A Unified HW/SW Interface Model to Remove Discontinuities between HW and SW Design

Ahmed Amine JERRAYA

TIMA Laboratory
46 Avenue Felix Viallet
38031 Grenoble CEDEX, France
Email: Ahmed.Jerraya@imag.fr
The SoC Era Challenges

- **SoC**: put on a chip what we used to put on one or several boards (ASIC, CPU, Memories, Analog/RF, MEMS, …)

- **Facts:**
  - 90% of new ASICs already include a CPU in 130nm.
  - Multimedia, network processors, mobile terminals and game applications are already multiprocessors.

- **Fundamental changes:**
  - SoC is different from ASIC
  - SoC is different from SW
  - SoC requires abstract HW-SW interfaces to allow fast integrations.

- **Challenges:**
  - Generic SoC platform (programmable, reconfigurable, …)
  - Specific SoCs using standard IP with specific interconnect.
Generic SoC Platform vs. Application-Specific MPSoC

Example: The GSM History/Roadmap

- 1986  Rack in a van
- 1990  PCB
- 1995  Chip set in a hand-set
- 2002  SoC: Specific HW + CPU executing SW
- 2006  SW component on a generic platform, e.g. Nomadic (ST)

Same roadmap for game computers, MP3, STB, NP, DVD
Why SoC Design is Needed

- Applications
  - Entertainment
  - Security
  - Networking
  - Model terminal
  - Productivity enhancements

- Example: MPEG2 encoding
  - 2000x1000 frame
  - Full motion search 128x128 search window
  - 32 TIPS (TERA instruction per second)
  - All software: 32 000 RISC CPU, 1Ghz
  - SoC Solution: 4 embedded specific CPU 200 Mhz
The key SoC design Issue: HW/SW Gap

- Different Concepts to Abstract Interfaces
  - HW communicates through wires.
  - SW communicates through APIs

- The gap: SW model hides a CPU.

- HW-SW interfaces includes HW, SW and CPU

![Diagram showing HW and SW components with a gap in between](image-url)
Outline

- Defining HW-SW Interfaces
- HW-SW Discontinuities
- The Service Dependency Graph (SDG) Model
- Building Custom HW-SW Interfaces
- Application example
Defining HW-SW Interfaces

- Application SW Designer: A set of system calls used to hide the underlying execution platform. Also Called Programming Model
- HW designer: A set of registers, control signals and more sophisticated adaptors to link CPU to HW subsystems.
- System SW designer: Low level SW implementation of the programming Model for a given HW architecture.
- CPU is the ultimate HW-SW Interface
- Sequential scheme assuming HW is ready to start low level SW design
- SOC requirements
  - Abstracts both HW and SW in addition to CPU
  - HW-SW interfaces tradeoff

Sequential SW program
... Call HW (x, y, z)

HW function
wait start
...
**HW/SW Interfaces Abstraction**

- **Different Abstraction Levels for Both HW and SW**
- **Key issue:** a unified model to represent CPU + SW + HW
- **Key benefit:** Formal reasoning, optimization partitioning and early validation of HW-SW interfaces.

---

Ahmed A. Jerraya
Outline

1. Defining HW-SW Interfaces
2. HW-SW Discontinuities
3. The Service Dependency Graph (SDG) Model
4. Building Custom HW-SW Interfaces
5. Application example
Classical SW design flow to interface HW

- **Programming Model:** Abstract HW at Different level
- **Discontinuities:**
  - **Compilation:** Generally ignore the CPU environment (Interrupts, Complex I/O)
  - **Sys.lib:** adapt for different HW
  - **MMAP:** Adapt to different CPU-memory architecture
  - **User.lib:** to make the flow efficient for the application

Diagram:
- Application
- Programming Model (API)
- Architecture
- Code + Calls
- Compiler
- Linker
- Executable Code
- Sys.lib
- MMAP
- User.lib
HW/SW Interfaces in Classical SoC
Design Flow

- System specification is a functional architecture: functional modules using specific programming models connected through abstract Interconnect

- Architecture implementation: heterogeneous components and sophisticated HW/SW interfaces

- HW/SW interfaces
  - Different Models (HW, SW, CPU)
  - Separate Design
  - Not efficient trade-off

- What is needed: A Unified Model for HW/SW Interfaces Codesign
Existing HW-SW Integration Technologies

- Custom HDS generation for existing HW
  - L. Gauthier, L. Benini, D. Gajski, ...

- Custom CPU generation for specific SW
  - Tensilica, ARC, Target, ...

- Rest of the CPU Subsystem and HW are not handled

Diagram:

- Custom HDS Generation
  - Generic HDS
  - Architecture Specification
  - Executable SW

- ASIP Generation
  - SW Program
  - Application Specific CPU
  - Generic CPU
  - Constraints
Key Innovation to Higher Level HW/SW Interface Abstraction: from CPU to SW Execution Subsystem

- Traditional view of SoC
  - CPU is HW-SW interface.
  - SW validation assumes HW ready.

- SoC Model with Abstract HW-SW Interfaces
  - Separate HW and SW design
  - Better HW & SW reuse.

- Unified Model to abstract HW/SW Interfaces
  - models HW, SW and CPU
  - Allows new HW/SW architecture trade-off.

Ahmed A. Jerraya
Outline

1. Defining HW-SW Interfaces
2. HW-SW Discontinuities
3. The Service Dependency Graph (SDG) Model
4. Building Custom HW-SW Interfaces
5. Application example
Service Dependency Graph Model

- Goal: adapting different interfaces
- Service Dependency Graph (SDG)
- Interface Component:
  - atomic unit that provides/requires services
- Previous Works
  - Protocol stack optimization [Zitterbart92]
  - Application specific OS Generation [Gauthier 1999]
Abstract Interface

- Interface is represented as 2 sets of logical ports connected through a SDG.
- Logical Ports: May be hierarchical and hold (provide) a list of services.
- Interface Element: Entity that provides (implements) and/or requires services. May be HW, SW, Functional or Mixed HW/SW.
- Service: Defines a function or communication primitive.
- Edges define the dependency relation between logical ports or interface elements and services.
SDG Combining HDS and CPU subsystem

Application software

Hardware-dependent software

Hardware

CPU subsystem

switch_banks

memory_put

memory_get

block

unblock

interrupt

MEMORY_DRIVER

INTERRUPT_MANAGER

MEMORY_IO

MULTI-BANK MEMORY CONTROLLER

ISA (e.g. load/store)

(other part of HW/SW interface)

CPU

MEMORY BANK (1 & 2)

BUS

NETWORK INTERFACE

mem_put

mem_get

bus_access

p2p_put

p2p_get

AHMED A. JERRAYA
Outline

1. Defining HW-SW Interfaces
2. HW-SW Discontinuities
3. The Service Dependency Graph (SDG) Model
4. Building Custom HW-SW Interfaces
5. Application example
Basic Concept: Composing Interfaces

- Abstract interface
  - Required/Provided services
  - Control and Synchronization services
  - Parameters
- Interface sub-system Design
  - User-extensible library
  - Services matching
  - Code specialization

Interface sub-system composition

Works for HW, SW, Functional and CPU sub-systems at Different abstraction levels
SB-Colif: Application with abstract Interfaces Specification
Library of Interface Components
Interface Generation Flow

Application/Architecture with Abstract Interfaces

Library

HW/SW interface generation tool
- Code selector
- Code generator
- Code integrator

Application/Architecture with explicit Interfaces

Validation/Design

Ahmed A. Jerraya
Outline

1. Defining HW-SW Interfaces
2. HW-SW Discontinuities
3. The Service Dependency Graph (SDG) Model
4. Building Custom HW-SW Interfaces
5. Application example
Application Example: MPEG4

Functional model

Input

Trans.1
Trans.2
Trans.3
Trans.4

DMA

Enco

der

Comb

er

Encoder

Comb iner

F_n
(current)

F_n-1
(ref.)

Motion
Estimate

Motion
Compensate

DCT

Quant

IDCT

IQuant

VLC

Coded Stream

HW/SW Interfaces to be designed
MPEG4 Architecture with Application Specific HW-SW Interfaces

Application software: encoder module

```
taskvlc::task_behavior(), ...
```

Hardware-dependent software

```
get_bank_address(), wait_event(), ...
```

Application software: transform module

```
task1::task_behavior(), ...
```

Hardware-dependent software

```
switch_bank(), wait_event(), ...
```

**CPU subsystem 1**

- **SW Execution Subsystem**
  - **Pvlc**
  - **SRAM**
  - **CPU subsystem**
  - **Ctrl**
  - **Hw**
  - **NI**

**CPU subsystem 2**

- **SW Execution Subsystem**
  - **SRAM**
  - **CPU subsystem**
  - **Ctrl**
  - **Hw**
  - **NI**

**DMA**

- **Video stream**
- **Input**
- **Combiner**
- **Coded Stream**

Ahmed A. Jerraya
Service Dependency Graph for SW2
Perspectives: HDS-CPU Codesign, New Design/Synthesis Area

- SDG to represent Abstract Interfaces, CPU and HDS
- CPU subsystem may require sophisticated architecture in addition to CPU.
- HW-SW trade-off.
Conclusions

- Classical design separates HW and SW interfaces
- SoC design requires to abstract CPU in addition to both HW and SW interfaces.
- Service dependency Graph is a working solution for HW/SW Interfaces abstraction

Perspectives

- HW-SW interfaces codesign
- HW/SW tradeoff between HdS and CPU
- Formal reasoning and synthesis
Thank You
"HW-SW interfaces" includes 3 layers: HDS, CPU, HW

- Use of predesigned layers

Use of custom HDS through personalization of a generic HDS
- Generic CPU but fixed

Use of a custom adaptation through unified HDS-CPU subsystem representation model
A Simple Application Example

Abstract HW/SW Interfaces

Input

DMA (HW)

Output

SW
Mixed Level HW/SW Interfaces

- SW
  - Boot
  - MemBank
  - TaskMgr
  - CPU
  - (Mem) Ctrl

- HW/SW Interfaces

Input

DMA (HW)

Output
HW-SW Interfaces abstraction Levels

Any intermediate level between TLM and RTL?

Ahmed A. Jerraya
Intermediate levels between TLM and RTL

- **Implicit : CPU SS organization**
  - SystemC 3.0: SW Native, BFM interface or CPU SS simulation

- **Implicit CPU organization**
  - BCA: SW Bin, BFM interface or CPU ISA simulation

- **Implicit Physical addressing (MMAP, Booting address, ...)**
  - MAXSIM: SW Bin, CPU ISS cycle accurate with explicit DATA Memory

- **All explicit**
  - Bin SW, RTL CPU + RTL HW + Physical memory

Ahmed A. Jerraya