System Architecture for Safety-Critical High-Performance Systems: Current Approaches and Future Perspectives

Bruno MONSUEZ
Department Of System and Computer Engineering

CONTEXT: A DEFINITION OF DEPENDABILITY

- Availability (“readiness for correct service”),
- Reliability (“continuity of correct service”),
- Integrity (“maintaining the consistency of data”),
- Maintainability (“ability for a process to undergo modifications and repairs”),
- Safety (“absence of catastrophic consequences on the users and the environment”)
- Security (“prevention of unauthorized disclosure of information”)
- Certificability (“capacity of to obtain safety certification from standard authority“).

CONTEXT: INCREASING COMPLEXITY

- SYSTEM COMPLEXITY
  - Numbers of Controlled System Components
  - Interaction between System Components
  - Numbers of monitored Variables
- FUNCTIONAL COMPLEXITY
  - Data & Sensor Fusion
  - Law of Control-Commands
  - Decision algorithms for Autonomous Systems
  - Shared Authority
- SYSTEM ARCHITECTURE COMPLEXITY
  - Heavily Distributed Systems
  - Communications & Network centric Systems
- HARDWARE COMPLEXITY
  - Mono & Multi-Core Processor Architecture with non-deterministic inter-connect
  - Distributed Multi-Level Cache Architecture
  - High-Performance DSP & FPGA

THE CURRENT PROBLEMS WHEN DESIGNING CRITICAL APPLICATIONS

- Being capable to start the verification at functional level

  Functional and specification errors are the most costly to detect and correct.

- Being capable to determine and guarantee the worst cases

  Non-determinism of the hardware.

  Model of the system environment.

- Being capable to replicate the faulty states

  Distribution and concurrency of sub-systems.

  Non-determinism of hardware components.

- Being capable to make the system maintainable

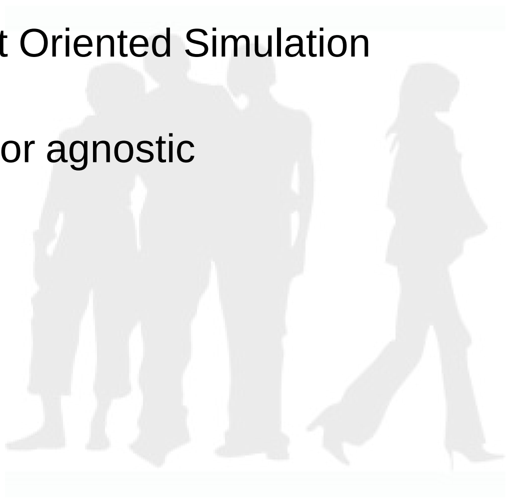
  Replace obsolete hardware with new hardware.

  Replace part of the code without revalidating all the code.

SYSTEM LEVEL VIRTUAL PROTOTYPE

- A System Level Virtual Prototype is a fully functional software model of a system that can run code
  - Provides all the abstraction of the various components
  - Allow to run the OS as well as the application code
  - Should run as close as to the real time as possible.
  - Allow to mix abstraction level

- Implementation & Availability
  - May be related to a platform like MOOSE (Motorola Object Oriented Simulation Environment)
  - May be proprietary like VaST, Virtio, Virtutech but processor agnostic
  - May rely on a standard API (SystemC and TLM 2.0)
  - May combine FPGA & Emulators to provide acceleration.



BENEFIT OF USING SYSTEM LEVEL VIRTUAL PROTOTYPE

- Early Verification and Validation

  Allows to start the Verification & Validation process at functional level.

- Architectural Analysis

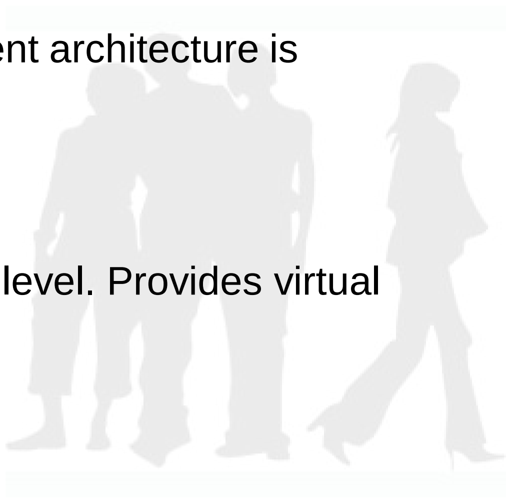
  Allows to explore the architectures and to determine if the performance is adequate for the application

- Software Development

  Allows to begin the software development before the current architecture is available.

- Debug

  Provide a set of virtual tools to allow debugging at system level. Provides virtual JTAG.



THE LIMIT OF SYSTEM LEVEL VIRTUAL PROTOTYPE

- **Accuracy of the SLVP**

  Compromise between the model accuracy & the simulation time.

- **Production Cost**

  The overhead cost of developing such a model and the possible reuse.

- **Time of availability**

  How long does it take to go from the specification to the SLVP

- **Execution control**

  Capacity of controlling the execution during debug.

- **System Interfaces**

  Connecting the SLVP to the external components.

USING ASIP AS AN ALTERNATIVE TO ISA PROCESSORS

- Can provide better performance than a standard ISA processor

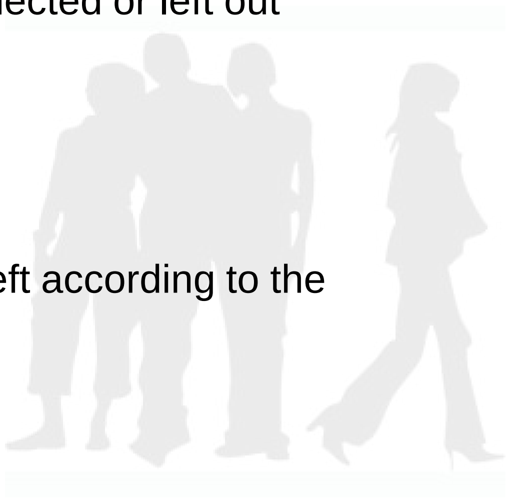
ASIP coarse-grained configuration can offer specific instructions dedicated for specific applications (data insensitive, DSP like,…)

Size, widths and organization of registers may be fine-tuned for an application domain.

Busses, Arbiters and interconnects may be configured, selected or left out according to the application domain.

Interruption may be configured according to the need.

Interfaces, caches, cache strategy may be configured or left according to the application domain.

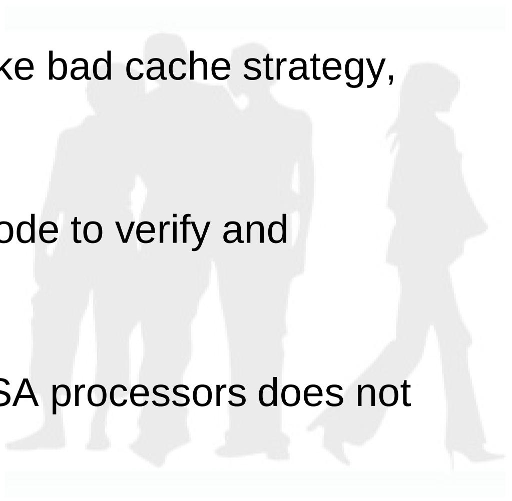


ASIP AS AN ALTERNATIVE TO ISA PROCESSORS

- Supports multi-core design
  - Homogeneous design where many cores are available
  - Heterogeneous design where different cores can be integrated

- Supports of multiple ASIP

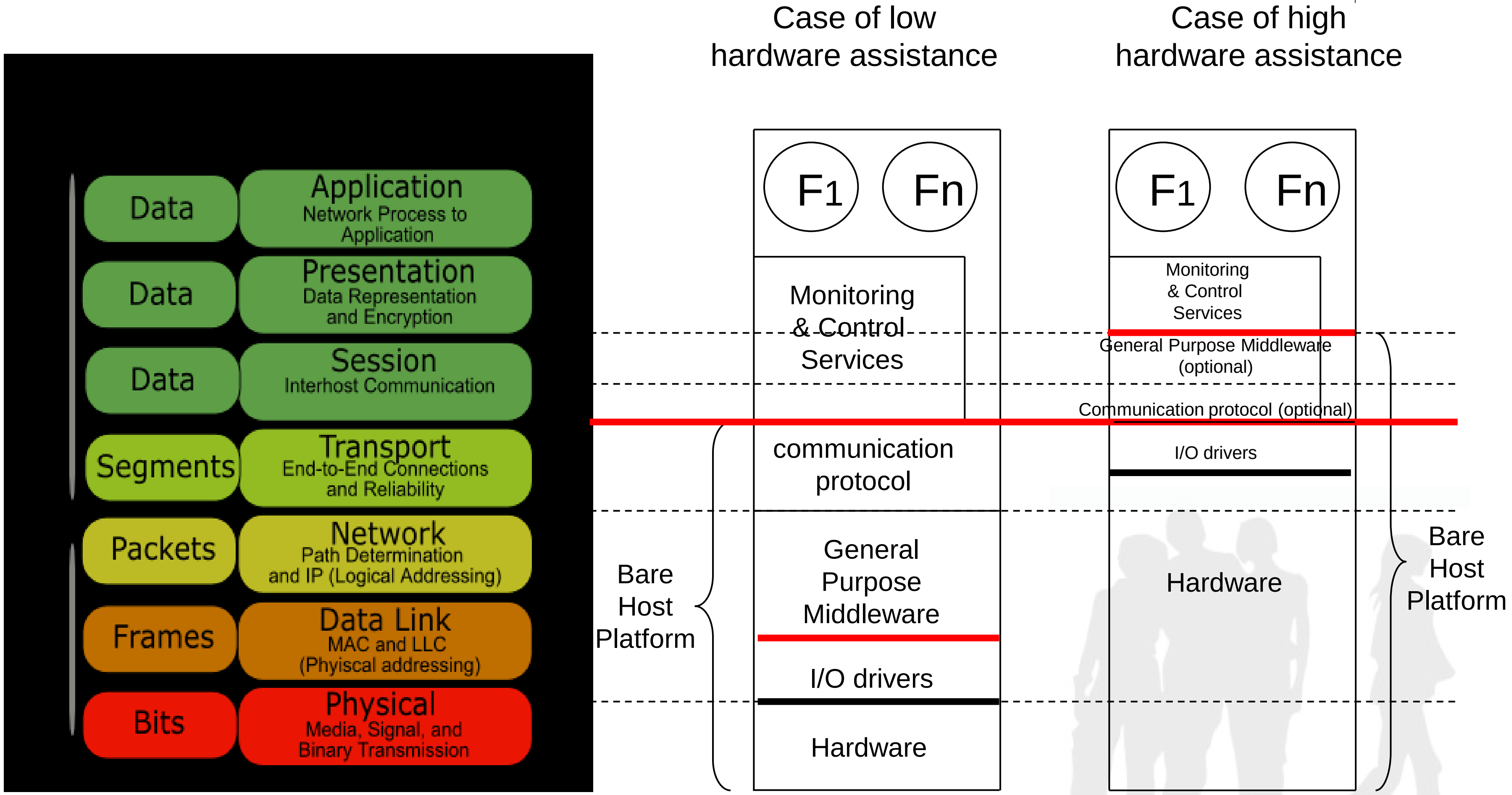
- Interest for Safety Critical Application
  - ASIP can be an alternative to ISA multi-core processors
    - May be configured so that non-deterministic features like bad cache strategy, best effort interconnect, … are left out.
  - ASIP may provide better performance and more simpler code to verify and validate.
  - ASIP may offer some additional monitoring features that ISA processors does not implement.



SYSTEM ORIENTED ARCHITECTURE FOR SAFETY CRITICAL APPLICATION

- Provides a Service Oriented Structure that offers
  1. System Virtualization
     - Hardware virtualization
     - Virtualizes the Input/Output
     - Virtualizes the communication
  2. Services
     - Generic Services
     - Monitoring
     - Configuration
     - Debugging
     - Power & Energy
  3. Application Management
     - OS like Arinc 653
     - Hypervisors

SYSTEM ORIENTED ARCHITECTURE MAY BE HARDWARE OR SOFTWARE DEFINED


Data
Application
Network Process to Application
Data
Presentation
Data Representation and Encryption
Data
Session
Interhost Communication
Segments
Transport
End-to-End Connections and Reliability
Packets
Network
Path Determination and IP (Logical Addressing)
Frames
Data Link
MAC and LLC (Phyiscal addressing)
Bits
Physical
Media, Signal, and Binary Transmission
Case of low hardware assistance
F1
Fn
Monitoring & Control Services
communication protocol
General Purpose Middleware
I/O drivers
Hardware
Bare Host Platform
Case of high hardware assistance
F1
Fn
Monitoring & Control Services
General Purpose Middleware (optional)
Communication protocol (optional)
I/O drivers
Hardware
Bare Host Platform

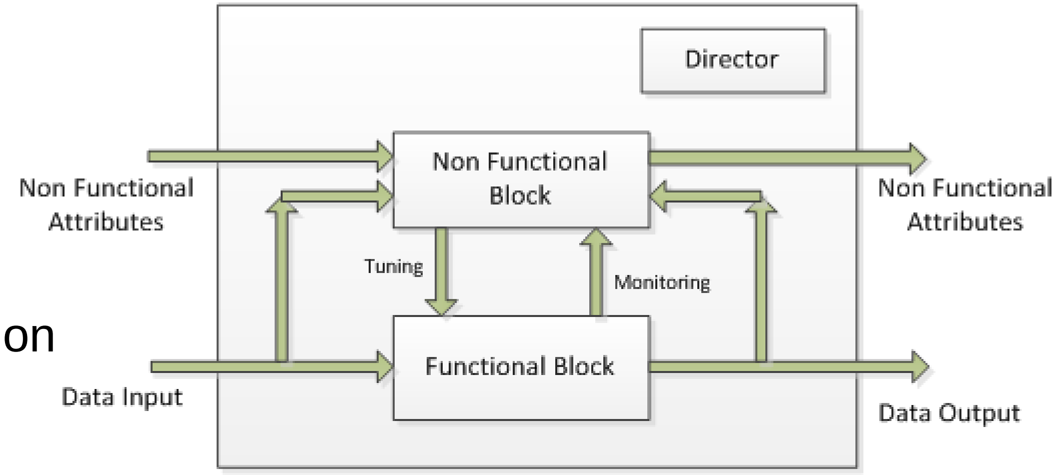
DEVELOPING SAFETY CRITICAL APPLICATIONS: CURRENT PROBLEMS: REMAINING PROBLEMS

- Lack of standard models, techniques, and languages for system engineering that support the structured specification of multi-functional systems and their mutual dependencies.
- Use of several platforms and tightly coupled HW/SW design methods that restricts portability and reuse.
- Lack of methodology for meta model analysis during design time.
- Lack of compositional approaches that allow to prove system correctness by construction.

CURRENT TRENDS: INTEGRATION OF MULTIPLE MODELS

Expressing all Architecture Requirements

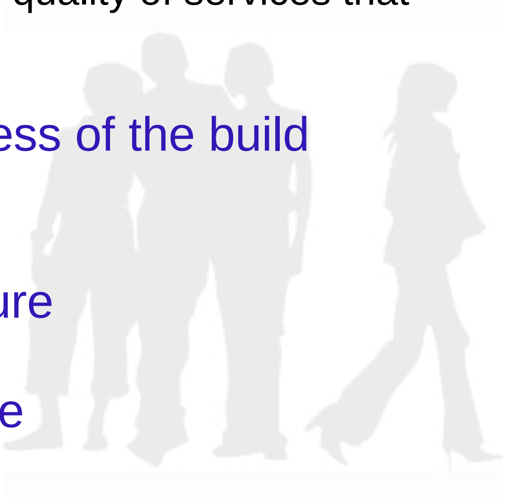
- Scalability
- Mixed Component
  Abstraction
- Quality Propagation
- Framework supported
  dynamic configuration
- Conflict Resolution
- Context Awareness
- Information Filter
- Reliability



First approach based on actor models like Ptolemy.

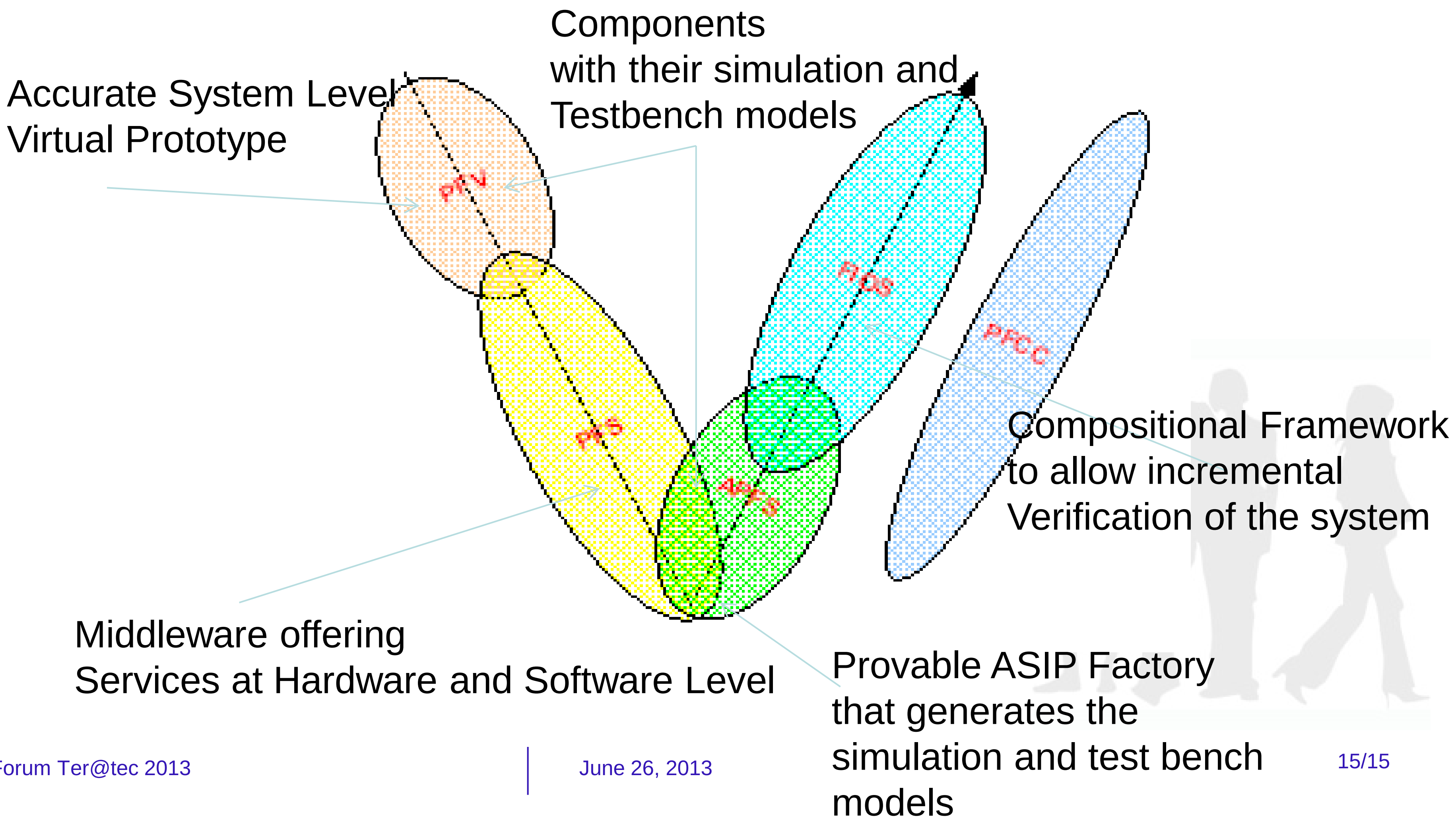
SIMPLIFYING THE MODEL BY DEFINING & USING ARCHITECTURE DESIGN PATTERNS

- Provides a way to implement a given functionality in a generic way by composing standard components.
  - Define requirements that the standard components must fulfill.
  - Define requirements on the services that must be provided to implement the functionality.
- Provides a pre-analysis of the system obtained by composition
  - Functional, Non-Functional, Dysfunctional.
  - Exposes the expected safety level as well as the expected quality of services that should be obtained.
- Defines the verification plan to prove the correctness of the build functionality
- Can be applied on Software Application Architecture
- Can be applied on System Application Architecture



A LA QUETE DU GRAAL

- Provides a complete open platforms that supports mixed abtractions and a common API


Components
with their simulation and
Testbench models
Accurate System Level
Virtual Prototype
PFV
POS
PFCC
PFS
APFS
Compositional Framework
to allow incremental
Verification of the system
Middleware offering
Services at Hardware and Software Level
Provable ASIP Factory
that generates the
simulation and test bench
models