Lecture 5 – System-Level Design Tools

Andreas Gerstlauer
Electrical and Computer Engineering
University of Texas at Austin
gerstl@ece.utexas.edu
Lecture 5: Outline

• Electronic system-level (ESL) design automation
  • Modeling and simulation
  • Refinement
  • Exploration
  • Synthesis

• System-On-Chip Environment (SCE)
  • Computation exploration and refinement
  • Communication exploration and refinement

• Design example
  • GSM Vocoder
Specify-Explore-Refine Methodology

System design

Capture

Specification model

Computation refinement

Computation model

Communication refinement

Communication model

Hardware synthesis

Interface synthesis

Software synthesis

Implementation model

Validation flow

Compilation

Validation Analysis Estimation

Simulation model

Compilation

Validation Analysis Estimation

Simulation model

Compilation

Validation Analysis Estimation

Simulation model

Compilation

Validation Analysis Estimation

Simulation model
Electronic System-Level (ESL) Design Tools

- **Simulation-centric system modeling**
  - Virtual system prototyping [CoWare, Vast]
    - C-based, abstracted computation/implementation models [TLM]
    - System-level design languages (SLDLs) [SystemC]
    - Processor instruction-set simulation (ISS) models [Tensilica, Lisatek]
  - Algorithmic specification [SPW, MATLAB, COSSAP]
    - Varying models of computation (MoC) [Ptolemy]
    - Model-based design [UML, MATLAB/Simulink]

  ➢ *Horizontal integration of different models / components*
  ➢ *Lack vertical integration for synthesis and verification*

- **High-level synthesis**
  - C-to-RTL [Forte]
    - Interface specification [SystemVerilog]

  ➢ Single hardware unit only
Automatic Model Refinement

- **Problem:** Writing of system models is
  - Time consuming, error-prone, tedious
- **Solution:**
  - Automatic model generation
- **Refinement-based approach**
  - System designer: makes design decisions
  - Refinement tool: automatically generates the model
- **Benefits**
  - No manual model writing, focus on design decisions
  - Low error rate by automating error-prone tasks
  - Easy change/upgrade for incremental/derivative design
  - No change in basic design methodology
  - Enables fast, extensive design space exploration
  - Productivity gains
Design Automation

- Synthesis = Decision making + model refinement

- Successive model refinement
- Layers of implementation detail
Design Environment (1): Modeling

- Specification model
- Computation model
- Communication model
- Implementation model

Validation GUI:
- Compile
- Simulate
- Test
- Simulate
- Test
- Simulate
- Test
- Simulate
- Test
Design Environment (2): Refinement

- Refinement GUI:
  - Alg. selection
  - Browsing
  - Spec. optimization
  - PE Allocation
  - Beh/Var Partitioning
  - Scheduling
  - Net Allocation
  - Channel partitioning
  - Prot. insertion
  - HW alloc/sched/bind
  - SW Compilation
  - IF generation

- Validation GUI:
  - Compile
  - Simulate
  - Test
  - Simulate
  - Verify
  - Test
  - Simulate
  - Verify
  - Test
  - Simulate
  - Verify
  - Test

- Diagram:
  - Capture
  - Specification model
  - Comp. refinement
  - Computation model
  - Comm. refinement
  - Communication model
  - Proc. refinement
  - Implementation model
Design Environment (3): Exploration

Refinement GUI
- Alg. selection
- Browsing
- Spec. optimization
- PE Allocation
- Beh/Var Partitioning
- Scheduling
- Net Allocation
- Channel partitioning
- Prot. insertion
- HW alloc/sched/bind
- SW Compilation
- IF generation

Capture
- Profiling
  - Profiling data
  - Design decisions
- Specification model
- Comp. refinement
  - Estimation results
- Computation model
- Proc. refinement
  - Estimation results
- Communication model
- Implementation model

Validation GUI
- Compile
  - Profile
- Simulate
- Test
- Estimate
- Simulate
- Verify
- Test
- Estimate
- Simulate
- Verify
- Test
- Simulate
- Test
- Design Environment (3): Exploration
  - Estimate
  - Estimate
  - Estimate
- Browsing
  - Alg. selection
- Refinement GUI
Design Environment (4): Synthesis

Refinement GUI

- Alg. selection
- Browsing
- Spec. optimization
- PE Allocation
- Beh/Var Partitioning
- Scheduling
- Net Allocation
- Channel partitioning
- Prot. insertion
- HW alloc/sched/bind
- SW Compilation
- IF generation

Validation GUI

- Compile
- Profile
- Simulate
- Test
- Estimate
- Simulate
- Verify
- Test
- Estimate
- Simulate
- Verify
- Test
Design Methodology

Computation Design

Capture → Specification model → Computation refinement → Computation model → Communication refinement → Communication model → Implementation model → Simulation model

Estimation → Validation → Analysis → Compilation

Algor. IP → Comp. IP → Proto. IP → RTL IP → RTOS IP → Hardware synthesis, Interface synthesis, Software compilation

Compilation → Simulation model
Computation Design (1)

- **Architecture exploration**
  - 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 complex channel to a PE
    - Map each variable to a PE

- **Architecture model**
  - Concurrent PEs
  - Abstract channels and memory interfaces
Computation Design (2)

- **Scheduling exploration**
  - Static scheduling of behaviors into sequential tasks
    - Group (flatten) behaviors into tasks
    - Determine fixed execution order of behaviors in each task
  - Dynamic scheduling of concurrent tasks by RTOS
    - Choose scheduling policy, i.e. round-robin or priority-based
    - For each set of tasks, determine task priorities

- **Scheduled model**
  - Abstract OS model in software PEs
Design Methodology
Communication Design (1)

• **Network exploration**
  • Allocation of system network
    – Type (protocols) and number of system busses
    – Type and number of CEs (bridges and transducers, if applicable)
    – System connectivity
  • Routing of channels over busses
    – Map each communication channel to a system bus
      (or an ordered list of busses, if applicable)

➢ **Network model**
  ➢ PEs + CEs
    ➢ Middleware stacks
  ➢ Point-to-point links
    ➢ Untyped packet transfers
    ➢ Untyped memory interfaces
Communication Design (2)

- Communication synthesis
  - Assignment of bus parameters
    - Address mapping for each channel and memory interface
    - Synchronization mechanism for each channel (dedicated or shared interrupts, polling)
    - Transfer mode for each channel/interface (regular, burst, DMA)

- Transaction-level model (TLM)
  - PEs + CEs + Busses
    - Protocol stacks
  - Abstract bus channels
    - Bus transactions + interrupts
Communication Design (2)

- **Communication synthesis**
  - Assignment of bus parameters
    - Address mapping for each channel and memory interface
    - Synchronization mechanism for each channel (dedicated or shared interrupts, polling)
    - Transfer mode for each channel/interface (regular, burst, DMA)

- **Transaction-level model (TLM)**
  - PEs + CEs + Busses
    - Protocol stacks
  - Abstract bus channels
    - Bus transactions + interrupts

- **Pin-accurate model (PAM)**
  - Physical bus structure
    - Bit-accurate pins and wires
**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*)
  - Network refinement (*scnr*)
  - Communication refinement (*sccr*)
  - RTL refinement (*scrtl*)
  - Software refinement (*sc2c*)
  - Scripting interface (*scsh*)
  - Tools and utilities ...

GUI / Scripting

- **Architecture model**
  - Specification model
  - Architecture Exploration
    - PE Allocation
    - Beh/Var/Ch Partitioning
  - Scheduled model
  - Scheduling Exploration
    - Static Scheduling
    - OS Task Scheduling
  - Network Exploration
    - Bus Network Allocation
    - Channel Mapping
  - Communication Synth.
    - Bus Addressing
    - Bus Synchronization

- **Network model**

- **Communication model**
  - Communication Synth.
  - Bus Addressing
  - Bus Synchronization

- **RTL Synthesis**
  - Datapath Allocation
  - Scheduling/Binding

- **SW Synthesis**
  - Code Generation
  - Compile and Link

- **RTL Synthesis**

- **Implementation model**

- **GUI / Scripting**

- **TLM**

- **RTL Synthesis**
  - Datapath Allocation
  - Scheduling/Binding

- **SW Synthesis**
  - Code Generation
  - Compile and Link

- **RTL DB**

- **SW DB**

- **Binary**
SCE Main Window
SCE Source Editor

```c
behavior Coder_13x2_Seq1{
  in  Word16 speech_proc[L_FRAME],
      Word16 old_speech[L_TOTAL],
      Word16 *speech,
  out Word16 *p_window,
      Word16 old_usp[L_FRAME + PIT_MAX],
      Word16 *usp,
      Word16 old_exc[L_FRAME + PIT_MAX + L_INTERPOL],
  out  Word16 *exc,
      Flag  *pitch,
      ITXctrl txtx_ctrl,
  in  Flag  reset_flag
}

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 */
  ...
  wsp = old_wsp + PIT_MAX;
  exc = old_exc + PIT_MAX + L_INTERPOL;
  ...
  Set_zero (old_speech, L_TOTAL);
  Set_zero (old_exc, PIT_MAX + L_INTERPOL);
  Set_zero (old_usp, PIT_MAX);
  txtx_ctrl = TX_SP_FLAG | TX_VAD_FLAG;
  pchm = 1;
}
SCE Hierarchy Displays
SCE Compiler and Simulator
SCE Profiling and Analysis
SCE Model Refinement

[Image of a software interface with details on hardware components and their specifications.]
SCE Tool Set

- **Server and accounts**
  - ECE LRC Linux servers
    - Labs (ENS 507?) or remote access (ssh)
  - SpecC software (© by CECS, UCI)
    - /home/project/courses/fall_08/ee382v-17295/sce-20080601
    - source /home/.../sce-20080601/bin/setup.{c}sh

- **System-On-Chip Environment**
  - GUI
    - sce
  - Scripting
    - sce_allocate / sce_map / sce_schedule / ...
  - Documentation ($SPECC/doc)
    - SCE Manual (online via Help->Manual)
    - SCE Tutorial (PDF or html)
    - SCE Specification Reference Manual (SpecRM.pdf)
Lecture 5: Outline

✓ Electronic system-level (ESL) design automation
  ✓ Modeling and simulation
  ✓ Refinement
  ✓ Exploration
  ✓ Synthesis

✓ System-On-Chip Environment (SCE)
  ✓ Computation exploration and refinement
  ✓ Communication exploration and refinement

• Design example
  • GSM Vocoder
GSM Vocoder

- **Enhanced Full Rate (EFR) standard (GSM 06.10, ETSI)**
  - Lossy voice encoding/decoding for mobile telephony
    - Speech synthesis model
      - Input: speech samples @ 104 kbit/s
      - Frames of 4 x 40 = 160 samples (4 x 5ms = 20ms of speech)
      - Output: encoded bit stream @ 12.2 kbit/s (244 bits / frame)
  - Timing constraint
    - 20ms per frame (total of 3.26s for sample speech file)

- **SpecC specification model**
  - 73 leaf, 29 hierarchical behaviors (9 par, 10 seq, 10 fsm)
  - 9139 lines of SpecC code (~13000 lines of original C)
Vocoder Specification Model

![Diagram of Vocoder Specification Model]
Vocoder Computation Model

- **Motorola DSP56600**
- **Custom hardware**
Vocoder Communication Model
Vocoder Implementation Model

Cycle-accurate co-simulation

- DSP instruction-set simulator (ISS)
  - 70,500 lines of assembly code (running @ 60MHz)
- RTL SpecC for hardware
  - 45,000 lines of VHDL RTL code (running @ 100MHz)
Vocoder Results

Simulation Speed

<table>
<thead>
<tr>
<th>Spec</th>
<th>Arch</th>
<th>Comm</th>
<th>Impl</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>100</td>
<td>1000</td>
</tr>
</tbody>
</table>

Refinement Effort

<table>
<thead>
<tr>
<th></th>
<th>Modified lines</th>
<th>Manual</th>
<th>Automated User / Refine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spec → Arch</td>
<td>3,275</td>
<td>3~4 month</td>
<td>15 min / &lt; 1 min</td>
</tr>
<tr>
<td>Arch → Comm</td>
<td>914</td>
<td>1~2 month</td>
<td>5 min / &lt; 0.5 min</td>
</tr>
<tr>
<td>Comm → Impl</td>
<td>6,146</td>
<td>5~6 month</td>
<td>30 min / &lt; 2 min</td>
</tr>
<tr>
<td>Total</td>
<td>10,355</td>
<td>9~12 month</td>
<td>50 min / &lt; 4 min</td>
</tr>
</tbody>
</table>

• Productivity gain: 12 months vs. 1 hour = 2000x
Speed and Accuracy Trade-Offs

Source: G. Schirner, A. Gerstlauer, R. Doemer
Lecture 5: Summary

• **System-level design automation**
  - Modeling and simulation
  - Exploration and refinement
  - Synthesis and verification

• **System-On-Chip Environment (SCE)**
  - Automatic model generation
  - Interactive exploration and decision entry