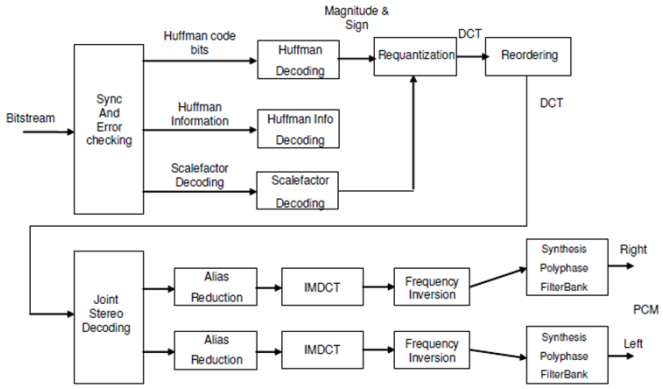
- Project Application: MP3 Audio Decoder
  - MP3 audio bit stream
    - organized in frames of bits
    - each frame contains 1152 encoded PCM samples
    - frame length depends on the bit rate (quality)
    - bit rate may vary in variable rate encoded streams
    - frame header contains information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



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

Project Review: Application Case Study

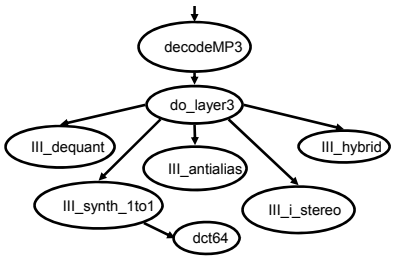
- Project Application: MP3 Audio Decoder
  - MP3 decoder block diagram



[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
    - **`gamma.eecs.uci.edu`** (NSF client)
    - **`omicron.eecs.uci.edu`** (NSF client)
    - Intel Pentium based workstations
    - 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
  - `cd ~`
  - `mkdir hw1`
  - `cd hw1`
  - `gtar xvzf /home/eecs222/EECS222C_S13/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

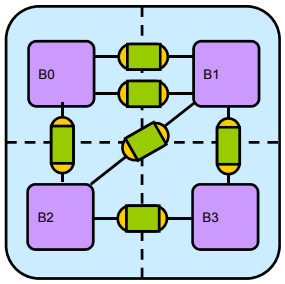
➢ Questions to study:

- 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. 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`**

Project Review: 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)

Project Review: 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`

Project Review: 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

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

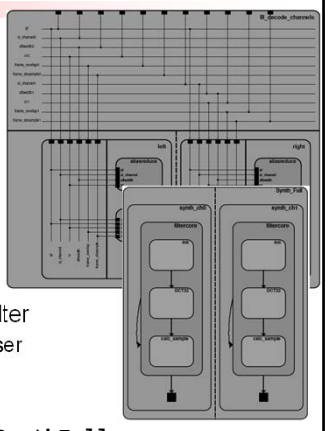
2. Setup your MP3 Decoder model in SCE
   - Setup SCE
     - Note that we will use the 2010 version of SCE:
     - **`source /opt/sce-20100908/bin/setup.csh`**
     - **`rm –rf ~/.sce`**
     - **`ln -s hw3 hw4`**
     - **`cd hw4`**
     - **`sce &`**
   - Create a new project in SCE
     - **`Project->New`**
     - **`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. Study your MP3 decoder model in SCE
   - … (continued from previous page)
   - Browse the structural hierarchy charts
     - Select a behavior in the behavior browser
     - Right-click **`->Chart`**
     - Double-click to add a level of hierarchy
     - **`View->Connectivity`**
     - **`...`**
   - Print the hierarchy chart for the Synthesis Filter
     - Select the **`Synth_Full`** behavior in the browser
     - Right-click **`->Chart`**
     - Add all levels of hierarchy, but no connectivity
     - **`Window->Print…`** in color (!) to file **`Chart_SynthFull.ps`**
   - Print the hierarchy chart for the Channel Decoding
     - Display the chart of the **`III_decode_channels`** behavior
     - Add all levels of hierarchy, including connectivity
     - **`Window->Print…`** in color (!) to file **`Chart_DecodeChannels.ps`**

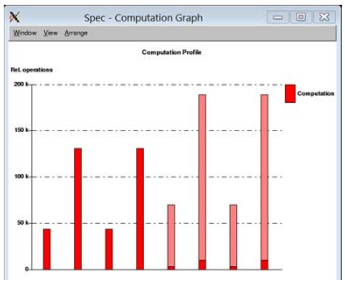


Project Review: Assignment 5

1. Profile your MP3 Decoder model in SCE
   - (continued from previous assignment)
   - Load your MP3 project in SCE
     - **`Project->Load "mp3.sce"`**
   - Open your "Spec" design model and validate it
     - Double-click on **`Spec.sir`** in the project window
     - **`Validation->Compile`**
     - **`Validation->Simulate`**
   - Profile your MP3 decoder in SCE
     - **`Validation->Profile`**

Project Review: Assignment 5

2. Analyze your Profiling Results
   - Use the SCE bar charts to compare the computational complexity of the behaviors in your MP3 decoder model
     - In the hierarchy browser, select behaviors of interest (use CTRL-LeftClick to select/deselect)
     - **`RightClick->Graphs->Computation`**
   - Identify the behavior instances with the most computational load
     - Goal is to find those components that make good candidates for hardware accelleration
       - Short code
       - Regular structure
       - High computation
     - Hint: There are 8 candidates as shown in the chart on the right!
   - Deliverable
     - **`ComputationProfile.pdf`**



Example Computation Profile (block names omitted)

Project Review: System Design Flow

- Step 1: Architecture Refinement
  - Allocation of Processing Elements (PEs)
    - Number and type of software processors
    - Number and type of custom hardware units
    - Number and type of system memories
  - Mapping to PEs
    - Map each behavior to a PE
    - Map each channel to a PE
    - Map each variable to a PE
  - Result:
    System architecture of concurrent PEs
    with abstract communication via channels
  - ➢ Estimated timing for computation specific to PE type

Project Review: System Design Flow

- Step 2: Scheduling Refinement
  - For each sequential PE (e.g. software processor),
    serialize the execution of behaviors to a single thread of control
  - Option (a): Static scheduling
    - For each set of concurrent behaviors,
      determine a fixed order of execution
  - Option (b): Dynamic scheduling by RTOS
    - Choose scheduling policy,
      i.e. round-robin or priority-based
    - For each set of concurrent behaviors,
      determine the scheduling priority
  - Result:
    System model with static or dynamic schedule
    in each sequential PE
  - ➢ Estimated total time of computation for each PE

Project Review: System Design Flow

- Step 3: Network Refinement
  - Allocation of system busses
    - Number and type of system busses
    - Number and type of communication elements (CEs)
      - Transducers: Routers or bridges
    - System connectivity
      - Masters and slaves
  - Mapping of channels to busses and transducers
    - Map each inter-PE communication channel to a system bus (or multiple busses, if applicable)
    - Routing
  - Result: Network model of the system
  - ➢ Accurate representation of top-level system components and their connectivity

Project Review: System Design Flow

- Step 4: Communication Refinement
  - Allocation and specification of communication protocol(s) for each communication link (bus)
    - Type of bus protocol for each link (if applicable)
    - Bus protocol parameters (e.g. bit width, etc.)
    - Synchronization policy and parameters
      - Polling vs. interrupt
  - Mapping of addresses
    - System-wide address mapping to registers and memories
    - Address translation in transducers (if needed)
  - Result: Bus-functional model of the system
    - Transaction Level Model (TLM)
    - Pin Accurate Model (PAM)
  - ➢ Accurate timing for computation and communication

Project Review: System Design Flow

- Step 5: Hardware Synthesis (for HW PEs)
  - Allocation of Register Transfer Level (RTL) components
    - Number and type of functional units (e.g. adder, multiplier, ALU)
    - Number and type of storage units (e.g. registers, register files)
    - Number and type of interconnecting busses (drivers, multiplexers)
  - Scheduling
    - Basic blocks assigned to super-states
    - Individual operations assigned to states (clock cycles)
  - Binding
    - Bind functional operations to functional units
    - Bind variables to storage units
    - Bind assignments/transfers to busses
  - Result:
    Synthesizable HDL description (Verilog, VHDL, or SystemC)
  - ➢ Clock-cycle accurate timing for each HW PE

Project Review: System Design Flow

- Step 6: Software Generation (for SW PEs)
  - Generation of custom C code
    - For selected target processor
    - Specific to the entire system (incl. communication layers)
  - RTOS targeting
    - Integration of selected target RTOS
  - Compilation to Instruction Set Architecture
    - Instruction Set Simulation (ISS) integration
      - Instruction-accurate or cycle-accurate
  - Assembly and Linking
  - Result:
    Downloadable binary image
  - ➢ Clock-cycle or instruction accurate timing for each SW PE

Project Review: Assignment 6

- Evaluate ARM7TDMI as a potential Processor for a SW-only Implementation of the MP3 Decoder
  - Continue from the "Spec" model of the previous assignment
  - Allocate an ARM_7TDMI processor for the entire decoder
    - Choose default port configuration (i.e. 20000ps bus cycle)
    - Choose 50 MHz (change it from default 100MHz)
  - Estimate the execution time and calculate the frame delay
  - Perform the following refinement steps
    - Architecture Refinement
    - Scheduling Refinement
    - Network Refinement
    - Communication Refinement
      - Transaction-level model (TLM)
        - » Code generation: TLM_C model
      - Pin-accurate model (PAM)
        - » Instruction Set Simulator (ISS) model
  - Details: **`/home/eecs222/EECS222C_S13/Assignment6.txt`**

Project Review: Assignment 6

- Evaluate ARM7TDMI as a potential Processor for a SW-only Implementation of the MP3 Decoder
  - Fill the following table with the estimated/simulated frame delays

Refinement Step	Model	Decode time per frame
Profiling estimation	Spec	
Architecture Refinement	Arm7Arch	
Scheduling Refinement	Arm7Sched	
Network Refinement	Arm7Net	
Transaction-Level Refinement	Arm7TLM	
C Code Generation	Arm7TLM_C	
Pin-Accurate Refinement	Arm7PAM	
Instruction Set Simulation	Arm7ISS	

  - Submit as file: **`hw6/ARM7_Evaluation.pdf`**

Project Review: Assignment 6

- Discussion
  - Test bench improvements
    - Stimulus: calls exit!? Should not!
    - Monitor: should quit the simulation and report the total time!
    - Simulation: Pass the number of frames (**8**) as 3rd argument
  - Corrections to compilation settings
    - Assertions are part of the DUT!? Must not!
    - Turn assertions (and debug) off: pass **–DNDEBUG** to compiler
  - Reported ISS cycles vs. reported decoding time!?
    - ISS model in SCE version 2010 has a bug: processor speed is fixed to 100Mhz!
    - Switch to "latest" SCE version 2012+ (`/opt/sce/bin/setup.csh`)
  - Profiling estimation is inaccurate!
    - Several times too optimistic for the ARM7
    - Calibrate profiler weight tables by corresponding factor

Project Review: Assignment 7

- Re-evaluate ARM7TDMI as a potential Processor for a SW-plus-HW Implementation of the MP3 Decoder
  - Allocate an ARM_7TDMI processor for the software
    - Choose desired clock period (change it from default 10000ps)
  - Allocate up to 6 custom HW units for acceleration
    - Keep default clock frequency of 100 MHz
  - Perform the system design refinement steps
    - Architecture Refinement
    - Scheduling Refinement
    - Network Refinement
    - Communication Refinement
      - Transaction-level model (TLM)
        - » Code generation: TLM_C model
      - Pin-accurate model (PAM)
        - » Instruction Set Simulator (ISS) model
  - Details: **`/home/eecs222/EECS222C_S13/Assignment7.txt`**

Project Review: Assignment 7

- Re-evaluate ARM7TDMI as a potential Processor for a SW-plus-HW Implementation of the MP3 Decoder
  - Fill the following table with the estimated/simulated frame delays

Refinement Step	Model	Decode time per frame
Profiling estimation	Spec	
Architecture Refinement	Arch	
Scheduling Refinement	Sched	
Network Refinement	Net	
Transaction-Level Refinement	TLM	
C Code Generation	TLM_C	
Pin-Accurate Refinement	PAM	
Instruction Set Simulation	ISS	

  - Submit as file: **`hw7/ARM7plusHW_Evaluation.pdf`**
  - Submit also the final ISS model: **`hw7/ISS.sir`**

Project Review: Problem Solving…

1. Slow server response and simulator run time
   - Symptoms:
     - Shell commands respond very slowly
     - Simulator runs unreasonably slow (e.g. more than a minute for models above TLM)
   - Analysis:
     - Shared server may be overloaded due to too many users and/or jobs
   - Possible Solutions:
     - Monitor the server load!
     - Use **`uptime`** and/or **`top`** commands
     - Login to server with lowest load
       - **`gamma:   ..., 14 users,  load average: 14.08, 13.47, 11.92`**
       - **`omicron: ...,  0 users,  load average: 0.04, 0.02, 0.00`**
       - **`iota:    ..., 10 users,  load average: 0.00, 0.02, 0.00`**

Project Review: Problem Solving…

2. Large frame delay in ISS model
   - Symptom:
     - Simulation of TLM/PAM models estimated frame delays such as **`Decode time per frame = 24.282 ms`**
     - Simulation of ISS model reports frame delays such as **`Decode time per frame = 64.174 ms`**
   - Analysis:
     - TLM/PAM computation time is only estimated (by SCE profiler)
     - Profiling produces only *fidelity* (not absolute *accuracy*!)
     - Profiling weight tables for ARM7 assume very optimistic cycles
       - Assumptions include zero cache-misses, pipeline stalls, etc.
   - Possible Solutions:
     - Trust the ISS, not the profiler!
     - Calibrate the weight tables at allocation by a factor (e.g. 64.174 / 24.282 ≈ 2.6x)

Project Review: Problem Solving…

3. Communication times can be 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 bus and/or high traffic
       - Indirect communication from slave to slave via master
   - Potential Solutions:
     - Review bus network 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…

4. Incorrect network setup
   - Symptoms:
     - ERROR #5535: Channel instance 'ar_cc__xr0__HW1__HW2' connects two PEs (HW1 and HW2) not reachable with given conectivity
     - ERROR #5533: Channel instance 'ar_cc__ar_tid_III_decode__ARM7__HW1' is not connected to only one PE, which is not allowed by network refinement
   - Analysis:
     - Network refinement fails to map channels to specified network
     - Network specification needs revision
   - Potential Solutions:
     - Ensure channel routing is possible (path exists)
     - Ensure master/slave relationship is obeyed
     - Ensure all behaviors involved are isolated

Project Review: Problem Solving…

5. IS Simulation is stuck
   - Symptoms:
     - PAM model simulates fine
     - Derived ISS model runs but shows no progress even after an hour
   - Analysis:
     - Difficult to diagnose without more data (needs code inspection, instrumentation, debugging)
     - Likely communication between CPU and slaves fails
   - Potential Solutions:
     - Ensure slaves listen to correct addresses
     - ➢ For the AMBA bus in SCE, slave1 listens to 0x1xxx xxxx, slave 2 listens to 0x2xxx xxxx, and so on
     - Ensure scheduling (execution order) across PEs matches (e.g. master sends X but slave waits for Y results in deadlock)

Project Review: Assignment 8

- ARM7-plus-HW Implementation of the MP3 Decoder
  that meets the real-time requirement
  and minimizes HW resources and power
  - Allocate an ARM_7TDMI processor for the software
    - Choose desired clock period (change it from default 10000ps)
  - Allocate up to 6 custom HW units for acceleration
    - Keep default clock frequency of 100 MHz
  - Perform the system design refinement steps
    - Architecture Refinement
    - Scheduling Refinement
    - Network Refinement
    - Communication Refinement
      - Transaction-level model (TLM)
        - » Code generation: TLM_C model
      - Pin-accurate model (PAM)
        - » Instruction Set Simulator (ISS) model
  - Details: **`/home/eecs222/EECS222C_S13/Assignment8.txt`**

Project Review: Assignment 8

- ARM7-plus-HW Implementation of the MP3 Decoder
  that meets the real-time requirement
  and minimizes HW resources and power
  - Fill the following table with the estimated/simulated frame delays

Refinement Step	Model	Decode time per frame
Profiling estimation	Spec	
Architecture Refinement	Arch	
Scheduling Refinement	Sched	
Network Refinement	Net	
Transaction-Level Refinement	TLM, TLM_C	
Pin-Accurate Refinement	PAM	
Instruction Set Simulation	ISS	

  - Submit as file: **`hw8/ARM7plusHW_Evaluation.pdf`**
  - Submit also the final ISS model: **`hw8/ISS.sir`**

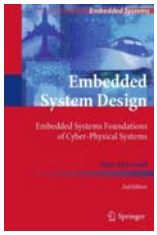
Project Discussion

- Q & A
  - …

Embedded Software

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

➢ Excerpts from Chapter 4.1 in
  - *"Embedded System Design"*
    Embedded Systems Foundations
    of Cyber-Physical Systems
    by P. Marwedel,
    2nd edition, Springer, 2011.



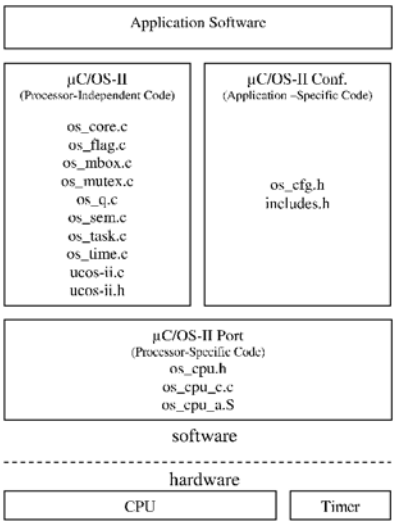
  ➢ **`Lecture10-subset-es-marw-4.1-rtos.pdf`**

Embedded Operating Systems

- Example: MicroC/OS-II
  - Supported 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)

EECS 222C Summary

- 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

Course Administration

- Final Exam
  - Date and time
    - **Wednesday, June 12, 2013, until 12pm (noon)**
  - Format
    - Delivery of Final Technical Report
    - Electronic submission (**`turnin`** on server)
      - **`final/ISS.sir`**
      - **`final/Report.pdf`**
  - ➢Hard deadline!
    - Wednesday, June 12, 2013, 12pm (noon)