EECS 222C: System-on-Chip Software Synthesis Lecture 5

Rainer Dömer

doemer@uci.edu

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

Lecture 5: Overview

- Assignment 3
  - Discussion
- System-on-Chip Design Environment
  - SoC Abstraction Levels
  - Top-down Design Methodology
  - System-on-Chip Environment (SCE)
  - Interactive Demonstration
    - GSM Vocoder, Model Analysis
- Assignment 4

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)

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`**

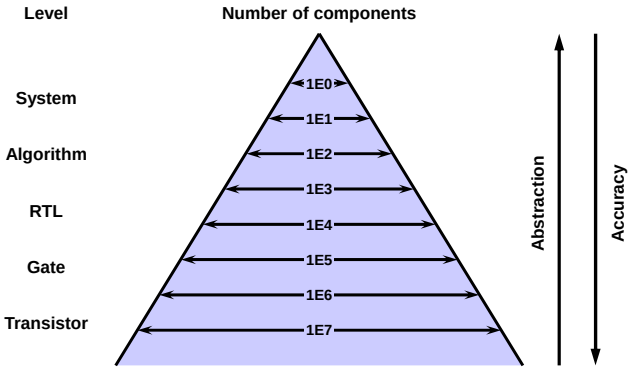
Assignment 3

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"

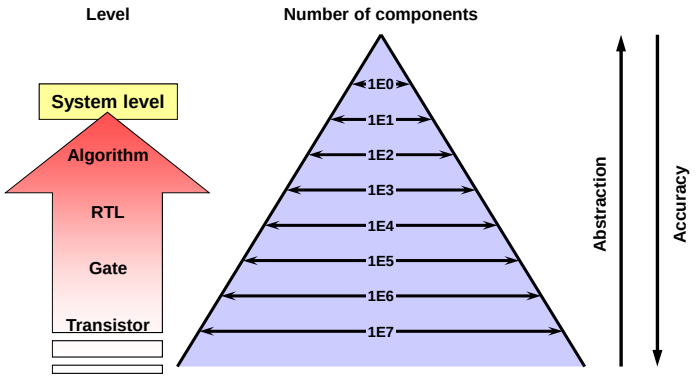
SoC Abstraction Levels

- Embedded system design faces tremendous increase of design complexity

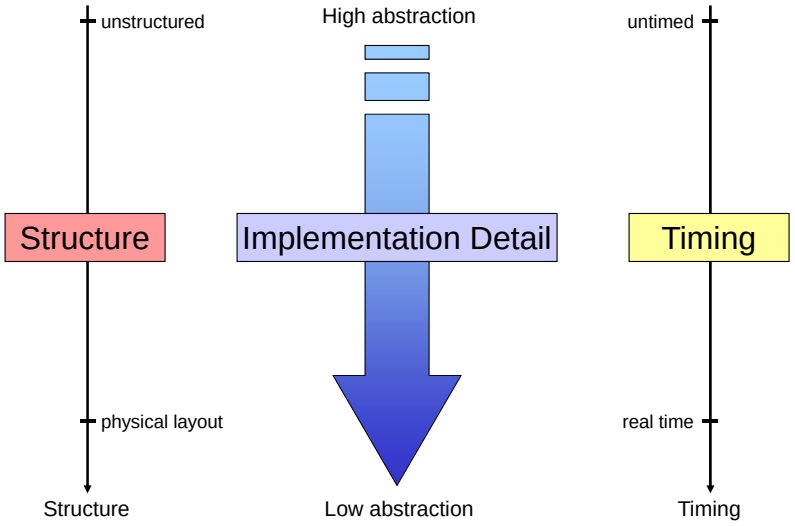


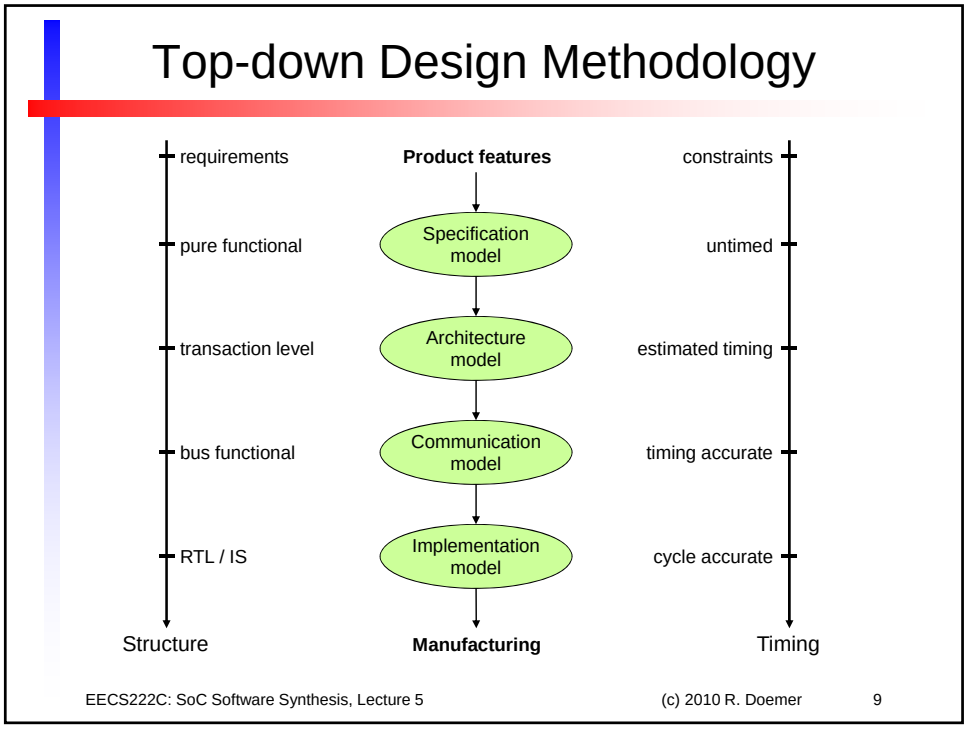
SoC Abstraction Levels

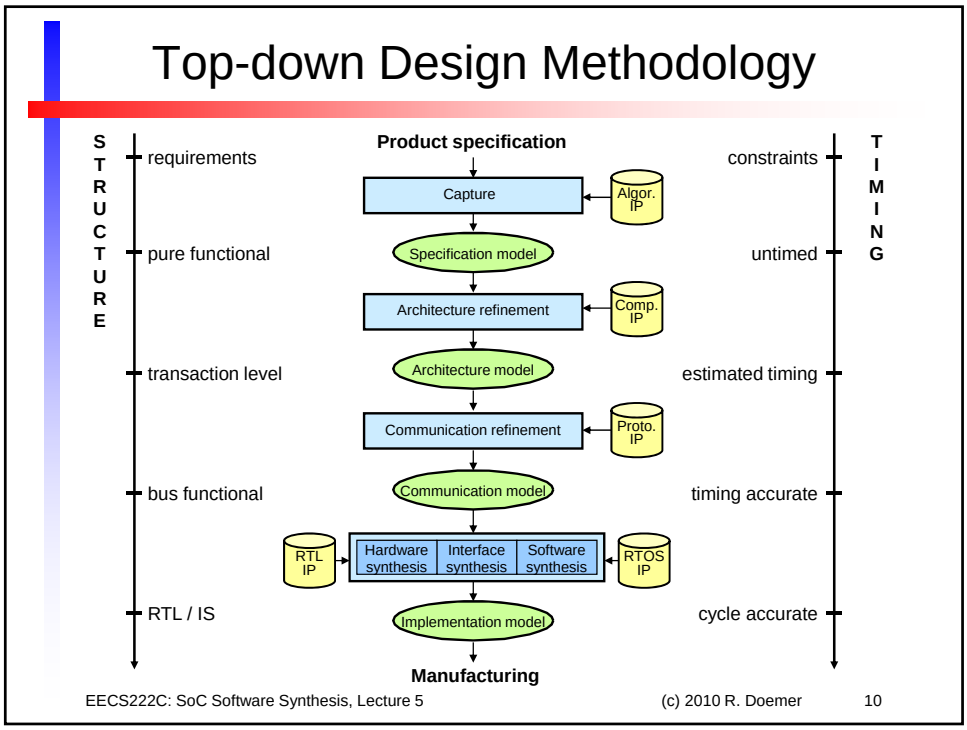
- Embedded system design faces tremendous increase of design complexity
- Move to higher levels of abstraction!



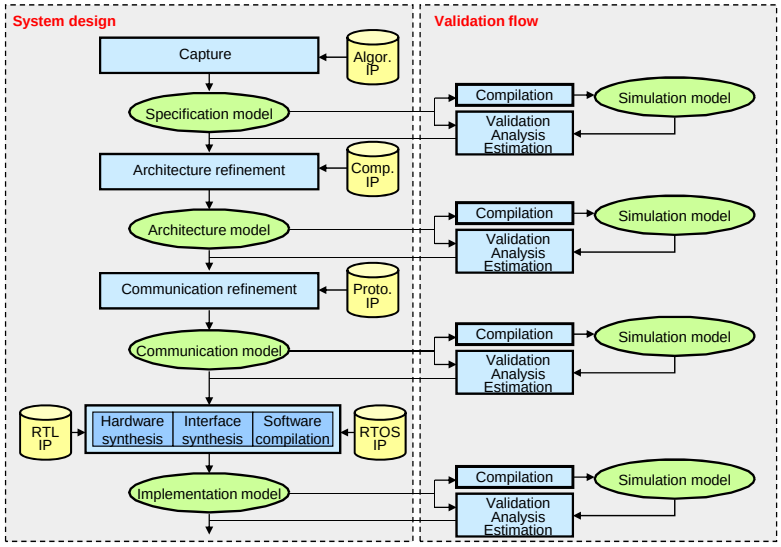
SoC Abstraction Levels




Top-down Design Methodology
requirements
pure functional
transaction level
bus functional
RTL / IS
Structure
Product features
Specification model
Architecture model
Communication model
Implementation model
Manufacturing
constraints
untimed
estimated timing
timing accurate
cycle accurate
Timing
EECS222C: SoC Software Synthesis, Lecture 5
(c) 2010 R. Doemer
9


Top-down Design Methodology
STRUCTURE
requirements
pure functional
transaction level
bus functional
RTL / IS
Product specification
Capture
Algor. IP
Specification model
Architecture refinement
Comp. IP
Architecture model
Communication refinement
Proto. IP
Communication model
RTL IP
Hardware synthesis
Interface synthesis
Software synthesis
RTOS IP
Implementation model
Manufacturing
constraints
untimed
estimated timing
timing accurate
cycle accurate
TIMING
EECS222C: SoC Software Synthesis, Lecture 5
(c) 2010 R. Doemer
10

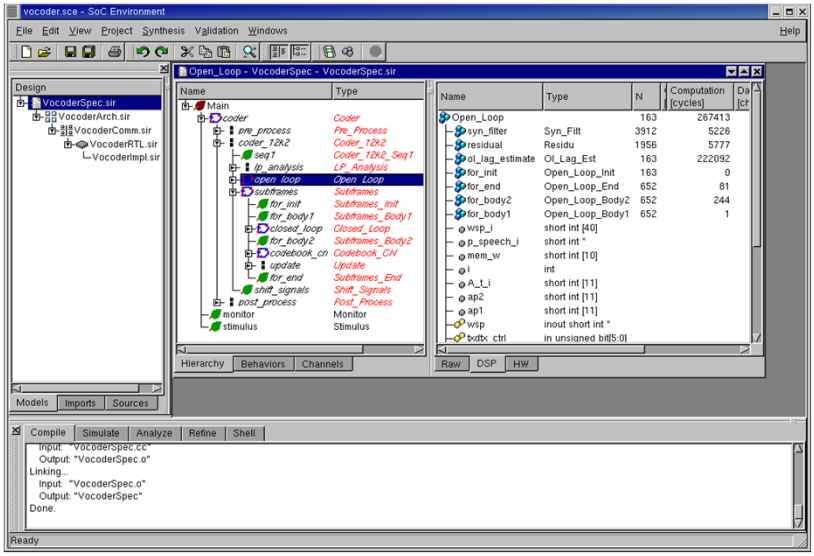
Top-down Design Methodology



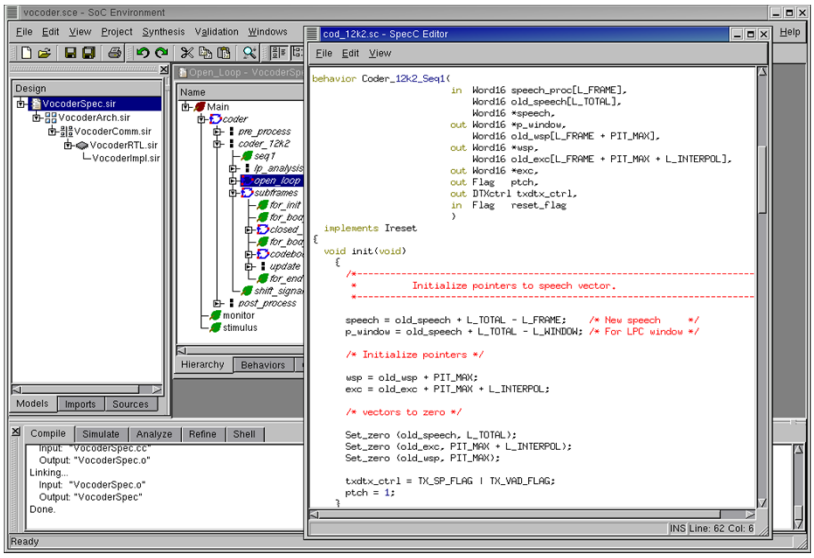
System-on-Chip Environment (SCE)

- Integrated Development Environment (IDE) with support of:
  - Graphical frontend (`sce, scchart`)
  - SLDL-aware editor (`sced`)
  - Compiler and simulator (`scc`)
  - Profiling and analysis (`scprof`)
  - Architecture refinement (`scar`)
  - RTOS refinement (`scos`)
  - Communication refinement (`sccr`)
  - RTL refinement (`scrtl`)
  - Software refinement (`sc2c`)
  - Scripting interface (`scsh`)
  - Tools and utilities ...

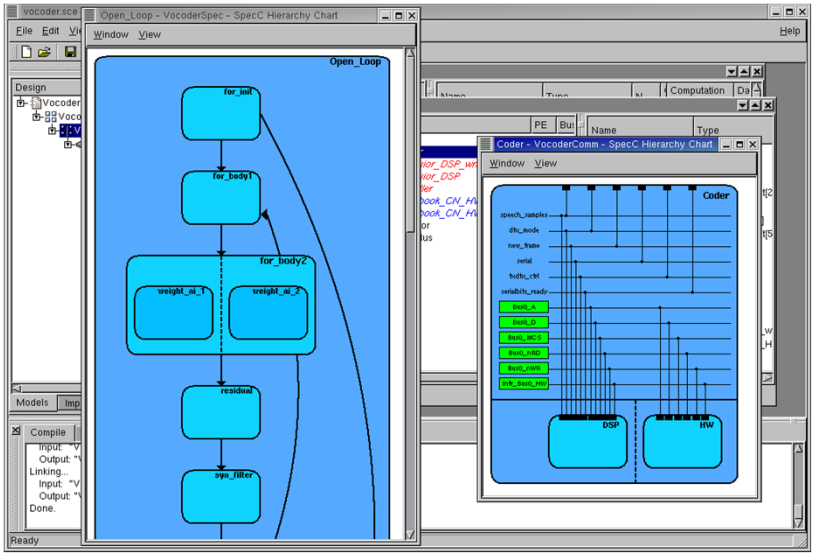
SCE Main Window



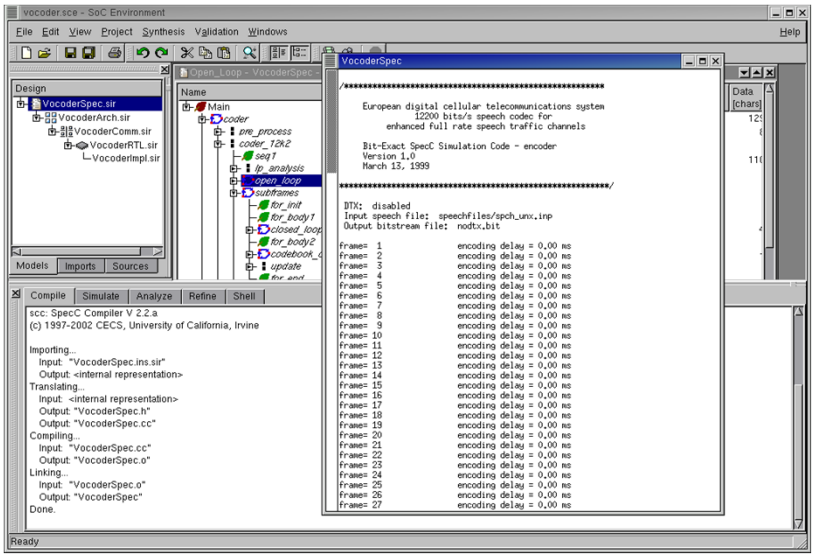
SCE Source Editor



SCE Hierarchy Displays



SCE Compiler and Simulator



SCE Profiling and Analysis

SCE Demonstration

- Design example: GSM Vocoder
  - Enhanced full-rate voice codec
  - GSM standard for mobile telephony (GSM 06.10)
  - Lossy voice encoding/decoding
    - Incoming speech samples @ 104 kbit/s
    - Encoded bit stream @ 12.2 kbit/s
    - Frames of 4 x 40 = 160 samples (4 x 5ms = 20ms of speech)
  - Real-time constraint:
    - max. 20ms per speech frame
      (max. total of 3.26s for sample speech file)
  - SpecC specification model
    - 29 hierarchical behaviors (9 par, 10 seq, 10 fsm)
    - 73 leaf behaviors
    - 9139 formatted lines of SpecC code
      (~13000 lines of original C code, including comments)

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`**

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 &`**

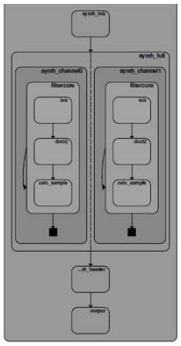
   - Create a new project in SCE
     - **`Project->New`**
     - **`Project->Settings`**
       - Set include path to "**`.`**" (current directory)
       - Set libraries to "**`-xl hufman.o`**"
       - Set both verbosity and warning level to 2
       - In the Simulator tab, set the simulation command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

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`**

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"`**



- Deliverable
  - Hierarchy chart **`"synth.ps"`** (in color!)
  - by Friday, Oct 24, 2008, at noon
  - by email to **`doemer@uci.edu`** with subject "EECS222C HW4"