

EECS 222A: System-on-Chip Description and Modeling Lecture 3

Rainer Dömer

doemer@uci.edu

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

Lecture 3: Overview

- Homework Assignment 1
 - Discussion, Q&A
- System-on-Chip Specification
 - Essential issues
 - Top-down SoC design flow
 - Specification Model
 - Specification Modeling Guidelines
- System-on-Chip Environment (SCE)
 - Demonstration
 - Example: GSM Vocoder
- Homework Assignment 2
 - Tasks

Homework Assignment 1

- Administration
 - Server
 - `epsilon.eecs.uci.edu`
 - Intel Pentium CPU, 3.0 GHz, 1GB RAM
 - RedHat Linux (Fedora Core 4)
 - 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
 - `/opt/sce-20041007/bin/setup.csh`

EECS222A: SoC Description and Modeling, Lecture 3

(c) 2007 R. Doemer

3

Homework Assignment 1

- Task
 - Become familiar with
 - the SpecC compiler, `scc`
 - the SpecC simulator
 - Use `scc` to
 - compile,
 - simulate and
 - modifythe examples found in
`/opt/sce-20041007/examples/simple/`
- Deliverables
 - none (but be prepared for the next assignment)
- Due
 - next week (Week 3)

EECS222A: SoC Description and Modeling, Lecture 3

(c) 2007 R. Doemer

4

Essential Issues in Specification

- An Example ...

Proposed by the project team Product specification Product design by senior analyst
 Product after implementation Product after acceptance by user What the user wanted

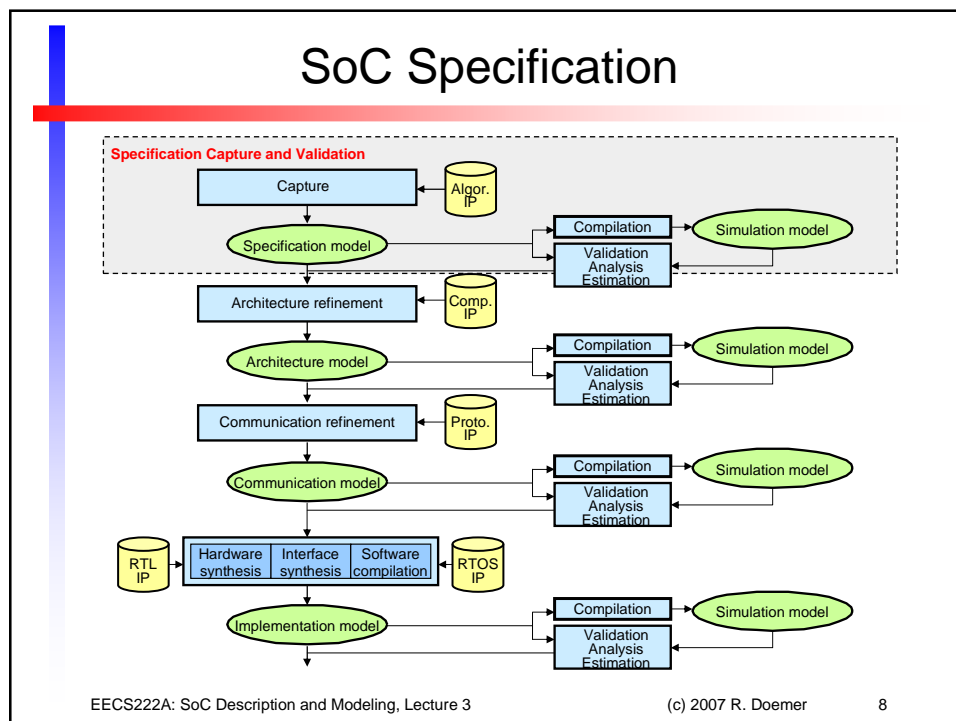
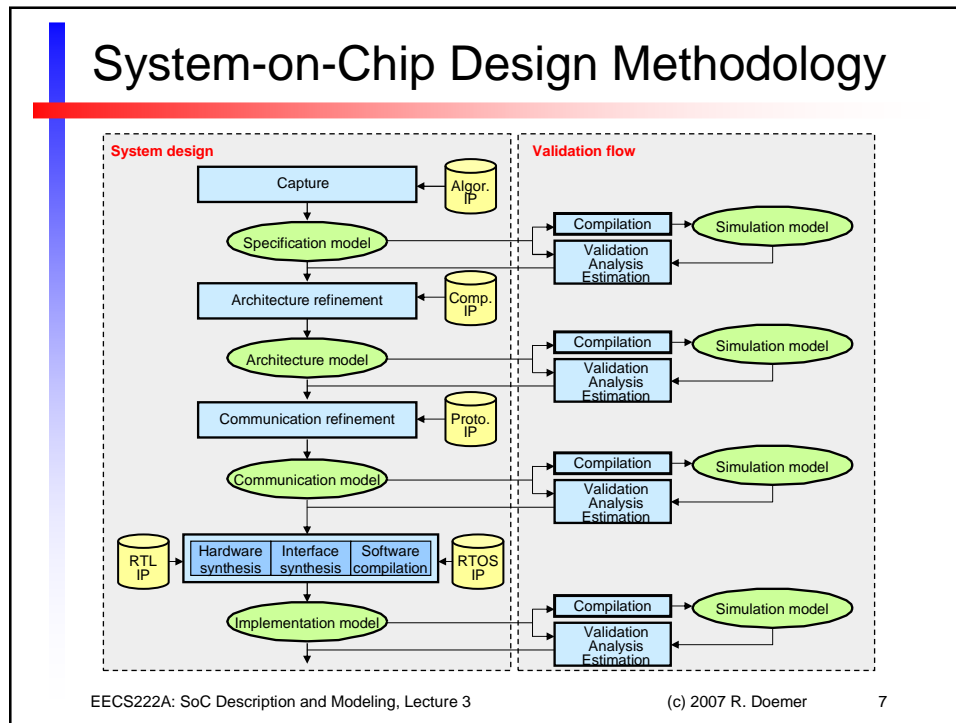
Source: unknown author

EECS222A: SoC Description and Modeling, Lecture 3 (c) 2007 R. Doemer 5

Top-Down SoC Design Flow

S T R U C T U R E	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 Hardware synthesis Interface synthesis Software synthesis ← RTOS IP Implementation model	constraints untimed estimated timing timing accurate cycle accurate	T I M I N G
Manufacturing				

EECS222A: SoC Description and Modeling, Lecture 3 (c) 2007 R. Doemer 6



Specification Model

- High-level, abstract model
 - Pure system functionality
 - Algorithmic behavior
 - No implementation details
- No implicit structure / architecture
 - Pure behavioral hierarchy
- Untimed
 - Execution in zero (logical) time
 - Causal ordering
 - Synchronization

(Source: A. Gerstlauer)

EECS222A: SoC Description and Modeling, Lecture 3
(c) 2007 R. Doemer
9

Specification Model

- Test bench
 - Main, Stimulator, Monitor
 - no restrictions in syntax and semantics (no synthesis)
- Design under test
 - DUT
 - restricted by syntax and semantic rules (synthesis!)

EECS222A: SoC Description and Modeling, Lecture 3
(c) 2007 R. Doemer
10

Specification Modeling Guidelines

- Functional and executable
 - “golden model” (first functional model in the design flow)
 - all other models will be derived from and compared to this one
- High abstraction level
 - no implementation details
 - unrestricted exploration of design space
- Separation of communication and computation
 - channels and behaviors
- Pure functional
 - no structural information
- No timing
 - exception: timing constraints

EECS222A: SoC Description and Modeling, Lecture 3

(c) 2007 R. Doemer

11

Specification Modeling Guidelines

- Computation: Behaviors
 - Hierarchy: explicit concurrency, state transitions, ...
 - Granularity: leaf behaviors = smallest indivisible units
 - Encapsulation: localization, explicit dependencies
 - Concurrency: explicitly specified (par, pipe, fsm, seq, ...)
 - Time: un-timed, partial ordering
- Communication: Channels
 - Semantics: abstract communication, synchronization (standard channel library)
 - Dependencies: explicit data dependency, partial ordering, port connectivity

EECS222A: SoC Description and Modeling, Lecture 3

(c) 2007 R. Doemer

12

Specification Modeling Guidelines

- Example rules for SoC Environment (SCE)
 - Clean behavioral hierarchy
 - hierarchical behaviors:
no code other than par, pipe, seq, fsm, try-trap, ... statements
 - leaf behaviors:
no child behavior calls (basically pure ANSI-C code)
 - Clean communication
 - point-to-point communication via standard channels
 - ports of plain type or interface type, no pointers!
 - port maps to local variables or ports only
- Detailed rules for SoC Environment
 - CECS Technical Report 03-21:
“System-on-Chip Specification Style Guide”
by A. Gerstlauer, K. Ramineni, R. Doemer, D. Gajski
 - http://www.ics.uci.edu/~doemer/publications/CECS_TR_03_21.pdf

EECS222A: SoC Description and Modeling, Lecture 3

(c) 2007 R. Doemer

13

Specification Modeling Guidelines

- Example: C code conversion to SpecC
 - Functions become behaviors or channels
 - Functional hierarchy becomes behavioral hierarchy
 - clean behavioral hierarchy required
 - if-then-else structure becomes FSM
 - while/for/do loops become FSM
 - Explicitly specify potential parallelism
 - Explicitly specify communication
 - avoid global variables
 - use local variables and ports (signals, wires)
 - use standard channels
 - Data types
 - avoid pointers, use arrays instead
 - use explicit SpecC data types if suitable

EECS222A: SoC Description and Modeling, Lecture 3

(c) 2007 R. Doemer

14

System-on-Chip Environment (SCE)

- SCE Components:
 - Graphical frontend (*sce*, *scchart*)
 - 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 ...

EECS222A: SoC Description and Modeling, Lecture 3

(c) 2007 R. Doemer

15

SCE Main Window

The screenshot shows the SCE Main Window interface. The top menu bar includes File, Edit, View, Project, Synthesis, Validation, Windows, and Help. The main workspace is divided into several panes:

- Design:** A tree view showing the project hierarchy for 'VocoderSpec.sir', including sub-projects like 'VocoderArch.sir', 'VocoderComm.sir', 'VocoderRTL.sir', and 'VocoderImpl.sir'.
- Component Table:** A table listing components and their properties. The table has columns for Name, Type, N, and Computation [cycles].
- Console:** A window at the bottom showing the output of the compilation process, including 'Compile', 'Simulate', 'Analyze', 'Refine', and 'Shell' stages.

Name	Type	N	Computation [cycles]
Open_Loop		163	257413
syn_filter	Syn_Filter	3912	5226
residual	Residu	1856	5777
ol_lag_estimate	Ol_Lag_Est	163	222092
for_init	Open_Loop_Init	163	0
for_end	Open_Loop_End	652	81
for_body2	Open_Loop_Body2	652	244
for_body1	Open_Loop_Body1	652	1
wp1	short int [80]		
p_speech_i	short int *		
mem_w	short int [10]		
i	int		
A_i	short int [11]		
ap2	short int [11]		
ap1	short int [11]		
wp	inout short int *		
btbx_ctl	in unsized bits[0]		

EECS222A: SoC Description and Modeling, Lecture 3

Copyright © 2003 CECS

16

SCE Source Editor

The screenshot shows the SCE Source Editor interface. On the left, a design tree displays the hierarchy of components, including 'VocoderSpec', 'VocoderArch', 'VocoderComm', 'VocoderRTL', and 'VocoderImpl'. The main window displays the source code for a behavioral block named 'Coder_12k2_Seq1'. The code includes input and output signals, initialization logic, and a main processing loop.

```

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_MK],
  out  Word16 *wsp,
  Word16 old_exc[L_FRAME + L_INTERPOL],
  out  Word16 *exc,
  out  Flag  patch,
  out  BitCtrl txdtx_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_MK;
  exc = old_exc + PIT_MK + L_INTERPOL;
  /* vectors to zero */
  Set_zero (old_speech, L_TOTAL);
  Set_zero (old_exc, PIT_MK + L_INTERPOL);
  Set_zero (old_wsp, PIT_MK);

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

EECS222A: SoC Description and Modeling, Lecture 3 Copyright © 2003 CECS 17

SCE Hierarchy Displays

The first screenshot shows a block diagram titled 'Open Loop' with a vertical flow of components: 'for_init', 'for_body1', 'for_body2' (containing two 'weight_m' blocks), 'residual', and 'sig_jitter'. The second screenshot shows a signal flow diagram titled 'Coder - VocoderComm - SpecC Hierarchy Chart' with a table of signals and their connections to 'DSP' and 'HW' blocks.

FE	Bus	Name	Type
speech_ctrls		speech_ctrls	
exc_wsp		exc_wsp	
wsp		wsp	
old_speech		old_speech	
weight_m		weight_m	
old_exc		old_exc	
old_wsp		old_wsp	
old_speech_ctrl		old_speech_ctrl	
old_exc_ctrl		old_exc_ctrl	
		DSP	
		HW	

EECS222A: SoC Description and Modeling, Lecture 3 Copyright © 2003 CECS 18

SCE Compiler and Simulator

European digital cellular telecommunications system
 enhanced full rate speech traffic channels
 Bit-Exact SpecC Simulation Code - encoder
 Version 1.0
 March 15, 1999

ITX: disabled
 Input speech file: speech_unv.inp
 Output bitstream file: rootv.bit

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

EECS222A: SoC Description and Modeling, Lecture 3 Copyright © 2003 CECS 19

SCE Profiling and Analysis

Type	N	Computation [cycles]
code	Q_Gain_Code	852 8817
upd_sh	Ex_Syn_Upd_Sh	852 1250
		852 7387

EECS222A: SoC Description and Modeling, Lecture 3 Copyright © 2003 CECS 20

Specification Model 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)

EECS222A: SoC Description and Modeling, Lecture 3

(c) 2007 R. Doemer

21

Homework Assignment 2

- Administration
 - Server
 - `epsilon.eecs.uci.edu`
 - Intel Pentium CPU, 3.0 GHz, 1GB RAM
 - RedHat Linux (Fedora Core 4)
 - 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)
 - System-on-Chip Environment
 - `/opt/sce-20041007/bin/setup.csh`
 - `/opt/sce-20060301/bin/setup.csh`

EECS222A: SoC Description and Modeling, Lecture 3

(c) 2007 R. Doemer

22

Homework Assignment 2

- Task
 - Become familiar with
 - the System-on-Chip Environment (SCE)
 - Follow the initial steps in the SCE Tutorial
 - `/opt/sce-20041007/doc/SCE_Tutorial/sce-tutorial.pdf`
- Deliverables
 - none (but be prepared for the next assignment)
- Due
 - next week (Week 4)