TEC-ED Presentation Days

ESL-day

Luca Fossati, Microelectronics Section (TEC-EDM)
ESA/ESTEC
19/09/2011
Program

- 10:00 – **Introduction** *(Luca Fossati, ESA/ESTEC)*
- 10:20 - **Hardware Modeling using Virtual Platforms** *(Thomas Schuster, IDA)*
- 11:05 – **LEON2/3 SystemC model** *(Luca Fossati, ESA/ESTEC)*
- 11:25 – **SpaceWire SystemC model** *(Nikos Mouratidis, Qualtek)*
- 11:45 – **SocROCKET Virtual Platform: Demo** *(Thomas Schuster, IDA)*
- 12:00 – **Socket: SoC toolKit for critical Embedded systems** *(Vincent Leftz, Astrium)*
- 13:30 – **Overview of the ASSERT and the TASTE toolchains** *(Maxime Perrotin, Julien Delange, ESA/ESTEC)*
- 14:00 - **The TASTE project: results and lessons learnt** *(Marc Pollina, M3 Systems)*
- 14:20 – **Extending TASTE through integration with SpaceStudio** *(Guy Bois, SpaceCodesign - Marc Pollina, M3 Systems)*
- 15:10 – **Hardware Design using High-Level synthesis** *(Laurent Hili, ESA/ESTEC)*
- 15:40 – **Conclusion and Open Discussion**
Electronic System Level design at ESA

*a hardware perspective*

Luca Fossati
ESA/ESTEC
19/09/2011
Outline

- ESL Design: Overview
- ESL at ESA: System Modeling
- ESL at ESA: Hardware Platform Modeling
- ESL at ESA: HW/SW co-design
- Conclusion
Definition [1]:

The utilization of **appropriate abstractions** in order to increase comprehension about a system, and to enhance the probability of a successful implementation of functionality in a cost-effective manner.

- Set of complementary methodologies that enable embedded system design, verification, and debugging through the hardware and software implementation of custom SoC, system-on-FPGA, system-on-board, and entire multi-board systems.
- Main goal is to *improve the productivity and quality of the complete chip deliverable, including both hardware and software...*
  - ... but also reduce time-to-market, cost, improve re-usability

[1]: ESL Design and Verification: A Prescription for Electronic System Level Methodology, G. Martin, B. Bailey, A. Piziali
Electronic System-Level (ESL)

- **ESL enables addressing the complexity of most Systems-on-Chip**, with requirements of flexibility, short time-to-market, tight interaction between hardware and software, low power consumption, etc.
  - Such complex designs are starting to emerge also in the space market (e.g. telecom, multi-core processors, VPUs)

- ESL helps moving forward in a project development, supporting the decision making process:
  - 80% of design decisions are taken in a project's first 20% of lifetime, when making the right choices will make the difference between project’s success or failure.
Overview

Traditional System Design Flow ...

Hardware:
- Functional specification
- Architectural Design
- Manual Coding
- HDL
- Tuning optimization

Software:
- Manual partitioning
- OS Choice
- Custom SW
- Functional specification
- Functional specification
- Execution on target HW
- Tuning optimization

Functionality
C/C++/Matlab
... and Related Problems

- Relies on a more extensive *designers’ experience* than ESL
  - Less automated steps
- Collection of unlinked tools
  - The implementation proceeds with “informal”, mostly *not automated techniques*
  - Manual conversion from concept to design to implementation
- Difficulties in crossing the technical domain boundaries
  - E.g. HW designers having little knowledge of SW
- No help in determining the optimal system configuration
  - There is *no global system model*
  - Choosing an architecture platform is more an art than a science
- System evaluation performed at the end of the flow(s)
  - *Problems in making-up for design flaws*
- *Limited re-usability* for new designs
  - Due to the lack of automated steps
Overview

Traditional System Design Flow

Hardware:

- Functional specification
  - Architectural Design
    - Manual Coding
      - HDL
        - Tuning optimization

Software:

- Functional specification
  - Manual partitioning
  - Functional specification
    - Hw datasheet
      - OS Choice
        - Custom SW
          - Execution on target HW
            - Tuning optimization
Overview

**ESL Design Flow**

Functionality refinement

- **Functional/Algorithmic Specification**
  - UML SySML
  - C/C++ AADL
  - SystemC
  - Matlab
  - Simulink

- **Base Platform**
  - Custom HW models

- **Software**
  - SW Synthesis
    - BSP creation
    - OS Choice

- **Hardware Platform**
  - Instantiation
    - after tuning

- **HDL**
  - Custom HW
    - Behavioural Synthesis

- **Implementation**
  - European Space Agency

**Corrections Improvements...**

- **Platform tuning /optimization**
  - Hw/Sw co-design

**Model Refinement**

**Initial Hw/Sw mapping**

ESL at ESA | Luca Fossati | ESA/ESTEC | 19/09/2011 | Slide 10

ESAS UNCLASSIFIED – For Official Use
Overview

ESL - Functional/Algorithmic Specification
Functionality refinement

**Functional/Algorithmic Specification**

UML SySML
C/C++ AADL
SystemC
Matlab
Simulink

Hw/Sw initial mapping

Corrections
Improvements

Base Platform
Custom HW models

Model Refinement

Platform tuning /optimization
Hw/Sw co-design

HDL - Custom HW

Behavioural Synthesis

SW Synthesis
BSP creation
OS Choice

Hardware Platform
Instantiation after tuning

Software

Implementation
ESL - *Functional/Algorithmic Specification*

- Specifies the **overall system functionality**
  - Considers *both HW and SW*: at this stage the partitioning hasn’t been performed yet
- Verifies that the **specification satisfies the requirements**
- Based on the use of a **high-level description**
  - C, C++, Matlab, UML, ...
- Shall **allow automatic code generation** / transition to more accurate descriptions
  - The whole ESL flow is based on the use of tools for (semi)automatic transitions between design steps
Overview

ESL - HW/SW co-design

Functionality refinement

Functional/Algorithmic Specification

UML SySML
C/C++ AADL
SystemC
Matlab
Simulink

Corrections
Improvements

... Hw/Sw initial mapping

Base Platform

Custom HW models

Software

Corrections
Improvements

Platform tuning /optimization

Hw/Sw co-design

HDL - Custom HW

Behavioural Synthesis

Implementation

Software

SW Synthesis

BSP creation

OS Choice

Hardware Platform

Instantiation

after tuning

HDL

Custom HW
Overview

ESL – HW/SW co-design

- Main part of the ESL flow
  - In the sense that it *shapes the overall system HW/SW structure*
- It is **not a single activity**
  - Composed of many iterations towards more accurate representations
- Often centered around the concept of *Platform-Based Design*
  - Base hardware platform, tuned and optimized during the flow
  - Hardware blocks created for the specific custom functionality
  - Optimal trade-off between design complexity and performance
- *Virtual Platforms* are often used
  - Simulatable models of hardware/software systems
  - Allow performance estimation and system refinement
ESL – Behavioural Synthesis

Functionality refinement

Overview

- **Functional/Algorithmic Specification**
  - UML SySML
  - C/C++ AADL
  - SystemC
  - Matlab
  - Simulink

- **Corrections Improvements**

- **Hw/Sw initial mapping**

- **Base Platform**
  - Custom HW models

- **Software**
  - **SW Synthesis**
    - BSP creation
    - OS Choice

- **Hardware Platform**
  - **Instantiation**
    - after tuning

- **HDL**
  - Custom HW
  - Behavioural Synthesis

- **Platform tuning /optimization**
  - Hw/Sw co-design

- **Implementation**

- **European Space Agency**
ESL – Behavioural Synthesis

- **Automated design process** that interprets an algorithmic description of a desired behavior and creates an RTL description that implements that behavior
  - Used for part of the system design, the rest is implemented by the chosen base platform
  - The process still requires user interaction to create optimal designs

- Shifts the focus from the implementation details to the high-level description of the desired functionality
  - We are interested in the *what* not in the *how*
Outline

- ESL Design: Overview
- ESL at ESA: System Modeling
- ESL at ESA: Hardware Platform Modeling
- ESL at ESA: HW/SW co-design
- Conclusion
System Modeling

- Based on the **ASSERT methodology** and the **TASTE toolset**

**Capture of the system properties**

1. **Modeling phase**

**Capture of the hardware architecture**

2. **Model transformation**

3. **Feasibility analysis**

So far mainly focused on the software portion of the system

- Based on **AADL** and **ASN.1**

- **System real-time architecture**

- **Run-time environment**

4. **Automatic code generation**

**Complete system**
System Modeling

1. Generate “application skeletons” in Simulink, SDL, C, and Ada

2. Generate a software real-time architecture (in AADL)

3. Generate glue code to put everything together on a real-time operating system
Outline

- ESL Design: Overview
- ESL at ESA: System Modeling
- ESL at ESA: Hardware Platform Modeling
- ESL at ESA: HW/SW co-design
- Conclusion
Hardware Modeling

- Models the *Base Hardware Platform* as required by the Platform-based Design methodology
- Can be performed at *various abstraction levels*:
Hardware Modeling

- Models the base *Hardware Platform* as required by the Platform-based Design methodology
- Can be performed at various abstraction levels

Current focus of the activities is *Instruction-* and *Cycle-* accurate models
  - Good tradeoff among flexibility, simulation performance, and performance (speed, power consumption, area ...) estimation accuracy

Models are based on SystemC and TLM libraries/methodologies
  - Built on the C++ language
  - Allow expressing concurrency, synchronization and timing

Not the best choice for Hw/Sw co-design
  - A preliminary partitioning has already been made
**SystemC**

- IEEE standard, implemented as *a set of C++ classes*
- Standard C++ compiler can be used to generate an executable specification
- Addresses various levels of abstraction down to RTL

**Transaction Level Modeling (TLM)**

- Well-established methodology for modeling complex systems
- Separates *communication* from *computation*
- Modules communicate with the rest of the world by performing transactions
  - Instead of modeling every single transferred bit, *data structures are exchanged*
Virtual Platform: Benefits

A Virtual Platform is a simulatable (software) model that can fully mirror the functionality of a target SoC or board.

- Virtual Platforms are not subject to the same constraints as physical implementations:
  - They are “easier” to create, modify, and observe.

- Enhanced modifiability of the models and of its parameters

- Observability, controllability and not intrusive monitoring
  - Performance measuring and debugging
Hardware Modeling: Activities

- SRAM
- SDRAM
- PROM
- Local I/O
- Mem-CTRL
- IRQMP-CTRL
- SpW-B Codec
- SoCWire Network
- GPTimer
- LEON3
- MMU/Cache
- UART
- CAN
- Bridge

Available
Under development
Not planned yet

Interconnection (based on the AMBA® AHB™/APB™ protocol)
ESA IP Cores

- [http://www.esa.int/TEC/Microelectronics/SEMVWLV74TE_0.html](http://www.esa.int/TEC/Microelectronics/SEMVWLV74TE_0.html)
- **SystemC and HDL** IP Cores to be used for designing, respectively the virtual platform and the base hardware platform
Outline

- ESL Design: Overview
- ESL at ESA: System Modeling
- ESL at ESA: Hardware Platform Modeling
- ESL at ESA: HW/SW co-design
- Conclusion
Meeting **system-level objectives** by exploiting the synergism of hardware and software through their **concurrent design**.

* Larger and larger portions of current systems are software:
  - Becoming the critical point in the schedule
  - Tight dependence between hardware and software
    - One of the peculiarities of embedded systems
  - **Software and Hardware designs have to proceed concurrently**
    - Functionalities can be moved among the two domains
    - Hardware has to be optimized for the execution of the selected class of software applications and, in turn, software should be customized to efficiently use hardware capabilities
    - Overall system can be evaluated only when HW and SW interact: they should not be separately addressed
Current Activities @ ESA

Functionality refinement

UML SySML
C/C++ AADL
SystemC
Matlab
Simulink

Functional/Algorithmic Specification

 ASSERT & TASTE

UML SySML
C/C++ AADL
SystemC
Matlab
Simulink

 ASSERT & TASTE

Base Platform
Custom HW
models

Model Refinement

Base Platform
Custom HW
models

 ASSERT & TASTE, SpaceStudio, Catapult C, HDL Coder...

Platform tuning /optimization
Hw/Sw co-design

 ASSERT & TASTE, Space Studio, ...

ASSERT & TASTE

Base Platform
Custom HW
models

 ASSERT & TASTE

So ft ware

 ASSERT & TASTE, SpaceStudio, ...

HDL
IP Cores

 ASSERT & TASTE

HDL

 ASSERT & TASTE

Behavioral Synthesis

Instantiation
after tuning

Hardware Platform

 ASSERT & TASTE

HDL

 ASSERT & TASTE

Implementation
HDL
- Custom HW

SocROCKET
 & SystemC Models

 ASSERT & TASTE

Software

 ASSERT & TASTE

So ft ware

 ASSERT & TASTE

Corrections
Improvements

 ASSERT & TASTE

 ASSERT & TASTE

 ASSERT & TASTE
Outline

✓ ESL Design: Overview
✓ ESL at ESA: System Modeling
✓ ESL at ESA: Hardware Platform Modeling
✓ ESL at ESA: HW/SW co-design
✓ Conclusion
Summary of the Day

- Hardware Platform modeling
  - SystemC and TLM hardware models
  - Virtual Platform infrastructure to manage simulation and the models

- SoCKET: an Integrated ESL Solution
  - SoC toolKit for critical Embedded sysTems
  - French project whose main goal is “to define a seamless design flow which integrates qualification and certification, from the system level to integrated circuits and to software”

- Early System Modeling - ASSERT and TASTE:
  - From a system-level description (with AADL and ASN.1) towards the implementation of the Sw system portion

- Behavioural Synthesis
  - Early experience with commercial tools

- HW/SW co-design
  - Integration of early performance estimation, platform-based design, hardware modeling, behavioural synthesis to guide the HW/SW system partitioning.