EE382V: Embedded System Design and Modeling

Lecture 8 – System Evaluation and Refinement

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

Lecture 8: Outline

• Evaluation and estimation
  • Static analysis
  • Profiling and simulation

• Refinement and implementation
  • Refinement flow
  • Automatic model generation

• System-on-Chip Environment (SCE)
  • Tools and flow
  • Vocoder tutorial
Evaluation and Estimation Methods

- **Measurement**
  - Fast (real time), exhaustive?
  - Requires physical implementation

- **Analysis**
  - Worst-case/best-case assumptions
  - Tightness of upper/lower bounds? Dynamic effects?

- **Simulation**
  - Speed vs. accuracy tradeoffs
  - Quality of testbench, corner cases?

### Analysis Methods

- **Static analysis**
  - Symbolic, mathematical models for best/worst case
    - Worst-case execution time analysis (WCET)
    - Real-time scheduling

- **Probabilistic analysis**
  - Statistical models, distributions for “average” case
    - Queuing theory

- **Deterministic dynamic analysis**
  - Min-plus/max-plus algebra, upper/lower bounds over time
    - Network calculus
    - Real-time calculus
      - Modular Performance Analysis (MPA)
**Static Analysis**

- **Worst-case execution time (WCET)**
  - Micro-architecture analysis
    - Compute bounds for each basic execution block
    - Symbolically simulate statements on processor model (pipeline)
    - Conservative assumptions for dynamic effects (caches, predictors)
  - Path analysis
    - Enumerate possible paths and take maximum of block sequence
    - Possible paths often highly dynamic (loop bounds, false paths)
    - Basis for back-annotation or static system analysis
    - Combine static code analysis with dynamic system simulation
    - Static or dynamic model of inter-process cross-dependencies

**Control/Data Flow Graph (CDFG)**

**Analytical System Evaluation**

- **Modular Performance Analysis (MPA)**
  - Network calculus, real-time calculus (RTC)
Simulation Methods

- **Static timing back-annotation**
  - Source level
  - Instructions, basic blocks or functions
  - Estimation of basic metrics

- **Dynamic system simulation**
  - System description language
  - Simulation host
  - Functionality & timing
  - Generate trace

- **Timing analysis**
  - Latency, throughput, response time, etc.
Trace-Driven Simulation

- **Drive simulation via pre-existing, static traces**
  - Traces for system block behavior
  - Traces obtained from fast functional-only simulation

- **Examples**
  - Trace-driven simulation
  - Arrival curve extraction from traces
  - Trace generation from arrival curves

---

Design Space Exploration

- **Explore and trim**
- Gradually prune design space
System Simulation

- **Runtime vs. accuracy**
  - Fast design space exploration
  - Fidelity: relative accuracy (vs. absolute accuracy)

- **Capabilities**
  - Various levels of abstraction: components, system
  - Wide range of metrics: power, timing, area, reliability
  - Wide variety of target implementations

---

Exploration Flow

1. **Spec model** → **Instrumentation** → **Simulation** → **Validation**
2. **Profiling**:
   - **Refinement** → **Back annotation** → **Component Estimates**
3. **Estimation**:
   - **Refinement** → **Retargeting** → **Design decision**
4. **Evaluation**:
   - **Refined model** → **Simulation/analysis** → **System Quality Metrics**
   - Feedback loop

---

**Exploration Flow Diagram**

Diagram showing the flow of exploration from spec model to system quality metrics, including profiling, estimation, and evaluation steps.
Profiling

- **Input specification MoC**
  - Hierarchy
  - Computation & communication

- **Multi-dimensional analysis**
  - Multi-entities
    - Behavior, channel, port, variable
  - Multi-metrics
    - Operation, traffic, storage
    - Static, dynamic
  - Multi-levels
    - Application, transaction, bus-functional

---

**Profiling**

- **Instrumentation-based profiling**
  - $B_b$: The execution counts of basic block $b$
    - Enumerate execution paths
  - $C_{b,i,d}$: No. of computed characteristics for item type $i$ and data type $d$ in the block $b$
  - Data type $i$: float, int, ..
  - Item type $d$: metric-dependent

**Specification metrics**

- $R_{i,d} = \Sigma_b C_{b,i,d} B_b$
- $R = \Sigma_i R_{i,d}$

Retargeting

- Target machine model
  - \( W_{i,d} \) : weights of components which the entity mapped to
    - Manual
    - Simulation
    - Complex cost function/algorithm

Implementation estimates

- \( E = \sum_i \sum_d W_{i,d} \)
- Time complexity: \( O(n) \)

\[ E(B1,PE1)_{x,int} = 7 \times 1 = 7 \]


Vocoder Profiling

- Computational complexity of top-level Vocoder behaviors:
  - LP_Analysis: 377.0 MOp
  - Open_Loop: 337.1 MOp
  - Closed_Loop: 478.7 MOp
  - Codebook: 646.4 MOp
  - Update: 43.6 MOp

- Codebook operation mix:
  - \([x, \text{int}]\): 46.2%
  - \([+, \text{int}]\): 33.5%
  - \([-\text{int}]\): 9.1%
  - \([/\text{int}]\): 7.1%
  - \([\text{others, int}]\): 4.1%

Floating-point not required
Dedicated hardware multipliers

HW acceleration

Source: EE382V: Embedded Sys Dsgn and Modeling, Lecture 8 © 2011 A. Gerstlauer
Lecture 8: Outline

- Evaluation and estimation
  - Static analysis
  - Profiling and simulation

- Refinement and implementation
  - Refinement flow
  - Automatic model generation

- System-on-Chip Environment (SCE)
  - Tools and flow
  - Vocoder tutorial

- Mapping of 8 top-level encoder behaviors onto ColdFire + DSP + HW
- 85:04h for 6561 alternatives (1.7s simulation + 3s refinement each)
- 100% fidelity
Virtual Platform Prototyping

Computation refinement

- Untimed
- TLM (LT/AT)
- PCAM

Communication refinement

Virtual Prototype

System Design Flow

- Abstraction based on level of detail & granularity
- Computation and communication

- System design flow
  - Path from model A to model F

- Design methodology and modeling flow
  - Set of models and transformations between models

A. System specification model
B. Timed functional model
C. Transaction-level model (TLM)
D. Bus cycle-accurate model (BCAM)
E. Computation cycle-accurate model (CCAM)
F. Cycle-accurate model (CAM)

The SpecC Methodology

System design
- Capture
  - Specification model
  - Computation refinement
  - Communication refinement
  - Implementation model

Validation flow
- Compilation
- Validation
- Analysis
- Estimation
- Simulation model

Computation model
- IP
  - Algorithmic behavior
  - No implementation details

Communication model
- IP
  - Untimed
  - Executes in zero (logical) time
  - Causal ordering
  - Events only for synchronization

Implementation model
- IP
  - Hardware synthesis
  - Interface synthesis
  - Software synthesis
  - RTOS

Specification Model
- High-level, abstract model
  - Pure system functionality
  - Algorithmic behavior
  - No implementation details

No implicit structure / architecture
- Behavioral hierarchy

Untimed
- Executes in zero (logical) time
- Causal ordering
- Events only for synchronization
### Specification Model Example

- **Synthesizable specification model**
  - Hierarchical parallel-serial composition
  - Communication through variables and standard channels

### Computation Refinement

- **PE allocation / selection**
- **Behavior partitioning**
- **Variable partitioning**
- **Scheduling**
PE Allocation, Behavior Partitioning

- Allocate PEs
- Partition behaviors
- Globalize communication

➢ Additional level of hierarchy to model PE structure

Model after Behavior Partitioning

➢ Synchronization to preserve execution order/semantics
Variable Partitioning

- Shared memory vs. message passing implementation
  - Map global variables to local memories
  - Communicate data over message-passing channels

Model after Variable Partitioning

- Keep local variable copies in sync
  - Communicate updated values at synchronization points
  - Transfer control & data over message-passing channel
Timed Computation

- Execution time of behaviors
  - Estimated target delay / timing budget
- Granularity
  - Behavior / function / basic-block level

➤ Annotate behaviors
  - Simulation feedback
  - Synthesis constraints

```c
behavior B2( in int v1, I5send c2 )
{
  void main( void ) {
    waitfor( B2_DELAY1 );
    c2.send( ... );
    ... 
    waitfor( B2_DELAY2 );
  }
}
```

Scheduling

➤ Serialize behavior execution on components

- Static scheduling
  - Fixed behavior execution order
  - Flatten behavior hierarchy
- Dynamic scheduling
  - Pool of tasks
  - Scheduler, abstracted OS
**Computation Model Example**

**Computation Model**

- **Component structure/architecture**
  - Top level of behavior hierarchy

- **Behavioral/functional component view**
  - Behaviors grouped under top-level component behaviors
  - Sequential behavior execution

- **Timed**
  - Estimated execution delays
Communication Refinement

- Network allocation / protocol selection
- Channel partitioning
- Protocol stack insertion
- Inlining

Network Allocation / Channel Partitioning

- Allocate busses
- Partition channels
- Update communication

➢ Additional level of hierarchy to model bus structure
Model after Channel Partitioning

Protocol Insertion

- Insert protocol layer
  - Bus protocol channel from database
- Create network layers
  - Implement message-passing over bus protocol
- Replace bus channel
  - Hierarchical combination of complete protocol stack
Model after Protocol Insertion

Master

Slave

Inlining: Transaction-Level Model (TLM)

• Create bus interfaces and drivers
Inlining: Pin-Accurate Model (PAM)

- Create bus interfaces and drivers
- Refine communication

Communication Model Example
Communication Model

- Component & bus structure/architecture
  - Top level of hierarchy
- Bus-functional component models
  - Timing-accurate bus protocols
  - Behavioral component description
- Timed
  - Estimated component delays
  - Timing-accurate communication

- Transaction-level model (TLM)
- Pin-accurate model (PAM)
  - Bus cycle-accurate model (BCAM)

Processor Refinement

- Cycle-accurate implementation of PEs
  - Hardware synthesis down to RTL
  - Software synthesis down to IS
  - Interface synthesis down to RTL/IS
**Hardware Synthesis**

- **Schedule operations into clock cycles**
  - Define clock boundaries in leaf behavior C code
  - Create FSMD model from scheduled C code
    - Controller + datapath

**Software Synthesis**

- **Implement behavior on processor instruction-set**
  - Code generation
  - Compilation
Interface Synthesis

- Implement communication on components
  - Hardware bus interface logic
  - Software bus drivers

Implementation Model

Software processor

Custom hardware
Implementation Model

- **Cycle-accurate system description**
  - RTL description of hardware
    - Behavioral/structural FSMD view
  - Object code for processors
    - Instruction-set co-simulation
  - Clocked bus communication
    - Bus interface timing based on PE clock

Lecture 8: Outline

- **Evaluation and estimation**
  - Static analysis
  - Profiling and simulation
- **Refinement and implementation**
  - Refinement flow
  - Automatic model generation
- **System-on-Chip Environment (SCE)**
  - Tools and flow
  - Vocoder tutorial
Design Automation

- Synthesis = Decision making + model refinement
- Successive, stepwise model refinement
- Layers of implementation detail

System-On-Chip Environment (SCE)

- Specification
- System Design
  - Architecture Exploration
  - Scheduling Exploration
  - Network Exploration
  - Communication Synthesis
- TLM
- System models
- Hardware Synthesis
- Software Synthesis
- Implementation Model

Design Decisions

Compile onto MPSoC platform

Synthesize target HW/SW
Abstract Programming Model

- Hierarchical process graph
  - Sequential processes
    - ANSI C code
  - Parallel-serial composition
    - Dependencies, fork-join
- Abstract inter-process communication
  - Communication channels
    - Message-passing, queues, etc.
  - Shared variables

System Transaction-Level Model (TLM)

- Compile onto multi-core/multi-processor platform
  - Implement computation on processors/cores and busses
  - Generate code for communication over bus network

- Generate MPSoC TLM simulation model
- Fast and accurate for exploration and verification
System Implementation

- Synthesize hardware and software for each processor
  - High-level/behavioral RTL and interface synthesis
    - Allocation, scheduling, binding
  - Software synthesis and RTOS targeting
    - Code generation, firmware synthesis, cross-compile & link

Cellphone Example: Hardware Platform

- 2 Subsystems
  - ARM7TDI
  - Motorola DSP 56600k
- 4 Busses
  - AMBA AHB
  - DSP bus
  - IP & memory busses
- 2 Accelerator HW/IP blocks
  - DCT IP
  - Custom codebook HW
- 10 I/O HW blocks
Cellphone Example: Software Platform

- Operating systems
  - ARM7
    - uCOS-II
  - DSP
    - Custom, interrupt-driven multi-tasking kernel

Cellphone Example: Application

- Computation
  - ARM7
    - MP3 Decoding
  - JPEG Encoding
  - DSP
    - GSM Transcoding
**Cellphone Example: Application**

- **Computation**
  - ARM7
  - MP3 Decoding
  - JPEG Encoding
  - DSP
  - GSM Transcoding

- **Communication**
  - Shared memory
    - Camera frames
  - Message-passing
    - MP3 and speech streaming

**Design Methodology**

- **Capture**
- **Algorithm IP**
- **Computation**
- **Specification model**
- **Validation**
- **Analysis**
- **Compilation**
- **Simulation model**
- **Implementation**
- **Hardware synthesis**
- **Software compilation**
- **RTL IP**
- **Implementation model**
- **Validation**
- **Analysis**
- **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

Cellphone Example: Architecture Model

- **Partitioning, synchronization, message-passing, RPC**
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

Cellphone Example: Scheduled Model

- **Scheduling, task refinement, OS model insertion**
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
Cellphone Example: Network Model

- Data conversion, channel merging
- CE insertion, packeting, routing

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

Baseband Example: TLM

- **Synchronization, addressing, media access**
- **Arbitration, data slicing, interrupt handling**
**Baseband Example: Pin-Accurate Model**

- Implementation synthesis in backend tools
  - Interface and high-level synthesis on hardware side
  - Firmware, RTOS and C synthesis on the software side

**Cellphone Example: Single-Processor Results**

- MP3 decoder on ARM
  - 55 MP3 frames
- JPEG encoder on ARM
  - 30 116x96 pictures
- GSM vocoder on DSP
  - 163 speech frames encoded/decoded

- P-TLM speed
  - 2000 Mcycles/s peak
  - 300-600 Mcycl/s sustained

- P-TLM accuracy
  - <3% average frame timing error
Cellphone Example: Multi-Processor Results

- **Experimental setup**
  - 1.5 second MP3
  - 640x480 picture
  - 1.5 speech GSM
- 3s / 300M ARM cycles / 180M DSP cycles

- **Simulation speed**
  - 300 Mcycles/s
- **Accuracy**
  - <3% error

Prototyping and exploration with 100% fidelity

Lecture 8: Outline

- **Evaluation and estimation**
  - Static analysis
  - Profiling and simulation

- **Refinement and implementation**
  - Refinement flow
  - Automatic model generation

- **System-on-Chip Environment (SCE)**
  - Tools and flow
  - Vocoder tutorial
System-on-Chip Environment (SCE)

- Server and accounts
  - ECE LRC Linux servers
    - Labs (ENS 507) or remote access (ssh)
  - SpecC software (© by CECS, UCI)
    - /usr/local/packages/sce-20100908
    - module load sce

- SCE tool set
  - GUI
    - sce, sced, scchart
  - 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)

SCE Tool Flow

- 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 ...
SCE Main Window

SCE Source Editor
SCE Hierarchy Displays

SCE Compiler and Simulator
SCE Profiling and Analysis

GSM Vocoder

- **Enhanced Full Rate (EFR) standard (GSM 06.10, ETSI)**
  - Lossy voice encoding/decoding for mobile telephony
    - Speech synthesis model
      - Long-term pitch filter
      - Delay/Adaptive codebook
      - Residual pulses
      - Fixed codebook
      - Short-term synthesis filter
      - 10th-order LP filter
      - Speech
      - 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

Vocoder Computation Model
Vocoder Communication 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

Code Size

Refinement Effort

<table>
<thead>
<tr>
<th>Spec → Arch</th>
<th>Modified lines</th>
<th>Manual</th>
<th>Automated User / Refine</th>
</tr>
</thead>
<tbody>
<tr>
<td></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

Lecture 8: Summary

- Evaluation and estimation
  - Analysis and simulation

- Refinement and implementation
  - Virtual prototyping at varying levels of detail

- System-on-Chip Environment (SCE)
  - Automatic model refinement and generation