Lecture 6: Outline

• **Synthesis**
  • System-level synthesis process

• **Refinement**
  • Modeling flow
  • The SpecC methodology

• **System-on-Chip Environment (SCE)**
  • Tool flow
  • Setup and tutorial
Synthesis

• Gajski's Y-Chart

System Synthesis

Behavior

Structure

Processor

Logic
Synthesis

• Platform-based design (the other Y Chart)

Synthesis

• X-Chart

Specification

Implementation
Synthesis

- System-Level Synthesis


Application Specification (MoC)

Computation
- Processes

Communication
- Channels
- Variables
Platform Architecture Template

Components:
- Processors
- Memories
- IPs, custom HW
- Buses, bridges

System Definition

Mapping decisions:
- Allocation
- Partitioning
- Scheduling

System Definition = Application + Platform + Mapping
Refined Transaction-Level Model

**Backend Hardware/Software Synthesis**
Lecture 6: Outline

- **Synthesis**
  - System-level synthesis process

- **Refinement**
  - Modeling flow
  - The SpecC methodology

- **System-on-Chip Environment (SCE)**
  - Tool flow
  - Setup and tutorial
Hardware vs. Software

- Double-roof model

**Software**
- system
- task
- instruction
- ISA

**Hardware**
- component
- logic
- RTL


Computation vs. Communication

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

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

Abstraction Levels

Structure

- unstructured
- physical layout
- Spatial order

Implementation Detail

- High abstraction
- Low abstraction

Timing

- untimed
- real time
- Temporal order

Top-Down Design Flow

- requirements
  - pure functional
  - bus functional
  - RTL / ISA
- Structure
- Product planning
  - Specification
  - System Design
  - Architecture
  - Processor Design
  - Implementation
- Logic Design
- Timing
  - constraints
  - untimed
  - timing accurate
  - cycle accurate
  - gate delays
### SpecC Design Methodology

**SpecC Design Methodology**

- **Requirements**
  - Product planning
  - Specification model
  - Computation design
  - Communication design
  - Processor design
  - Implementation model

- **Logic design**
  - Structure
  - Timing

**Pure functional**
- **Computation model**
- **Communication model**
- **RTL / IS**

**Partitioned**
- **Computation model**
- **Communication model**
- **RTL / IS**

**Bus functional**
- **Computation model**
- **Communication model**
- **RTL / IS**

**RTL / IS**
- **Hardware synthesis**
- **Interface synthesis**
- **Software synthesis**
- **RTOS IP**

**Logic design**
- **Product planning**
- **Capture**
- **Aspr. IP**
- **Comp IP**

**Product planning**
- **Capture**
- **Aspr. IP**
- **Comp IP**

**Constraints**
- **Untimed**
- **Scheduled**
- **Timing accurate**
- **Cycle accurate**

**Requirements**
- **Capture**
- **Aspr. IP**
- **Comp IP**

**Product planning**
- **Capture**
- **Aspr. IP**
- **Comp IP**

**Constraints**
- **Untimed**
- **Scheduled**
- **Timing accurate**
- **Cycle accurate**
SpecC Design Methodology

System design validation flow:
- Specification model
- Computation model
- Communication model

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

Backend:
- Estimation
- Validation
- Analysis
- Compilation
- Simulation model

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, ISend c2 )
{
    void main( void ) {
        
        waitfor( B2_DELAY1 );
        c2.send( ... );
        
        waitfor( B2_DELAY2 );
    }
}
```

Scheduling

➤ **Serialize behavior execution on components**

- **Static scheduling**
  - Fixed behavior execution order
  - Flattened 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

Bus1

PE1

BusProtocol

PE2

BusSlave

B1

B13snd

B2

B34rcv

B13rcv

B3

B34snd

PE1

PE2

Bus1

BusProtocol

IBusSlave

IBusMaster

addr[16]
data[32]
ready
ack

BusProtocolTLM

PE1Bus

PE2Bus

create bus interfaces
and drivers

• Create bus interfaces
and drivers

Inlining: Transaction-Level Model (TLM)
**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 6: Outline

- Synthesis
  - System-level synthesis process

- Refinement
  - Modeling flow
  - The SpecC methodology

- System-on-Chip Environment (SCE)
  - Tool flow
  - Setup and 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$j$
- Hardware Synthesis
- Software Synthesis
- Implementation Model

Design Decisions

Compile onto MPSoC platform

Synthesize target HW/SW

PE/OS Models
CE/Bus Models

RTL DB
HW.v

CPU.bin

SW DB

RTL

© 2014 A. Gerstlauer
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
  - ARM7TDMI
  - 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
  - Specification model
    - Computation refinement
      - Computer IP
    - Computation model
      - Validation
        - Analysis
      - Estimation
    - Communication refinement
      - Protob IP
      - Communication model
        - Validation
          - Analysis
        - Estimation
      - Implementation model
        - Validation
          - Analysis
        - Estimation
      - 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

**Cellpone 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

Cellphone Example: TLM

- Synchronization, addressing, media access
- Arbitration, data slicing, interrupt handling
Cellphone 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

- **TLM speed**
  - 2000 Mcycles/s peak
  - 300-600 Mcycl/s sustained
- **TLM accuracy**
  - <3% average frame timing error
Cellphone Example: Multi-Processor Results

- **Experimental setup**
  - 1.5 second MP3
  - 640x480 picture
  - 1.5 speech GSM

- **TLM simulation speed**
  - 300 Mcycles/s

- **TLM accuracy**
  - <3% error

- Prototyping and exploration with 100% fidelity

SCE Exploration Results

- **Suite of industrial size examples**

<table>
<thead>
<tr>
<th>Example</th>
<th>System Architecture (masters → slaves)</th>
<th>Model size (LOC)</th>
<th>Generation time</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPEG A1</td>
<td>CF→HW</td>
<td>1806</td>
<td>1.01 s</td>
</tr>
<tr>
<td>Vocoder A1</td>
<td>DSP→HW</td>
<td>7385</td>
<td>4.77 s</td>
</tr>
<tr>
<td>Vocoder A2</td>
<td>DSP→HW1,HW2</td>
<td></td>
<td>5.21 s</td>
</tr>
<tr>
<td>Vocoder A3</td>
<td>DSP→HW1,HW2,HW3</td>
<td></td>
<td>5.76 s</td>
</tr>
<tr>
<td>Mpdfloat A1</td>
<td>CF→HW1,HW2,HW3</td>
<td>6800</td>
<td>5.86 s</td>
</tr>
<tr>
<td>Mpdfloat A2</td>
<td>CF→HW1,HW2,HW3</td>
<td></td>
<td>6.18 s</td>
</tr>
<tr>
<td>Mpdfloat A3</td>
<td>CF→HW1,HW2,HW3</td>
<td></td>
<td>17.69 s</td>
</tr>
<tr>
<td>Mpdfix A1</td>
<td>ARM→2 I/O</td>
<td>13363</td>
<td>3.85 s</td>
</tr>
<tr>
<td>Mpdfix A2</td>
<td>ARM→2 I/O, LDCT, RDCT</td>
<td></td>
<td>7.01 s</td>
</tr>
<tr>
<td>Mpdfix A3</td>
<td>ARM→2 I/O, LDCT, RDCT</td>
<td></td>
<td>7.40 s</td>
</tr>
<tr>
<td>Baseband A1</td>
<td>DSP→HW, 4 I/O, CF, DMA→Mem, BR, T, DMA</td>
<td>11481</td>
<td>8.99 s</td>
</tr>
<tr>
<td>Cellphone A1</td>
<td>ARM→4 I/O, 2 DCT, DSP→HW, 4 I/O, T</td>
<td>16441</td>
<td>9.49 s</td>
</tr>
</tbody>
</table>
Lecture 6: Outline

- **Synthesis**
  - System-level synthesis process

- **Refinement**
  - Modeling flow
  - The SpecC methodology

- **System-on-Chip Environment (SCE)**
  - Tool flow
    - Setup and 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 (*sc*)
  - Profiling and analysis (*scprof*)
  - Architecture refinement (*scar*)
  - RTOS refinement (*scos*)
  - Network refinement (*scnr*)
  - Communication refinement (*scrr*)
  - RTL refinement (*scrl*)
  - 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 Tutorial

- Enhanced Full Rate (EFR) standard (GSM 06.10, ETSI)
  - Lossy voice encoding/decoding for mobile telephony
    - Speech synthesis model
      - Input: speech samples at 104 kbit/s
      - Frames of $4 \times 40 = 160$ samples ($4 \times 5$ms = 20ms of speech)
      - Output: encoded bit stream at 12.2 kbit/s (244 bits per 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
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

- **Productivity gain**: 12 months vs. 1 hour = 2000x
Lecture 6: Summary

- System-level synthesis
  - X-chart

- System-level refinement
  - SpecC methodology
  - Flow of models at increasing level of detail

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