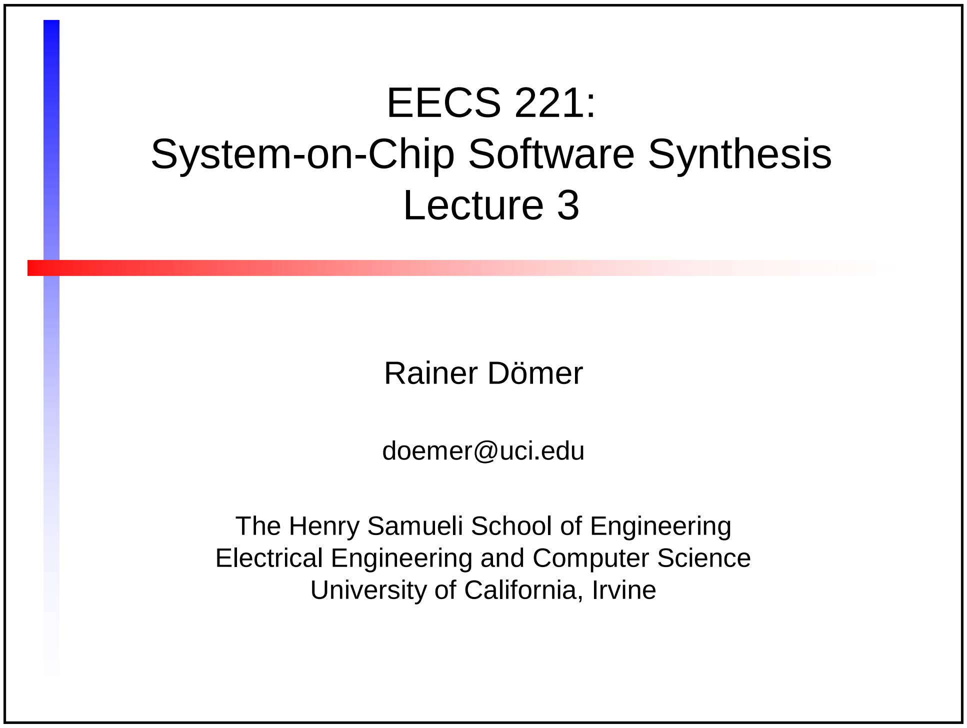
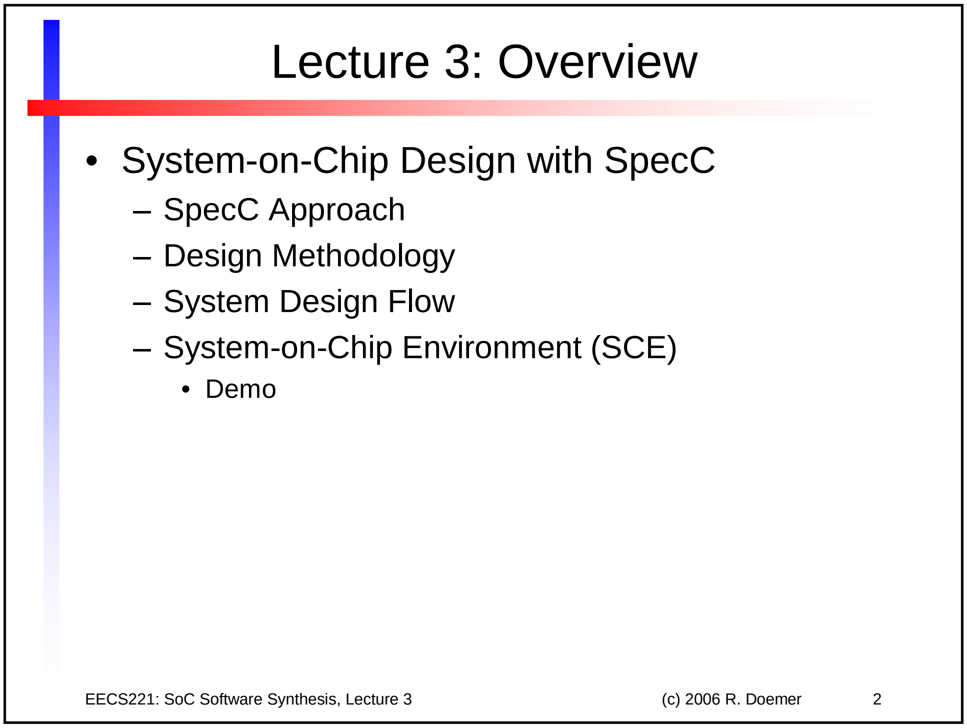


EECS 221:
System-on-Chip Software Synthesis
Lecture 3

Rainer Dömer

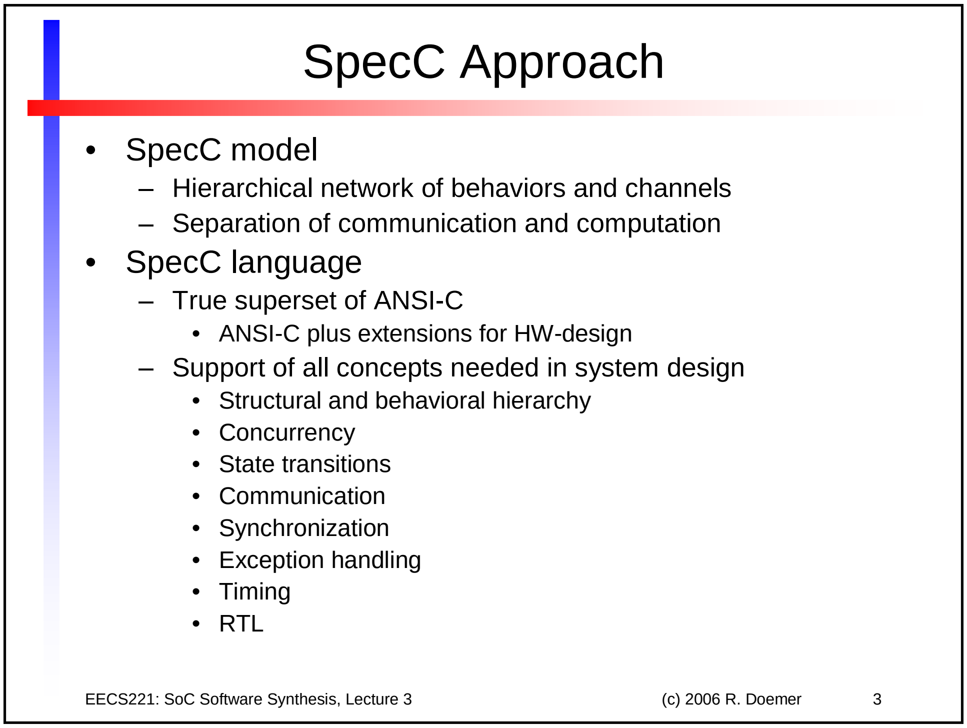
doemer@uci.edu

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



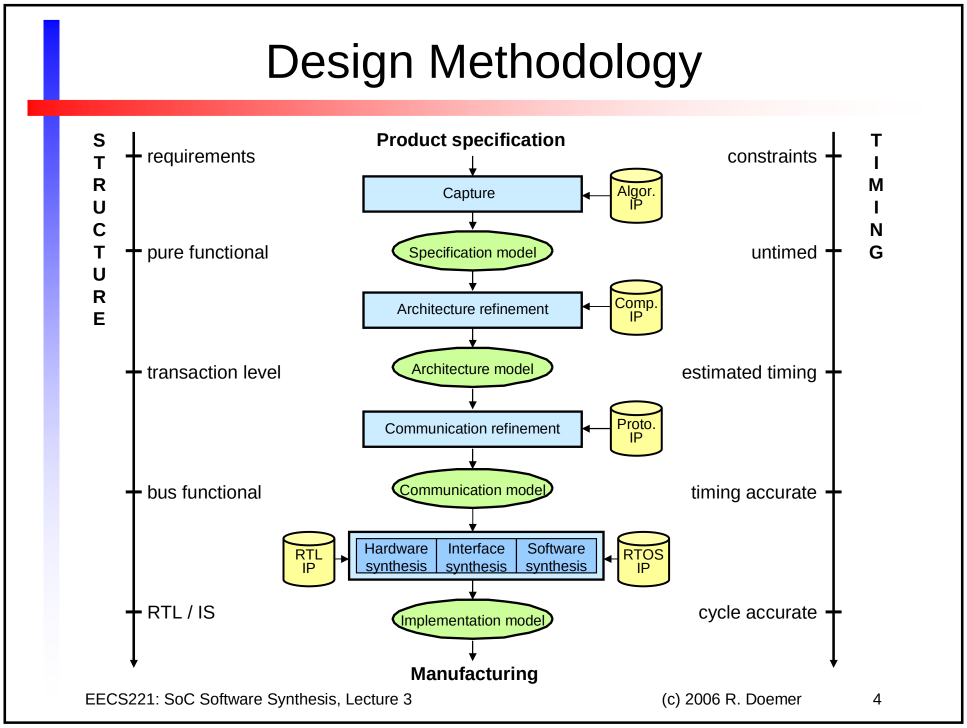
Lecture 3: Overview

- System-on-Chip Design with SpecC
  - SpecC Approach
  - Design Methodology
  - System Design Flow
  - System-on-Chip Environment (SCE)
    - Demo



SpecC Approach

- SpecC model
  - Hierarchical network of behaviors and channels
  - Separation of communication and computation
- SpecC language
  - True superset of ANSI-C
    - ANSI-C plus extensions for HW-design
  - Support of all concepts needed in system design
    - Structural and behavioral hierarchy
    - Concurrency
    - State transitions
    - Communication
    - Synchronization
    - Exception handling
    - Timing
    - RTL



Design Methodology

STRUCTURE

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

RTL IP

Hardware synthesis

Interface synthesis

Software synthesis

RTOS IP

Implementation model

Manufacturing

constraints

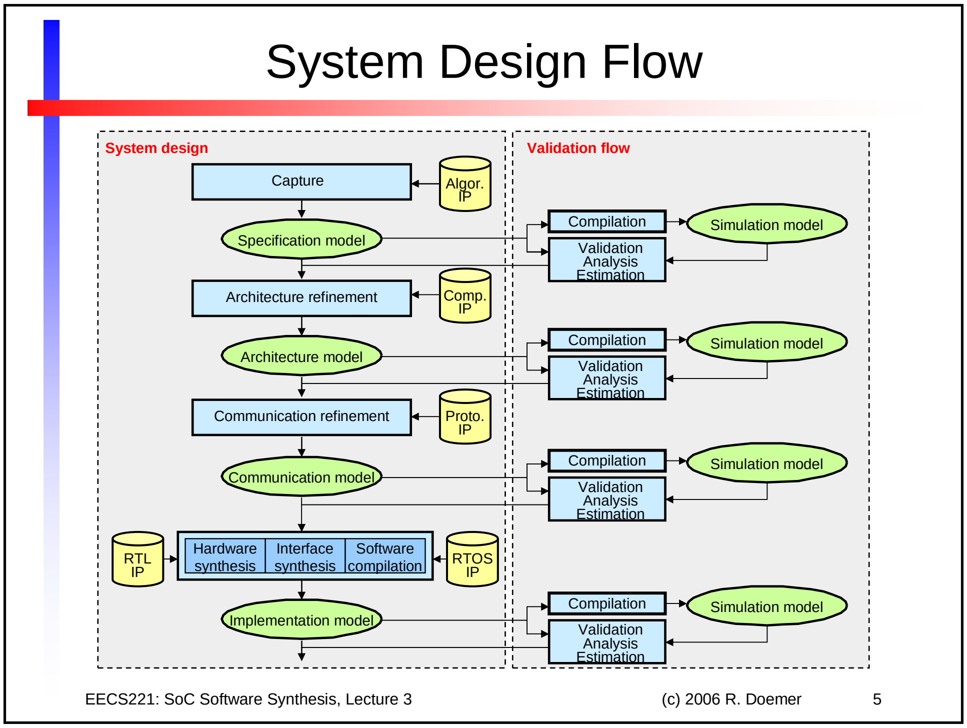
untimed

estimated timing

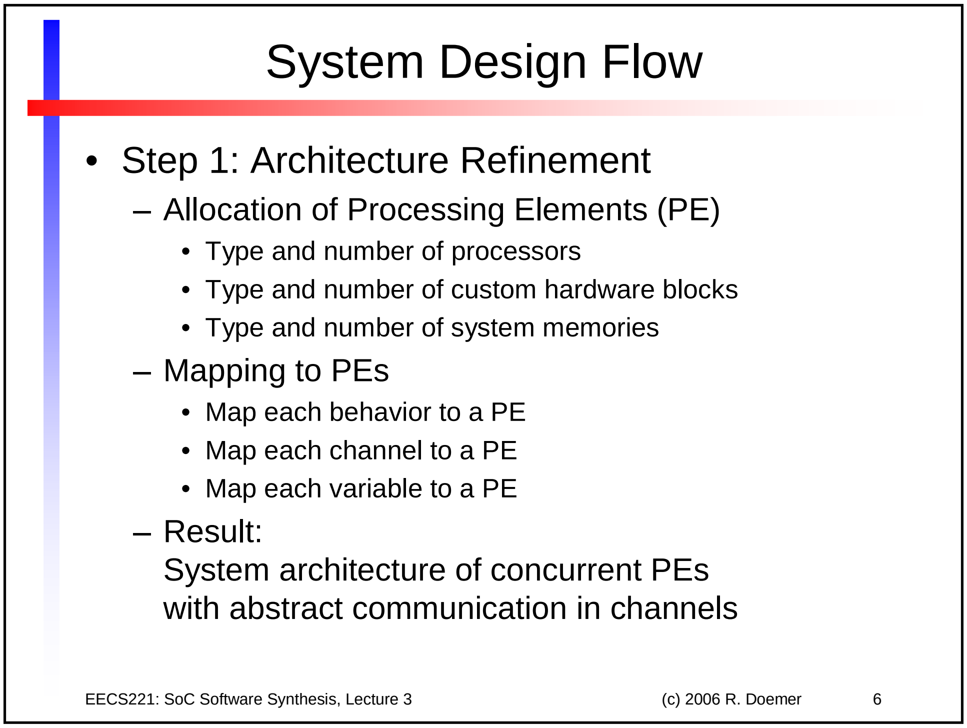
timing accurate

cycle accurate

TIMING

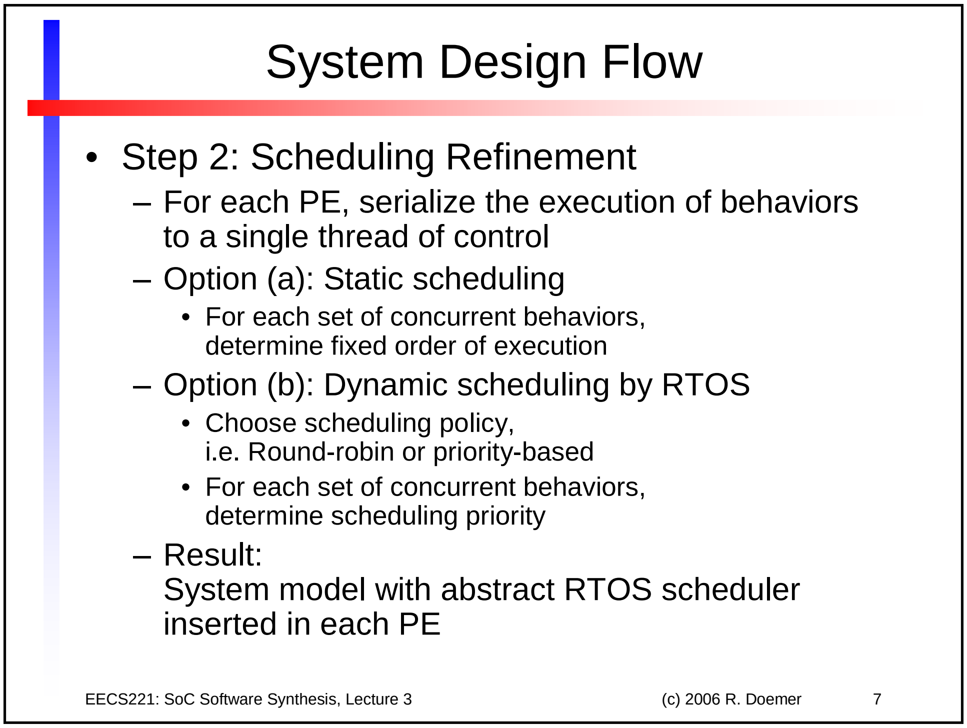


System Design Flow



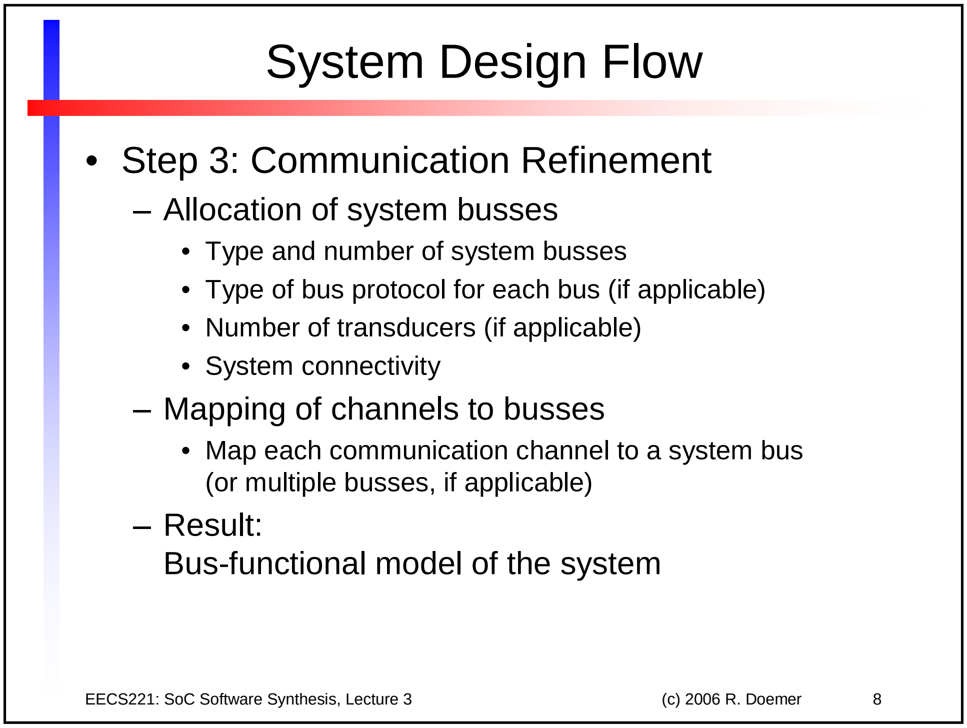
System Design Flow

- 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



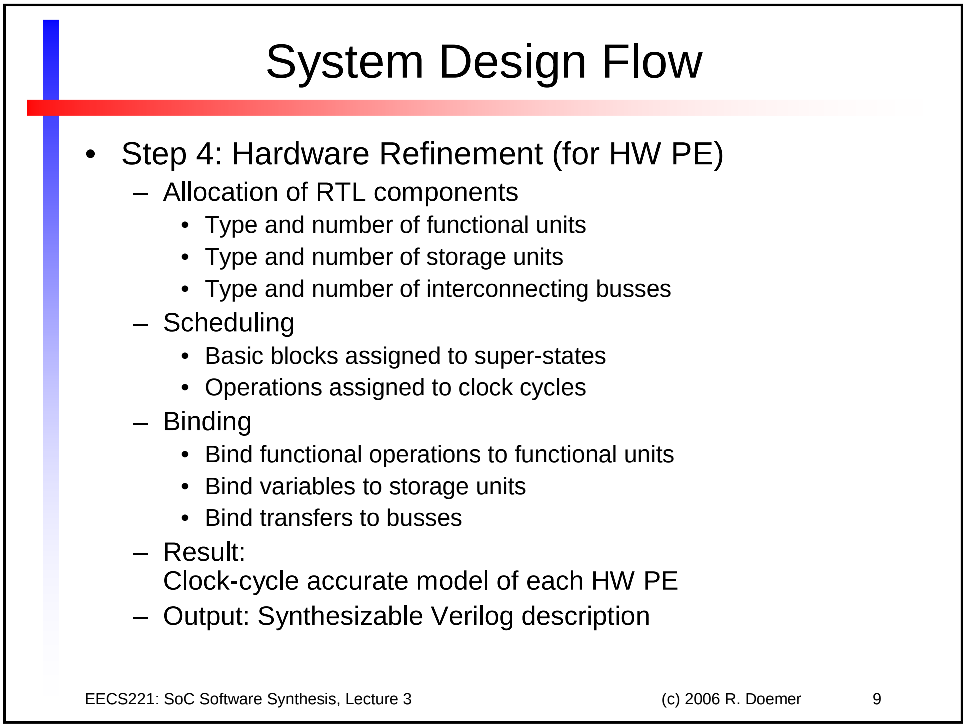
System Design Flow

- 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



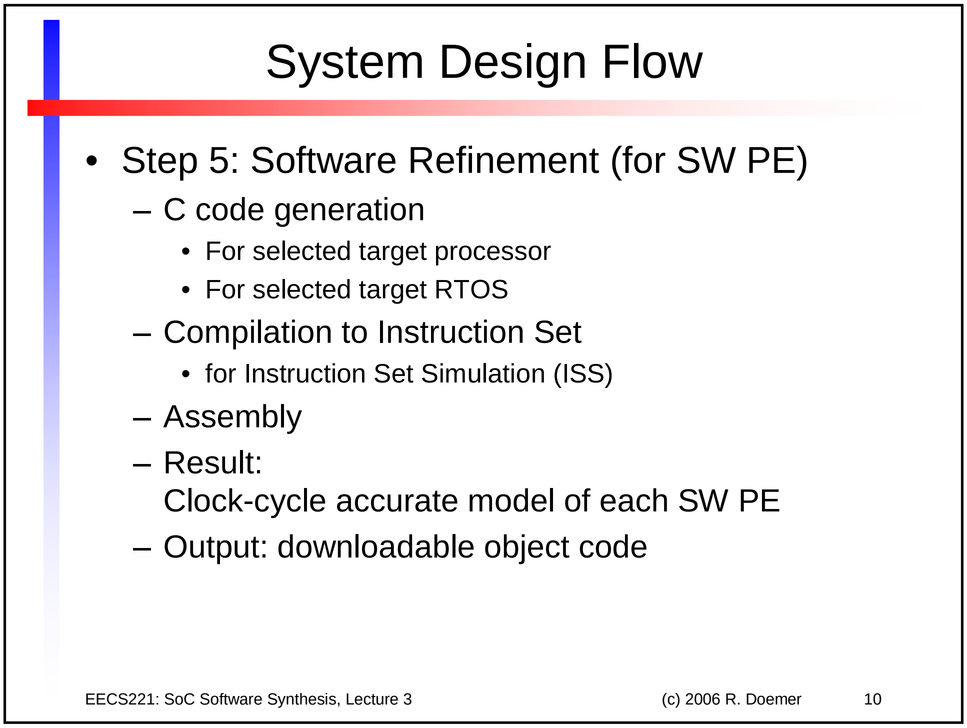
System Design Flow

- 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



System Design Flow

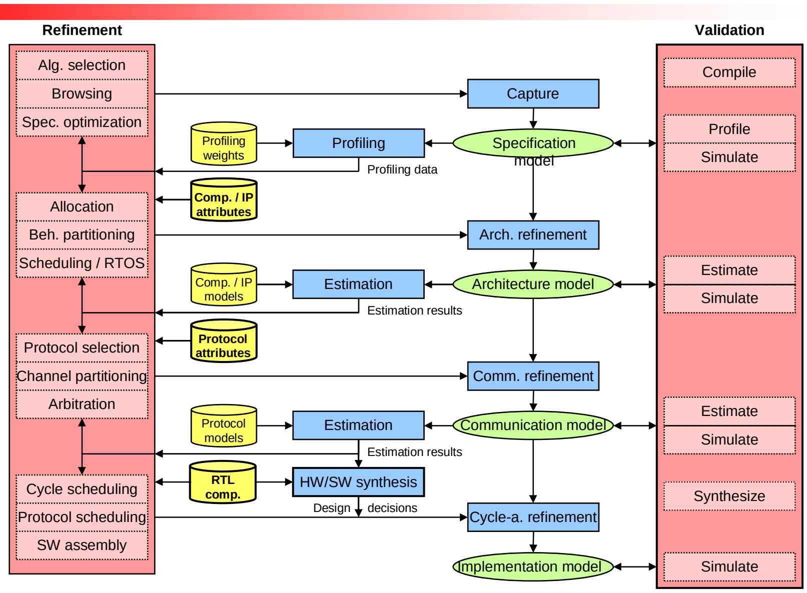
- Step 4: Hardware Refinement (for HW PE)
  - Allocation of RTL components
    - Type and number of functional units
    - Type and number of storage units
    - Type and number of interconnecting busses
  - Scheduling
    - Basic blocks assigned to super-states
    - Operations assigned to clock cycles
  - Binding
    - Bind functional operations to functional units
    - Bind variables to storage units
    - Bind transfers to busses
  - Result:
    Clock-cycle accurate model of each HW PE
  - Output: Synthesizable Verilog description



System Design Flow

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

System-on-Chip Environment (SCE)



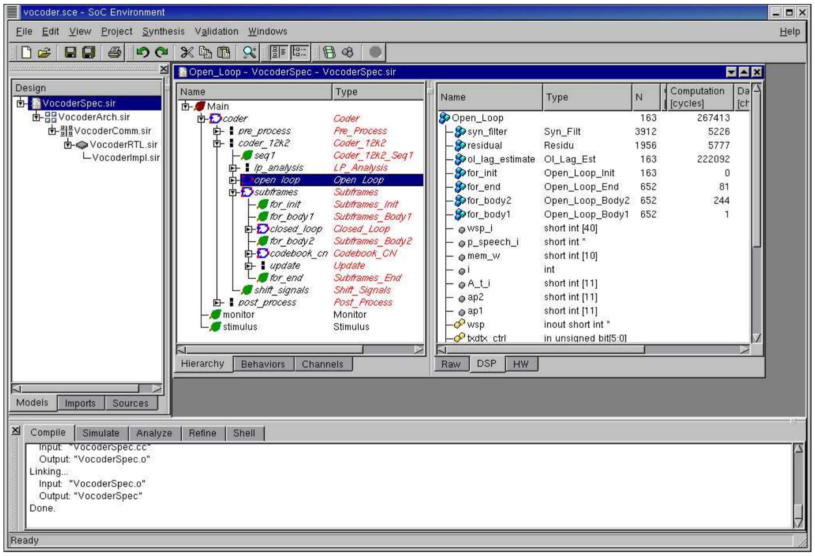
EECS221: SoC Software Synthesis, Lecture 3 Copyright © 2003 CECS 11

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 ...

EECS221: SoC Software Synthesis, Lecture 3 (c) 2006 R. Doemer 12

SCE Main Window


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

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

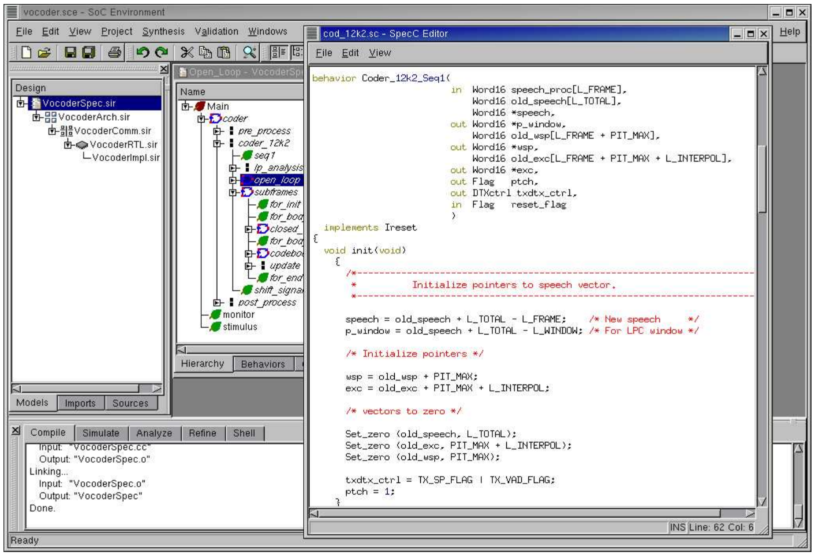
Open_Loop - VocoderSpec - VocoderSpec.sir

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]	Da [ch
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.
```

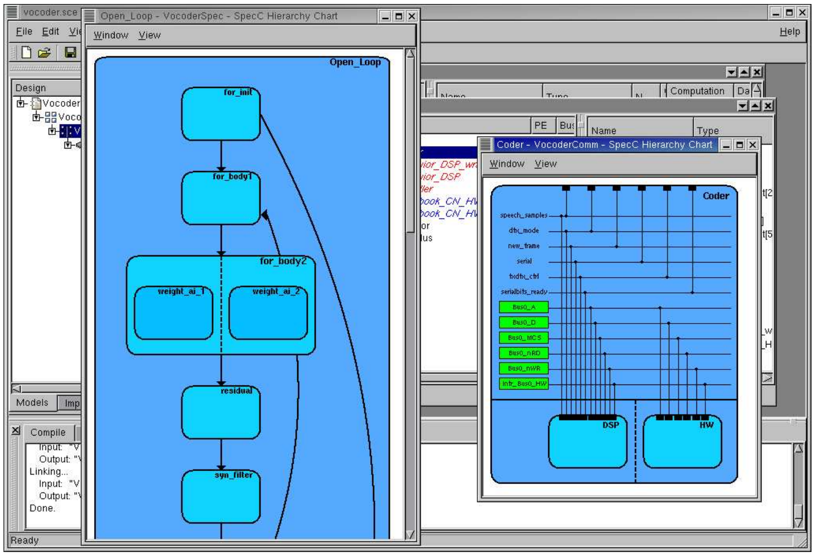
SCE Source Editor


vocoder.sce - SoC Environment
File Edit View Project Synthesis Validation Windows Help
Design
Models Imports Sources
Hierarchy Behaviors
Compile Simulate Analyze Refine Shell
Ready
cod_12k2.sc - SpecC Editor
File Edit View
INS Line: 62 Col: 6

```
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;
    }
```

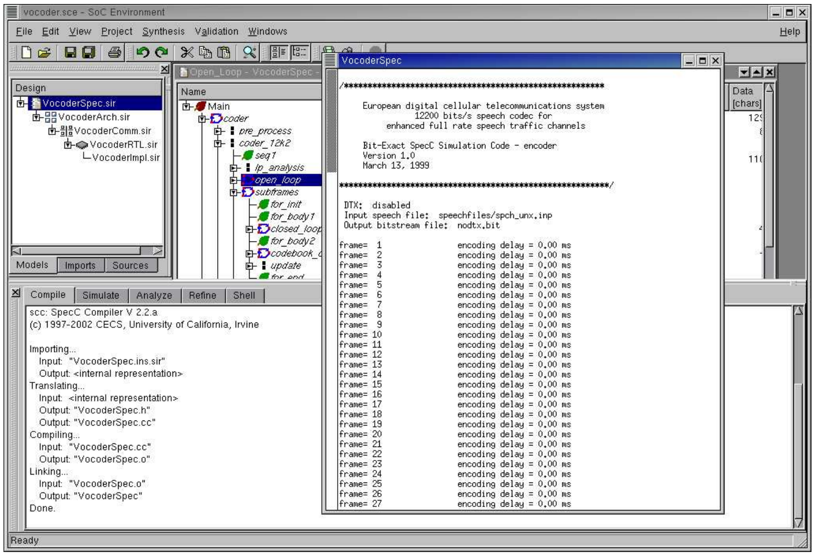
```
  Input:  "VocoderSpec.cc"
  Output: "VocoderSpec.o"
Linking...
  Input:  "VocoderSpec.o"
  Output: "VocoderSpec"
Done.
```

SCE Hierarchy Displays



```
  Input:  "V
  Output: "V
Linking...
  Input:  "V
  Output: "V
Done.
```

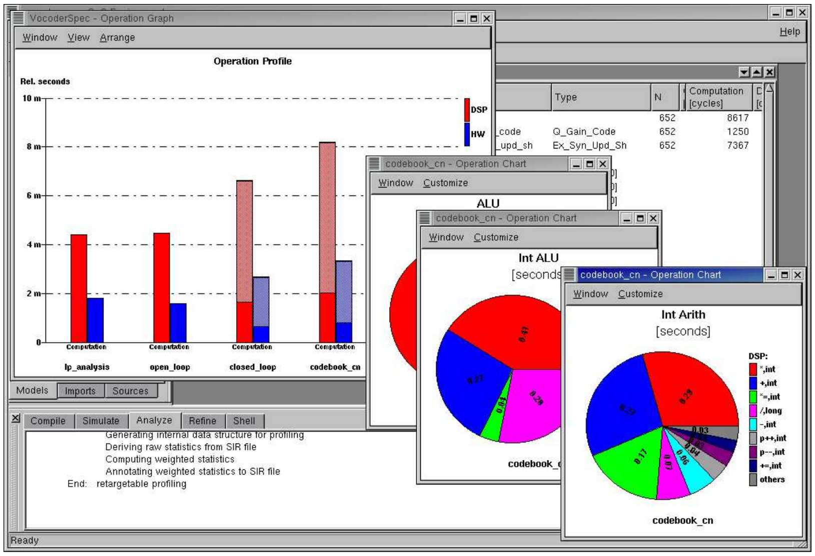
SCE Compiler and Simulator



```
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.
```

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

    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
```

SCE Profiling and Analysis



	Type	N	Computation [cycles]
		652	8617
_code	Q_Gain_Code	652	1250
_upd_sh	Ex_Syn_Upd_Sh	652	7367

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

EECS221: SoC Software Synthesis, Lecture 3 Copyright © 2003 CECS 17

Application Example

- 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)

EECS221: SoC Software Synthesis, Lecture 3 (c) 2006 R. Doemer 18