

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`

EECS222C: SoC Software Synthesis, Lecture 5

(c) 2013 R. Doemer

5

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

EECS222C: SoC Software Synthesis, Lecture 5

(c) 2013 R. Doemer

6

SoC Abstraction Levels

- Embedded system design faces tremendous increase of design complexity

Level	Number of components
System	1E0
Algorithm	1E1
RTL	1E2
Gate	1E3
Transistor	1E4
	1E5
	1E6
	1E7

Abstraction ↑

Accuracy ↓

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

SoC Abstraction Levels

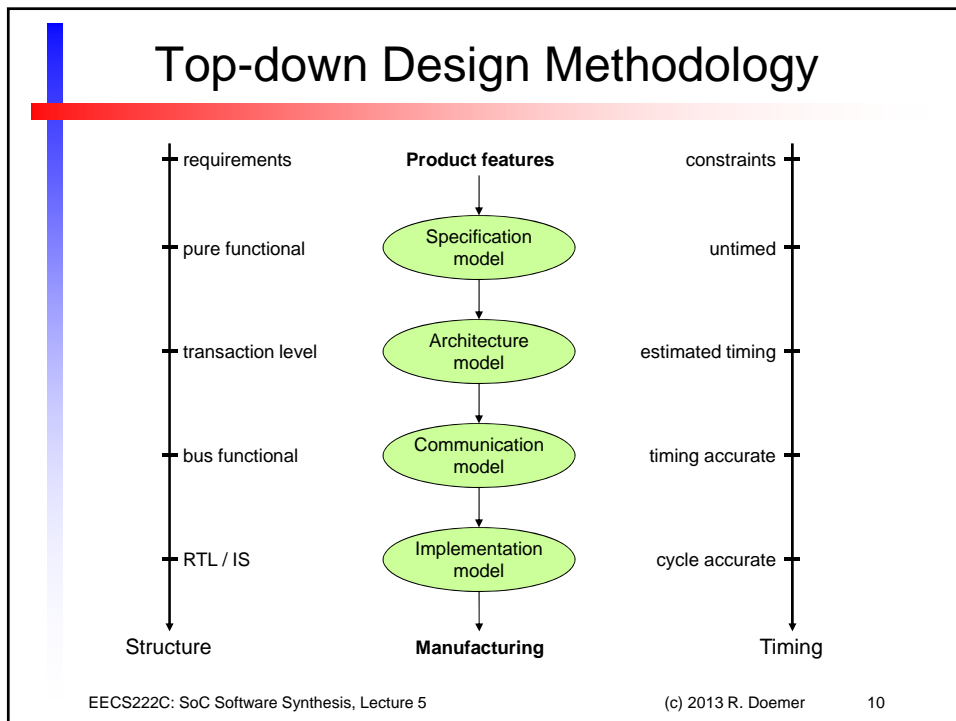
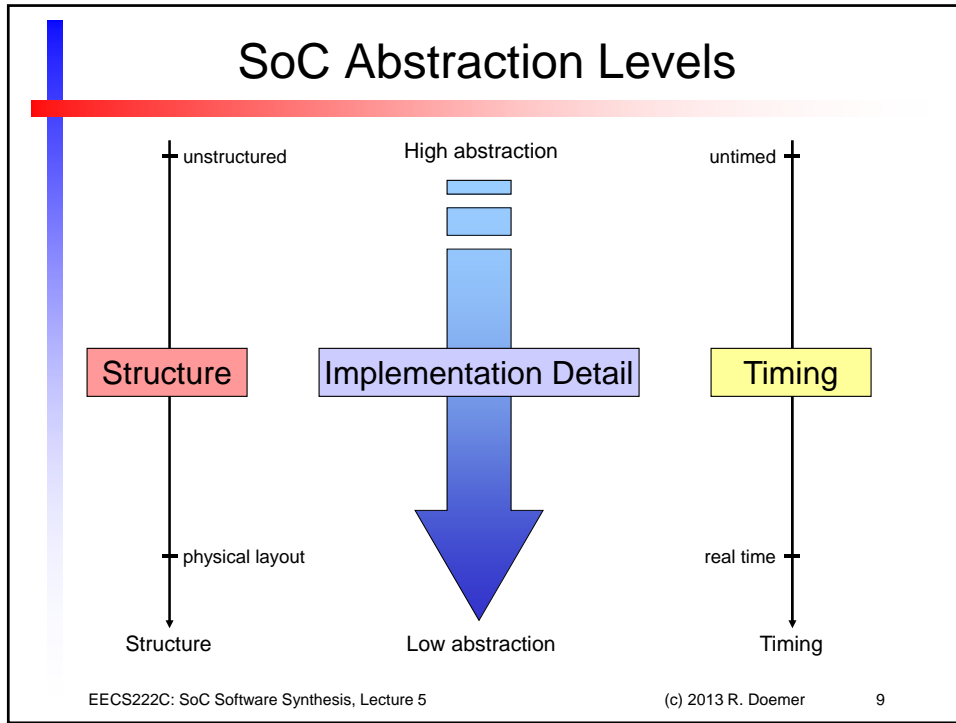
- Embedded system design faces tremendous increase of design complexity
- Move to higher levels of abstraction!

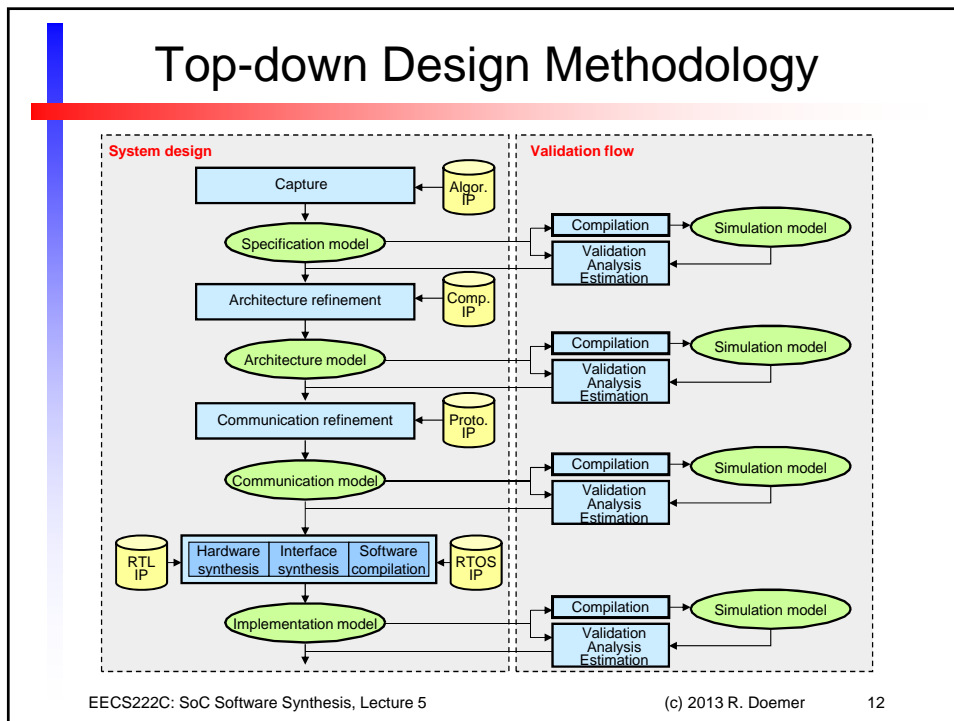
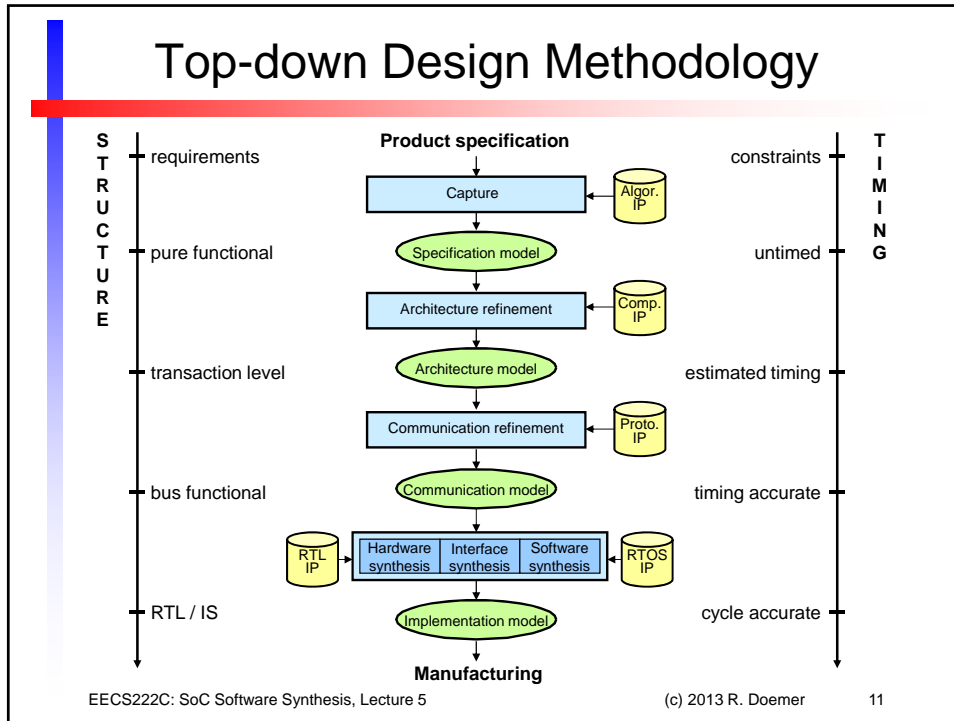
Level	Number of components
System level	1E0
Algorithm	1E1
RTL	1E2
Gate	1E3
Transistor	1E4
	1E5
	1E6
	1E7

Abstraction ↑

Accuracy ↓

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





System-on-Chip Environment (SCE)

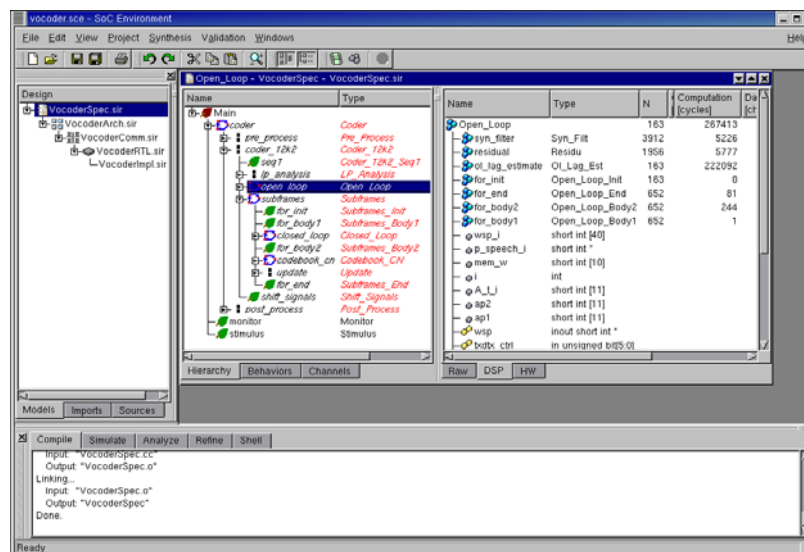
- Integrated Development Environment (IDE) with support of:
 - Graphical frontend (*sce*, *scchart*)
 - SLDL-aware editor (*sced*)
 - Compiler and simulator (*scc*)
 - Profiling and analysis (*scprof*)
 - Architecture refinement (*scar*)
 - RTOS refinement (*scos*)
 - Communication refinement (*sccr*)
 - RTL refinement (*scrt1*)
 - Software refinement (*sc2c*)
 - Scripting interface (*scsh*)
 - Tools and utilities ...

EECS222C: SoC Software Synthesis, Lecture 5

(c) 2013 R. Doemer

13

SCE Main Window



EECS222C: SoC Software Synthesis, Lecture 5

Copyright © 2003 CECS

14

SCE Source Editor

The screenshot shows the SCE Source Editor interface. On the left, a project tree displays the hierarchy: **VocoderSpec.sir** (Main) containing **pre_process**, **coder_12k2**, **post_process**, **monitor**, and **stimulus**. The **coder_12k2** block is expanded to show sub-frames: **for_init**, **for_body**, **for_body2**, **for_body3**, **for_body4**, **for_body5**, **for_body6**, **for_body7**, **for_body8**, **for_body9**, **for_body10**, **for_body11**, **for_body12**, **for_body13**, **for_body14**, **for_body15**, **for_body16**, **for_body17**, **for_body18**, **for_body19**, **for_body20**, **for_body21**, **for_body22**, **for_body23**, **for_body24**, **for_body25**, **for_body26**, **for_body27**, **for_body28**, **for_body29**, **for_body30**, **for_body31**, **for_body32**, **for_body33**, **for_body34**, **for_body35**, **for_body36**, **for_body37**, **for_body38**, **for_body39**, **for_body40**, **for_body41**, **for_body42**, **for_body43**, **for_body44**, **for_body45**, **for_body46**, **for_body47**, **for_body48**, **for_body49**, **for_body50**, **for_body51**, **for_body52**, **for_body53**, **for_body54**, **for_body55**, **for_body56**, **for_body57**, **for_body58**, **for_body59**, **for_body60**, **for_body61**, **for_body62**, **for_body63**, **for_body64**, **for_body65**, **for_body66**, **for_body67**, **for_body68**, **for_body69**, **for_body70**, **for_body71**, **for_body72**, **for_body73**, **for_body74**, **for_body75**, **for_body76**, **for_body77**, **for_body78**, **for_body79**, **for_body80**, **for_body81**, **for_body82**, **for_body83**, **for_body84**, **for_body85**, **for_body86**, **for_body87**, **for_body88**, **for_body89**, **for_body90**, **for_body91**, **for_body92**, **for_body93**, **for_body94**, **for_body95**, **for_body96**, **for_body97**, **for_body98**, **for_body99**, **for_body100**.

The main editor window shows the following code:

```

behavior Coder_12k2_Seq1(
  in  Word16 speech_proc[L_FRAME],
      Word16 old_speech[L_TOTAL],
      Word16 *speech,
  out Word16 *p_window,
      Word16 old_wsp[L_FRAME + PIT_MAK],
  out Word16 *wsp,
      Word16 old_exe[L_FRAME + PIT_MAK + L_INTERPOL],
  out Word16 *exc,
  out Flag patch,
  out DTCtrl txtx_ctrl,
  in  Flag reset_flag
)

implements Ireset
{
  void init(void)
  {
    /* Initialize pointers to speech vector. */

    speech = old_speech + L_TOTAL - L_FRAME; /* New speech */
    p_window = old_speech + L_TOTAL - L_WINDOW; /* For LPC window */

    /* Initialize pointers */

    wsp = old_wsp + PIT_MAK;
    exc = old_exe + PIT_MAK + L_INTERPOL;

    /* vectors to zero */

    Set_zero(old_speech, L_TOTAL);
    Set_zero(old_exe, PIT_MAK + L_INTERPOL);
    Set_zero(old_wsp, PIT_MAK);

    txtx_ctrl = TX_SP_FLAG | TX_VAD_FLAG;
    patch = 1;
  }
}
    
```

At the bottom of the screenshot, the text "EECS222C: SoC Software Synthesis, Lecture 5" and "Copyright © 2003 CECS" is visible.

SCE Hierarchy Displays

The screenshot displays two hierarchy charts. The left chart, titled "Open_Loop - VocoderSpec - SpecC Hierarchy Chart", shows a vertical flow of components: **for_init** → **for_body** → **for_body2** (which contains **weight_at_1** and **weight_at_2**) → **residual** → **spe_filter**. The right chart, titled "Coder - VocoderComm - SpecC Hierarchy Chart", shows a more complex structure with **for_DSP** and **for_DSP2** blocks at the top, connected to a central **Coder** block, which is further connected to **DSP** and **HW** blocks at the bottom.

At the bottom of the screenshot, the text "EECS222C: SoC Software Synthesis, Lecture 5" and "Copyright © 2003 CECS" is visible.

SCE Compiler and Simulator

The screenshot shows the SCE Compiler and Simulator interface. The main window displays the compilation status for 'VocoderSpec'. The output window shows the following text:

```

European digital cellular telecommunications system
12200 bit/s speech codec for
enhanced full rate speech traffic channels

Bit-Exact Spec: Simulation Code - encoder
Version 1.0
March 15, 1999

BITx: disabled
Input speech file: speechfiles/spch_uns.inp
Output bitstream file: routx.bit

Frames 1 encoding delay = 0,00 ms
Frames 2 encoding delay = 0,00 ms
Frames 3 encoding delay = 0,00 ms
Frames 4 encoding delay = 0,00 ms
Frames 5 encoding delay = 0,00 ms
Frames 6 encoding delay = 0,00 ms
Frames 7 encoding delay = 0,00 ms
Frames 8 encoding delay = 0,00 ms
Frames 9 encoding delay = 0,00 ms
Frames 10 encoding delay = 0,00 ms
Frames 11 encoding delay = 0,00 ms
Frames 12 encoding delay = 0,00 ms
Frames 13 encoding delay = 0,00 ms
Frames 14 encoding delay = 0,00 ms
Frames 15 encoding delay = 0,00 ms
Frames 16 encoding delay = 0,00 ms
Frames 17 encoding delay = 0,00 ms
Frames 18 encoding delay = 0,00 ms
Frames 19 encoding delay = 0,00 ms
Frames 20 encoding delay = 0,00 ms
Frames 21 encoding delay = 0,00 ms
Frames 22 encoding delay = 0,00 ms
Frames 23 encoding delay = 0,00 ms
Frames 24 encoding delay = 0,00 ms
Frames 25 encoding delay = 0,00 ms
Frames 26 encoding delay = 0,00 ms
Frames 27 encoding delay = 0,00 ms
    
```

EECS222C: SoC Software Synthesis, Lecture 5 Copyright © 2003 CECS 17

SCE Profiling and Analysis

The screenshot shows the SCE Profiling and Analysis interface. The main window displays the 'Operation Profile' bar chart, which shows the relative seconds for various operations. The chart shows that 'codebook_cn' is the most time-consuming operation, followed by 'closed_loop', 'open_loop', and 'ip_analysis'.

The 'codebook_cn' pie charts show the distribution of operations within this codebook. The 'Int ALU' pie chart shows the following distribution:

Operation	Percentage
codebook_cn	~45%
Int ALU	~35%
Int Arith	~15%
others	~5%

The 'Int Arith' pie chart shows the following distribution:

Operation	Percentage
codebook_cn	~45%
Int Arith	~35%
Int ALU	~15%
others	~5%

EECS222C: SoC Software Synthesis, Lecture 5 Copyright © 2003 CECS 18

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 $4 \times 40 = 160$ samples ($4 \times 5\text{ms} = 20\text{ms}$ 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)

EECS222C: SoC Software Synthesis, Lecture 5

(c) 2013 R. Doemer

19

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`

EECS222C: SoC Software Synthesis, Lecture 5

(c) 2013 R. Doemer

20

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 "-x1 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"**

EECS222C: SoC Software Synthesis, Lecture 5

(c) 2013 R. Doemer

21

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**

EECS222C: SoC Software Synthesis, Lecture 5

(c) 2013 R. Doemer

22

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`

