EEM870 Embedded System and Experiment
Lecture 5: Platform Architect Overview and Labs

Wen-Yen Lin, Ph.D.
Department of Electrical Engineering
Chang Gung University
Email: wylin@mail.cgu.edu.tw
March 2013
Outlines

- Platform Architect Overview
- System Level Design
- Analysis Functions
- Labs
Platform Architect Overview

- ESL Concept
- The PA Family
- Tool Components
- Usage
- Benefits
System Design Process

Algorithm Design & Analysis → Matlab

Dataflow Analysis → Simulink, SPW, ADS, etc.

Architecture Design → System Verification

Constraints

HW Implementation → Constraints → SW/FW Implementation

System Integration

*Dr. Alen Su’s lecture
Typical Project Schedule

Time to Market

System Design
Hardware Design
Prototype Build
Hardware Debug
Software Design
Software Coding
Software Debug
Project Complete

Embedded System and Experiment, 101/2, EE/CGU, W.Y. Lin
HW/SW Co-design

The modeling, synthesis (or partitioning), and simulation of systems that include interacting HW/SW modes

- **Concurrent** HW & SW development
- Interaction/interface between HW & SW
- HW/SW co-verification

**Concurrent:** hardware and software developed at the same time on *parallel* paths
HW/SW Co-design Benefits

- Integrate earlier
- Debug SW sooner
- Iterate changes faster
- Reduce project risk

- Early architecture closure reduces risk by 80%
- Start software development 6 months earlier
The CoWare product family is a suite of tools for system architecture and design.

<table>
<thead>
<tr>
<th>Main feature</th>
<th>Other features</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Virtual Platform Designer</td>
<td>• Processor Designer</td>
</tr>
<tr>
<td>• Platform Architect</td>
<td>• Signal Processing Designer</td>
</tr>
<tr>
<td>• Model Designer</td>
<td>• Model Library</td>
</tr>
</tbody>
</table>
## Similar/Related Tools

<table>
<thead>
<tr>
<th>Vendors</th>
<th>Product</th>
<th>Function/Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM</td>
<td>Core Generator</td>
<td>LISA</td>
</tr>
<tr>
<td></td>
<td>System Generator</td>
<td>LISA, C, C++</td>
</tr>
<tr>
<td>Carbon Design Systems</td>
<td>SOC Designer</td>
<td>SystemC</td>
</tr>
<tr>
<td></td>
<td>Model Studio</td>
<td>Verilog-&gt;SystemC</td>
</tr>
<tr>
<td>Tenison Design Automation</td>
<td>VTOC</td>
<td>Verilog-&gt;SystemC</td>
</tr>
<tr>
<td>Mentor Graphics</td>
<td>ModelSim SE</td>
<td>Verilog, VHDL, SystemVerilog, SystemC</td>
</tr>
<tr>
<td></td>
<td>Vista Architect</td>
<td>SystemC/TLM 2.0</td>
</tr>
<tr>
<td></td>
<td>Catapult C Synthesis</td>
<td>High Level Synthesis</td>
</tr>
<tr>
<td>Chip Vision</td>
<td>PowerOpt</td>
<td>Power-Optimizing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>High Level Synthesis</td>
</tr>
</tbody>
</table>
The Platform Architect consists of

- **System Verifier**
  (SystemC IDE, SystemC Explorer, scsh)

- **System Designer**
  (Analysis API, sdviewer)

- **Advanced System Designer**
  (Platform Creator, pcsh)

**Simulation and Debugging**

**Analysis**

**Platform Assembly and Configuration**
The Platform Architect consists of:

Software Debuggers
(Virtual Platform Analyzer, xarmsgd, ...)

Other Capabilities

• SystemC/HDL co-simulation
• ARM ELF GCC
CoWare IP Library consists of

| *ARM 7 Family  | Processor | MIPS32 Family | Processor |
| *ARM 9 Family  | Processor | IBM PPC405/440 | Processor |
| ARM 10 Family  | Processor | IBM PPC750     | Processor |
| *ARM 11 Family | Processor | IBM CoreConnect| Bus       |
| ARM 11 MPCore  | Processor |              |           |
| ARM Cortex R4/A8| Processor |              |           |
| *AXI           | Bus       | SONICS SMX    | Bus       |
| *AMBA          | Bus       | SONICS DDR    | Memory    |
| *AMBA Bridges  | Bus       | SONICS MemMax | Controller |
| *AHB PrimeCells| Peripherals|              |           |
| AXI PrimeCells | Peripherals| LSI ZSP400    | DSP       |
| *ARM ELF GCC   | Software  | LSI ZSP500    | DSP       |
| *HW/SW Scenario| PA Option | Tensilica Diamond | Processor |
| *Architecture View | PA Option | Tensilica Xtensa | Processor |
| Framework      |           |              |           |
| *Generic Library | PA Default | CEVA Teaklite | DSP       |
System Verifier provides the simulation and debugging environment

Integrated Development Environment

Command Line Interface

% scsh
scsh> load names.fof
scsh> sbreak ei
scsh> run

Stack and control

Design Browser

Source Code

Outline

Input and Output

Embedded System and Experiment, 101/2, EE/CGU, W.Y. Lin 13
System Designer adds analysis capabilities

```
% scsh
scsh> load names.fof
scsh> analysis_config_gui on
scsh> run
```

Hardware

Software

Bus

Memory
Advanced System Designer is for architecture optimization and system integration

- **Import of SystemC and HDL Blocks**
- **Block Diagram Editor**
- **Scripting Interface**
- **HW/SW Partitioning and Interface Synthesis**
- **IP Reuse based on XML metadata**
- **Export of SystemC and HDL Blocks (Bus Interface)**
ASD components consist of Platform Creator, XML libraries, and the transactional bus simulator.
Use the Platform Architect to build platform-based designs

User’s tasks:
- Create system design
- Configure memory map
- Set parameters
- Perform error checking
- Export Hardware

User:
- Builds simulation
- Configures analysis

Bus Library Handler automatically generates:
- address decoders
- muxes
- arbiters
- bridges

Platform Creator

User models

Platform Model .cpp .h .maf

SV/SD

Generate Analysis Views

Postproc

db

Simulate

Embedded System and Experiment, 101/2, EE/CGU, W.Y. Lin
The Platform Architect provides:

**Simulation**
- Fully compliant SystemC simulator
- Optimizes design for high speed execution
- TCL scripting language for simulation control
- SystemC aware debugger

**TLM Modeling**
- High speed, cycle accurate bus architecture models

**Analysis**
- Waveforms & event tracing
- Performance statistics
- Profiling & coverage

**HDL Co-Simulation**

**System Level Design**
- Easy assembly & configuration of bus topologies
- Block diagram editor
- Platform development
- HW/SW partitioning and Interface Synthesis
- Export of SystemC and HDL blocks
System Level Design

A. Overview
- What is Advanced System Designer?
- Working with Platform Creator
- Components and Operations

B. Working With Libraries
- Library Descriptions
- Importing SystemC
- Importing HDL
- Working with Multiple Encapsulations
- Portability

C. Platform–based Design
- System Description
- Steps
Advanced System Designer is an assembly tool for creating and manipulating systems containing hardware and software components.
Menus and the toolbar provide access to commonly used commands:

- **File**
  - New Ctrl+N
  - Open... Ctrl+O
  - Open Recent
  - Save Ctrl+S
  - Save As... Ctrl+Shift+S
  - Save As Tcl Script...
  - Save System Library As...
  - Print... Ctrl+P
  - Exit Ctrl+Q

- **Edit**
  - Undo
  - Redo

- **Library**
  - Open Library File
    - Open From Search Path...
    - Close Ctrl+Alt+C
    - Library Set

- **Diagram**
  - Open Library (read only)
  - Update System Library
  - Import SystemC Modules
  - Reload SystemC Modules
  - Import HDL...
  - Import SPIRIT...
  - Import Managed Make Project...

- **Check**
  - Rename
  - Update System Library from Selection Ctrl+Alt+U

- **Export**
  - Import SystemC Modules...

- **Tools**
  - Show Messages

- **View**
  - Build Run Debug

- **Simulation**

- **Plugins**

- **Help**

**Drawing**
- Selection
- Connect
- Text Edit
- Zoom
- Pan

**Add**
- Initiator Port
- Target port

**Preferences**
- Diagram Navigator
A closer look at the Library Drawer shows that it is divided into two parts

• System Library
  — Your system workspace, the system under construction
  — Displays the system blocks, nodes, and protocols of your design

• Currently Open IP Libraries
  — AMBA Bus Library
    — Nodes
    — Bridges
    — Protocols
  — Processor Library
System Library
IP Library
This is the empty working area
Watch messages in the Console

Run/debug
Software debugger
CA model: xarmsgd/rvdebug
IA model: vpa

Hardware debugger
SystemC IDE
SystemC Explorer
Being aware of the components of Platform Creator will help you understand the commands and usage of the tool.
The entities you assemble with Platform Creator are System Blocks

- **Platform Creator**
  - Memory_AHB
  - Untimed Transaction Level Model (UT, TLM, CA, RTC, VHDL, Verilog ...)
  - SystemC/ HDL source
- **System Block**
  - Memory_AHB
  - Untimed
  - Transaction Level Model
  - Cycle Accurate
- **Identifies pin to port mapping, protocol assignment, and source location**
- **Supports multiple views**
  - (UT, TLM, CA, RTC, VHDL, Verilog ...)
- **Properties of Encapsulation Ports**
- **Mapping of System Block Ports to Encapsulation Ports**
- **All System Block Ports and Protocols**

(Display caused by selecting an editable system block in the Library Drawer)
Platform-based design starts with architecting a processing platform for a given vertical application space.

- User models
- Block Library
- Processor Library
- Bus Library

Platform Creator

- User’s tasks:
  - Create system design
  - Configure memory map
  - Set parameters
  - Perform error checking
  - Export Hardware

User models

Bus Generator

- Bus generator automatically generates:
  - address decoders
  - muxes
  - arbiters
  - bridges

Platform Model

- .cpp
- .h
- .maf

SV/SD

Simulate

IP Library

Generate Analysis Views

Post proc

Db

User:

- Builds simulation
- Configures analysis
Top-down design starts with functional validation of the system spec.

**Platform Creator**

**User’s tasks:**
- Load platform or PSP
- Import UT system spec
- Block partitioning
- HW/SW scenario selection
- Configure memory map
- Set parameters
- Perform error checking
- Export System

**User:**
- Compiles SW system
- Builds simulation
- Configures analysis

**Part of the System Spec may run as software on the platform while the rest is implemented as hardware.**

**The Scenario Library is used to build communication between HW and SW portions of the System Spec:**
- memory-mapped scenarios
- interrupt-driven scenarios
- HW/SW drivers
- IPEs

**Bus Library**

**Protocol Library**

**Scenario Library**

**SW Systems**
- .cpp .h

**IP Library**

**Bus Generator**

**Compile**

**Simulate**

**SV/SD**

**db**

**Generate Analysis Views**

**Post proc**
The directory organization for a typical installation looks like this:

```
<install_dir>/
  |__ CoWare
  |__ V<release>/
  |     |__ Documentation
  |     |__ docs/pdf/
  |     |__ training/svsd/index.htm
  |     |__ CovergenSC documentation
  |     |__ Self-study training
  |__ IP /
  |    |__ ARM926EJS_AHB_PSP /
  |    |__ AMBA_BL /
  |    |__ Processor Support Packages
  |    |__ Bus Library
  |__ AMBA_BL/documentation/
  |__ PAMD
  |__ linux /
     |__ any /
     |__ license
     |__ tests
     |__ examples
     |__ IP @
     |    |__ plugins /
     |    |    |__ (for HDL simulators)
     |    |__ tools /
     |    |__ common /
     |    |__ pc /
     |    |__ sd /
     |__ PlatformArchitect.csh

$COWAREHOME points here
```
To use Advanced System Designer, you only need to source the *setup* file in the CoWare installation directory

```
source <install_dir>/CoWare/V<release>/PAMD/linux/PlatformArchitect.csh
```

To start Platform Creator:

**GUI version**

```
% pct
```

**Command Line version**

```
% pcsh
```

*Add to your .cshrc file.*

*This will set a number of environment variables such as COWAREHOME and add paths to your executable and library path variables.*
Using Analysis

- Introduction
- Use Model – four phases
- Post Processing
System development consists of putting together a system from a collection of hardware and software components…then iteratively measuring and modifying the system for optimum performance

**Software**
- Startup code
- Device drivers
- RTOS
- Application code

**Hardware**
- Processors
- Bus
- Memory
- Peripherals
- Interfaces

---

**Simulation**

**Analysis**
Areas of Interest for Analysis are:

- Memory
- Hardware
- Bus
- Software

Diagram showing connections between:
- ARM Core
- DMA Ctrl
- Interrupt Ctrl
- Clock Gen.
- Reset Ctrl
- APB
- AHB
- Custom Peripheral
- Input Device

Signals include:
- Instruction
- Data
- IRQ
- FIQ
- Slave
- Master 1
- Master 2
- DMA
- Int
- Rom
- RAM
CoWare’s processor support packages (PSP) come pre-instrumented for software, memory access, and counter analysis. Only busmasters need to be instrumented.
The CoWare bus libraries are pre-instrumented with views to analyze contention, latencies, and transactions automatically generated by the BusCompiler.
The analysis usage model proceeds through four phases:

1. **Pre-sim phase**: Analysis switched on. User starts simulation run.
2. **Config phase**: At ACC user determines which views are enabled (via CLI commands or GUI).
3. **Sim run phase**: Simulation runs and data is collected in CoWare Dynamic Database.
4. **Post process phase**: Post processing utility sdviewer reads database and displays data.

**ACC** = analysis-configuration-callback breakpoint
1. The pre-simulation phase consists of running the simulation up to the analysis-configuration-callback breakpoint.
2. At the analysis-configuration-callback breakpoint, the analysis view objects are available for configuration.

Notice the sequence does not include enabling analysis until this point.
The File Setup dialog is for defining analysis related files.

- Set database name and location
- Set memory architecture file name and location
- Set Symbol file name and location

These settings are saved in a file called “.cwrSimParams” in the current working directory.
Enable views with the configuration dialog

- Enable analysis
- Double click toggles enable/disable
- Click to expand or collapse selected node
- Select view by name
- Expand or collapse tree
- Selected view
- Enable, disable, or toggle selected view
- Full path: /HARDWARE/_ARM926/FuncGroup/FuncGroup/FuncLoad
Set configuration values in the View Settings section

View settings can be modified in a separate window.

Double clicking on the view setting to modify.

Press “ok” button when done.
3. Continue the simulation to collect the data for the enabled views.

The analysis settings are now defined and the simulation is stopped at the analysis-configuration-callback breakpoint.

The next step is to “continue.”

When `sc_main` returns due to `exit(1)`, `sc_stop()`, or `ctrlZ`, the database is automatically closed.
4. The post processing utility, sdviewer, reads and displays the analysis data.
The `sdviewer` interface provides access to all of the views that were selected in the configuration.

The “Views tree” contains all views that were selected in Configuration:

- ‘checked’ means will be displayed in case one presses the “Display” button.
- ‘open’ means contains data
- ‘grey’ means contains no data

Note: in case more than one view is selected, they will be grouped.
Display views by selecting in the Views tree window

Click to select a view

display
The view window provides basic control features such as zoom, fit, and scroll.

To zoom into a range:
left-click and drag

Mouse shortcuts:
Zoom Fit Pop-up menu

Save View
Go Last
Go Next
Go set …
Zoom Fit
Zoom In
Zoom Out

Left Window:
Text & Spread sheets

Right Window:
Graphic Displays

Notes:
- Go set …
- Mouse shortcuts:
  - Zoom Fit
  - Pop-up menu
Selecting more than one view to display will cause the views to be grouped.
Later, you can view the groupings you created by name.
Labs