EECS 222A: System-on-Chip Description and Modeling Lecture 6

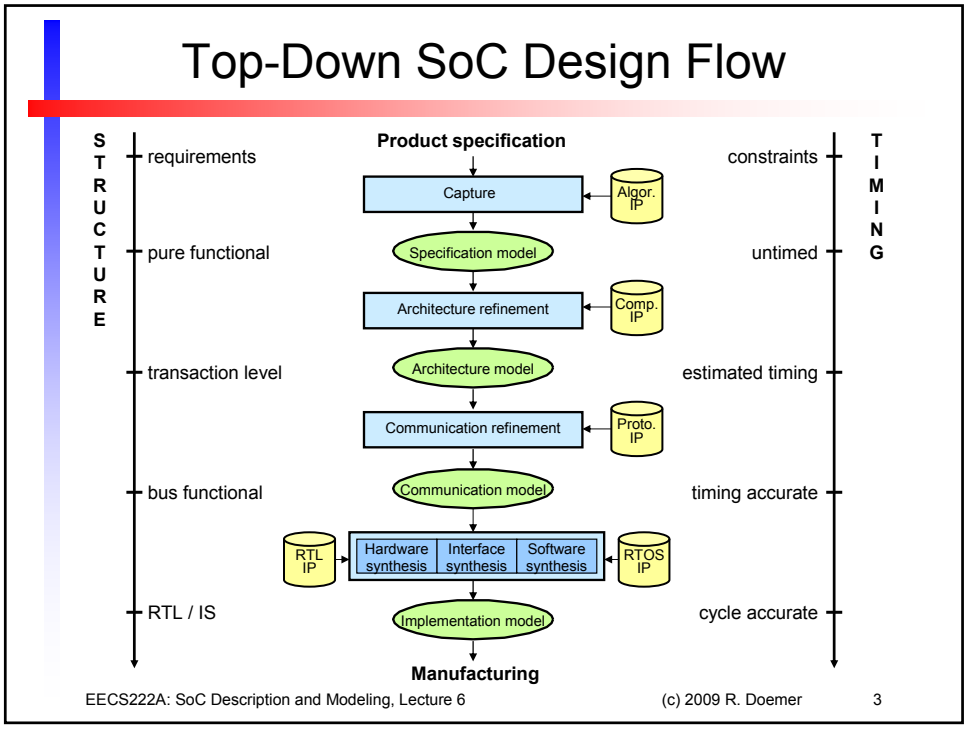
Rainer Dömer

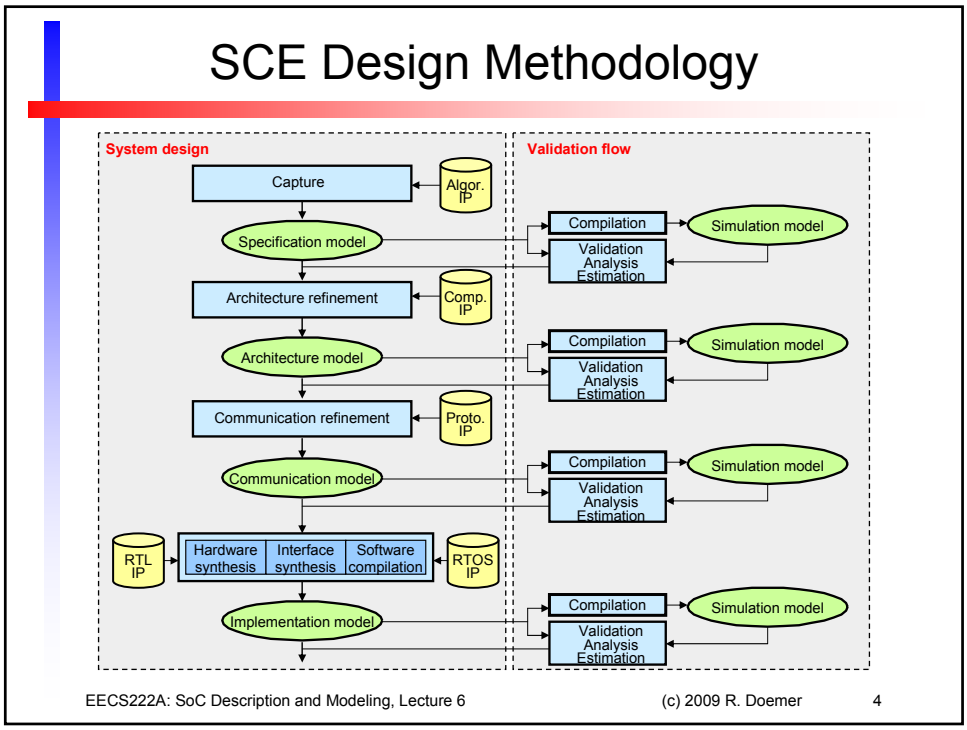
doemer@uci.edu

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

Lecture 6: Overview

- System-on-Chip Design Flow
  - Top-down design methodology
  - Refinement-based design flow
    - Architecture and Scheduling
    - Communication
    - Implementation
      - Hardware synthesis
      - Software generation
- System-on-Chip Environment (SCE)
  - Interactive Demonstration
  - Design Example: GSM Vocoder
- Homework Assignment 3
  - Discussion, Q&A
- Homework Assignment 4


Top-Down SoC Design Flow
STRUCTURE
TIMING
requirements
pure functional
transaction level
bus functional
RTL / IS
constraints
untimed
estimated timing
timing accurate
cycle accurate
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
EECS222A: SoC Description and Modeling, Lecture 6
(c) 2009 R. Doemer
3


SCE Design Methodology
System design
Validation flow
Capture
Algor. IP
Specification model
Architecture refinement
Comp. IP
Architecture model
Communication refinement
Proto. IP
Communication model
RTL IP
Hardware synthesis
Interface synthesis
Software compilation
RTOS IP
Implementation model
Compilation
Validation Analysis Estimation
Simulation model
EECS222A: SoC Description and Modeling, Lecture 6
(c) 2009 R. Doemer
4

SCE Design Methodology

- Four levels of abstraction
  - Specification model: untimed, functional
  - Architecture model: estimated, structural
  - Communication model: timed, bus-functional
  - Implementation model: cycle-accurate, RTL/IS
- Five data bases
  - Algorithms for specification
  - Components for architecture
  - Busses for communication
  - RTOS for SW
  - RTL components for HW
- Three refinement steps
  - Architecture refinement (Specification -> Architecture)
  - Communication refinement (Architecture -> Communication)
  - Cycle-accurate refinement (Communication -> RTL/IS)
    - HW / SW / interface synthesis

Refinement-based System Design Flow

- Step 1: Architecture Refinement
  - Allocation of Processing Elements (PE)
    - Type and number of processors
    - Type and number of custom hardware blocks
    - Type and number 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 in channels

Refinement-based System Design Flow

- Step 2: Scheduling Refinement
  - For each PE, serialize the execution of behaviors to a single thread of control
  - Option (a): Static scheduling
    - For each set of concurrent behaviors, determine 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 scheduling priority
  - Result:
    System model with abstract RTOS scheduler inserted in each PE

Refinement-based System Design Flow

- Step 3: Communication Refinement
  - Allocation of system busses
    - Type and number of system busses
    - Type of bus protocol for each bus (if applicable)
    - Number of transducers (if applicable)
    - System connectivity
  - Mapping of channels to busses
    - Map each communication channel to a system bus (or multiple busses, if applicable)
  - Result:
    Bus-functional model of the system

Refinement-based System Design Flow

- Step 4: Hardware Refinement (for HW PE)
  - Allocation of Register Transfer Level (RTL) components
    - Type and number of functional units (e.g. adder, multiplier, ALU)
    - Type and number of storage units (e.g. registers, register files)
    - Type and number 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:
    Clock-cycle accurate model of each HW PE
  - Output: Synthesizable Verilog description

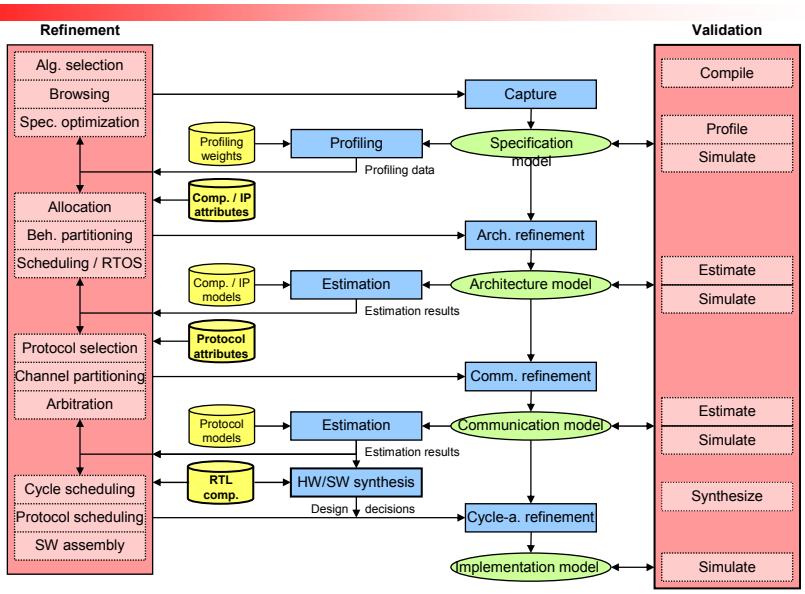
Refinement-based System Design Flow

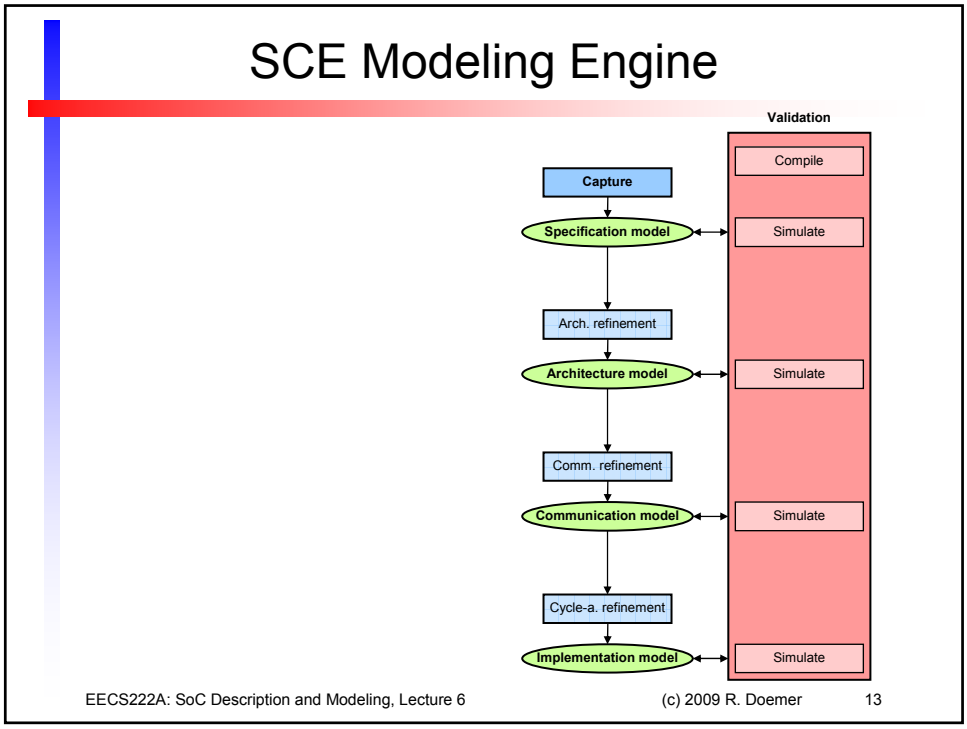
- Step 5: Software Refinement (for SW PE)
  - C code generation
    - For selected target processor
  - RTOS targeting
    - For selected target RTOS
  - Compilation to Instruction Set Architecture
    - for Instruction Set Simulation (ISS)
  - Assembly
  - Result:
    Clock-cycle accurate model of each SW PE
  - Output: downloadable binary image

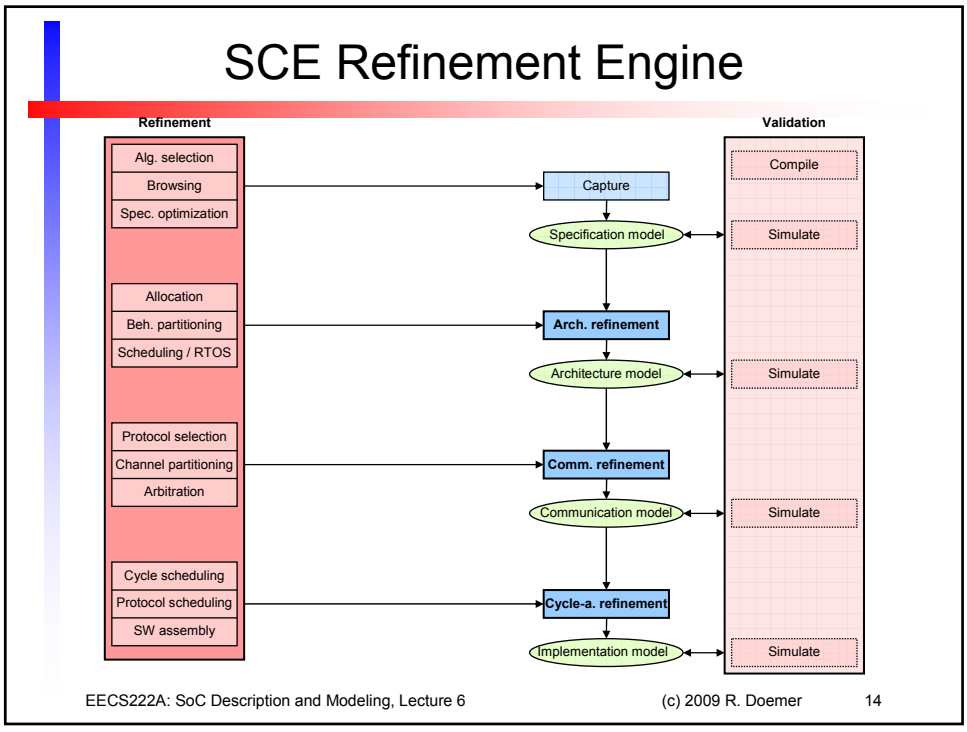
System-on-Chip Environment (SCE)

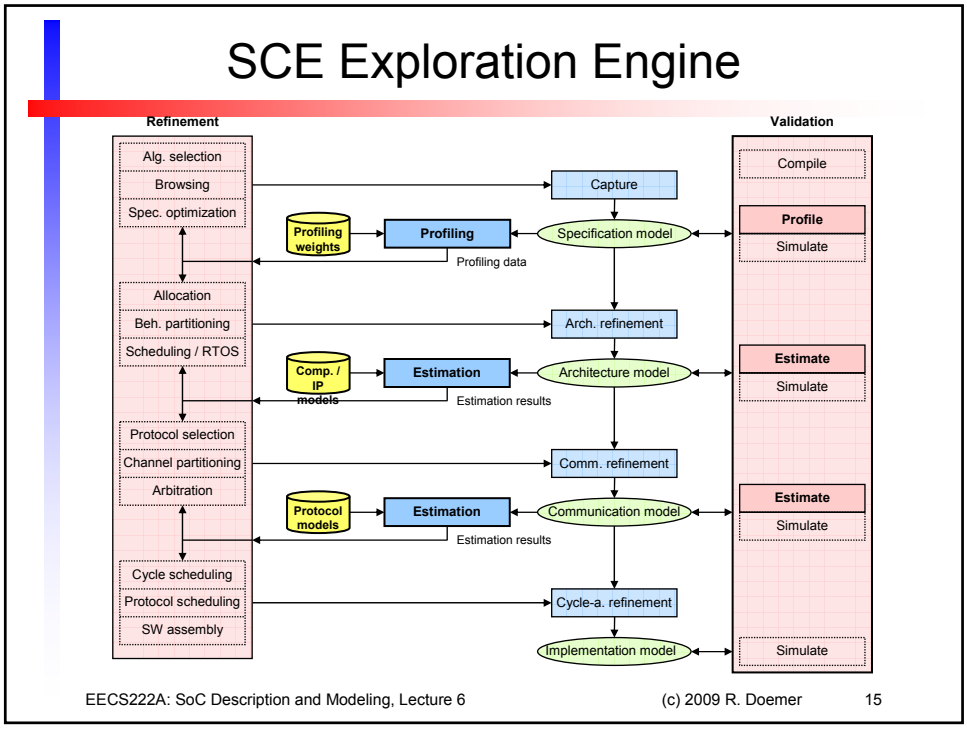
- Integrated Development Environment (IDE) with support of:
  - Graphical frontend (`sce, scchart`)
  - SLDL-aware editor (`sced`)
  - Compiler and simulator (`scc`)
  - Profiling and analysis (`scprof`)
  - Architecture refinement (`scar`)
  - RTOS refinement (`scos`)
  - Communication refinement (`sccr`)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

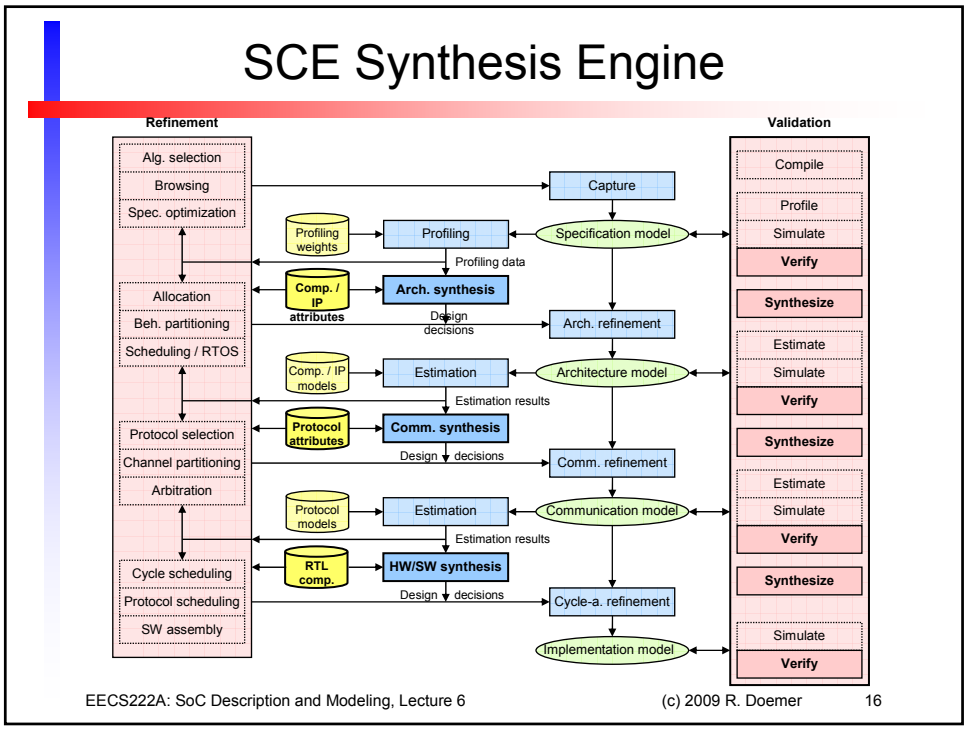
System-on-Chip Environment (SCE)



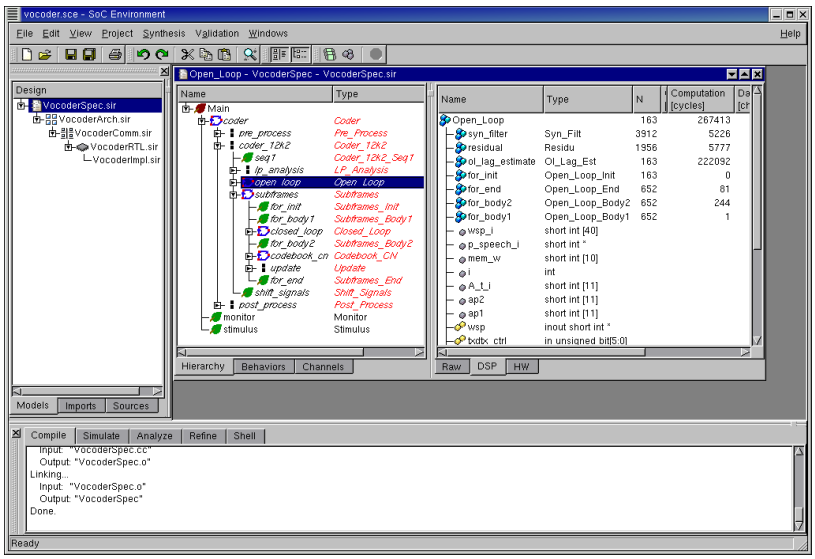

SCE Modeling Engine
Validation
Compile
Capture
Specification model
Simulate
Arch. refinement
Architecture model
Simulate
Comm. refinement
Communication model
Simulate
Cycle-a. refinement
Implementation model
Simulate
EECS222A: SoC Description and Modeling, Lecture 6
(c) 2009 R. Doemer
13


SCE Refinement Engine
Refinement
Validation
Alg. selection
Browsing
Spec. optimization
Allocation
Beh. partitioning
Scheduling / RTOS
Protocol selection
Channel partitioning
Arbitration
Cycle scheduling
Protocol scheduling
SW assembly
Compile
Capture
Specification model
Simulate
Arch. refinement
Architecture model
Simulate
Comm. refinement
Communication model
Simulate
Cycle-a. refinement
Implementation model
Simulate
EECS222A: SoC Description and Modeling, Lecture 6
(c) 2009 R. Doemer
14

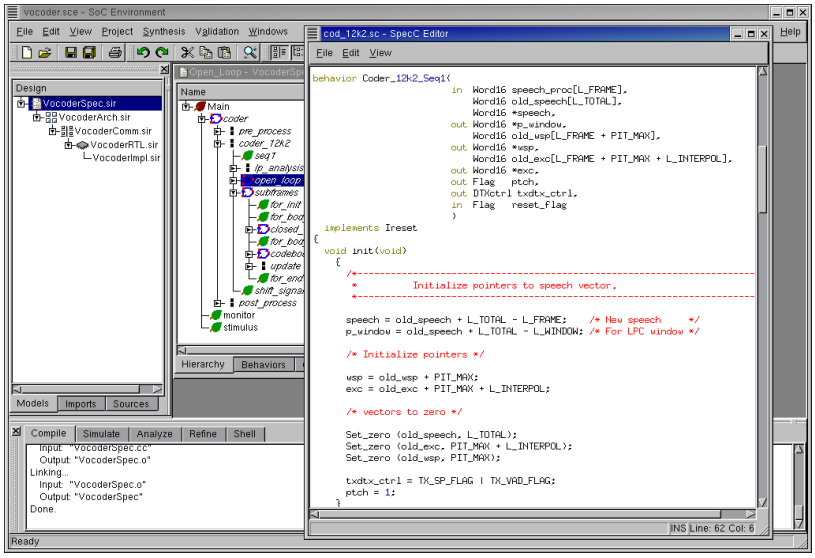

SCE Exploration Engine
Refinement
Validation
Alg. selection
Browsing
Spec. optimization
Allocation
Beh. partitioning
Scheduling / RTOS
Protocol selection
Channel partitioning
Arbitration
Cycle scheduling
Protocol scheduling
SW assembly
Capture
Profiling weights
Profiling
Specification model
Profiling data
Arch. refinement
Comp. / IP models
Estimation
Architecture model
Estimation results
Comm. refinement
Protocol models
Estimation
Communication model
Estimation results
Cycle-a. refinement
Implementation model
Compile
Profile
Simulate
Estimate
Simulate
Estimate
Simulate
Simulate
EECS222A: SoC Description and Modeling, Lecture 6
(c) 2009 R. Doemer
15


SCE Synthesis Engine
Refinement
Validation
Alg. selection
Browsing
Spec. optimization
Allocation
Beh. partitioning
Scheduling / RTOS
Protocol selection
Channel partitioning
Arbitration
Cycle scheduling
Protocol scheduling
SW assembly
Capture
Profiling weights
Profiling
Specification model
Profiling data
Comp. / IP attributes
Arch. synthesis
Design decisions
Arch. refinement
Comp. / IP models
Estimation
Architecture model
Estimation results
Protocol attributes
Comm. synthesis
Design decisions
Comm. refinement
Protocol models
Estimation
Communication model
Estimation results
RTL comp.
HW/SW synthesis
Design decisions
Cycle-a. refinement
Implementation model
Compile
Profile
Simulate
Verify
Synthesize
Estimate
Simulate
Verify
Synthesize
Estimate
Simulate
Verify
Synthesize
Simulate
Verify
EECS222A: SoC Description and Modeling, Lecture 6
(c) 2009 R. Doemer
16

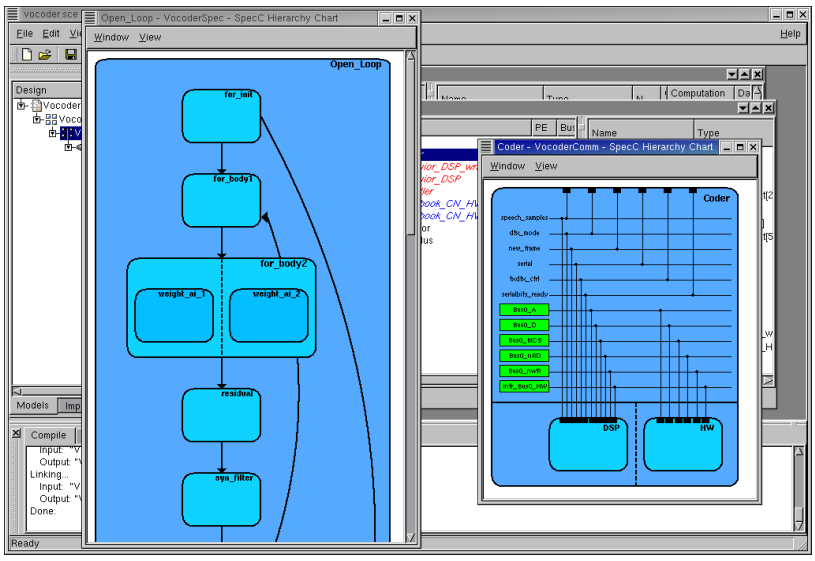
SCE Main Window



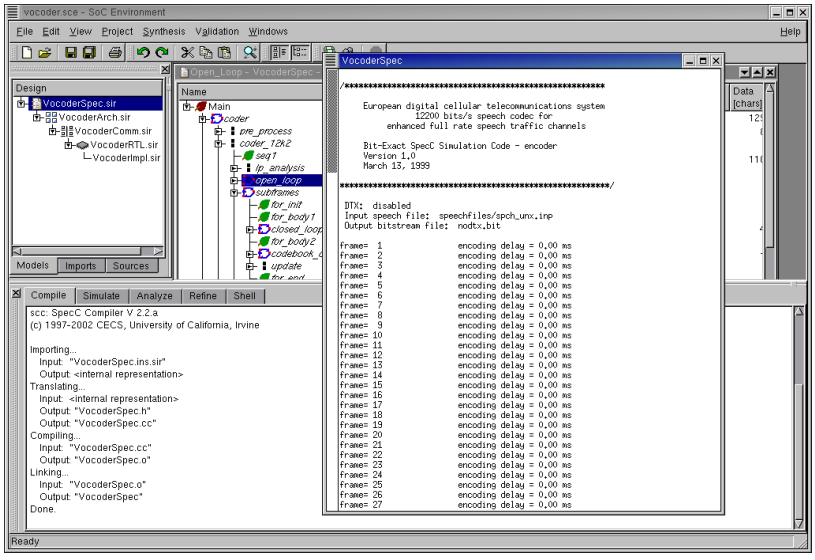
SCE Source Editor



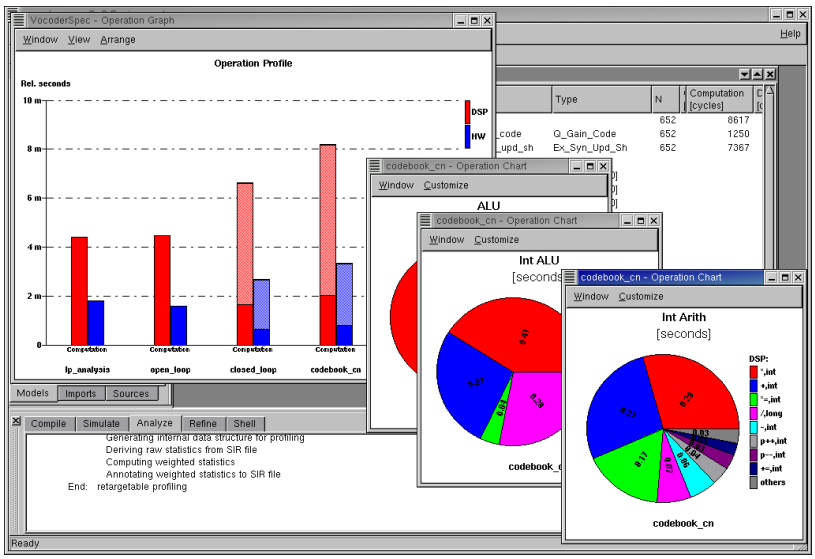
SCE Hierarchy Displays



SCE Compiler and Simulator



SCE Profiling and Analysis



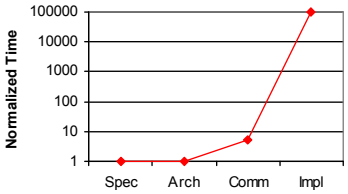
SCE Application Design Example

- 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)

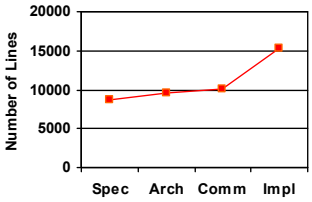
SCE Experimental Results

- Design Example 1: GSM Vocoder

Simulation Speed



Code Size



Refinement Effort

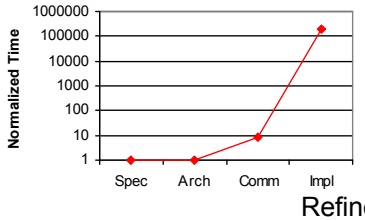
	Modified lines	Manual	Automated User / Refine
Spec → Arch	3,275	3~4 month	15 min / < 1 min
Arch → Comm	914	1~2 month	5 min / < 0.5 min
Comm → Impl	6,146	5~6 month	30 min / < 2 min
Total	10,355	9~12 month	50 min / < 4 min

- Productivity gain: 12 months vs. 1 hour = 2000x !

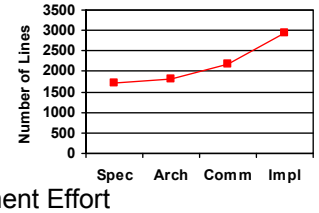
SCE Experimental Results

- Design Example 2: JPEG Encoder

Simulation Speed



Code Size



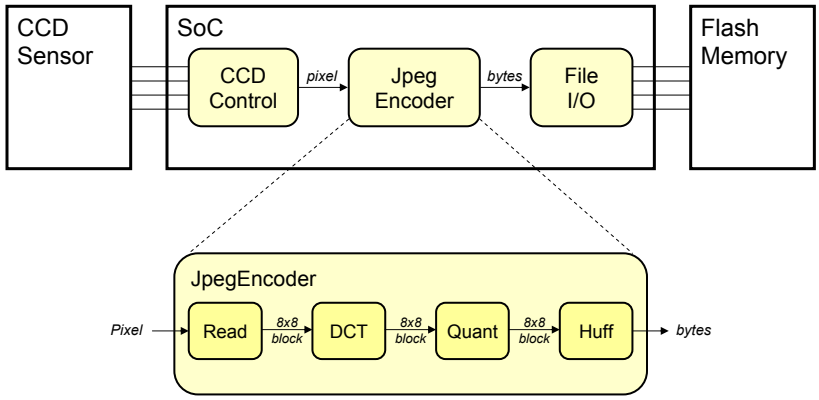
Refinement Effort

	Modified lines	Manual	Automated User / Refine
Spec → Arch	751	1~2 month	5 min / < 0.5 min
Arch → Comm	492	~1 month	3 min / < 0.5 min
Comm → Impl	1,278	3~4 month	20 min / < 1 min
Total	2,521	5~7 month	28 min / < 2 min

- Productivity gain: 6 months vs. 1/2 hour = 2000x !

Project Discussion

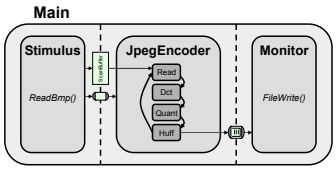
- Digital Camera Example
  - Component Model



EECS222A: SoC Description and Modeling, Lecture 6 (c) 2009 R. Doemer 25

Homework Assignment 3

- Task
  - Convert JPEG Encoder Application to a proper SpecC Specification Model



- Deliverables
  - Email to **`doemer@uci.edu`** with subject "EECS222A Assignment 3"
    - Brief status description (in body of your email)
    - Source code **`jpegencoder.tar.gz`** (attachment)
- Due
  - Next week: October 30, 2009, 12pm (noon)

EECS222A: SoC Description and Modeling, Lecture 6 (c) 2009 R. Doemer 26

Homework Assignment 3

- Hint:
  - Use the **`sir_tree`** tool to validate your hierarchy
  - The final model should look like this:

```
doemer@epsilon.eecs.uci.edu:3 > sir_tree -blt digicam.sir
B i o   behavior Main
B i o   |------ JpegEncoder jpeg
B i s   |       |------ Dct dct
B i l   |       |       |------ Bound bound
B i l   |       |       |------ ChenDct chendct
B i l   |       |       \------ Preshift preshift
B i s   |       |------ Huff huff
B i l   |       |       |------ Huffencode huffencode
B i l   |       |       \------ Zigzag zigzag
B i l   |       |------ Quantize quantize
B i l   |       \------ ReadBlock readblock
B i l   |------ Monitor monitor
B i l   |------ Stimulus stimulus
C i l   |------ c_queue data
C i l   \------ c_handshake start
```


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 &`**
     - Please *do not print* the tutorial, it contains 250 pages!
   - Follow the SCE Tutorial instructions
     until the architecture model is successfully created (page 94)
     - **`sce &`**
   - Cleanup
     - When done (or to start over), clean up your demo directory
     - **`cd ..`**
     - **`rm -rf demo`**

Assignment 4

2. Simulate your digital camera model in SCE
   - Setup
     - Note that we will use the 2008 version of SCE for the JPEG Encoder:
     - **`source /opt/sce-20080601/bin/setup.csh`**
     - **`rm -rf ~/.sce`**
     - **`cd jpegencoder2`**
     - **`sce`**
   - Create a new project in SCE
     - **`Project->New`**
     - **`Project->Settings`**
       - Set verbosity and warning levels to 2
       - Adjust any other options the compiler may need to compile your model
     - **`Project->SaveAs "digicam.sce"`**
   - Load your design model into SCE
     - **`File->Import "digicam.sc"`**
     - **`Project->AddDesign`**
     - Right-click on **`digicam.sir`** in the project window, and **`Rename`** the model to **`DigicamSpec`**
   - Compile and simulate your model in SCE
     - **`Validation->Compile`**
     - **`Validation->Simulate`**

Assignment 4

3. Analyze your digital camera model in SCE
   - Setup
     - …continued from step 2 (previous page)
   - View the structural hierachy chart
     - Select the **`Main`** behavior in the behavior browser
     - Right-click **`->Chart`**
     - Double-click the chart to add further levels of hierarchy
     - Turn on connectivity **`View->Connectivity`**
     - **`Window->Print…`** to file **`"digicam.ps"`**
     - In your shell window, convert the PostScript file to PDF: **`ps2pdf digicam.ps`**
     - Check the PDF file: **`acroread digicam.pdf`**
- Deliverables
  - Hierarchy chart
    - **`"digicam.pdf"`**
- Due
  - by Friday, Nov 6, 2009, at noon
  - by email to **`doemer@uci.edu`** with subject "EECS222C HW4"