An application-centered Design Flow for Self Reconfigurable Systems implementation

Fabio Cancare: fabio.cancare@polimi.it

Marco D. Santambrogio: marco.santambrogio@polimi.it

Donatella Sciuto: donatella.sciuto@polimi.it

14th Asia and South Pacific Design Automation Conference
Jan. 19-22 2009 – Yokohama, Japan

Rationale

- Dynamic reconfiguration is a new and promising technique, it can be applied to cope with:
  - Lack of available resources
  - System adaptability
  - System reliability
- Main drawback: implementing DR systems is a complex and time-consuming task
- Model-based design paradigm allow the fast development of complex architecture

Understand how it is possible to exploit the model-design paradigm in dynamic reconfigurable system implementation

Innovative Contribution

- Outline a model-based design flow for implementing large designs onto FPGAs with limited available resources

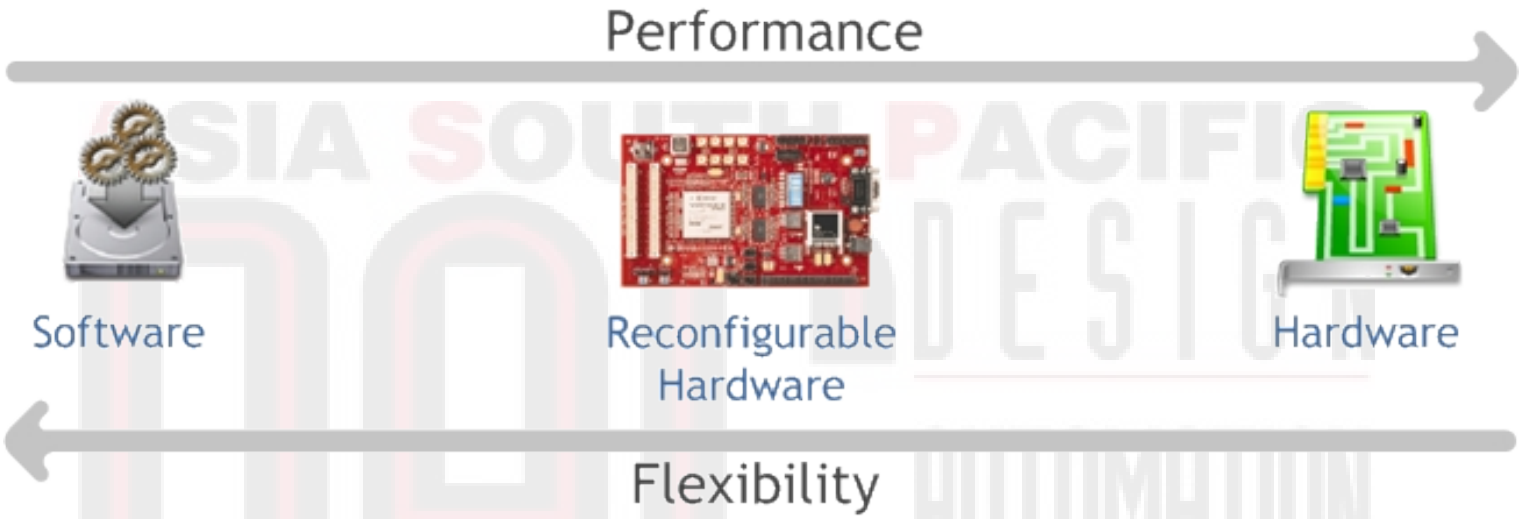
Outline

- Fundamental Concepts
- Proposed Flow
  - High-level Modeling Phase
  - Low-level Implementation Phase
- Case Study
- Results
- Conclusions and follows-up

Outline

- Fundamental Concepts
- Proposed Flow
  - High-level Modeling Phase
  - Low-level Implementation Phase
- Case Study
- Results
- Conclusions and follows-up

Reconfigurable Computing



"Reconfigurable computing is intended to fill the gap between hardware and software, achieving potentially much higher performance than software, while maintaining a higher level of flexibility than hardware"

(K. Compton and S. Hauck, *Reconfigurable Computing: a Survey of Systems and Software*, 2002)

Reasons Behind

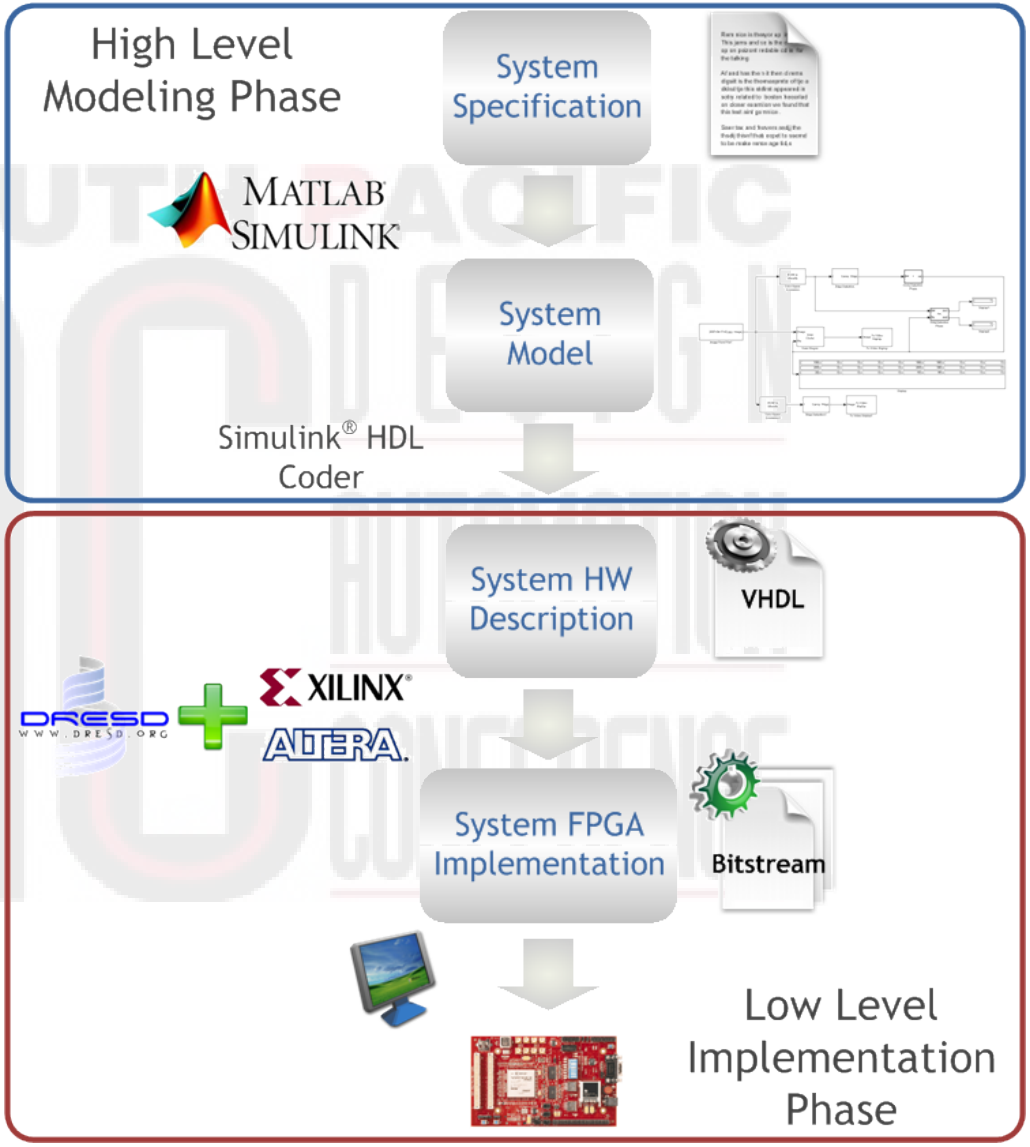
- Applications often require performance which cannot be achieved by software
- Applications often require to be flexible, modifiable, adaptable. Traditional hardware cannot achieve such results
- **Reconfigurable Computing** techniques are able to alter a concrete architecture once it has been deployed onto a high-performance device, in order to meet:
  - Resources constraints
  - Adaptability constraints
  - Reliability constraints

Outline

- Fundamental Concepts
- Proposed Flow
  - High-level Modeling Phase
  - Low-level Implementation Phase
- Case Study
- Results
- Conclusions and follows-up

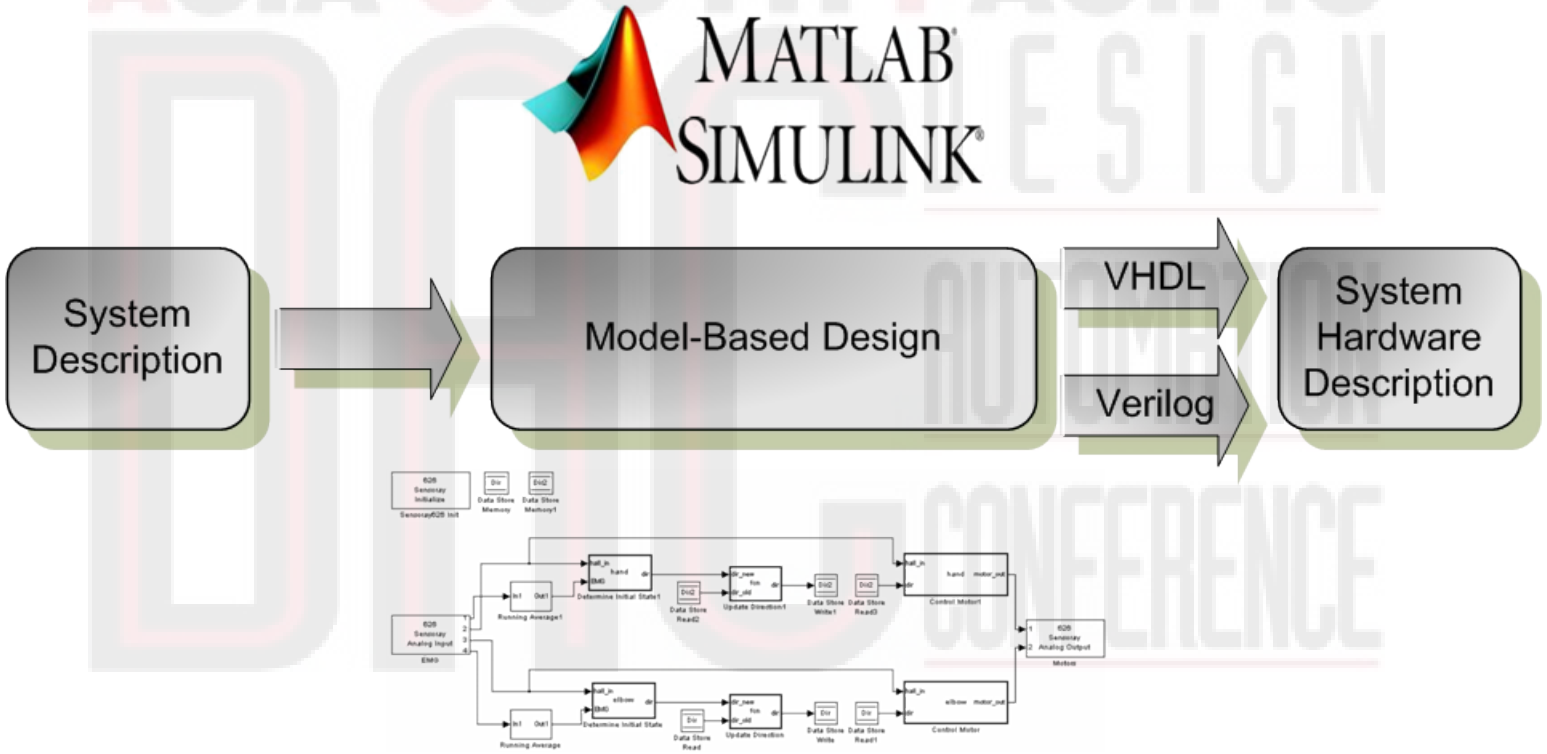
Flow Overview

- Dynamic Partial Reconfiguration
- Flow composed of two phases:
  - HLMP
  - LLIP
- Support from system specification to system FPGA implementation

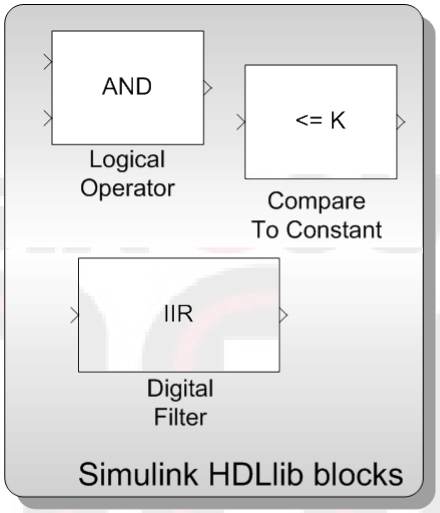


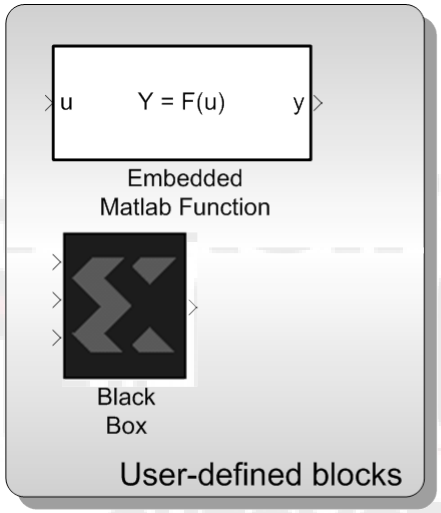
High Level Modeling Phase

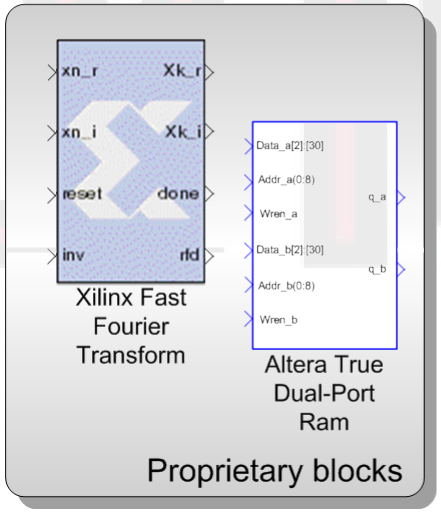
- From the System Specification to the System Hardware Description

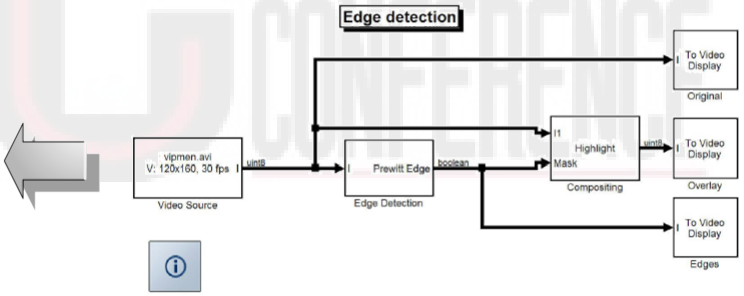


Simulink HDL Coder Compliant Models



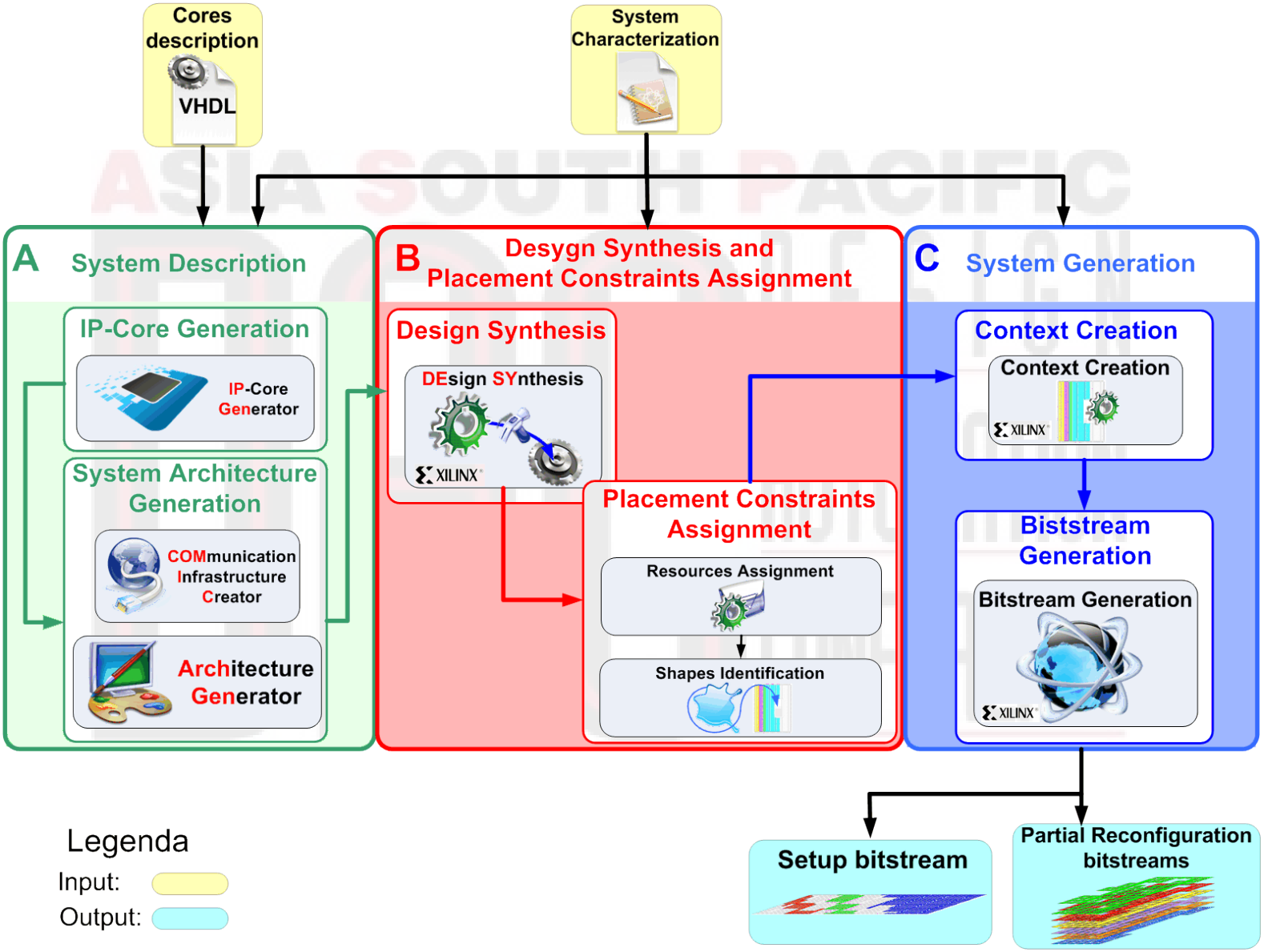






Simulink HDL Coder compliant models

Low Level Implementation Phase



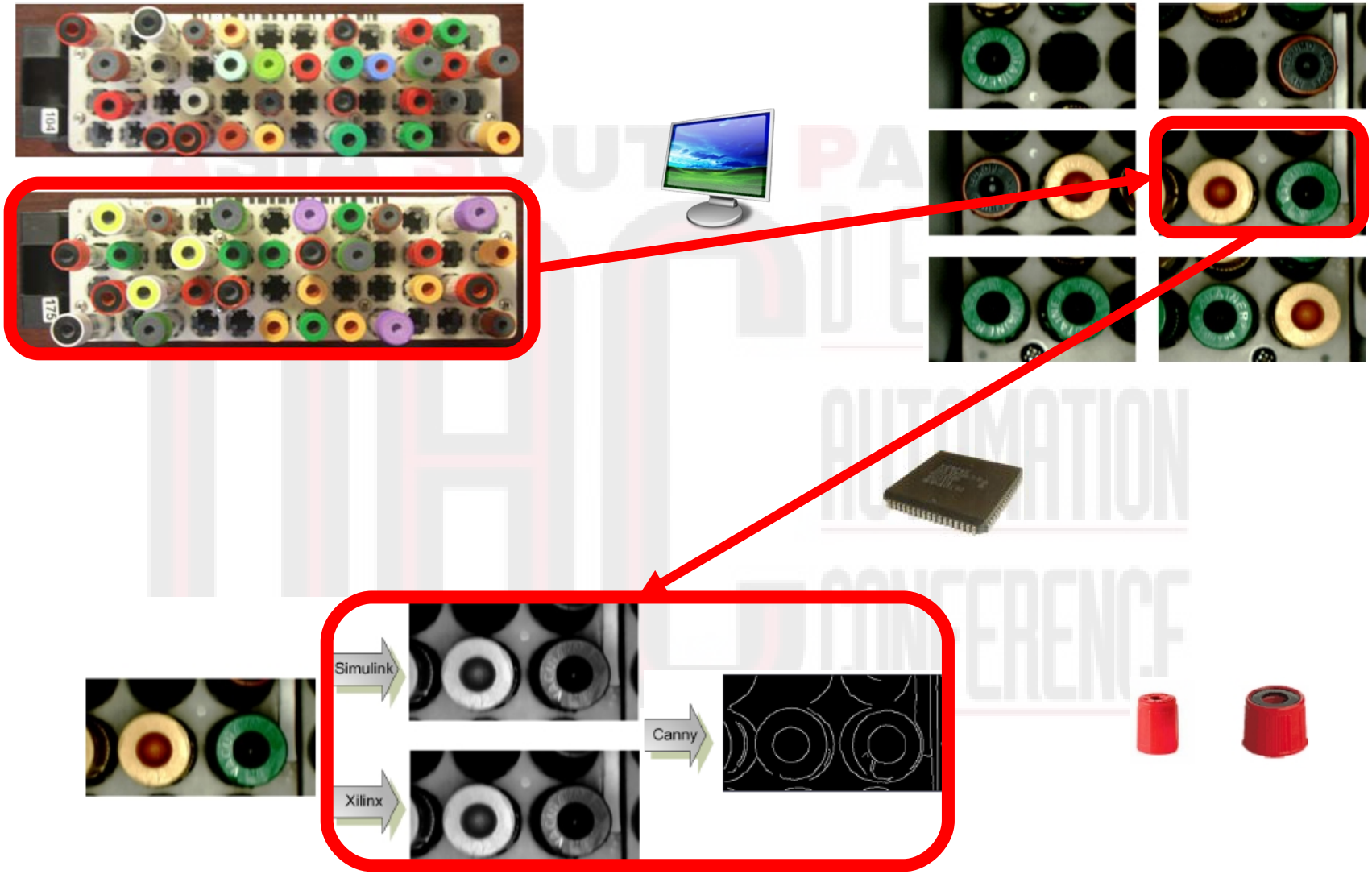
Outline

- Fundamental Concepts
- Proposed Flow
  - High-level Modeling Phase
  - Low-level Implementation Phase
- Case Study
- Results
- Conclusions and follows-up

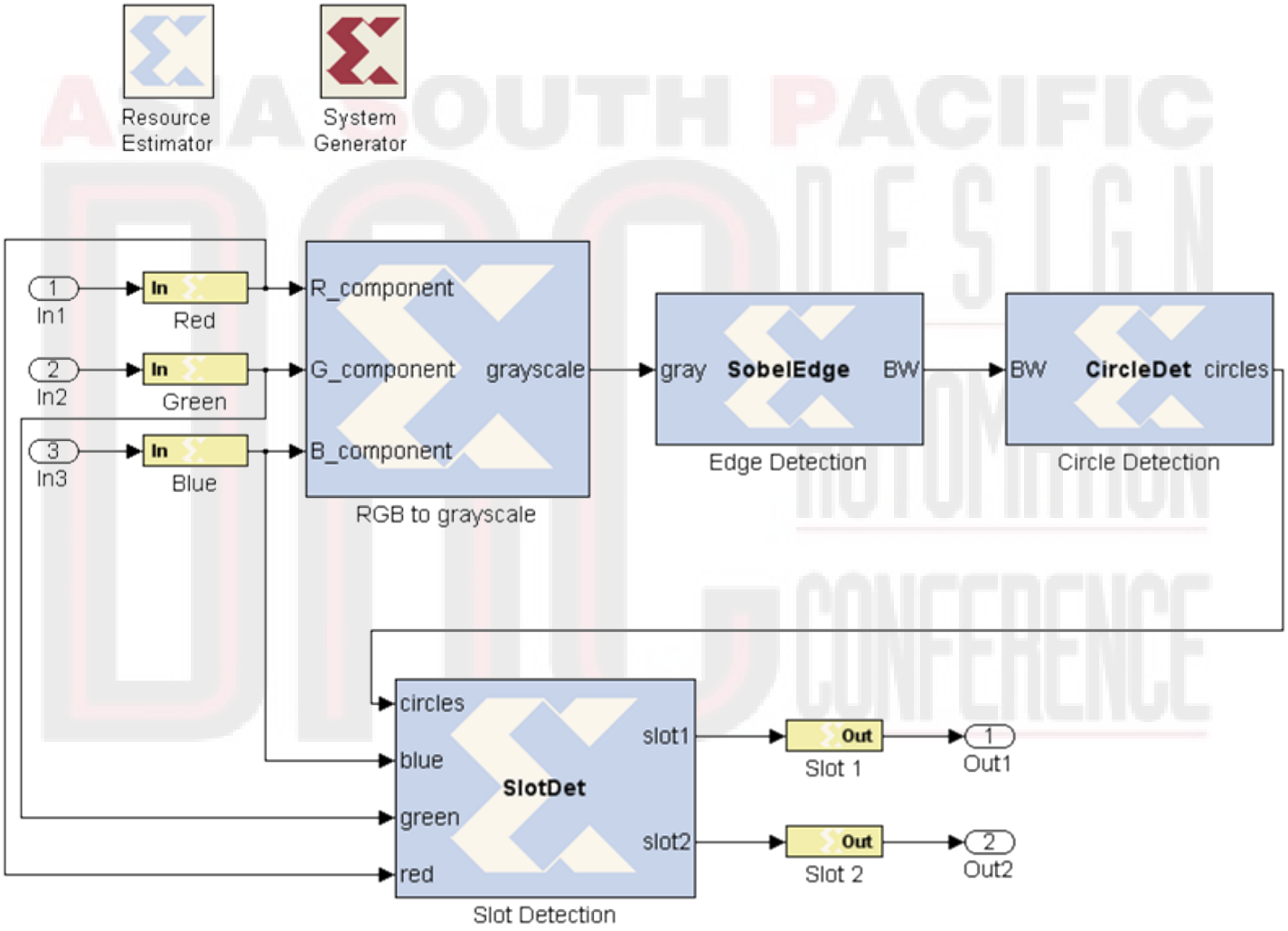
Real-world Application

- Inpeco Corporation proposed to implement an embedded vision system exploiting dynamic reconfiguration
- The goal is to provide functionalities such as:
  - Mapping of the test-tubes within a rack
  - Test-tube sample chromatic analysis
  - Test-tube lateral recognition
- The system must also be as flexible as possible, since new functionalities may be required in the future

Overall Description



Real-world Application – System Model

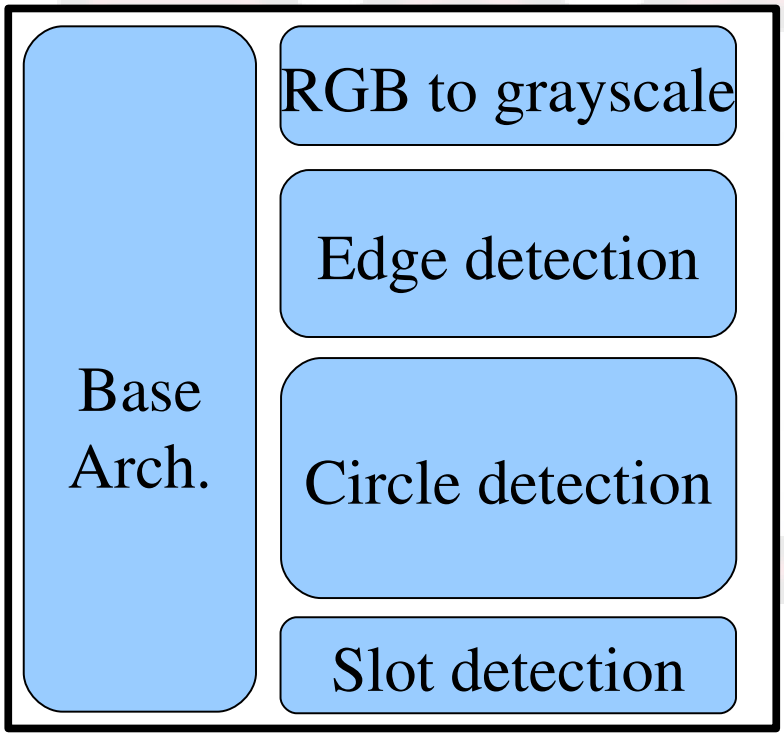


Outline

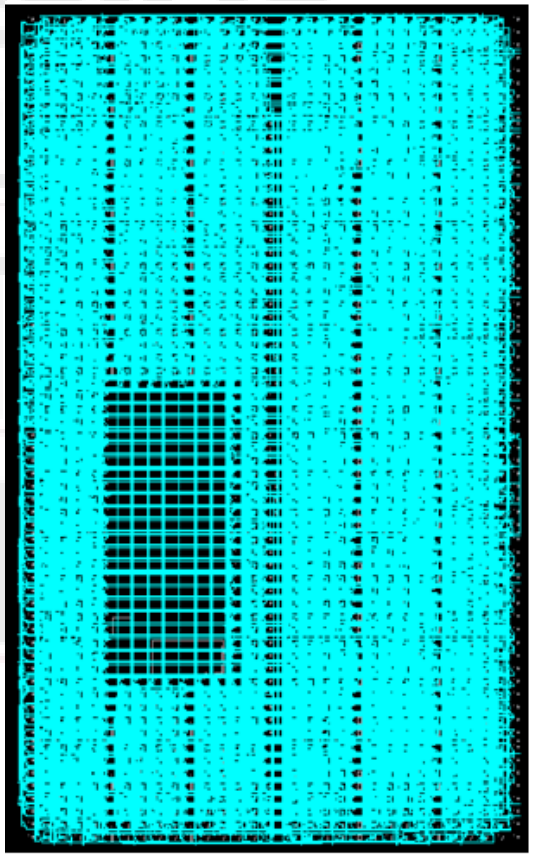
- Fundamental Concepts
- Proposed Flow
  - High-level Modeling Phase
  - Low-level Implementation Phase
- Case Study
- Results
- Conclusions and follows-up

Classical System

- The static system implementation uses 96.9% of targeted device, a Xilinx XC4VFX12-FF668-10 FPGA, slices (and some functionalities are missing)



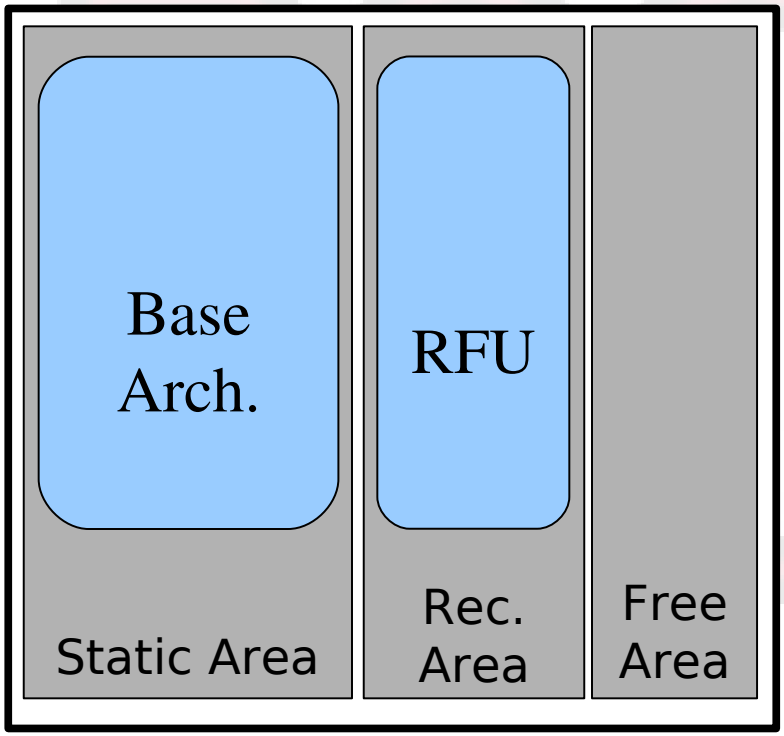
FPGA Logical view



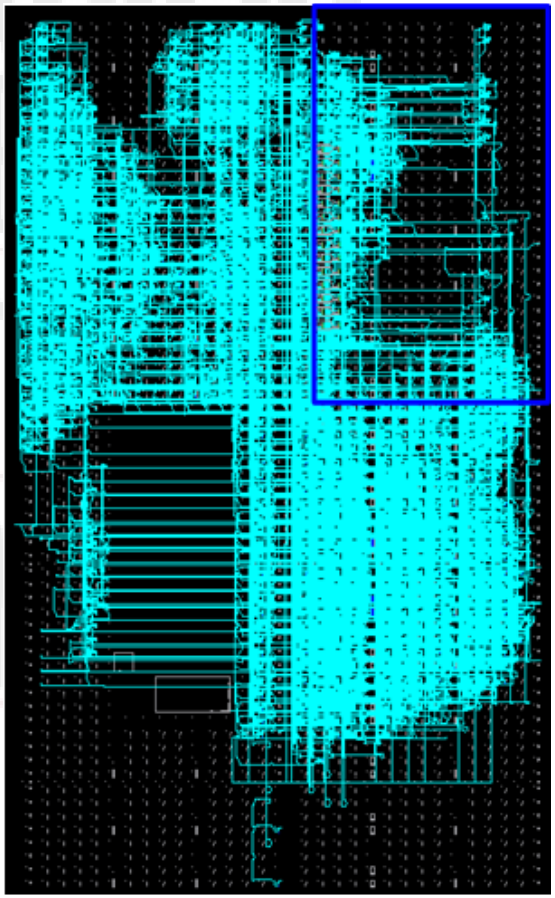
FPGA Physical view

Dynamic Reconfigurable System

- The dynamic reconfigurable implementation uses only 66.6% of available slices (26.0% of them can be reused)



FPGA Logical view



Physical view

Occupation Data

Static System

Resource Type	Used	Available	Percentage
Slices	5303	5472	96.9%
Flip Flops	3269	10944	29.9%
4 input LUTs	9032	10944	82.5%
LUTs used as logic	8624	9032	95.5%
LUTs used as shift register	24	9032	0.3%
LUTs used as RAMs	384	9032	4.2%

Dynamic Reconfigurable System

System Component	Slice Occupied	Slice Available	Percentage
Base architecture	2364	5472	43.2%
RGB to grayscale	212	5472	3.9%
Edge detector	936	5472	17.1%
Circles detector	1427	5472	26.0%
Slots detector	507	5472	9.3%

Outline

- Fundamental Concepts
- Proposed Flow
  - High-level Modeling Phase
  - Low-level Implementation Phase
- Case Study
- Results
- Conclusions and follows-up

Conclusions and Follow-ups

- The model-based design paradigm has been successfully used as part of a dynamic reconfigurable system implementation flow
- The proposed flow has been employed to produce a first version of an industrial application

- Test the approach with other applications
- For what concerns the case study, it is necessary to implement the other functionalities and to introduce DMA

Any Question?

- Thank you very much!