CONTENTS

1. HW/SW co-design study
2. Definition of the proposed HW/SW co-design approach
3. HW/SW methodology
4. Implementation of the tool support
5. Case study
6. Conclusions and future lines of work
HW/SW Co-design

HW/SW CO-DESIGN STUDY
STUDY OBJECTIVES (I)

- **Main Objective**: to address the HW/SW co-design phase to integrate this engineering task as part of the ASSERT process, compatible with the existing ASSERT approach, process and tools.

- HW/SW co-design exploits trade-offs between hardware and software in a system through their concurrent design:
  - Concurrent development of HW and SW, and
  - Integrated design.

- Implementation of a process compatible with the AADL methodology.

- The study approach will be validated on the proposed industrial case study building the models in full compliance with the ASSERT process.

**Note**: The ASSERT project was completed in 2007, since then, the ASSERT framework has evolved to TASTE (The ASSERT Set of Tools for Engineering).
STUDY OBJECTIVES (II)

HW/SW Codesign process

HW Engineering
TEAM PRESENTATION

- GMV, prime contractor
  - Expertise on on-board SW development and Model-based Engineering
- Polytechnic University of Madrid (UPM)
  - Expertise on ASSERT process and Ravenscar Computational Model
- University of Cantabria (UC)
  - Expertise on system simulation on System C and AADL, and HW/SW co-design
- GMV Sistemas
  - Expertise on design, development, manufacturing and commercialisation of HW for embedded systems
STUDY MAJOR TASKS

- **Task 1** - Definition of the proposed HW/SW co-design approach
  - ASSERT process assessment.
  - HW/SW co-design survey.
  - Preliminary design of the HW/SW co-design process.

- **Task 2** - Implementation of the tool support
  - AADS-T/SCoPE: AADS-T (AADS Taste) generation of SystemC model from the AADL specification, SCoPE tool for simulation and performance analysis of the AADL model.
  - ASSERT Model Transformation (AMT) tool supported by AADL ASSERT-based extension.

- **Task 3** - Demonstration on a case study
  - Demonstration of methodology and toolset on a case study:
    - FPGA design board: Payload image processing system.
    - LEON design board: On-Board SW (OBSW).
STUDY MILESTONES

- **Kick Off (KO)**  Nov/2009
- **M1 - Review of Task1**  Jun/2010
  - Review of the HW/SW co-design definition.
  - Review of main agreements for the toolset preparation.
  - Selection for the proposed approach.
- **M2 – Review of Task2**  Dec/2010
  - Review of the implementation of the toolset.
  - Review of main agreements for the framework demonstrator.
- **M3 - Final Review**  Jul/2011
  - Review of the demonstrator.
  - Review of main results of the project.
HW/SW Co-design

DEFINITION OF THE PROPOSED HW/SW CO-DESIGN APPROACH
DEFINITION OF THE PROPOSED HW/SW CO-DESIGN APPROACH

Activities carried out for the definition of HW/SW co-design approach:

1. ASSERT Process assessment
   Focusing on the achievement of maximum compatibility with the ASSERT methods and tools.

2. HW/SW Co-design survey
   Methodologies and associated tools for system specification, performance estimation, HW/SW partitioning and HW synthesis.

3. Preliminary Design of the HW/SW Co-design Process
   Supported by intense assessment on the findings of the previous tasks.
Investigation of current technical solutions: AADL track
- Study of the AADL implementation track: concepts and tools
- Study of the different PIM views: Data, Functional, Interface and Deployment
- Study of the development flow

Investigation of current ASSERT toolset constraints
- Data, Functional, Interface and Deployment editors
- Relationships between different tools
- Constraints

Investigation of model transformations
- Different modelling levels: System, Software (ASSERT) and Hardware levels
MDA uses models to abstract away from technological choices and to facilitate the port to the specific target.

- Platform Independent Model (PIM) that specifies the solution in a way that does not depend on any particular implementation;
- PIM is transformed into a Platform Specific Model (PSM), by feeding the transformation with information about the chosen execution platform and its characterizing parameters;
- PSM is automatically generated and users are not allowed to directly modify it;
- Analyses on the PSM to validate its feasibility and adequacy against a range of criteria;
- When satisfied, the final stage of transformation from PSM to code.
ASSERT PROCESS ASSESSMENT – AADL TRACK (3/4)

AADL track and tool chain:

- General text editor: Data view editor
  - asncc compiler

- Labassert tool: Functional view editor
  - Development editor

- Labassert tool: Interface view editor

- Labassert tool: Deployment view editor
HW/SW CO-DESIGN SURVEY (1/2)

- Analysis of design methodologies and tools for Electronic System-Level (ESL):
  - System specification, performance estimation, HW/SW partitioning and HW synthesis.
  - Both commercial and academic tools are covered.
  - A detailed list of advantages, weak points and use case scenarios on languages and tools for HW/SW co-design.
Due to the high use of C (and C++), a large number of design languages have been developed based on these languages, such as SystemC, SpecC and ImpulseC among others.

SystemC (IEEE std. 1666) has become predominant for ESL specification and design.

Generating SystemC from system specification and modeling languages is a very active research area today.

Platform Virtualization is becoming a very widely accepted as a SW simulation technology.

Native simulation is a promising SW simulation technology combining both accuracy and speed.

SCoPE provides the technology to perform HW/SW co-simulation and performance estimation of MPSoC with NoC.

Lack of tools for partitioning and synthesis of HW and SW.
PRELIMINARY DESIGN OF THE HW/SW CO-DESIGN PROCESS (1/2)

- Definition of a **HW/SW methodology** that ensures **consistency** between SW and HW design:
  - Definition of an unbiased, unified **system specification**, compatible with the ASSERT component model
  - Implementation of a **performance estimation analysis** based on fast SW simulation, compliant with the ASSERT computational model
  - Definition of a **common integration modelling substrate** that would allow the propagation of changes between HW and SW development paths

- Building a prototype version of the **tool support** compatible with:
  - The interfaces with the ASSERT toolset
    - OCARINA properties, structure and semantics of ASSERT views
  - ASSERT Virtual Machine communication mechanisms and protocols
    - OCARINA code generation, polyORB-HI communication protocols
PRELIMINARY DESIGN OF THE HW/SW CO-DESIGN PROCESS (2/2)

- **System Model:**
  - **Logic View**
    - Interface, Functional and Data views: ASSERT/TASTE views.
  - **Platform View**
    - Description of HW elements available to implement the system (i.e., devices, buses, memories and processing resources).
- **Partition View:**
  - Processing nodes of the platform view are annotated specifying which components are mapped to HW and SW.
HW/SW Co-design

HW/SW METHODOLOGY
Co-specification: development of the system model:
- Modelling of system functions
- Definition of the system platform architecture

Co-design: allocation of function onto HW resources and estimation of performance figures

Co-synthesis: generation of the communication interfaces between the HW and SW systems, and the synthesis of the HW and SW systems
System Specification: modelling of structural and behavioural characteristics of the system. Two model viewpoints:

- **System Logic View**: the data model, the system functions, and system components and their interconnection
  - Inherited from the ASSERT Data, Functional and Interface views
  - Unbiased either to HW or SW: Platform Independent Model (PIM)
  - Modelled with the ASSERT toolset
  - Imported into the AADL project by the ASSERT Model Transformer (AMT)

- **System Platform View**: description of HW elements and the system platform architecture.
  - Describes the system platform architecture: Platform Description Model (PDM)
  - Provides implementation details of HW elements: read time, word size, implementation technology
  - Modelled with the OSATE editor
CO-DESIGN (1/2)

- HW/SW partitioning and performance analysis:
  - **HW/SW partitioning**: allocation of system components to HW computational resources
    - Model to model transformation: **System Partitioned View**
  - **Performance analysis**: fast SW simulation of system model (AADS-T and SCoPE tools)
    - Transformation of AADL system model into executable **SystemC model**
    - Compatible with the Ravenscar Computational Model (ASSERT computational model)
    - Tailored for the Leon2 microprocessor
    - Performance metrics of interest: CPU load, cache misses/hits, power consumption, bus load
    - Required input: ASSERT Concurrency View
  - **ASSERT Model Transformer (AMT)**
CO-DESIGN (2/2)

- **HW component generation, simulation and optimization**
  - HW components developed in **ImpulseC**
  - HW functions developed in **ANSI C**
  - Additional components to communicate with the SW system: **ASSERT HW broker**
    - Encoding/Decoding messages
    - Dispatching HW services
    - Multiplexing/De-multiplexing data streams

- **SW implementation**
  - No changes with regard to the ASSERT process
  - SW interfaces implemented as HW components are already included in the ASSERT Interface and Deployment views
HW/SW TOOL CHAIN

- HW/SW methodology is supported by the HW/SW tool chain:
  - OSATE
  - GMV’s ASSERT Model Transformer (AMT)
  - University of Cantabria’s AADS-T and SCoPE tools
  - ASSERT toolset
HW/SW TOOL CHAIN – AADS-T/SCOPE

- System-Level Performance Tool AADS-T (AADS for TASTE):
  - Implementing the generation of SystemC in SCoPE.
  - Implementing bus load in SCoPE. Modelling the LEON2 processor for SCoPE.
  - Ravenscar Computational Model.
  - Generation a SystemC file with the system description.
HW/SW TOOL CHAIN - ECLIPSE

- All tools are integrated in Eclipse environment:
HW/SW Co-design

DEMONSTRATION ON A CASE STUDY
CASE STUDY: OVERVIEW

- **Goal**: To exercise the HW/SW co-design methodology
- **Case-study**: Simplified space application for digital image processing
  - Representative for the space domain
  - Exercises all phases: from system specification to system integration
CASE STUDY: IMAGE APPLICATION

- **High-pass filter:**
  \[
  h = \frac{1}{8} \cdot \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}
  \]

- **Low-pass filter:**
  \[
  h = \frac{1}{9} \cdot \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}
  \]

- **Vertical frequencies enhancement filter:**
  \[
  h = \frac{1}{3} \cdot \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}
  \]

BMP Images
Gray scale
256Kbytes
CO-SPECIFICATION (1/2)

- **System Logic View**: Data Model, Interface and Functional Views
  - Unbiased either to HW or SW: Platform Independent Model (PIM)
  - Modelled with the ASSERT toolset
CO-SPECIFICATION (2/2)

- **System Platform View**: description of HW elements and the system platform architecture
  - Describes the system platform architecture (PDM)
  - Provides implementation details of HW elements: read time, word size, implementation technology
  - Modelled with the OSATE editor
AMT generates the Partition View automatically from the system model

- **SW-centric approach:**

  - **properties**
    - `Actual_Processor_Binding` => `reference Leon2 applies to Housekeeping;`
    - `Actual_Processor_Binding` => `reference Leon2 applies to Event_Generation;`
    - `Actual_Processor_Binding` => `reference Leon2 applies to Connection_Test;`
    - `Actual_Processor_Binding` => `reference Leon2 applies to Image_Filter;`
    - `Actual_Processor_Binding` => `reference Leon2 applies to Startup;`
    - `Actual_Processor_Binding` => `reference Leon2 applies to HW_Controller;`
    - `Actual_Processor_Binding` => `reference Leon2 applies to Image_Processing;`
    - `Actual_Processor_Binding` => `reference Leon2 applies to Stub_Simulating_GROUND;`
    - `Actual_Processor_Binding` => `reference Leon2 applies to Image_Processing_Management;`
    - `Actual_Processor_Binding` => `reference Leon2 applies to Monitoring;`
    - `Actual_Processor_Binding` => `reference Leon2 applies to Control;`
For each partition scheme, the following steps are executed:

1. Generate with AMT:
   - ASSERT Deployment View
   - AADS-T Deployment View
2. Generate and import to OSATE the ASSERT Concurrency View (AMT)
3. Obtain the performance analysis results (AADS-T)
4. Evaluate the performance analysis results
CO-DESIGN: AADS-T PERFORMANCE ANALYSIS

AADS-T Console

- Use of cpu: 54.3057%
- Instructions executed: 41827564
- Instruction cache misses: 54393
- Data cache hits: 0
- Data cache misses: 0
- Data cache write backs: 0
- Core Energy: 1.2816e+09 nJ
- Core Power: 256.319 mW
- Instruction Cache Energy: 1.09567e+09 nJ
- Data Cache Energy: 0 nJ
- Instruction Cache Power: 391.134 mW
- Data Cache Power: 0 mW
- Bus access time: 866120 ns
- Idle time: 2273287400 ns
- Number of interrupts: 8819

Total instruction miss transfers: 638
Total data miss transfers: 0

processor_Ltos_0
- Number of thread switches: 12102
- Number of context switches: 0
- Running time: 2055786540 ns
- Use of cpu: 41.158%
- Instructions executed: 31666202
- Instruction cache misses: 38682
- Data cache hits: 0
- Data cache misses: 0
- Data cache write backs: 0
- Core Energy: 9.70252e+08 nJ
- Core Power: 194.05 mW
- Instruction Cache Energy: 1.47908e+09 nJ
- Data Cache Energy: 0 nJ
- Instruction Cache Power: 295.817 mW
- Data Cache Power: 0 mW
- Bus access time: 6205120 ns
- Idle time: 2995233780 ns
- Number of interrupts: 14790

Total instruction miss transfers: 463
Total data miss transfers: 0

Bus Lan_0
- Bytes transferred: 2077960
Main problem: lack of support of the serial lines by the TASTE tool chain.
- ORK UART driver were not integrated
- Workaround: developing the code manually but
  - using enumerations auto-generated from Ocarina (component and interface IDs)
  - coding following the Ravenscar computational model
  - packaging data on the serial bus as done by polyORB-Ada
CO-DESIGN: HW IMPLEMENTATION (1/2)

- **HW component generation, simulation and optimization**
  - HW components developed in **ImpulseC**
    - HW components
    - ASSERT Broker
  - HW functions implemented in **ANSI C**
  - Additional components to communicate with the SW system: **ASSERT HW broker**
    - Encoding/Decoding messages
    - Dispatching HW tasks
    - Multiplexing/De-multiplexing data streams
  - Additional components developed in HW description languages
Implementation and optimization of the HW system:

- HW components (ImpulseC), derived from the system model, independent of the target platform
- Pure HW elements (VHDL), dependent of the target platform:
  - IP Core for interface management/memory access
  - Development of HW drivers for interfaces
  - Development of interfaces between pure HW and HW components
**ImpulseC follows a similar programming model based on modified Communicating Sequential Processes (CSP)**

- Fixed number of sequential tasks communicating with each other through synchronous message-passing
- Asynchronous message-passing is also available (non-blocking reads)
- Primitives: events (stream and signals), and processes (functional behaviour)
Assimilation of the System-level component model for provided interfaces:

- Cyclical interfaces: periodic execution of HW tasks (A)
- Sporadic interfaces: event-driven execution of HW tasks (B)
  - Events: data available in the incoming data streams or signals
- Protected interfaces: event-driven execution of HW tasks (C)
  - Events: data available in the incoming data streams or signals
  - Barrier: internal process semaphores
Communication with the SW system:

- Interfacing the SW system means:
  - **Marshalling/un-marshalling** messages from/to the SW system
  - **Multiplexing/de-multiplexing** data to/from the SW system

- HW ASSERT Broker: dispatch HW tasks associated to components mapped to the HW system
HW IMPLEMENTATION: ASSERT BROKER

Message structure:
- Version 2010-01-07
- Component and Interface IDs: auto-generated code by OCARINA
- Interface parameters: data types defined in data view, marshalling and un-marshalling algorithms auto-generated by ASSERT toolset
HW IMPLEMENTATION: INTEGRATION

- HW components and ASSERT Broker (ImpulseC)
  - Transformed into VHDL using the ImpulseC compiler

- Integration of signals, streams and other staff performed manually
  - Integration of VHDL code
HW IMPLEMENTATION: OPTIMIZATION

- Pre-conditions to the optimization:
  - The data model cannot be modified
  - The HW component interface cannot be modified, unless data streams are not used

- Two levels of optimization:
  - At ImpulseC level (pre-integration):
    - Deleting unused resources like data streams, signals
    - Pipelining sections of the ImpulseC code
    - Modifying the programming defines
  - At VHDL level (post-integration):
    - Signals and port numerical ranges
    - Improvements on the pure HW elements
    - Synthesis options (i.e. XILINIX tool)
Objectives for HWSWCO:
- Improvements on Design Speed for the Image processing functions
- If possible, reduction of the amount of resources used

Results:
- Reduction of the Design speed by 31%
- Reduction of occupied slices by 28%
- Increment of the RAM memory used by 8%
USE-CASE: EXECUTION (1/2)

Linux PC: Camera simulation

Linux PC: Grmon

Virtex 5

Leon2
USE-CASE: EXECUTION (2/2)

- Camera simulation: java application sending images sequentially
HW/SW Co-design

CONCLUSIONS AND FUTURE WORK
CONCLUSIONS

- Two different analyses must drive the HW/SW partitioning process:
  - Performance and timing analysis (SW simulation)
  - Worst Case Execution Time and Schedulability Analysis

- A common integration substrate must be defined, enabling the propagation of changes between the HW and SW development paths
  - A common data model, especially in terms of data size and bit ordering
  - The HW system should be compliant with the ASSERT component model, and have a similar programming model
  - The HW system should use a similar programming language
  - The HW system must be aware of the deployment information generated by the ASSERT toolset:
    - Identification of system components and their interfaces
FUTURE LINES OF WORK (1/2)

This study does not cover all aspects of the HW/SW co-development:

- Design Space Exploration: automatic multi-objective optimization, key in the HW/SW process
- Automatic generation of HW wrappers and broker(s), part of the common integration substrate

Regarding to the ASSERT process:

- Implications of the ASSERT computational model in the HW system
- Timing analysis of the HW/SW system
FUTURE LINES OF WORK (2/2)

- **Adopting SystemC** as a HW description language:
  - Developing SystemC libraries that enforce a particular HW/SW computational model
    - SystemC is an open standard
    - The functional code is still ANSI C
    - Availability of EDA tools to transform SystemC to VHDL

- Improvement of **support processes** like Verification & Validation, and Dependability & Safety.

- Porting AMT and AADS-T to **AADL v2**
Thank you!

Ana-Isabel Rodríguez
airodriguez@gmv.com

Francisco Ferrero
fferrero@gmv.com

http://hwswcodesign.gmv.com
http://www.gmv.com