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. 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`**

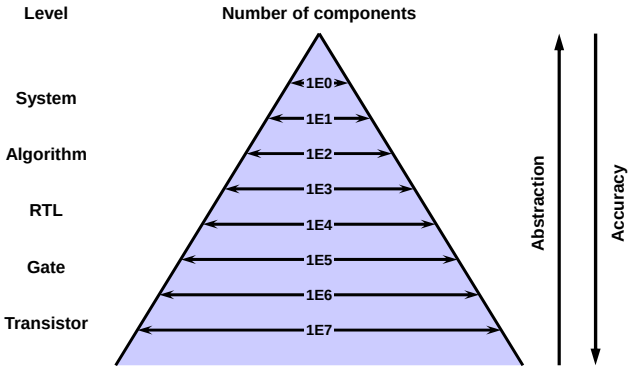
Assignment 3

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

SoC Abstraction Levels

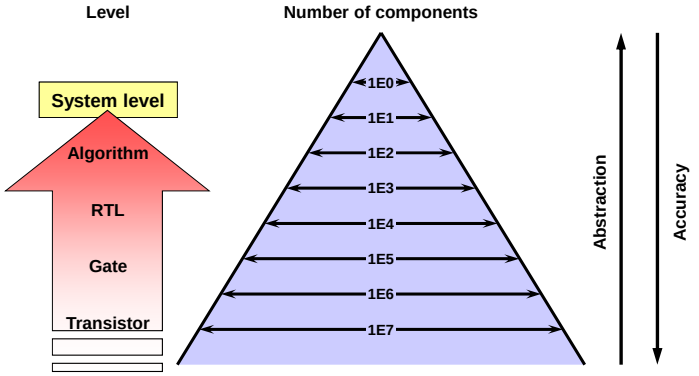
- Embedded system design faces tremendous increase of design complexity



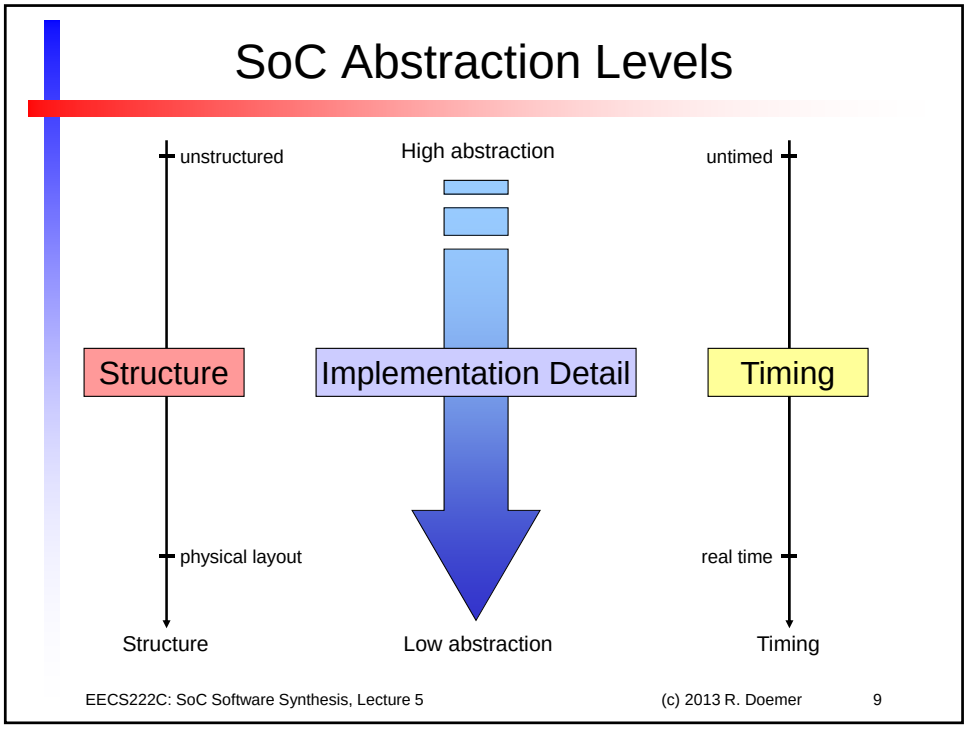
EECS222C: SoC Software Synthesis, Lecture 5 (c) 2013 R. Doemer 7

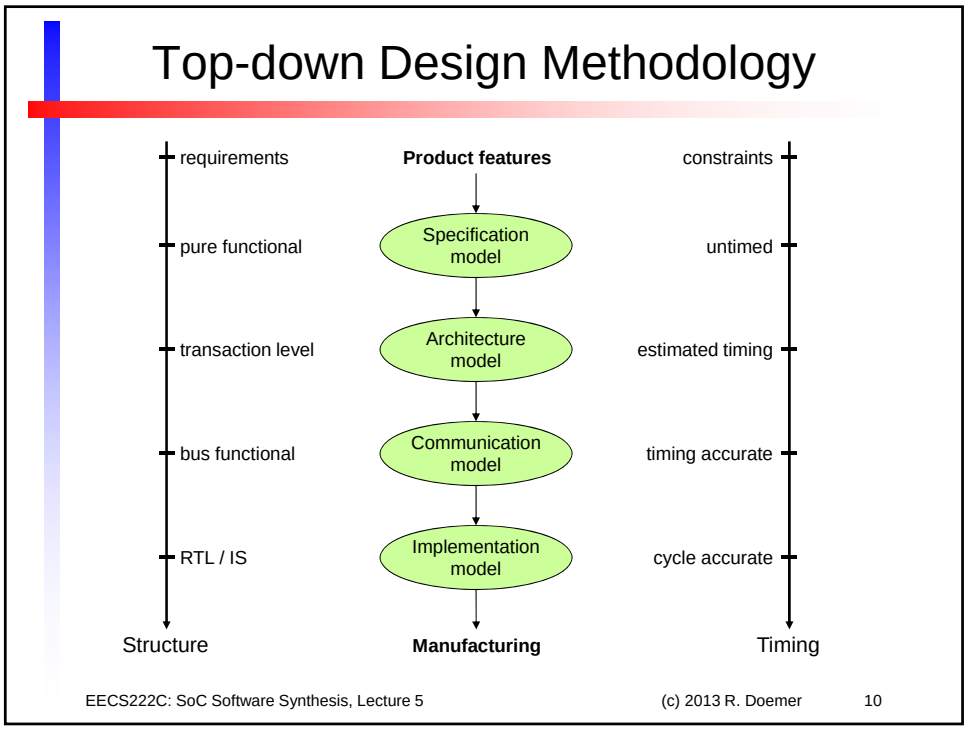
SoC Abstraction Levels

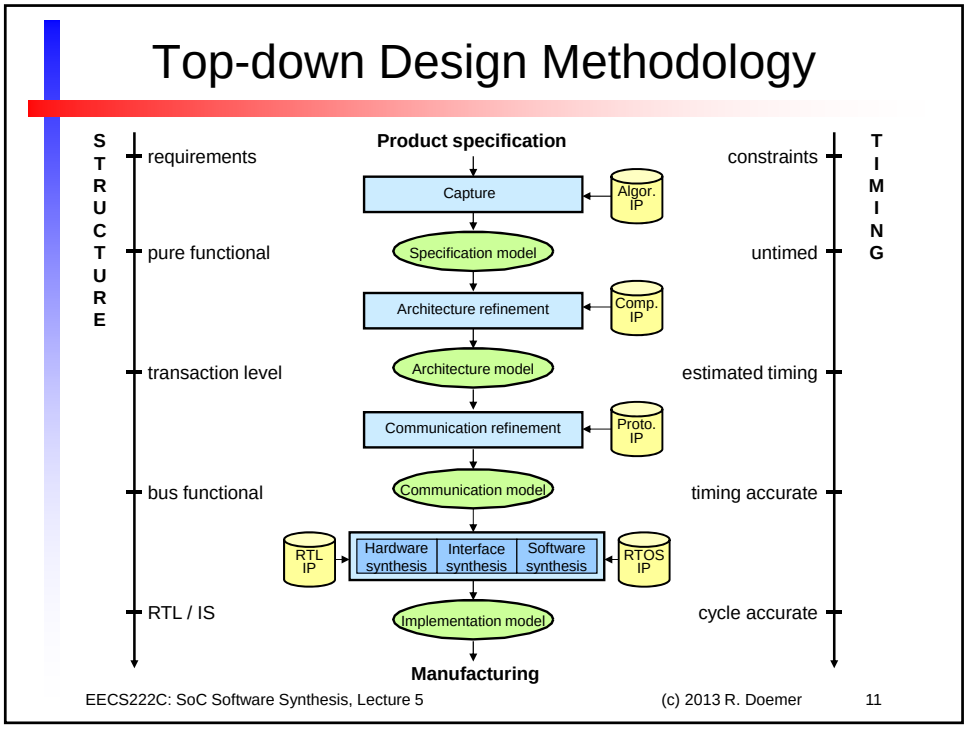
- Embedded system design faces tremendous increase of design complexity
- Move to higher levels of abstraction!

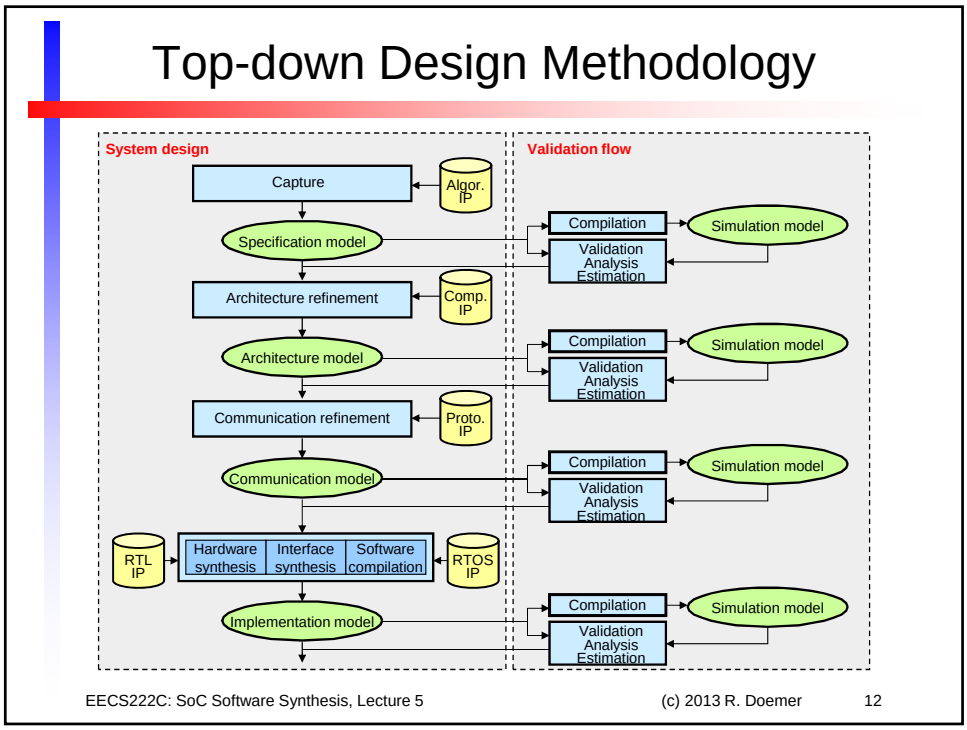


EECS222C: SoC Software Synthesis, Lecture 5 (c) 2013 R. Doemer 8


SoC Abstraction Levels
unstructured
High abstraction
untimed
Structure
Implementation Detail
Timing
physical layout
real time
Structure
Low abstraction
Timing
EECS222C: SoC Software Synthesis, Lecture 5
(c) 2013 R. Doemer
9


Top-down Design Methodology
requirements
Product features
constraints
pure functional
Specification model
untimed
transaction level
Architecture model
estimated timing
bus functional
Communication model
timing accurate
RTL / IS
Implementation model
cycle accurate
Structure
Manufacturing
Timing
EECS222C: SoC Software Synthesis, Lecture 5
(c) 2013 R. Doemer
10

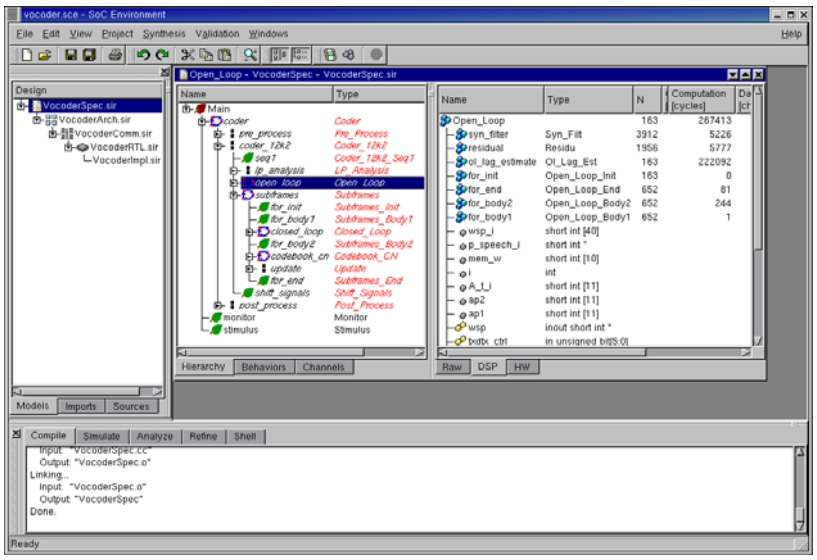

Top-down Design Methodology
STRUCTURE
TIMING
Product specification
requirements
constraints
Capture
Algor. IP
pure functional
untimed
Specification model
Architecture refinement
Comp. IP
transaction level
estimated timing
Architecture model
Communication refinement
Proto. IP
bus functional
timing accurate
Communication model
RTL IP
Hardware synthesis
Interface synthesis
Software synthesis
RTOS IP
RTL / IS
cycle accurate
Implementation model
Manufacturing
EECS222C: SoC Software Synthesis, Lecture 5
(c) 2013 R. Doemer
11


Top-down Design Methodology
System design
Validation flow
Capture
Algor. IP
Specification model
Compilation
Simulation model
Validation Analysis Estimation
Architecture refinement
Comp. IP
Architecture model
Communication refinement
Proto. IP
Communication model
RTL IP
Hardware synthesis
Interface synthesis
Software compilation
RTOS IP
Implementation model
EECS222C: SoC Software Synthesis, Lecture 5
(c) 2013 R. Doemer
12

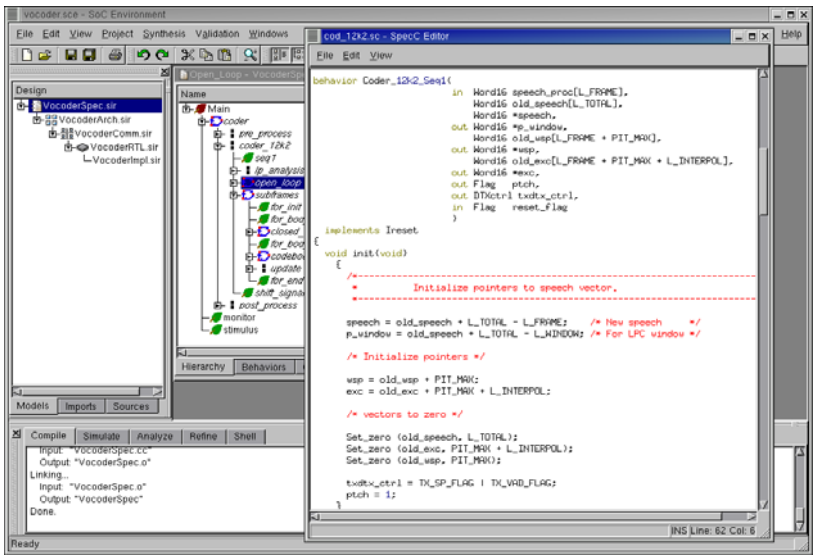
System-on-Chip Environment (SCE)

- 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 ...

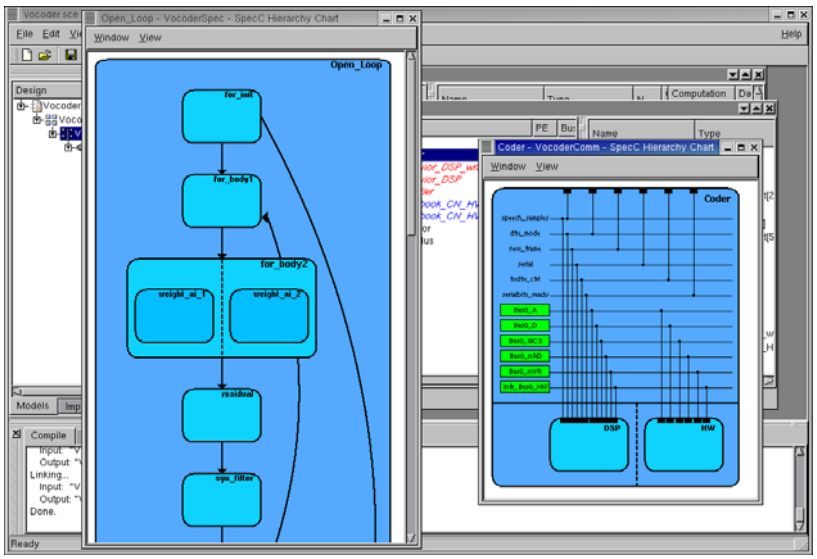
SCE Main Window



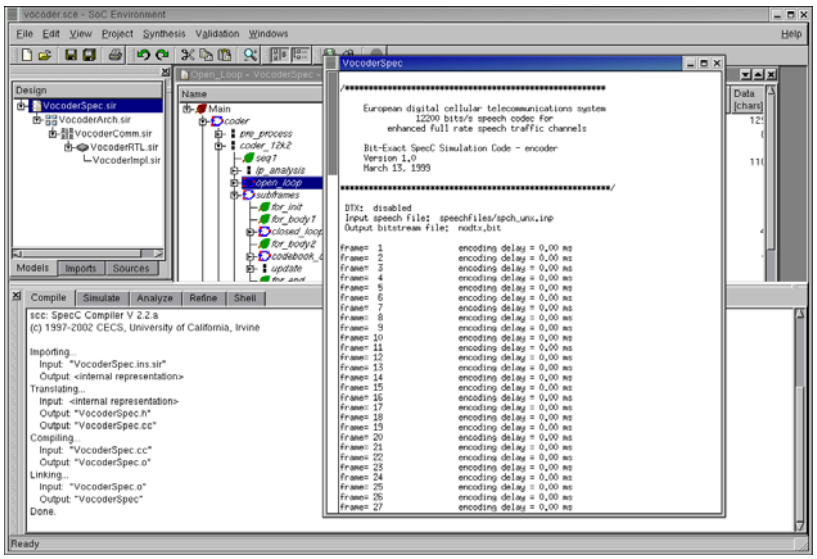
SCE Source Editor



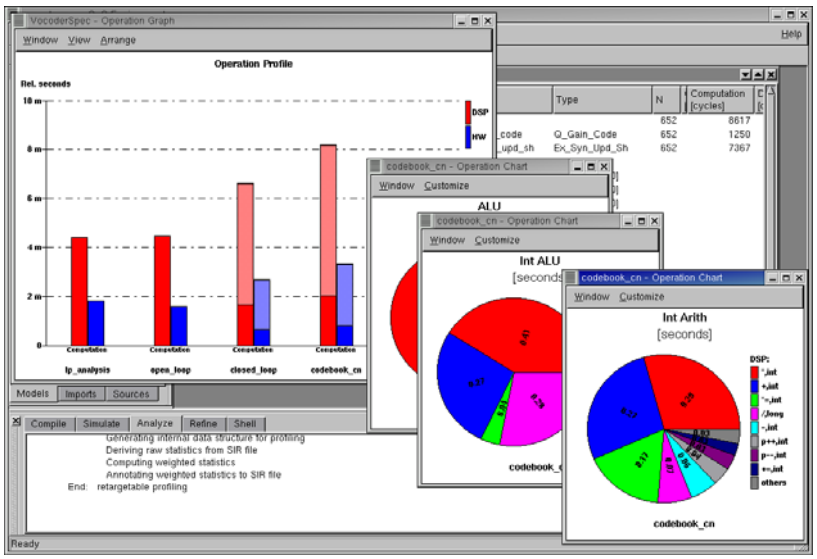
SCE Hierarchy Displays



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`**

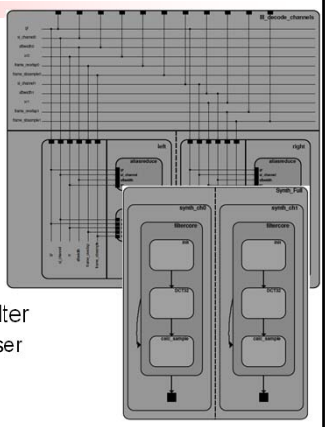
Assignment 4

2. Setup your MP3 Decoder model in SCE
   - Setup SCE
     - Note that we will use the 2010 version of SCE:
     - **`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"`**

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. 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`**