Hardware-defined Software: Concepts & Architecture

Christian FABRE – LETI – christian.fabre1@cea.fr
Eugenio VILLAR – Univ. of Cantabria – villar@teisa.unican.es
Emmanuel VAUMORIN – MAGILLEM – vaumorin@magillem.com
1. **HW/SW and SW/SW basic dependencies**
   1. SW Taxonomies
   2. Case studies
   3. Basic SW Dependencies Conclusions

2. **Layered Hardware-dependant-Software**
   1. Introduction to HdS Stack
   2. Linux
   3. WinCE
   4. HdS Stack Conclusion

3. **IP Xact & HdS**
Let's Focus on Binary...

• SW IP can be delivered in source or binary form
• Source code is more abstract – Source code is actually a mean to shield from some hardware dependencies
• Binary only delivery won't go away anyway: binary-only-deliveries, specialized compute intensive code, etc
• In this part of the talk, software will mean binary software
What a piece of software does vs. what a piece of software is

• Software for what it does. Keywords are
  – Signal processing, Video compression, Codecs, Baseband radio, UMTS layers, CRC algorithms, etc.

• Software for what it is. Keywords are
  – ARM binary, Libraries, intel ABI, POSIX API, Instruction Set, x86 IS, DSP IS, VxWorks, eCos, etc.
• Provides a business function taxonomy
  – FFT, IFFT, CRC, dithering, etc.

• Provides relevant information to build systems from functions

• Answers the question
  – What function do I have to write/reuse/buy to develop my application?
SOFTSOC

SW for « What it is »

• Details the dependencies of (binary) software towards
  – Hardware
    • Processor: instruction set, register set
    • Memory: Memory map, cache sizes & parameters
    • IP: registers banks, interruption
  – Software
    • Compiler tool chain: ABI (appl. binary interface)
    • Other software: API (appl. programming interface)
  – Operating system:
    • Driver framework
• Basis for assembly rules of SW dedicated on a given hardware
Structure Before Business

• IP-Xact is all about structural description of HW IP
  – So should be its extensions for software

• We need first to express the necessary information required to decide if and how software can be assembled
  – That's structural description of software
  – Such description are mute about the business functions implemented by a piece of software
SOFTSOC

Plan

1. HW/SW and SW/SW basic dependencies
   1. SW Taxonomies
   2. Case studies
   3. Basic SW Dependencies Conclusions

2. Layered Hardware-dependant-Software
   1. Introduction to HdS Stack
   2. Linux
   3. WinCE
   4. HdS Stack Conclusion

3. IP Xact & HdS
• Software that compute cosine(x)
  – The software exported API is named “mycosine.h”
  – Exports a single function “double mycosine(double);”

• The SW IP
  – Is named “mycosine”
  – Is compiled for ia32 with SSE instructions
SOFTSOC

A Simple Software IP (2/2): Detailed Dependencies

- Dependency towards the processor
  - Requires a ia32 processor with SSE instructions
- Dependency towards the calling protocol
  - Export its API through ABI v3.3 of GCC for ia32
- Dependency towards API descriptions
  - Is an implementation of “mycosine.h”
• The software exported API
  – Is named “myfft.h”
  – Exports a single function
    “double[] myfft(double[], size_t n);”

• The software binary implementation
  – Is named “myfft”

• Requires another SW IP
  – mycosine

• Is compiled for ia32 with SSE instructions
Another Simple Software IP (2/2): Detailed Dependencies

- Dependency towards the processor
  - Requires a ia32 processor with SSE instructions
- Dependency towards the calling protocol
  - Obeys ABI v3.3 of GCC for ia32
- Dependency towards API descriptions
  - Is an implementation of “my-fft.h”
- Depends on “my-cosine”
HW IP-dependent SW IP (1/3): User Documentation

- A never-ending countdown (9 to 0) made of

- Hardware
  - A 386 processor
  - A HW IP: 7 segment decoder for LCD display
    - Named "7-seg-lcd-decoder"
    - Exports a single 8 bit register with bits 0-6 writeable
  - The register is mapped at 0x00000140

- Software
  - Closed software
  - No operating system
  - No RTC: Calibrated software loops
SOFTSOC

HW IP-dependent SW IP (2/3): Detailed Dependencies

• Dependency towards the processor
  – Requires a ia32 processor

• Dependency towards the calling protocol
  – Obeys ia32 ABI v3.3

• Dependency towards API descriptions
  – <none>

• Dependency towards the HW IP
  – One instance of hardware IP “7-seg-lcd-decoder”
  – The memory map has it at 0x00000140
HW IP-dependent SW IP (3/3): More Detailed Dependencies

• “One instance 7-seg-lcd-decoder”, means:
  – The SW IP shall drive the IP through registers whose format is defined by the IP

• “The memory map has the decoder at 0x0140”
  – The register is memory mapped
  – The SW IP code shall have the rights to access it
  – Its address is known and constant at 0x0140

• These are strong dependencies on the way the software is architectured/built
• A Processor access several Memory Banks
• Each Memory Bank is accessed at a single Base Address
• A Memory Bank is made of several Memory Cells
• A Register is a Memory Cell
• A HW IP exports several Memory Cells
SOFTSOC

Plan

1. HW/SW and SW/SW basic dependencies
   1. SW Taxonomies
   2. Case studies
   3. Basic SW Dependencies Conclusions

2. Layered Hardware-dependant-Software
   1. Introduction to HdS Stack
   2. Linux
   3. WinCE
   4. HdS Stack Conclusion

3. IP Xact & HdS
Summary of SW/SW dependencies

• Automatic assembly is based on structural dependencies
  – Not easy: Move progressively from simple to more complex cases
  – Otherwise the README file will strike back

• Structural dependencies of HdS are
  – Not only about dependencies over HW IPs
  – But also about deeper SW dependencies
    • Processors
    • Compilers ABI
    • Memory map
    • Boot ordering
Plan

1. HW/SW and SW/SW basic dependencies
   1. SW Taxonomies
   2. Case studies
   3. Basic SW Dependencies Conclusions

2. Layered Hardware-dependant-Software
   1. Introduction to HdS Stack
   2. Linux
   3. WinCE
   4. HdS Stack Conclusion

3. IP Xact & HdS
HdS: The HW IP case

Application – Platform Interface

Hardware – Software Interface

Application and Middleware Software

Operating System

Hardware Dependent Software

Hardware platform

CPU

Memory

IO

HW IP

Prog. unit

Interconnect

Hardware Abstraction

OS Abstraction

Hardware Dependent

OS Dependent
• More complex architectures
  – More complex HdS
  – More complex HdS models
  – Increased dependencies w.r.t. OS, HW, \( \mu \)Processor…

  – Current ‘hand-written’ approach no longer valid
HdS: An structured approach

• Objectives
  – Automate the HdS generation
  – Automate the HdS model generation
  – Capsulate dependencies w.r.t. OS, HW, µProcessor...
HdS: The Linux case

Application SW

HdS1 - iowr, iord

uProcessor | HW Platform | H264
• Depends on:
  – Specific HW (H.264 HW component)
  – uProcessor (only if assembler code)
  – OS (through its register access functions)

• Objectives:
  – Basic functionality
  – Support low-level accesses to HW platform
    • iord → read request
    • iowr → write request
HdS2: The Linux case

Application SW

OS API:
- mem_req
- request_irq
- Scheduler

requests_irq

HdS2:
- dev_init, dev_exit, dev_open,
- dev_close, dev_read, dev_write,
- dev_ioctl, dev_irq_handler[n]

HdS1 - iowr, iord

uProcessor

HW Platform

H264
HdS2: The Linux case

- Depends on:
  - OS/RTOS (Linux in example)
  - Specific HW (H.264 HW component)
  - HdS1

- Objectives:
  - Establishes communication with OS/RTOS
  - Defines OS/RTOS specific access functions
  - Provides services through OS/RTOS general access functions
### HdS3: The Linux case

<table>
<thead>
<tr>
<th>OS API</th>
<th>Open, close, read, write, ioctl</th>
</tr>
</thead>
<tbody>
<tr>
<td>mem_req</td>
<td>HdS2 dev_init, dev_exit, dev_open, dev_close, dev_read, dev_write, dev_ioctl, dev_irq_handler[n]</td>
</tr>
<tr>
<td>request_irq</td>
<td></td>
</tr>
<tr>
<td>Scheduler</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Boot</td>
<td></td>
</tr>
</tbody>
</table>

#### Application SW
- **HdS1 - iowr, iord**
- **HdS2**
- **HdS3 API config, reset, encode**

#### Hardware Platform
- **H264**

#### Components
- **uProcessor**
- **HW Platform**
- **OS**
HdS3: The Linux case

• Depends on:
  – Specific HW (H.264 HW component)
  – HdS2

• Objectives:
  – Defines specific functions according to HW capabilities
  – Encapsulate complex functionality in specific functions
  – Provides abstraction from HW platform
Middleware: The Linux case

Application SW

- GStreamer Plug-in
- CORBA Plug-in
- ...

HdS3 API config, reset, encode

OS API

- open, close, read, write, ioctl

OS

- mem_request_irq
- Scheduler
- ...
- Boot

HdS2

- dev_init, dev_exit, dev_open,
- dev_close, dev_read, dev_write,
- dev_ioctl, dev_irq_handler[n]

HdS1 - iowr, iord

TSE

OS

Sensor

uProcessor

HW Platform

H264
• Depends on:
  – HW platform capabilities
  – HdS3

• Objectives:
  – Adapts HdS3 to libraries and specific software
  – Allows to merge HW platform capabilities
  – In example: camera with H.264 video-encoded streaming
    • Sensor + H.264 encoder + TSE
  – Provides full abstraction from HW platform
    • Only performance-dependent
HdS: The WinCE case

HdS1 - INREG, OUTREG

HdS2
XXX_Init, XXX_Deinit, XXX_Open, XXX_Close, XXX_Read, XXX_Write, XXX_IOControl, XXX_Init

HdS3 API config, reset, encode

Application SW

GStreamer Plug-in
CORBA Plug-in
...

Open, Close, Read, Write, IOControl

OS API

mem_req
request_irq
Scheduler
...

Boot

uProcessor

HW Platform

H264

TSE
Sensor

OS

SSE

HD3 SW

HdS API config, reset, encode

GStreamer Plug-in
CORBA Plug-in
...

OS API

mem_req
request_irq
Scheduler
...

Boot

uProcessor

HW Platform

H264

HdS: The WinCE case

HdS1 - INREG, OUTREG

HdS2
XXX_Init, XXX_Deinit, XXX_Open, XXX_Close, XXX_Read, XXX_Write, XXX_IOControl, XXX_Init

HdS3 API config, reset, encode

Application SW

GStreamer Plug-in
CORBA Plug-in
...

Open, Close, Read, Write, IOControl

OS API

mem_req
request_irq
Scheduler
...

Boot

uProcessor

HW Platform

H264

HdS: The WinCE case

HdS1 - INREG, OUTREG

HdS2
XXX_Init, XXX_Deinit, XXX_Open, XXX_Close, XXX_Read, XXX_Write, XXX_IOControl, XXX_Init

HdS3 API config, reset, encode

Application SW

GStreamer Plug-in
CORBA Plug-in
...

Open, Close, Read, Write, IOControl

OS API

mem_req
request_irq
Scheduler
...

Boot

uProcessor

HW Platform

H264

HdS: The WinCE case

HdS1 - INREG, OUTREG

HdS2
XXX_Init, XXX_Deinit, XXX_Open, XXX_Close, XXX_Read, XXX_Write, XXX_IOControl, XXX_Init

HdS3 API config, reset, encode

Application SW

GStreamer Plug-in
CORBA Plug-in
...

Open, Close, Read, Write, IOControl

OS API

mem_req
request_irq
Scheduler
...

Boot

uProcessor

HW Platform

H264

HdS: The WinCE case

HdS1 - INREG, OUTREG

HdS2
XXX_Init, XXX_Deinit, XXX_Open, XXX_Close, XXX_Read, XXX_Write, XXX_IOControl, XXX_Init

HdS3 API config, reset, encode

Application SW

GStreamer Plug-in
CORBA Plug-in
...

Open, Close, Read, Write, IOControl

OS API

mem_req
request_irq
Scheduler
...

Boot

uProcessor

HW Platform

H264

HdS: The WinCE case

HdS1 - INREG, OUTREG

HdS2
XXX_Init, XXX_Deinit, XXX_Open, XXX_Close, XXX_Read, XXX_Write, XXX_IOControl, XXX_Init

HdS3 API config, reset, encode

Application SW

GStreamer Plug-in
CORBA Plug-in
...

Open, Close, Read, Write, IOControl

OS API

mem_req
request_irq
Scheduler
...

Boot

uProcessor

HW Platform

H264

HdS: The WinCE case

HdS1 - INREG, OUTREG

HdS2
XXX_Init, XXX_Deinit, XXX_Open, XXX_Close, XXX_Read, XXX_Write, XXX_IOControl, XXX_Init

HdS3 API config, reset, encode

Application SW

GStreamer Plug-in
CORBA Plug-in
...

Open, Close, Read, Write, IOControl

OS API

mem_req
request_irq
Scheduler
...

Boot

uProcessor

HW Platform

H264

HdS: The WinCE case

HdS1 - INREG, OUTREG

HdS2
XXX_Init, XXX_Deinit, XXX_Open, XXX_Close, XXX_Read, XXX_Write, XXX_IOControl, XXX_Init

HdS3 API config, reset, encode

Application SW

GStreamer Plug-in
CORBA Plug-in
...

Open, Close, Read, Write, IOControl

OS API

mem_req
request_irq
Scheduler
...

Boot

uProcessor

HW Platform

H264

HdS: The WinCE case

HdS1 - INREG, OUTREG

HdS2
XXX_Init, XXX_Deinit, XXX_Open, XXX_Close, XXX_Read, XXX_Write, XXX_IOControl, XXX_Init

HdS3 API config, reset, encode

Application SW

GStreamer Plug-in
CORBA Plug-in
...

Open, Close, Read, Write, IOControl

OS API

mem_req
request_irq
Scheduler
...

Boot

uProcessor

HW Platform

H264

HdS: The WinCE case

HdS1 - INREG, OUTREG

HdS2
XXX_Init, XXX_Deinit, XXX_Open, XXX_Close, XXX_Read, XXX_Write, XXX_IOControl, XXX_Init

HdS3 API config, reset, encode

Application SW

GStreamer Plug-in
CORBA Plug-in
...

Open, Close, Read, Write, IOControl

OS API

mem_req
request_irq
Scheduler
...

Boot

uProcessor

HW Platform

H264

HdS: The WinCE case

HdS1 - INREG, OUTREG

HdS2
XXX_Init, XXX_Deinit, XXX_Open, XXX_Close, XXX_Read, XXX_Write, XXX_IOControl, XXX_Init

HdS3 API config, reset, encode

Application SW

GStreamer Plug-in
CORBA Plug-in
...

Open, Close, Read, Write, IOControl

OS API

mem_req
request_irq
Scheduler
...

Boot

uProcessor

HW Platform

H264

HdS: The WinCE case

HdS1 - INREG, OUTREG

HdS2
XXX_Init, XXX_Deinit, XXX_Open, XXX_Close, XXX_Read, XXX_Write, XXX_IOControl, XXX_Init

HdS3 API config, reset, encode

Application SW

GStreamer Plug-in
CORBA Plug-in
...

Open, Close, Read, Write, IOControl

OS API

mem_req
request_irq
Scheduler
...

Boot

uProcessor

HW Platform

H264

HdS: The WinCE case

HdS1 - INREG, OUTREG

HdS2
XXX_Init, XXX_Deinit, XXX_Open, XXX_Close, XXX_Read, XXX_Write, XXX_IOControl, XXX_Init

HdS3 API config, reset, encode

Application SW

GStreamer Plug-in
CORBA Plug-in
...

Open, Close, Read, Write, IOControl

OS API

mem_req
request_irq
Scheduler
...

Boot

uProcessor

HW Platform

H264

HdS: The WinCE case

HdS1 - INREG, OUTREG

HdS2
XXX_Init, XXX_Deinit, XXX_Open, XXX_Close, XXX_Read, XXX_Write, XXX_IOControl, XXX_Init

HdS3 API config, reset, encode

Application SW

GStreamer Plug-in
CORBA Plug-in
...

Open, Close, Read, Write, IOControl

OS API

mem_req
request_irq
Scheduler
...

Boot

uProcessor

HW Platform

H264

HdS: The WinCE case

HdS1 - INREG, OUTREG

HdS2
XXX_Init, XXX_Deinit, XXX_Open, XXX_Close, XXX_Read, XXX_Write, XXX_IOControl, XXX_Init

HdS3 API config, reset, encode

Application SW

GStreamer Plug-in
CORBA Plug-in
...

Open, Close, Read, Write, IOControl

OS API

mem_req
request_irq
Scheduler
...

Boot

uProcessor

HW Platform

H264

HdS: The WinCE case

HdS1 - INREG, OUTREG

HdS2
XXX_Init, XXX_Deinit, XXX_Open, XXX_Close, XXX_Read, XXX_Write, XXX_IOControl, XXX_Init

HdS3 API config, reset, encode

Application SW

GStreamer Plug-in
CORBA Plug-in
...

Open, Close, Read, Write, IOControl

OS API

mem_req
request_irq
Scheduler
...

Boot

uProcessor

HW Platform

H264

HdS: The WinCE case

HdS1 - INREG, OUTREG

HdS2
XXX_Init, XXX_Deinit, XXX_Open, XXX_Close, XXX_Read, XXX_Write, XXX_IOControl, XXX_Init

HdS3 API config, reset, encode

Application SW

GStreamer Plug-in
CORBA Plug-in
...

Open, Close, Read, Write, IOControl

OS API

mem_req
request_irq
Scheduler
...

Boot

uProcessor

HW Platform

H264

HdS: The WinCE case

HdS1 - INREG, OUTREG

HdS2
XXX_Init, XXX_Deinit, XXX_Open, XXX_Close, XXX_Read, XXX_Write, XXX_IOControl, XXX_Init

HdS3 API config, reset, encode

Application SW

GStreamer Plug-in
CORBA Plug-in
...

Open, Close, Read, Write, IOControl

OS API

mem_req
request_irq
Scheduler
...

Boot

uProcessor

HW Platform

H264
HdS Stack Conclusions

• The HdS structured approach
  – Identification of the different components
  – Maximization of the reusability

• OS
  – Linux & WinCE are very similar
  – Opportunities for HdS standardization
  – Facilitates the development of tools
    • Simulation,
    • Synthesis,
    • Verification, ...
• IP-XACT
  – Current IP-XACT is the “level 0” layer
    • Only describes the HW platform
  – SoftSoC opens IP-XACT to “n” levels of abstraction
  – Easy identification of IP services at different levels
  – Higher level → less knowledge of concrete services
Plan

1. HW/SW and SW/SW basic dependencies
   1. SW Taxonomies
   2. Case studies
   3. Basic SW Dependencies Conclusions

2. Layered Hardware-dependant-Software
   1. Introduction to HdS Stack
   2. Linux
   3. WinCE
   4. HdS Stack Conclusion

3. IP Xact & HdS
Extensions in IP-XACT for HDS

• New schema
  – Structured and standardized electronic documentation dedicated to HDS structure (layers)
  – in relation with HW platform in IP-XACT schema (level 0)

• Description of SW blocks and structure assembly
  – Dependencies with HW processors, compilers, OS, etc.
  – Interfaces: SW-SW, inter layers, SW-HW + definition of services
  – Specificities for HDS1, 2, 3, middleware?
  – Views for several implementations

• Extensions of existing IP-XACT
  – Reference to several drivers
  – Description of performances (power, timing)
  – Provided services toward the SW layers (list of standard services?)
Solutions and tooling for HDS

Design environment (base on IP-XACT)
• SW architecture assembly (hierarchical) in a reuse and multi site context
• Manage though a comon cockpit the heterogenity of tools, methods and languages
• Automate the HW/SW codesign and mapping

Tools and engines
• Import/export to/from IP-XACT
  – UML description of HDL layers
  – Doc generation
• Design Verification
  – Required services available on HW platform?
  – Verify the access of HW registers by SW (though buses, bridges...)
• Etc...