Part 1: Introduction to Hardware/Software Codesign

Ahmed A. Jerraya
TIMA Laboratory
46 Avenue Felix Viallet
38031 Grenoble Cedex France
Tel: +33 476 57 47 59
Fax: +33 476 47 38 14
Email: Ahmed.Jerraya@imag.fr
Hardware Software Codesign

- Principle: Joint development of hardware and software starting from a high level specification

- Application: Telecommunication, Automotive, Consumer, ...

- Challenges:
  - Reducing the design time
  - Reducing the cost of modification/evolution
  - Optimizing the hardware software partitioning
Objectives

- Mixed HW/SW Systems
- Co-design techniques
- Models for Co-design
- Tools and Methods: State of the Arts
Outline

Part 1

1. Introduction
   - System design
   - Models for codesign

2. Codesign techniques
   - Refinement from system-level specification
   - HW/SW co-simulation

3. Example: Codesign from SDL
   - SDL
   - Design example

4. Future directions and conclusion
Video Codec (H261)

- SW PROCESSORS
- HARDWARE PROCESSORS

Ahmed A. Jerraya Tutorial DATE 9-03-99
Mobile Telecom Terminal
(e.g. GSM)

Different Kind of Expertises

Different Kind of Languages

- Protocol : SDL
- DSP : COSSAP, Matlab
- RF : VHDL-AMS, SABER
- OCI : SDL, VHDL
Electronic parts
- may amount to > 10% of the total cost
- critical part of car design
Mixed HW-SW Design

MAIN PROBLEM: - Different cultures.

(H/W design tools (COSSAP, VHDL, Verilog, VHDL-AMS, ...)

(System Design Tools SDL, StateCharts, Java, Matlab, ...)

HW + SW

System-Level Model

LINKING CASE TOOLS & IC CAD TOOLS

Ahmed A. Jerraya Tutorial DATE 9-03-99
Outline
Part 1

1. Introduction
   - System design
   - Models for codesign

2. Codesign techniques
   - Refinement from system-level specification
   - HW/SW co-simulation

3. Example : Codesign from SDL
   - SDL
   - Design example

4. Future directions and conclusion
Models and Steps for Codesign

System-Level Modeling

1- System Level Modeling Language

System-Level Synthesis

2- System Level Synthesis Intermediate Form

3- Architectural Model

Protocol e.g. IEEE 488

Other Subsystems

Analog Subsystems

ASIPS μ controllers DSP

Existing Hardware & ASICs
Architecture for Codesign

- MONO-PROCESSOR:
  - A top Controller
    - FSM
    - Programmable Controller (ASIP), (standard Processor)
  - Execution Part
    - Fus: (Operators, Coprocessors, Memory, Interface Units)
    - Communication (Muxes, Buses, Registers, …)

- MULTI-PROCESSOR:
  - Set Of Communicating Processors
  - Communication Network
    - (Bus-Controllers, arbiters, layered Networks …)
  - Very difficult to model using a pure synchronous language

Ahmed A. Jerraya

Tutorial DATE 9-03-99
Model

- Components: Processors
- Links: Communication
- Composition: Hierarchy

Example

Software Processors
- CPU+
- ROM+
- I/O
- Existing Components

HW Processors
- ASIC
- FPGA
- I/O

Communication Processors
- FPGA
- Memory
- FIFO
# Models for Hardware/Software Specification

<table>
<thead>
<tr>
<th>System Level</th>
<th>Time Unit</th>
<th>HW/SW Models</th>
<th>HW Language</th>
<th>SW Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithmic</td>
<td>Computation step</td>
<td>CFG-DFG</td>
<td>VHDL, VERILOG, C</td>
<td>C, C++</td>
</tr>
<tr>
<td>RTL Level</td>
<td>Clock cycle</td>
<td>FSM-Bool.eq Low Level code</td>
<td>VHDL</td>
<td>ASM, RTL-C</td>
</tr>
</tbody>
</table>

- Task graph
- Communicating Processes

- SPW, COSSAP, StateCharts, SDL, Java, ...
- VHDL, VERILOG, C
- SDL, Matlab, SART, Java, ...

Ahmed A. Jerraya

Tutorial DATE 9-03-99
Increasing the Abstraction Level

- System-level Specification
  - Architecture exploration
    - SW High Level C
      - Software Compilation & Processor Design
        - Processor
        - CODE
    - HW Behaviour VHDL
      - Behavioral Synthesis
        - HW

- 50k SLDL Transactions
- 150k VHDL computation steps
- 350k RTL VHDL clock cycles
- 1000k Gates
Outline
Part 1

1. Introduction
   - System design
   - Models for codesign

2. Codesign techniques
   - Refinement from system-level specification
   - HW/SW co-simulation

3. Example: Codesign from SDL
   - SDL
   - Design example

4. Future directions and conclusion
Refinement from System Level Specification

- **Partitioning:**
  - Allocation: fixes number and types of processors
  - Binding: assign function to processors

- **Communication Synthesis:**
  - Protocol selection: select a protocol to execute abstract communication
  - Interface synthesis: adapt the unit interfaces to the selected protocols

- **Interface synthesis:** adapts the application to the interface of the processor

- **Software synthesis:**
  - fixes the execution order of parallel tasks
  - Targets a specific processor

- **Hardware Synthesis:** decomposes computation steps into clock cycles
Partitioning

- Allocation: Fixes the number and types of processors
- Binding: Assign functions to processors
- Main Problems
  - Cost function
  - Granularity
  - Communication estimation
Communication Synthesis

**STEPS:**
- Protocol Selection
- Interface Synthesis

Pi : Processor
Li : Logical Channel
Ri : Physical Channel

Ahmed A. Jerraya Tutorial DATE 9-03-99
HW/SW Interfaces Synthesis

SW-Model

HW Model

Processor Library

Low-Level C code

Micro-Processor

IO

HW

RTL HDL

 interrupts

ADR

Data
SW Code Generation

- **The problem:**
  - Input = A task graph
  - Output = Code for a specific architecture

- **The solutions:**
  - RTOS-based
  - Software synthesis
  - RTOS synthesis
Problems to be solved

- Task inter-dependence = avoids dead locks
- I/O = adapts to different I/O schemes
- Synchronization = avoids active waiting
- Performances = code size, execution speed
- Quality
Hardware Synthesis

System-Level model
Transition = Computation step

Synthesizable
Transition = clock cycle
Outline
Part 1

1. Introduction
   - System design
   - Models for codesign

2. Codesign techniques
   - Refinement from system-level specification
   - HW/SW co-simulation

3. Example: Codesign from SDL
   - SDL
   - Design example

4. Future directions and conclusion
Co-simulation

- **Principle:** Simulation of a multilanguage model

- **Engine Model**
  - Single-engine: Code-level simulation
  - Multi-engine: Source-level simulation

- **Synchronization**
  - Master-slave model
  - Distributed model

- **Timing model**
  - Zero delay operations: functional validation
  - Timed operations: timing validation
**Co-simulation**

Single engine vs multi-engine Co-simulation Approach

**Single Simulation engine**
- Code-level simulation
- High speed
- Efficient when restricted to specific application domain

**Multiple Simulation engine**
- Source-level simulation
- Modular design simulation & debug
- Keeps the model semantic

**Unified Model**

**Simulator 1**

**Simulator 2**

**Simulator n**

**COSIMULATION BUS**
Co-simulation (C-VHDL)

**MASTER SLAVE MODEL**
- VHDL
  - HW BEHAVIOR
- CLI
- C- PROGRAM

**DISTRIBUTED MODEL**
- VHDL
- CLI
- C- PROGRAM
- SW BUS: UNIX (IPC/Sockets)

- Makes use of VHDL extension (CLI)
- C program is called as a procedure
- When distributed computation required, the C code needs to be organized as an FSM (scheduled)

- Uses UNIX-like Communication Model
- Fully distributed Computation Model
- Needs a Synchronization Model
- Allows to debug HW and SW using specific debug Tools
- Multi-C Modules

Ahmed A. Jerraya

Tutorial DATE 9-03-99
Implementing Distributed C - VHDL Co-simulation

main() {
    int a, b;
    exin(a);
    b = f(a);
    exout(b);
}

Each simulator communicates via a fragment of shared memory (or queues or ...).

The router secures the coherence between several memories via sockets.
Multilanguage Co-simulation (Distributed Model)

VHDL

CLI

C- PROGRAM

LX

INT

Co-simulation BUS (e.g. UNIX/IPC)
Automatic Generation of Co-simulation Interfaces

Module 1

Module 2

... Module n

Configuration file

Co-simulation Interfaces generation

Module 1

Interface 1

Module 2

Interface 2

... Module n

Interface n

Co-simulation Bus
Control of Motors
1. Introduction
   - System design
   - Models for codesign

2. Codesign techniques
   - Refinement from system-level specification
   - HW/SW co-simulation

3. Example: Codesign from SDL
   - SDL
   - Design example

4. Future directions and conclusion
**SDL**

- **SDL** (Specification and Description Language): an ITU-T Standard for the Specification of telecommunications systems
- **Concurrency**: Distributed systems
- **Hierarchy**: System, Block, Process
- **Communication**: message passing
- **Synchronization**: Asynchronous Communication Implicit Queues

Ahmed A. Jerraya

**Tutorial DATE 9-03-99**
The Process: Behavior

PROCESS SENDER;

DCL Next_to_Send INTEGER;
...
START;
  TASK Next_to_Send := 0;
  TASK Data := Next_to_Send;
  OUTPUT Data;
  NEXT STATE WAIT_Ack;

STATE WAIT_Ack;
  INPUT Nack;
  ...
  INPUT Ack;
  ...

ENDPROCESS SENDER;

Next_to_Send := 0
Data := Next_to_Send

WAIT_Ack

Nack

Data

Ack

Next_to_Send := Next_to_Send + 1
Data := Next_to_Send

Data

Ahmed A. Jerraya Tutorial DATE 9-03-99
Communication in SDL

- Two Communication Levels:
  - Inter blocks: Channels
  - Inter Processes: SignalRoutes
- Processes exchange messages

(ADT)
- Layered Communication:
  A channel may be specified as a system (blocks + channels)

Ahmed A. Jerraya
Tutorial DATE 9-03-99
Outline
Part 1

1. Introduction
   - System design
   - Models for codesign

2. Codesign techniques
   - Refinement from system-level specification
   - HW/SW co-simulation

3. Example: Codesign from SDL
   - SDL
   - Design example

4. Future directions and conclusion
CODESIGN from SDL

- Initial model SDL
- SYSTEM REFINEMENT
- Virtual Prototype
  - C
  - VHDL
- Architecture

- Rapid prototyping
- Flexible Synthesis Flow: Designer in the Loop
Co-design flow

System-Level Specification

Functional Partitioning

Communication Synthesis

Virtual Processor Allocation

HW-SW Co-Generation

Ahmed A. Jerraya
Example: ATM Network Interface Card

Application

TCP
IP
AAL
ATM

Flexible Partitioning

Application

TCP
IP
AAL
ATM

Physical layer

SW
HW
System Specification in SDL

Software
166Mhz Pentium Processor

Hardware
ASIC
### Architecture Exploration

**Resources:** Pentium based SW / Dedicated Hardware  
**Constraints:** Performance (5k cycles / cell = 25 Mb/s)

<table>
<thead>
<tr>
<th></th>
<th>TCP</th>
<th>IP</th>
<th>AAL</th>
<th>ATM</th>
<th>Performance (cycles/frame)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW</td>
<td>SW</td>
<td>SW</td>
<td>HW</td>
<td></td>
<td>6 Mb/s</td>
</tr>
<tr>
<td>SW</td>
<td>SW</td>
<td>SW</td>
<td>SW</td>
<td></td>
<td>12 Mb/s</td>
</tr>
<tr>
<td>SW</td>
<td>SW</td>
<td>HW</td>
<td></td>
<td></td>
<td>19 Mb/s</td>
</tr>
<tr>
<td>HW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>60 Mb/s</td>
</tr>
<tr>
<td>SW</td>
<td>HW</td>
<td></td>
<td></td>
<td></td>
<td>41 Mb/s</td>
</tr>
</tbody>
</table>

Ahmed A. Jerraya  
Tutorial DATE 9-03-99
Outline
Part 1

1. Introduction
   - System design
   - Models for codesign

2. Codesign techniques
   - Refinement from system-level specification
   - HW/SW co-simulation

3. Example: Codesign from SDL
   - SDL
   - Design example

4. Future directions and conclusion
State of the Arts

- Plethora of academic tools:
  COSYMA, POLIS, COSMOS, VULCAN, LYCOS, ...

- Emerging commercial solutions:
  Eagle, Felix, Seamless, Archgen, C-level-design, CoWare, Arexsys
Gap between Dreams & Reality

Early codesign tools have created exaggerated hopes because:

1- An under-estimation of the gap between system specification and state of the art tools: system-level is far from RTL.

2- An over-estimation of language’s capabilities: systems are heterogeneous.

3- An under-estimation of the importance of behavioral synthesis.

4- An under-estimation of multiprocessor architecture models.
What is needed: linking system design tools to implementation

System-level modelling and design tools
- Matlab
- SDL
- IP (C, VHDL, ...)
- SPW, Cossap ...

MISSING LINKS
- Global validation: System, Environment, IP
- Architecture exploration
- HW-SW Co-generation

SW DESIGN ???? HW DESIGN

Co-simulation
Conclusion

KEY POINTS

- Codesign applications: Circuits & large distributed systems
- Codesign models:
  - One processor vs multiprocessor
  - Homogeneous models vs Heterogeneous models
- Codesign techniques:
  - Refinements
  - Cosimulation
- State of the Arts
  - Emerging solutions