EECS 298: System-on-Chip Description and Modeling Lecture 3

Rainer Dömer

doemer@uci.edu

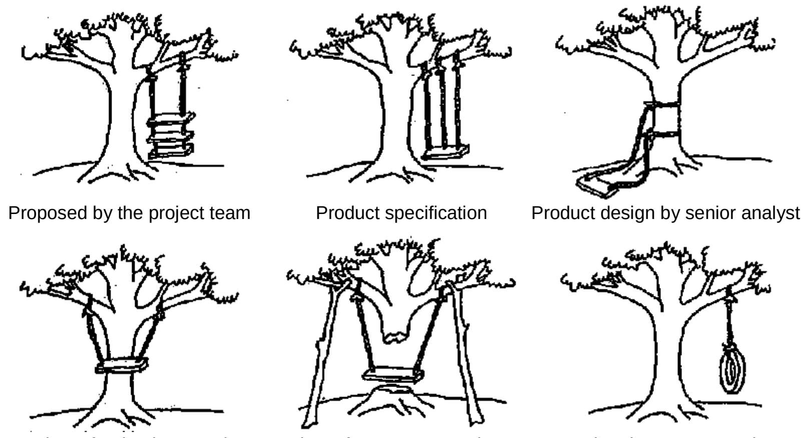
The Henry Samueli School of Engineering
Electrical Engineering and Computer Science
University of California, Irvine

Lecture 3: Overview

- 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 1
  - Administration
  - Tasks

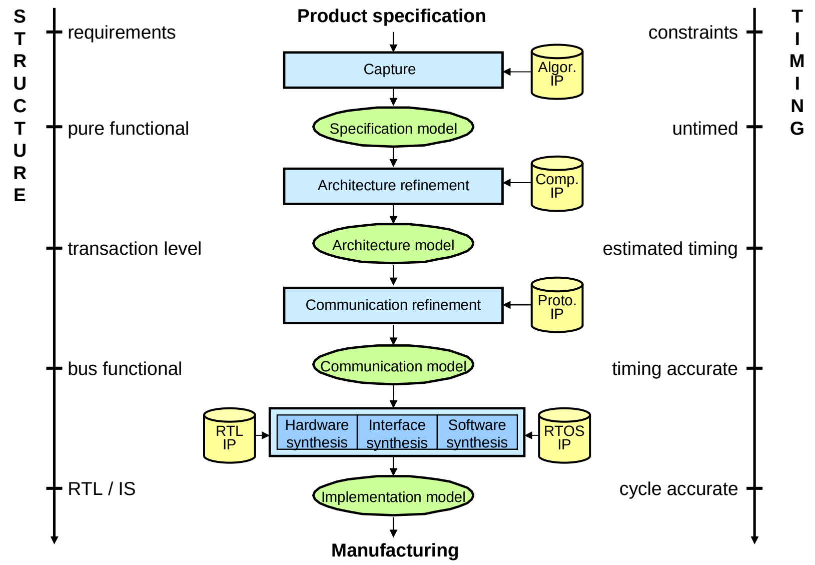
Essential Issues in Specification

- An Example ...

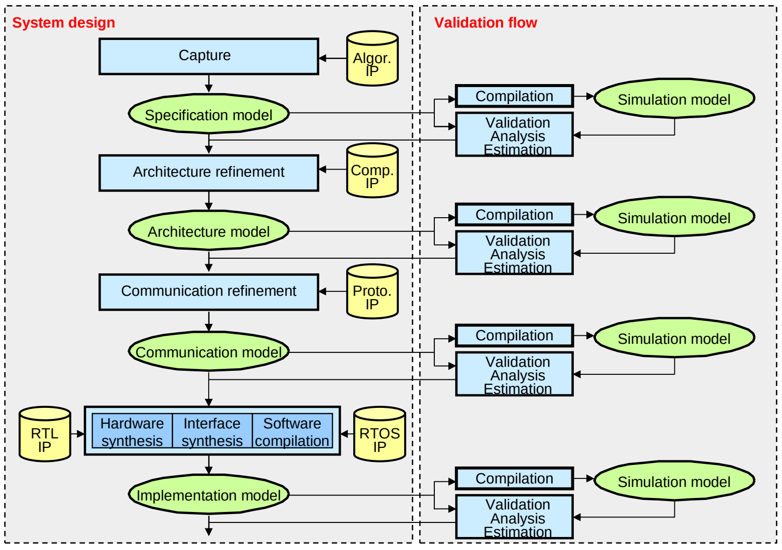

Proposed by the project team
Product specification
Product design by senior analyst
Product after implementation
Product after acceptance by user
What the user wanted

Source: unknown author

Top-Down SoC Design Flow

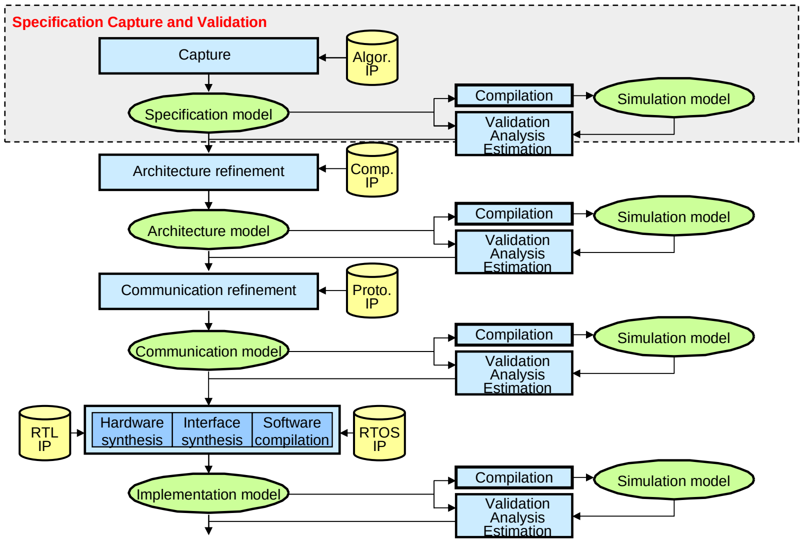


System-on-Chip Design Methodology



EECS298: SoC Description and Modeling, Lecture 3 (c) 2006 R. Doemer 5

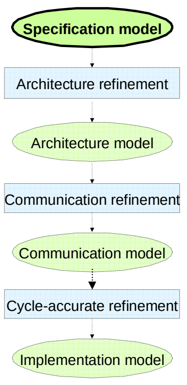
SoC Specification



EECS298: SoC Description and Modeling, Lecture 3 (c) 2006 R. Doemer 6

Specification Model

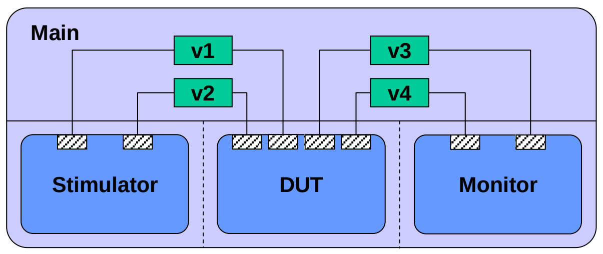
- High-level, abstract model
  - Pure system functionality
  - Algorithmic behavior
  - No implementation details
- No implicit structure / architecture
  - Pure behavioral hierarchy
- Untimed
  - Execution in zero (logical) time
  - Causal ordering
  - Synchronization

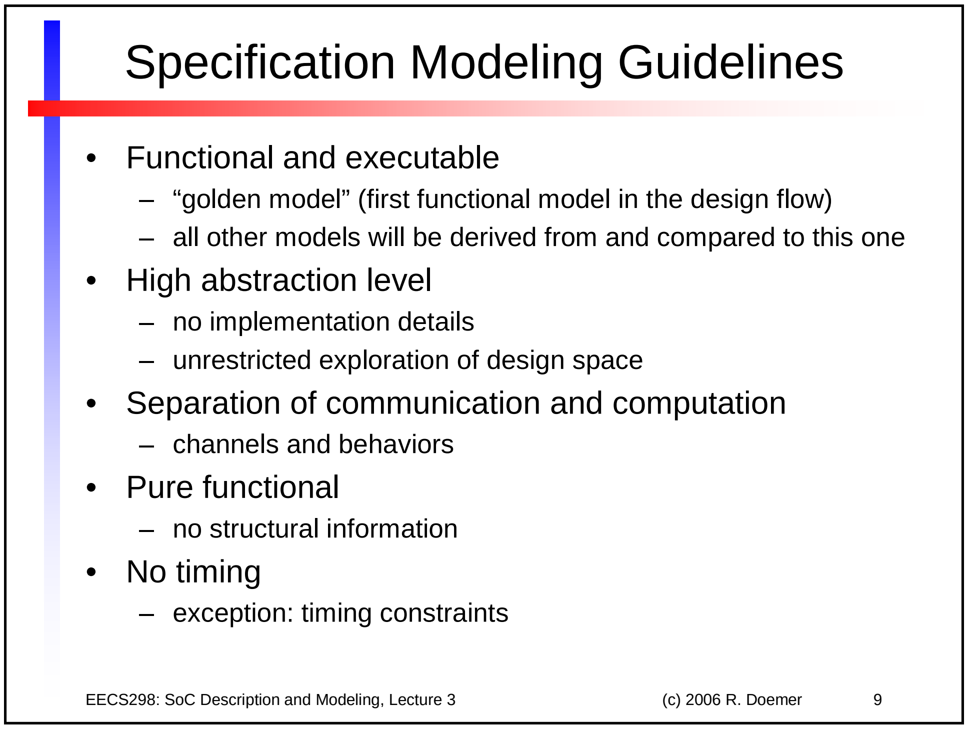


(Source: A. Gerstlauer)

Specification Model

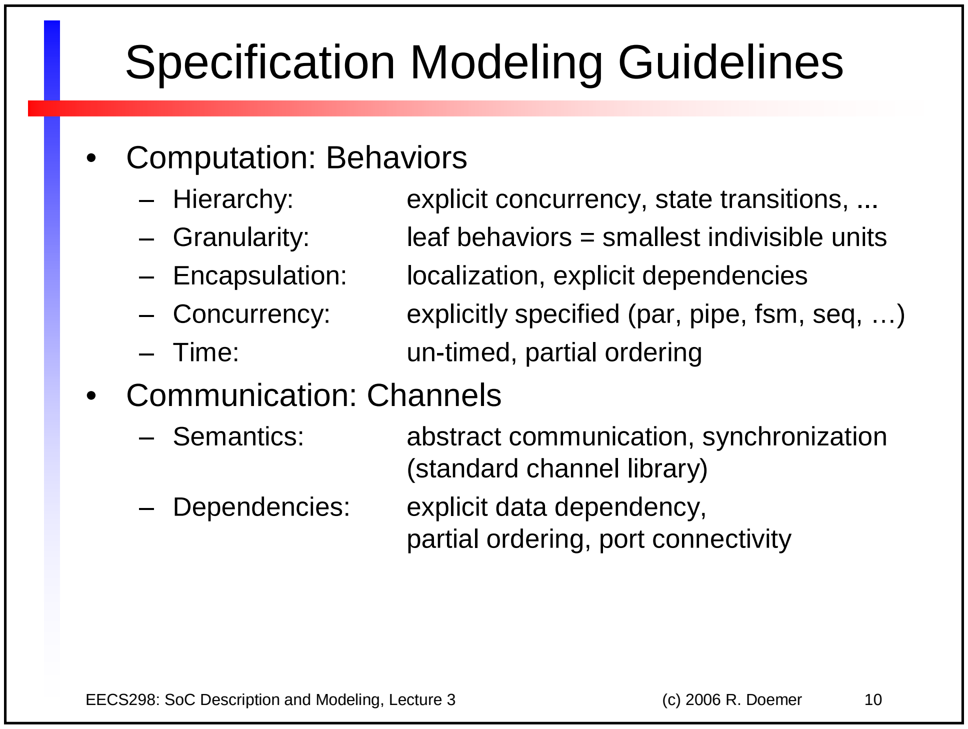
- Test bench
  - Main, Stimulator, Monitor
  - no restrictions in syntax and semantics (no synthesis)
- Design under test
  - DUT
  - restricted by syntax and semantic rules (synthesis!)





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



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

Specification Modeling Guidelines

- Example rules for SpecC 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 SpecC 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

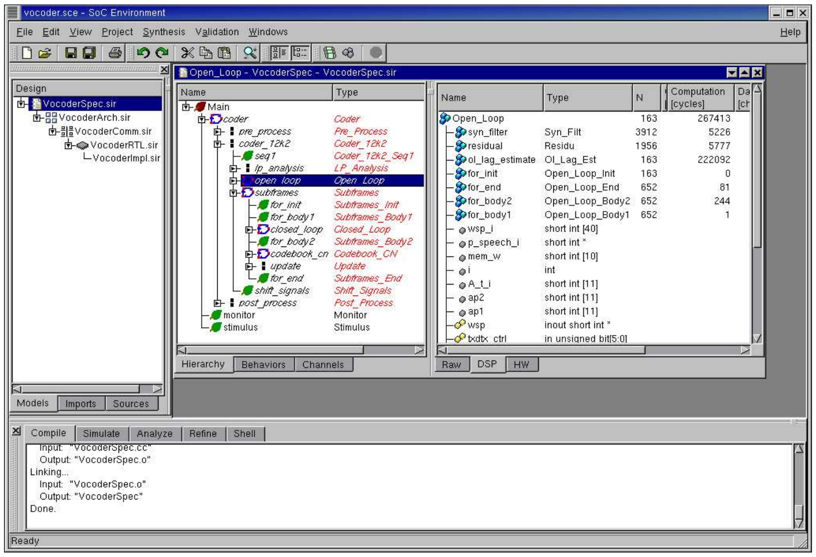
Specification Modeling Guidelines

- 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

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 ...

SCE Main Window


vocoder.sce - SoC Environment
File Edit View Project Synthesis Validation Windows Help
Open_Loop - VocoderSpec - VocoderSpec.sir
Models Imports Sources
Hierarchy Behaviors Channels
Raw DSP HW
Compile Simulate Analyze Refine Shell
Ready

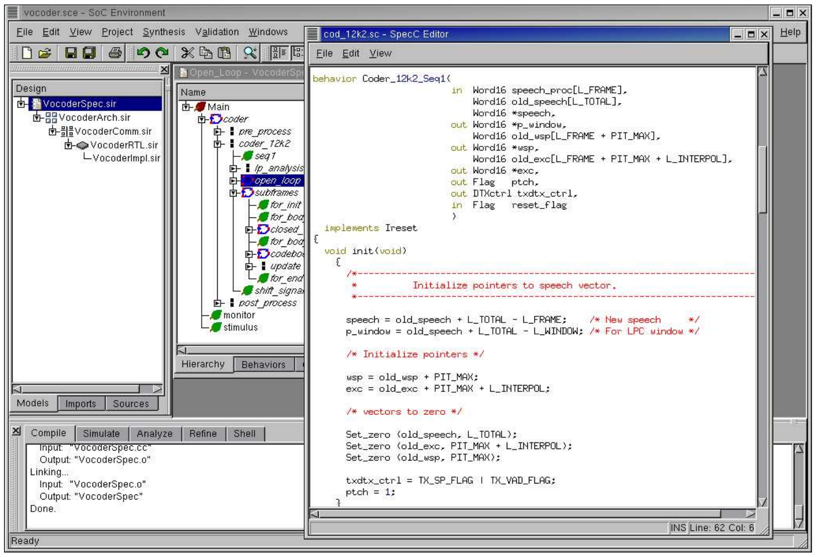
```
Design
VocoderSpec.sir
  VocoderArch.sir
    VocoderComm.sir
      VocoderRTL.sir
        VocoderImpl.sir
```

```
Name                    Type
Main
  coder                 Coder
    pre_process         Pre_Process
    coder_12k2          Coder_12k2
      seq1              Coder_12k2_Seq1
      lp_analysis       LP_Analysis
      open_loop         Open_Loop
      subframes         Subframes
        for_init        Subframes_Init
        for_body1       Subframes_Body1
        closed_loop     Closed_Loop
        for_body2       Subframes_Body2
        codebook_cn     Codebook_CN
        update          Update
        for_end         Subframes_End
      shift_signals     Shift_Signals
    post_process        Post_Process
  monitor               Monitor
  stimulus              Stimulus
```

Name	Type	N	Computation [cycles]
Open_Loop		163	267413
syn_filter	Syn_Filt	3912	5226
residual	Residu	1956	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
wsp_i	short int [40]		
p_speech_i	short int *		
mem_w	short int [10]		
i	int		
A_t_i	short int [11]		
ap2	short int [11]		
ap1	short int [11]		
wsp	inout short int *		
txdtx_ctrl	in unsigned bit[5:0]		

```
  Input:  "VocoderSpec.cc"
  Output: "VocoderSpec.o"
Linking...
  Input:  "VocoderSpec.o"
  Output: "VocoderSpec"
Done.
```

SCE Source Editor

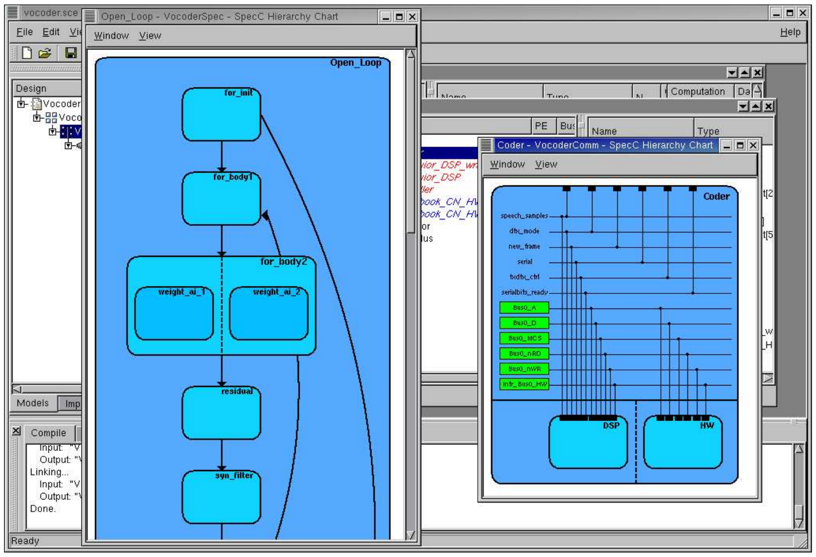


```
behavior Coder_12k2_Seq1(
                          in  Word16 speech_proc[L_FRAME],
                              Word16 old_speech[L_TOTAL],
                              Word16 *speech,
                          out Word16 *p_window,
                              Word16 old_wsp[L_FRAME + PIT_MAX],
                          out Word16 *wsp,
                              Word16 old_exc[L_FRAME + PIT_MAX + L_INTERPOL],
                          out Word16 *exc,
                          out Flag   ptch,
                          out DTXctrl 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_MAX;
      exc = old_exc + PIT_MAX + L_INTERPOL;

      /* vectors to zero */

      Set_zero (old_speech, L_TOTAL);
      Set_zero (old_exc, PIT_MAX + L_INTERPOL);
      Set_zero (old_wsp, PIT_MAX);

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

```
   Input:  "VocoderSpec.cc"
   Output: "VocoderSpec.o"
Linking...
   Input:  "VocoderSpec.o"
   Output: "VocoderSpec"
Done.
```

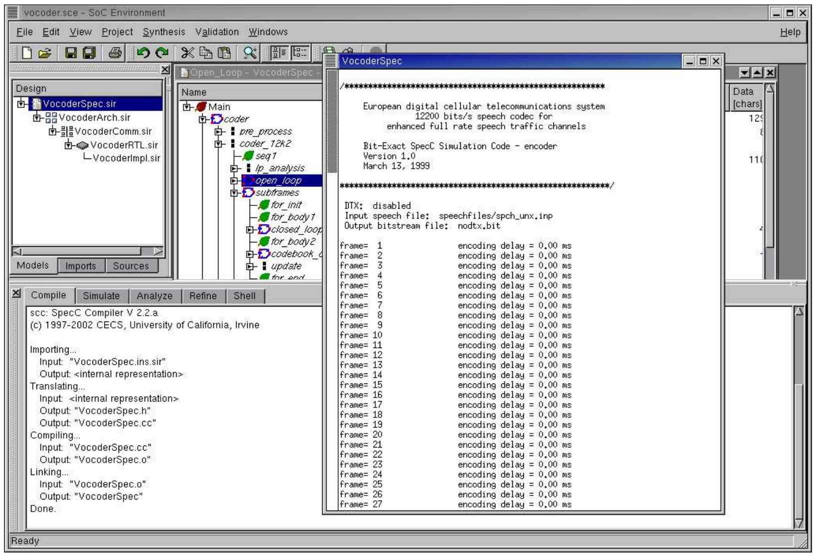
EECS298: SoC Description and Modeling, Lecture 3   Copyright © 2003 CECS   15

SCE Hierarchy Displays



EECS298: SoC Description and Modeling, Lecture 3   Copyright © 2003 CECS   16

SCE Compiler and Simulator

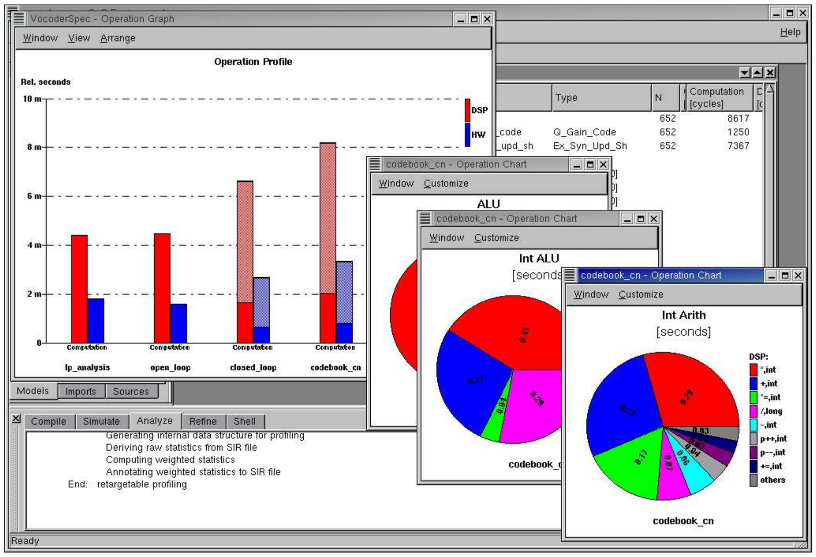


```
/*********************************************************

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

     Bit-Exact SpecC Simulation Code - encoder
     Version 1.0
     March 13, 1999

**********************************************************/

 DTX:  disabled
 Input speech file:  speechfiles/spch_unx.inp
 Output bitstream file:  nodtx.bit

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
```

```
scc: SpecC Compiler V 2.2.a
(c) 1997-2002 CECS, University of California, Irvine

Importing...
   Input:  "VocoderSpec.ins.sir"
   Output: <internal representation>
Translating...
   Input:  <internal representation>
   Output: "VocoderSpec.h"
   Output: "VocoderSpec.cc"
Compiling...
   Input:  "VocoderSpec.cc"
   Output: "VocoderSpec.o"
Linking...
   Input:  "VocoderSpec.o"
   Output: "VocoderSpec"
Done.
```

EECS298: SoC Description and Modeling, Lecture 3 Copyright © 2003 CECS 17

SCE Profiling and Analysis



```
            Generating internal data structure for profiling
            Deriving raw statistics from SIR file
            Computing weighted statistics
            Annotating weighted statistics to SIR file
      End:  retargetable profiling
```

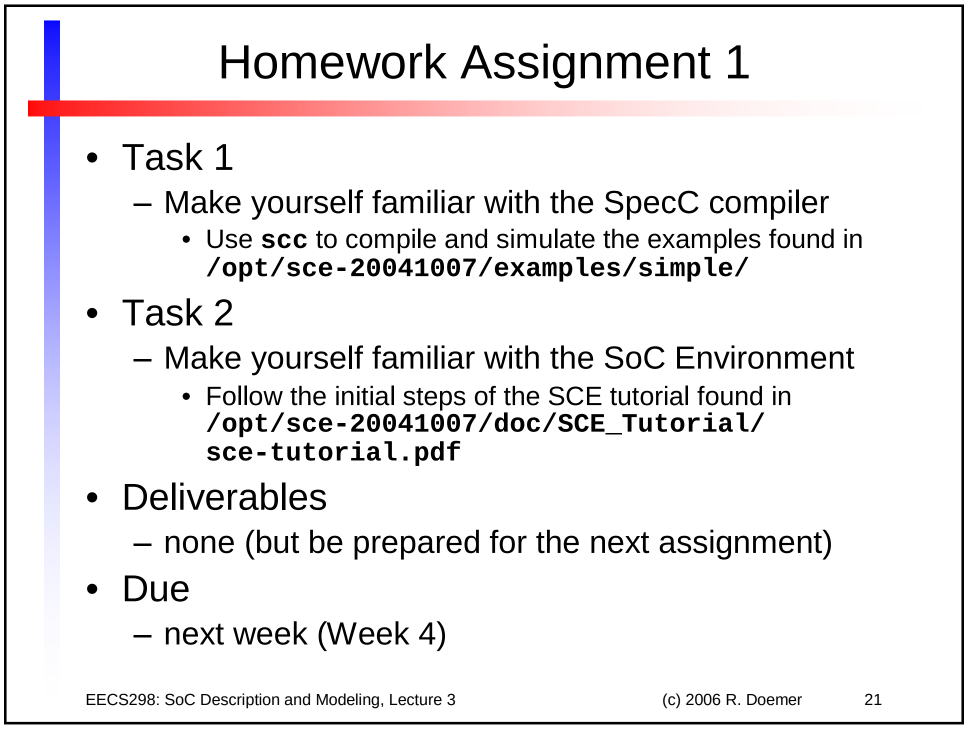
EECS298: SoC Description and Modeling, Lecture 3 Copyright © 2003 CECS 18

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 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)

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
  - Software (© by CECS, UCI)
    - SpecC Compiler and Simulator
      - **`/opt/sce/bin/setup.csh`**
    - System-on-Chip Environment
      - **`/opt/sce-20041007/bin/setup.csh`**



Homework Assignment 1

- Task 1
  - Make yourself familiar with the SpecC compiler
    - Use **`scc`** to compile and simulate the examples found in **`/opt/sce-20041007/examples/simple/`**
- Task 2
  - Make yourself familiar with the SoC Environment
    - Follow the initial steps of the SCE tutorial found in **`/opt/sce-20041007/doc/SCE_Tutorial/sce-tutorial.pdf`**
- Deliverables
  - none (but be prepared for the next assignment)
- Due
  - next week (Week 4)

EECS298: SoC Description and Modeling, Lecture 3 (c) 2006 R. Doemer 21