Bundesministerium für Bildung und Forschung

DLR

Modeling, Analysis and Refinement of Heterogeneous Interconnected Systems Using Virtual Platforms

O. Bringmann (FZI), J. Gerlach (BOSCH),
U. Nageldinger (Infineon), J. Stellmacher (Cadence)

BOSCH cadence FZI infineon

Eberhard Karls Universität Tübingen OFFIS Universität der Bundeswehr München

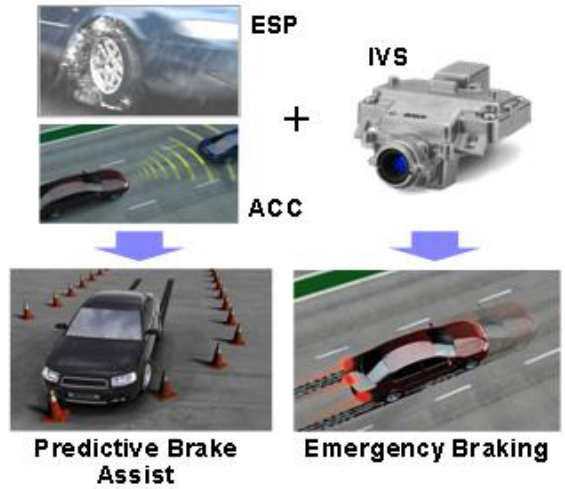
The VISION project (FKZ: 01 M 3078) is supported in accordance with the BMBF „Nanoelectronics“ sponsorship scheme within the scope of the Ekompass sponsorship complex of the Federal Ministry of Education and Research (BMBF).

Motivation

- **New applications by synergetic networking**
  - New functionality is less and less the sum of dedicated components, but the intelligent interconnection of these components
    - innovation as result of networks of interconnected ECUs
    - added value results more and more from the networked functionality
- **Suppliers are faced with an increasing system responsibility**
  - Supplier is not only responsible for the designed subsystem, but additionally for a safe integration of the subsystem into the entire system
    - today: Test of the requirements of a single component
    - in future: Validation of the entire system requirements under consideration of the designed component
- **Consideration of the effects of a component on the entire system already in the design stage**
  - comprehensive modeling of distributed systems
  - early analysis and simulation of the system integration

Requirements for Different Application Domains

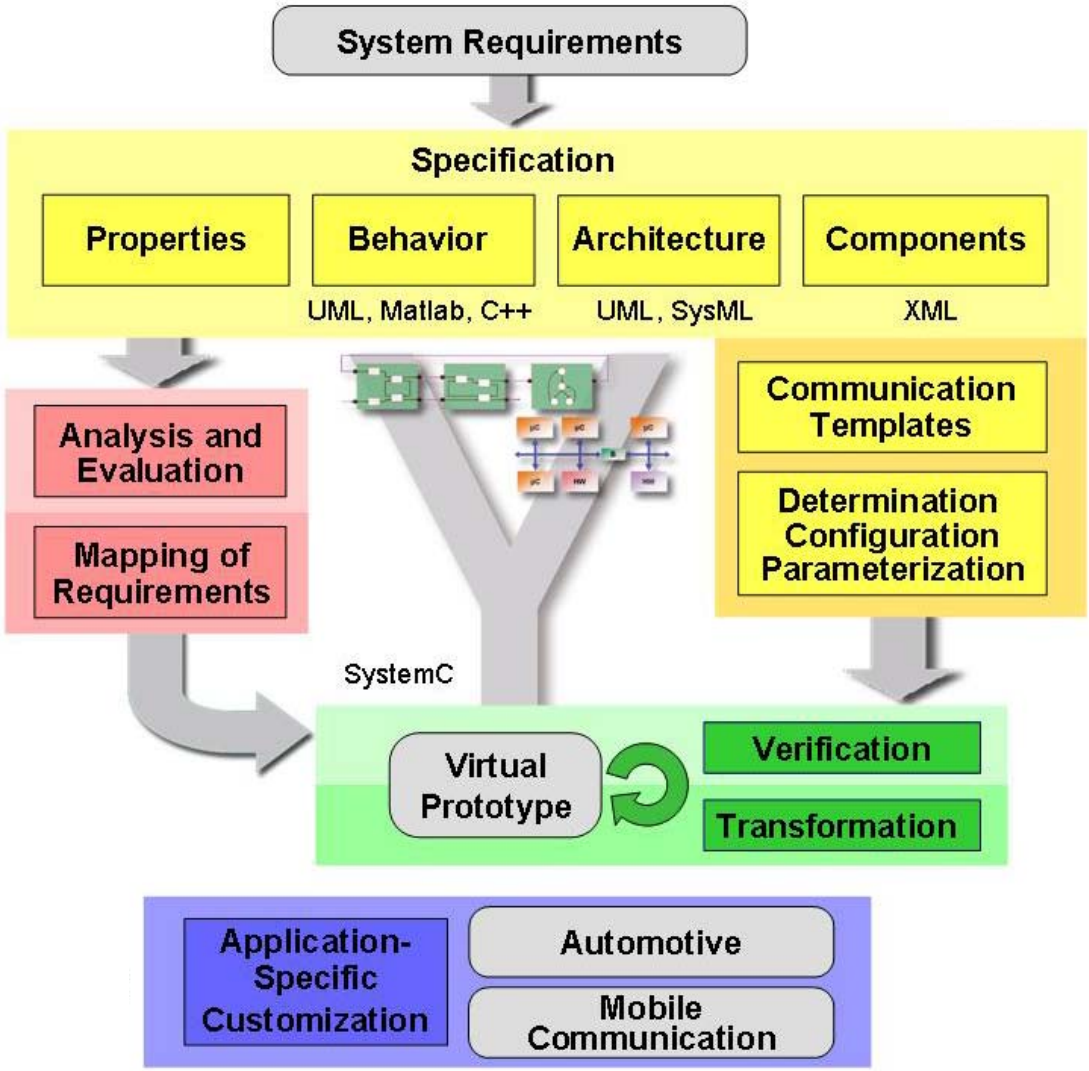
- **Automotive**
  - Transition from passive to active safety
  - Passive systems: Innovation by increasing the complexity and number of ECUs
  - Active systems: Innovation by interaction of ECUs, added-value by synergetic networking



- **Mobile Communication**
  - New application scenarios for mobile phones
  - Mobile phone is not a dedicated application, but an integrated communication device in the embedding system
  - Trend to complex product scenarios

- **New challenges**
  - Early consideration of the component behavior in the context of the entire system
  - Early analysis of global safety requirements
  - Flexible product variants
  - Fast reaction on varying market requirements

VISION Design Flow



Virtual Platform Modeling

- Requirements
  - Graphical and textual model-based platform composition
  - Easy incorporation of user-defined component models
  - Semi-automated platform refinement
  - Mapping of embedded software
  - Automated generation of virtual prototypes
- Status
  - Many commercial tools already available
    - Mentor Platform Express/Vista Architect, CoWare Virtual Platform, Synopsys Innovator, Magillem MPA, VAST CoMET, CoFluent Studio, …
    - Proprietary component characterization
      - Design services are offered for integration of user-defined components
    - IP-XACT-based component characterization
      - Limited support for vendor extension
    - Control on the internal data representation is missing
    - Limited support for refinement across multiple levels of abstraction
    - Design entry starts mostly on a very detailed level
-

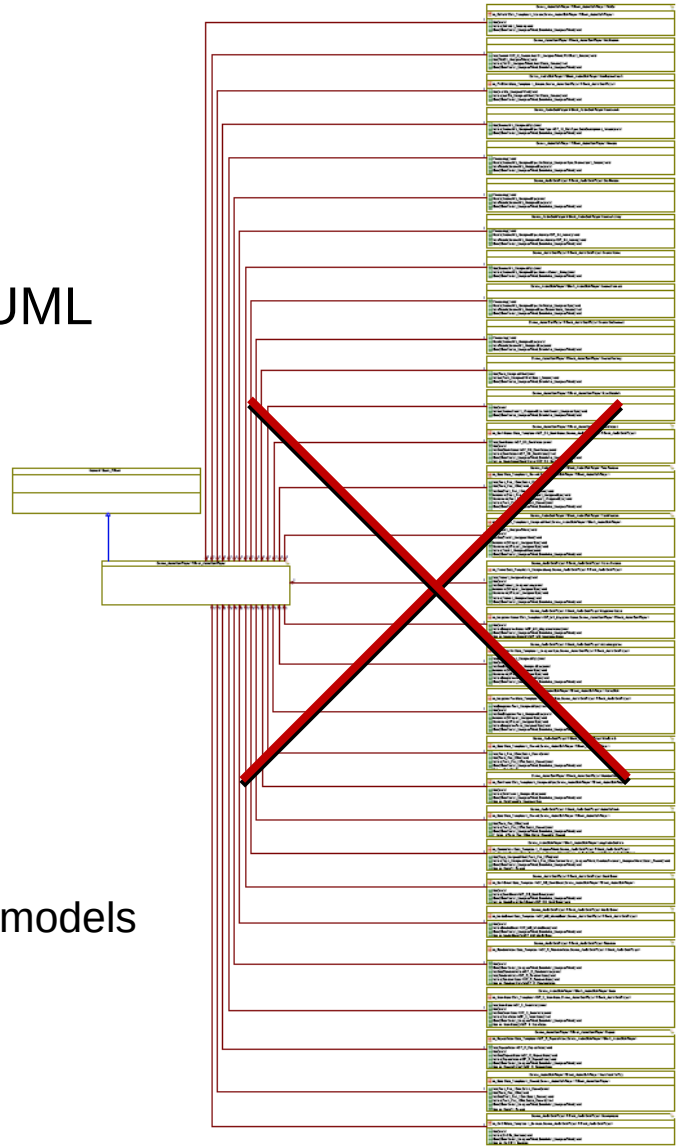
UML-based Platform Modeling and Refinement

Model-based platform composition: Advantages

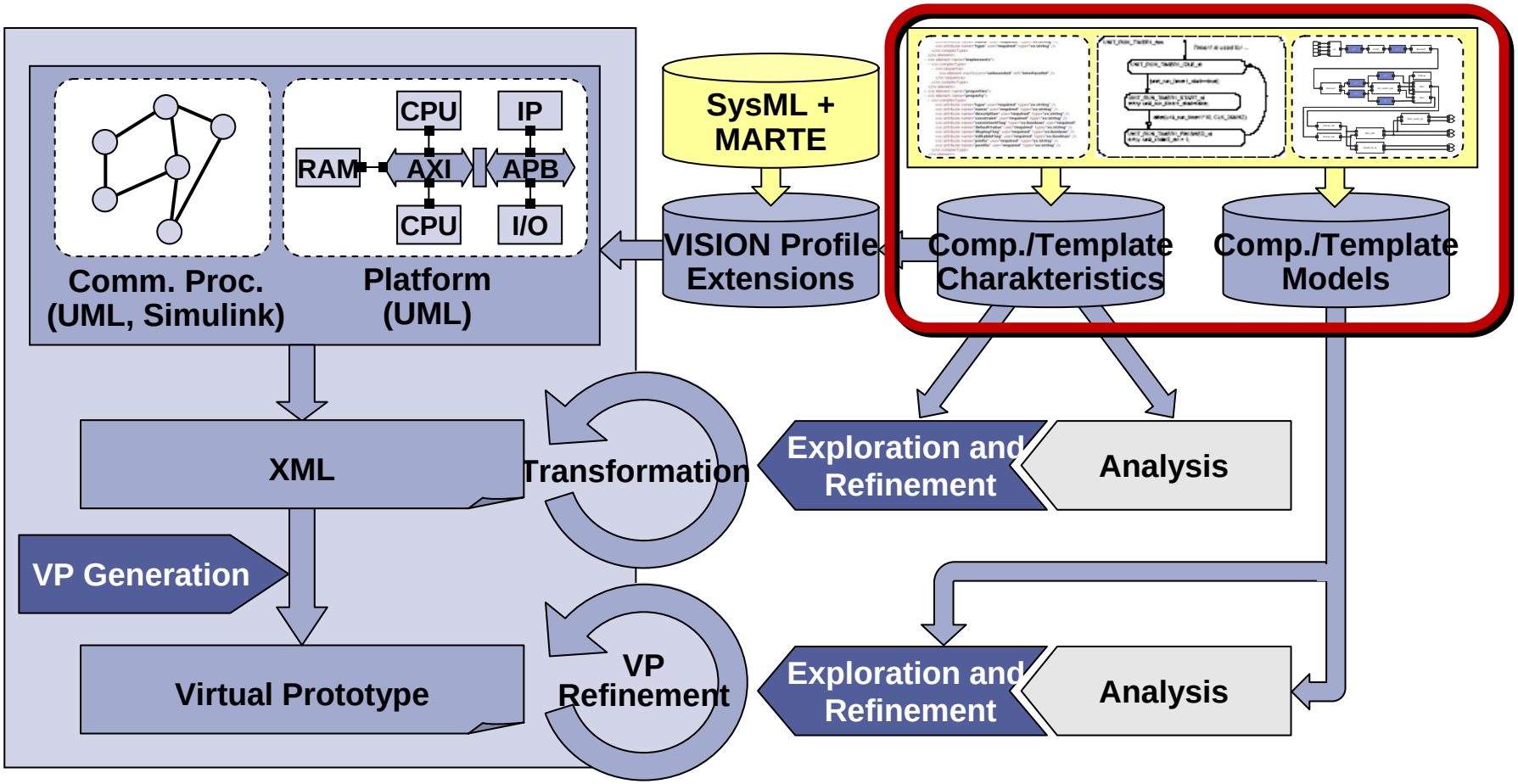
- High flexibility
- Easy software integration
- Modeling at arbitrary levels of abstraction

Recommendations for appropriate usage of the UML

- Do **not** use the UML for
  - Comprehensive IP block modeling including
    - hundreds of registers
    - dozens of RT level signal ports
  - Platform composition at RT level
  - Modeling of cycle-accurate behavioral description of hardware functionality
- Do use the UML for
  - Platform composition at transaction level
  - Incorporation of externally specified component models
  - Mapping of functions onto platform components
  - Refinement by model to model transformation
  - Automatic generation of virtual prototypes

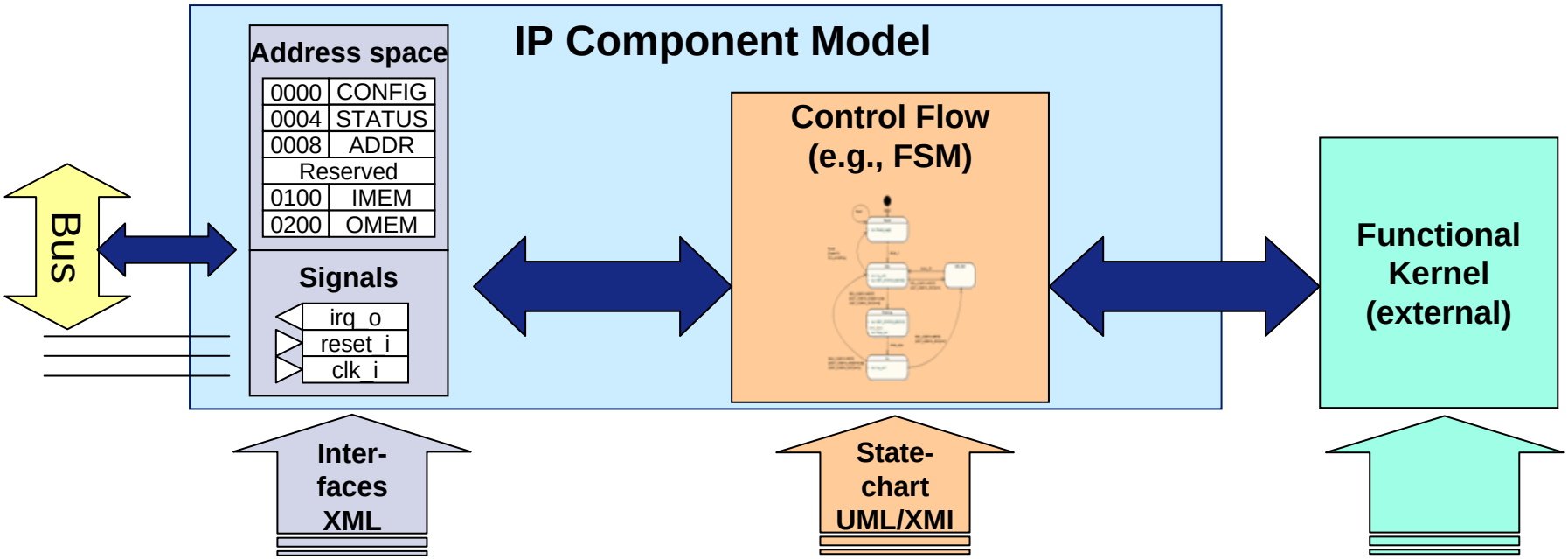


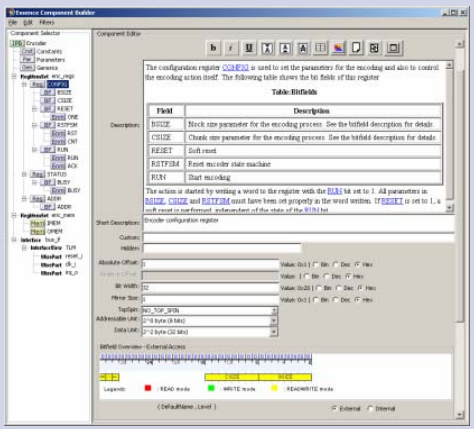
Platform Composition and Refinement



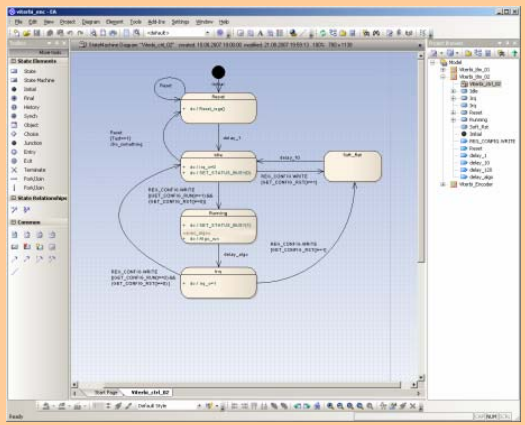
- Modeling techniques providing a holistic system
- Derivation of an optimized network architecture
- Generation of abstract executable models (virtual prototypes)

Component Modeling Methodology

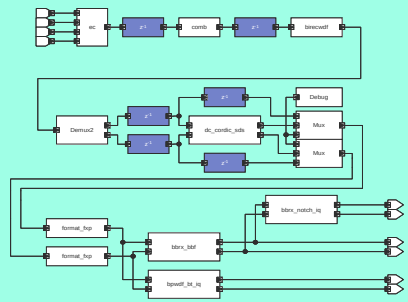




Component Description

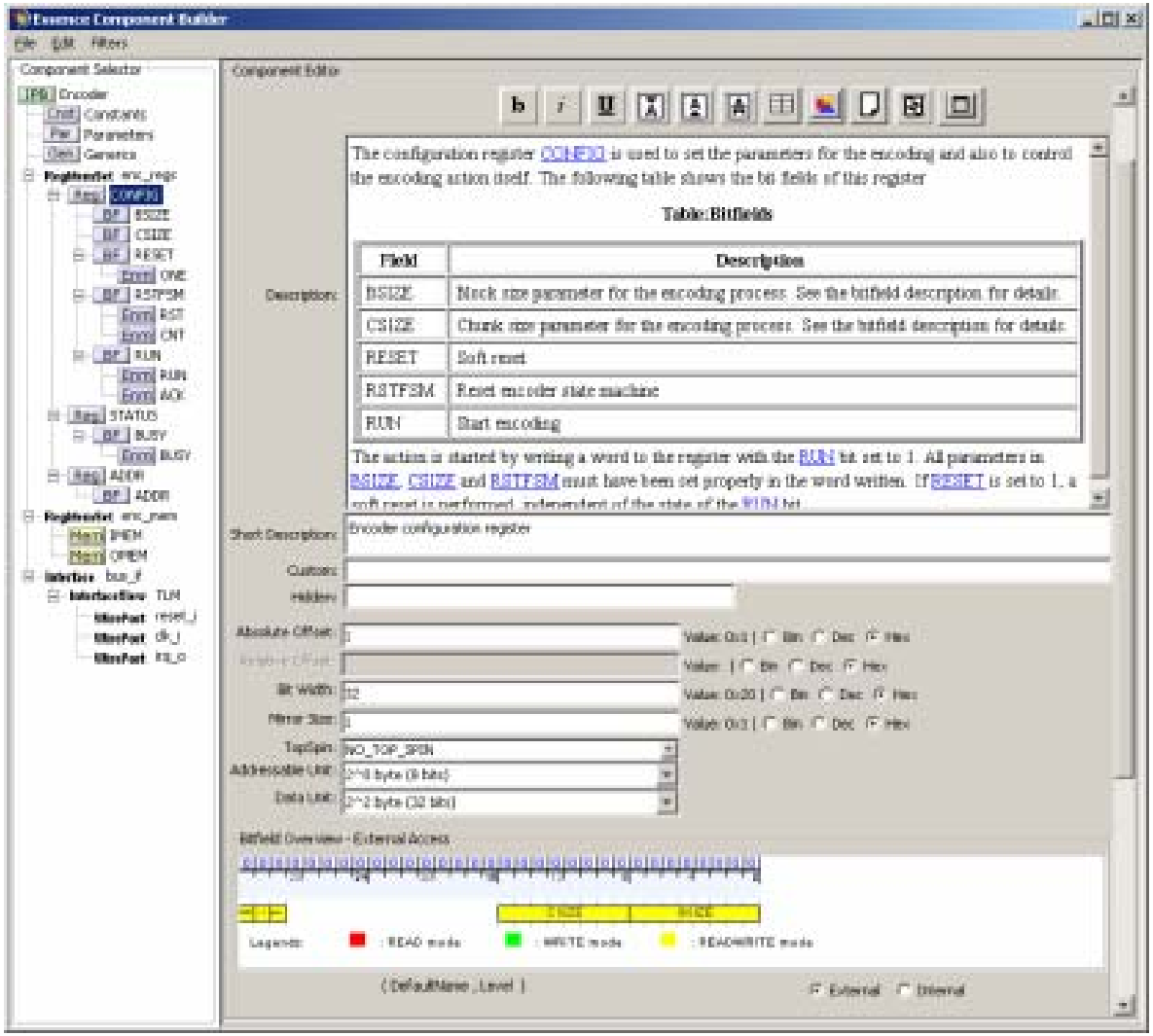


Statechart Model



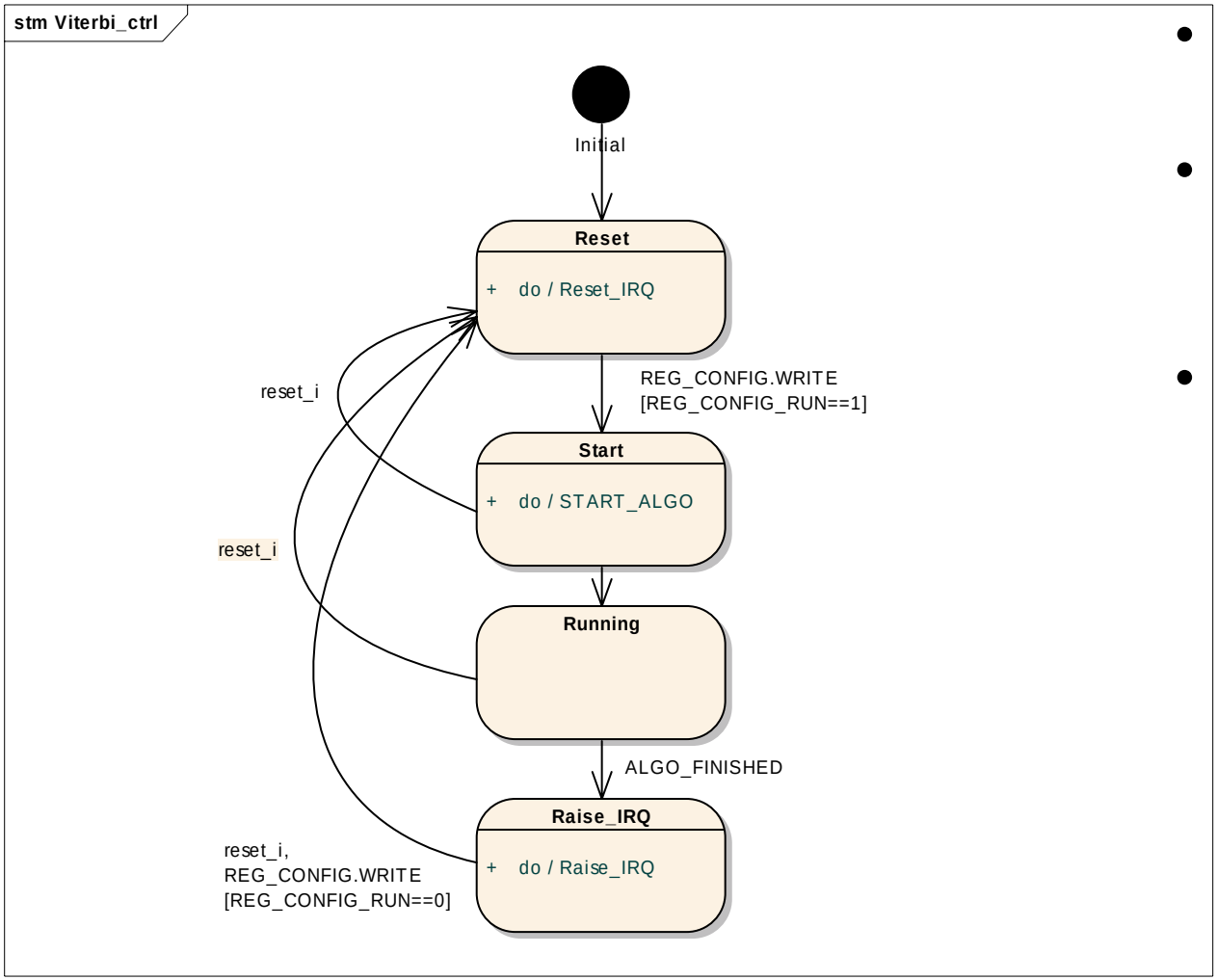
Component Function

Component Specification Using XML



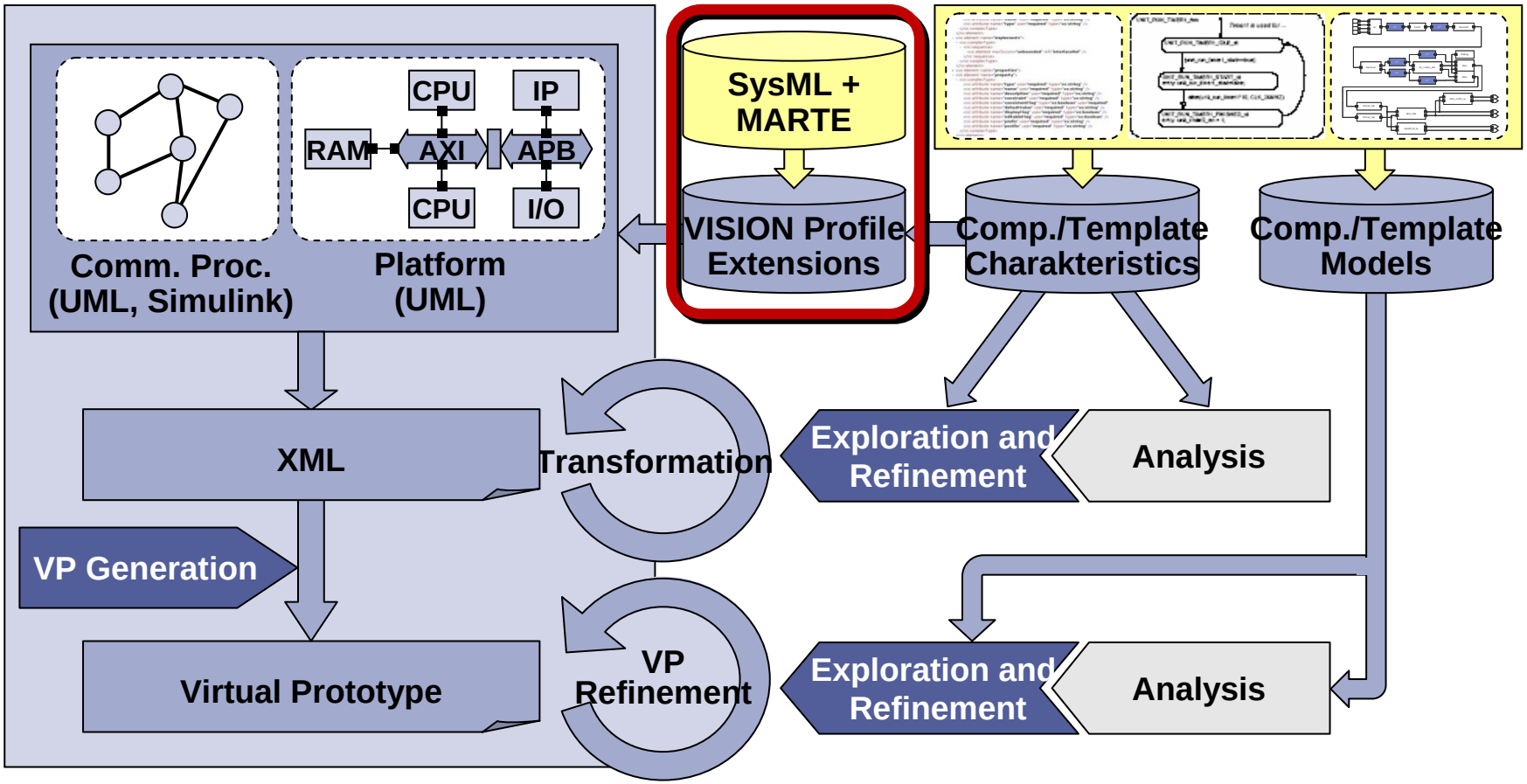
- Interface information in XML format
  - Address spaces
  - Registers
  - Signal ports
- Concept similar to IP-XACT with enhancements
  - Documentation
  - Internal elements
  - Company-specific modifications

Specification of the Control-Flow



- Module control flow: e.g., Statechart
- Triggers:
  - Register accesses
  - Input signals (e.g., reset)
- Output:
  - Algorithm activation
  - Control signals (e.g., irq)

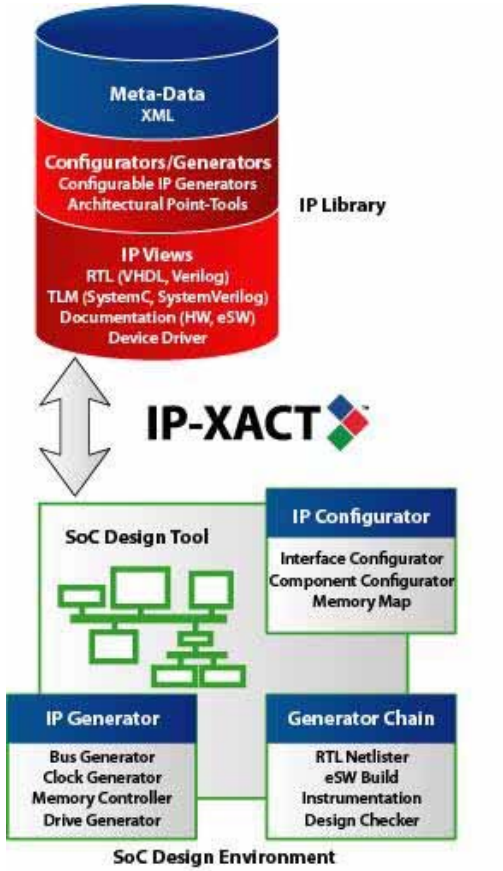
Platform Composition and Refinement



- Modeling techniques providing a holistic system
- Derivation of an optimized network architecture
- Generation of abstract executable models (virtual prototypes)

IP-XACT Interface Specification

- Broad support and acceptance in industry
  - ARM, Infineon, STM, Philips Semiconductor, Cadence, Mentor, Synopsys, …
- XML-based, IEEE standard
- ESL extensions since v1.4
- Basic elements:
  - *component*
    - Description of IP components (interfaces, implementation views, internal channels for bus components, …)
  - *busDefinition*
    - High-level functionality of an interconnect (Protocol, …)
  - *design*
    - Platform design (initialization on components, interconnections of interfaces)
  - *generator integration*
    - Generator API



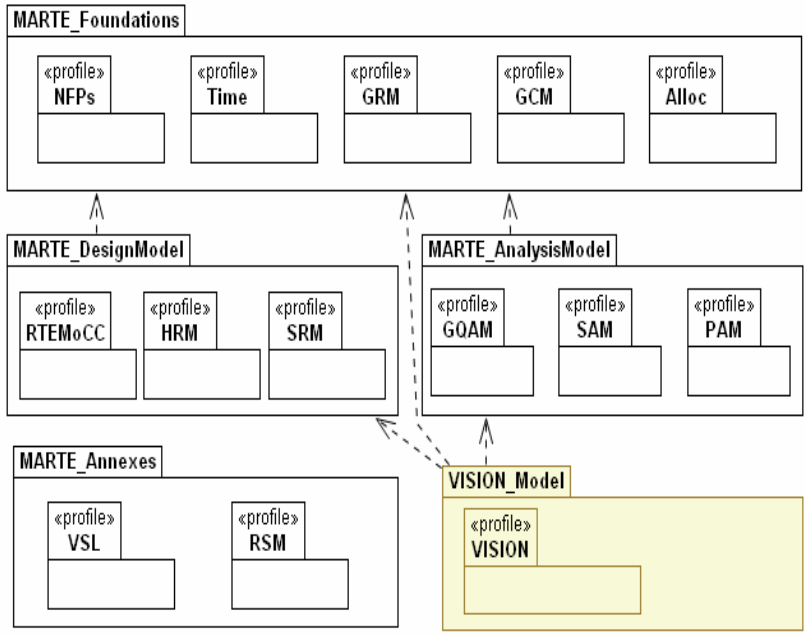
source: SPIRIT Consortium

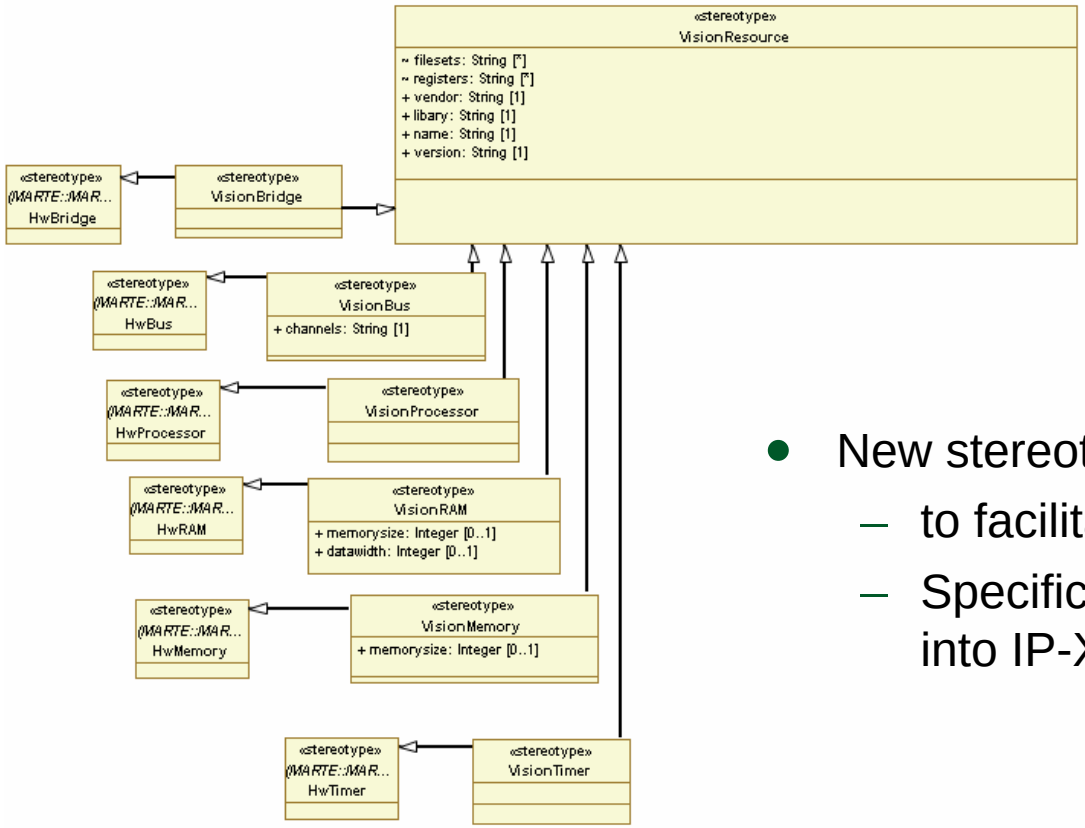
Example: IP-XACT *busDefinition*

```
<?xml version="1.0" encoding="UTF-8"?>
<spirit:busDefinition xmlns:spirit="http://www.spiritconsortium.org/XMLSchema/SPIRIT/1.4" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.spiritconsortium.org/XMLSchema/SPIRIT/1.4
Y:\workspace\GreenSocs\SPIRIT\IP-XACT_ESL_alpha_2\schema\index.xsd">
    <spirit:vendor>GreenSocs</spirit:vendor>
    <spirit:library>busdef.greensocs.greenbus</spirit:library>
    <spirit:name>genericrouter_1.0</spirit:name>
    <spirit:version>1.4</spirit:version>
    <spirit:directConnection>false</spirit:directConnection>
    <spirit:ports>
        <spirit:port>
            <spirit:logicalName>TRANSACTION</spirit:logicalName>
            <spirit:onMaster>
                <spirit:presence>required</spirit:presence>
                <spirit:initiative>requires</spirit:initiative>
                <spirit:protocol>
                    <spirit:type>cable</spirit:type>
                    <spirit:name>generic</spirit:name>
                    <spirit:parameters>
                        <!-- add set/get-access to master and get-access to slave/system-->
                        <spirit:parameter spirit:name="MAddr" spirit:format="long"/>
                        <spirit:parameter spirit:name="MCmd" spirit:format="choice" spirit:choiceRef="MCmdType"/>
                        <spirit:parameter spirit:name="MID" spirit:format="long"/>
                        <spirit:parameter spirit:name="MBurstLength" spirit:format="long"/>
                        <spirit:parameter spirit:name="MData" spirit:format="string"/>
                    </spirit:parameters>
                </spirit:protocol>
            </spirit:onMaster>
            <spirit:onSlave>
                <spirit:presence>required</spirit:presence>
                <spirit:initiative>provides</spirit:initiative>
                <spirit:protocol>
                    <spirit:type>cable</spirit:type>
                    <spirit:name>generic</spirit:name>
                    <spirit:parameters>
                        <!-- add set/get-access to slave and get-access to master -->
                        <spirit:parameter spirit:name="SData" spirit:format="string"/>
                        <spirit:parameter spirit:name="SResp" spirit:format="choice" spirit:choiceRef="SRespType"/>
                    </spirit:parameters>
                </spirit:protocol>
            </spirit:onSlave>
        </spirit:port>
    </spirit:ports>
```

VISION Profile Extension

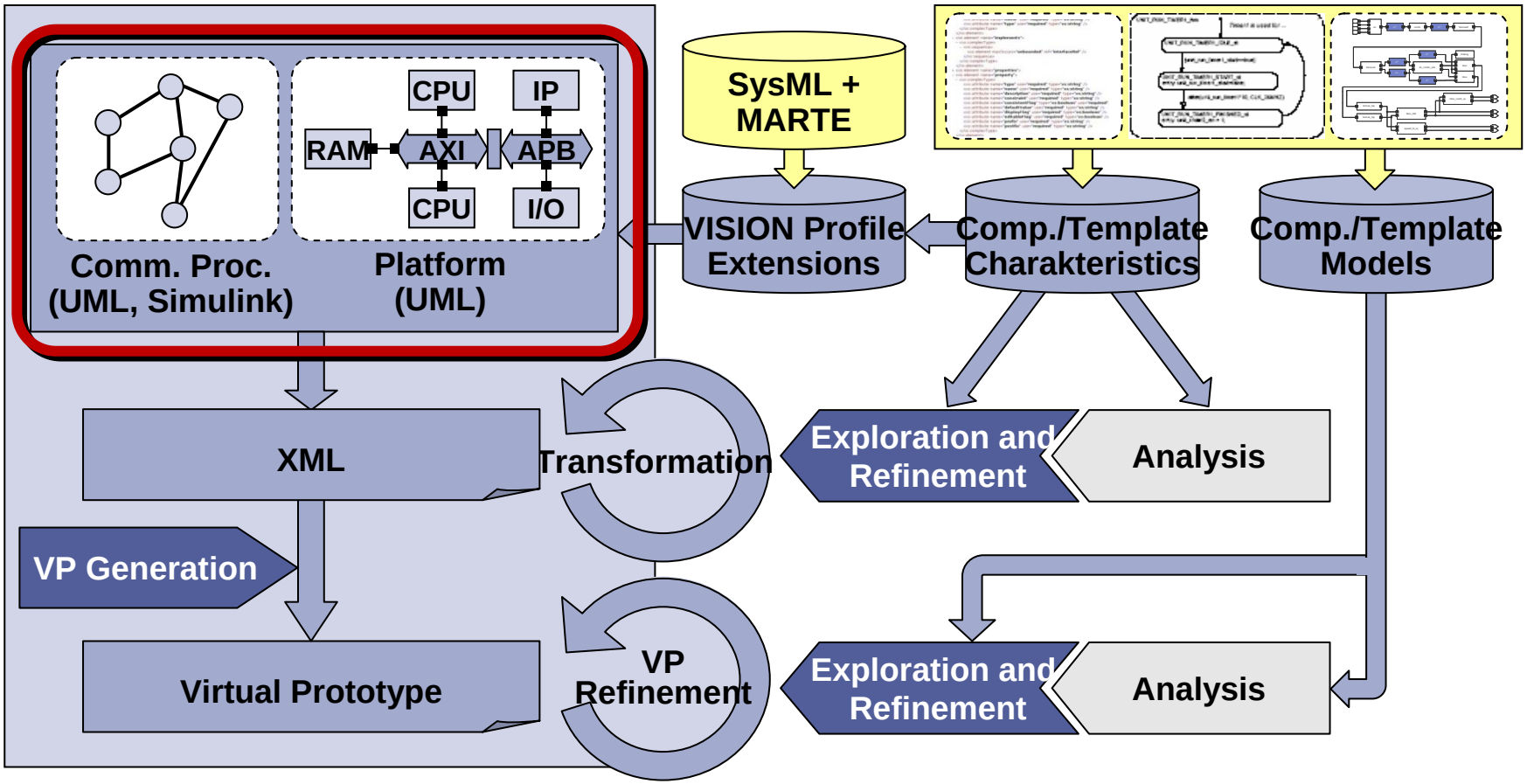
- Extension of SysML and MARTE for flexible platform composition
  - by use of the VISION component model
  - which is based on the IP-XACT component model





- New stereotypes
  - to facilitate SoC platform composition
  - Specific attributes for back transformation into IP-XACT have to be taken into account

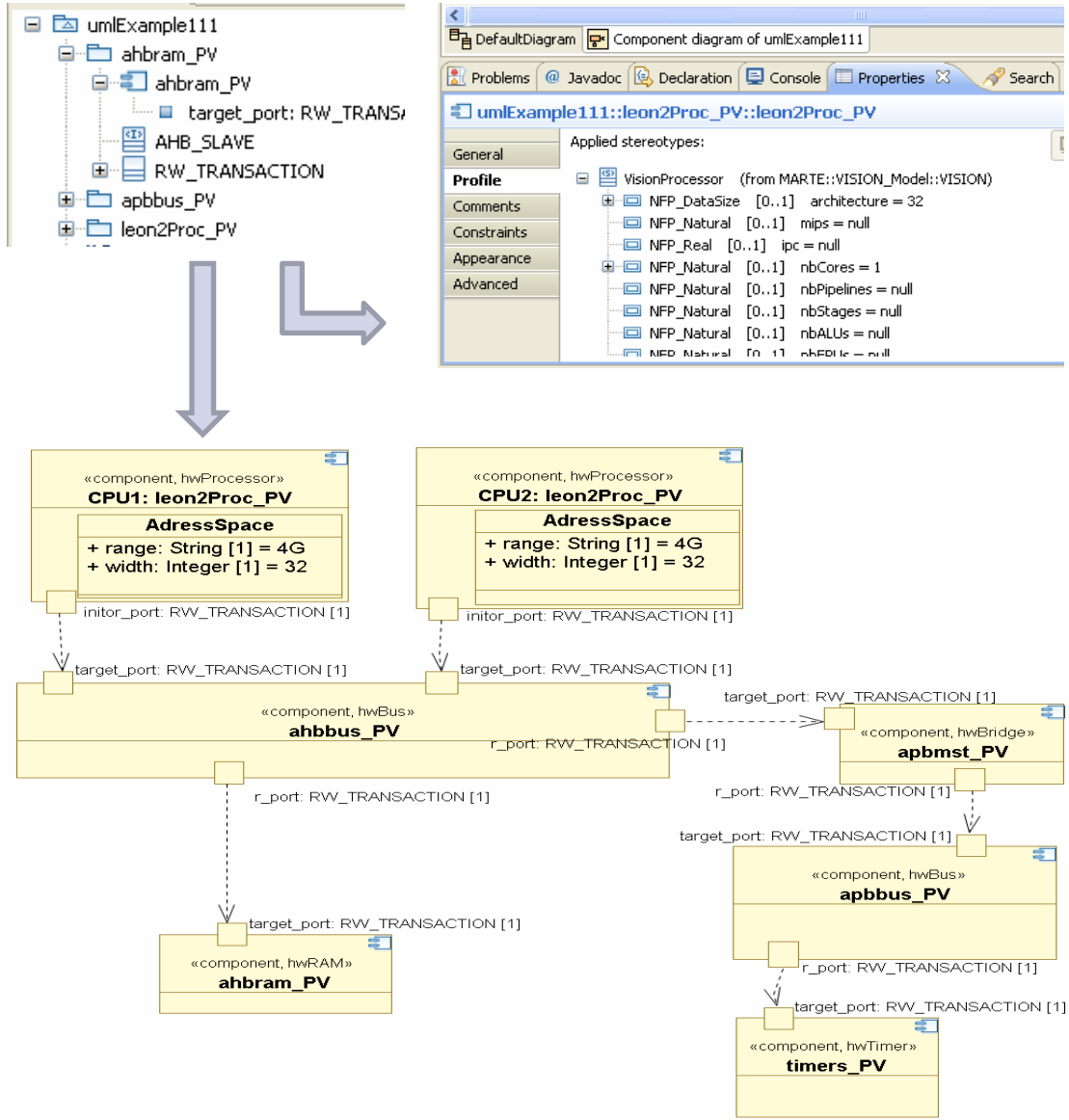
Platform Composition and Refinement



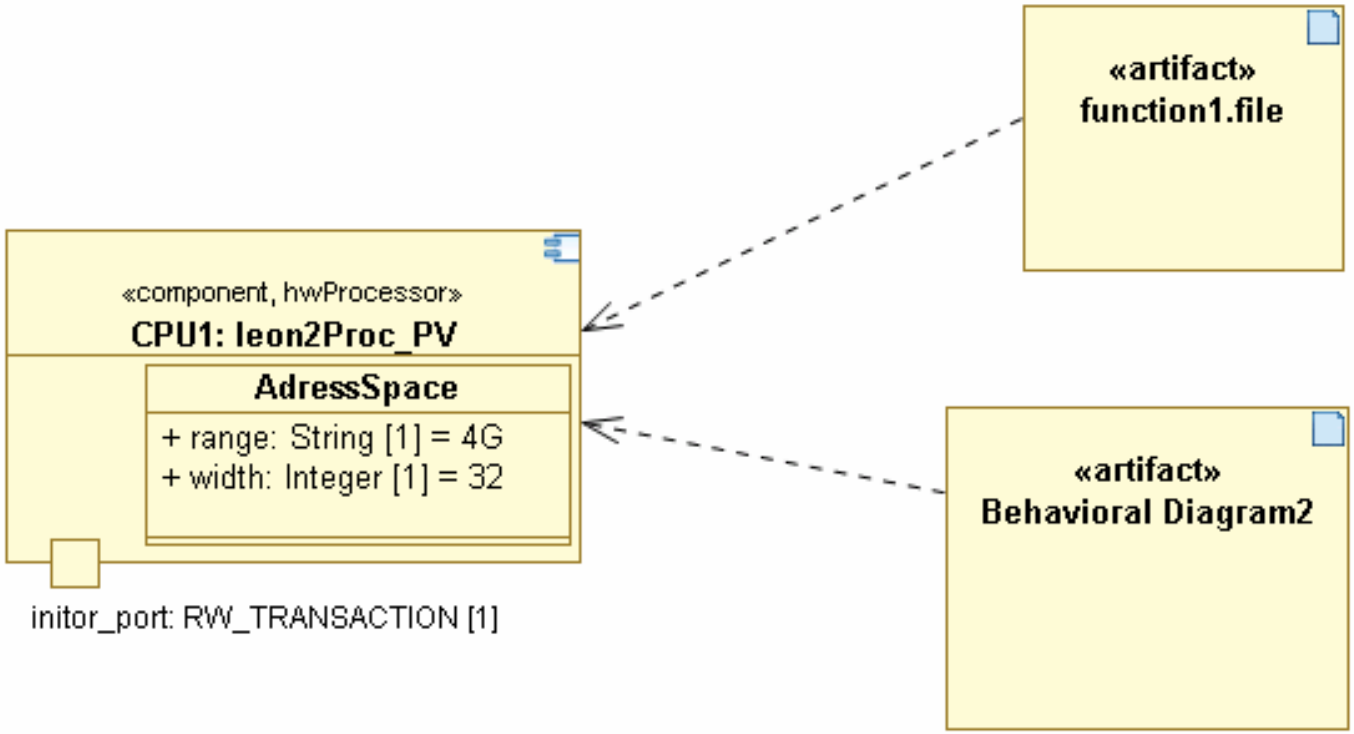
- Modeling techniques providing a holistic system
- Derivation of an optimized network architecture
- Generation of abstract executable models (virtual prototypes)

Platform Composition (Abstract)

- Flexible model-based platform composition using the VISON component model
- Import and export for IP-XACT-based IP libraries and platform templates
- „Drag-and-Drop“ platform composition
- Modeling at multiple levels of abstraction
  - “Un-typed” modeling:
    - Easy and fast
    - w/o consideration of interfaces, protocols, etc.
  - “Typed” modeling:
    - Consideration of interfaces and protocols
    - Supporting semi-automated refinement

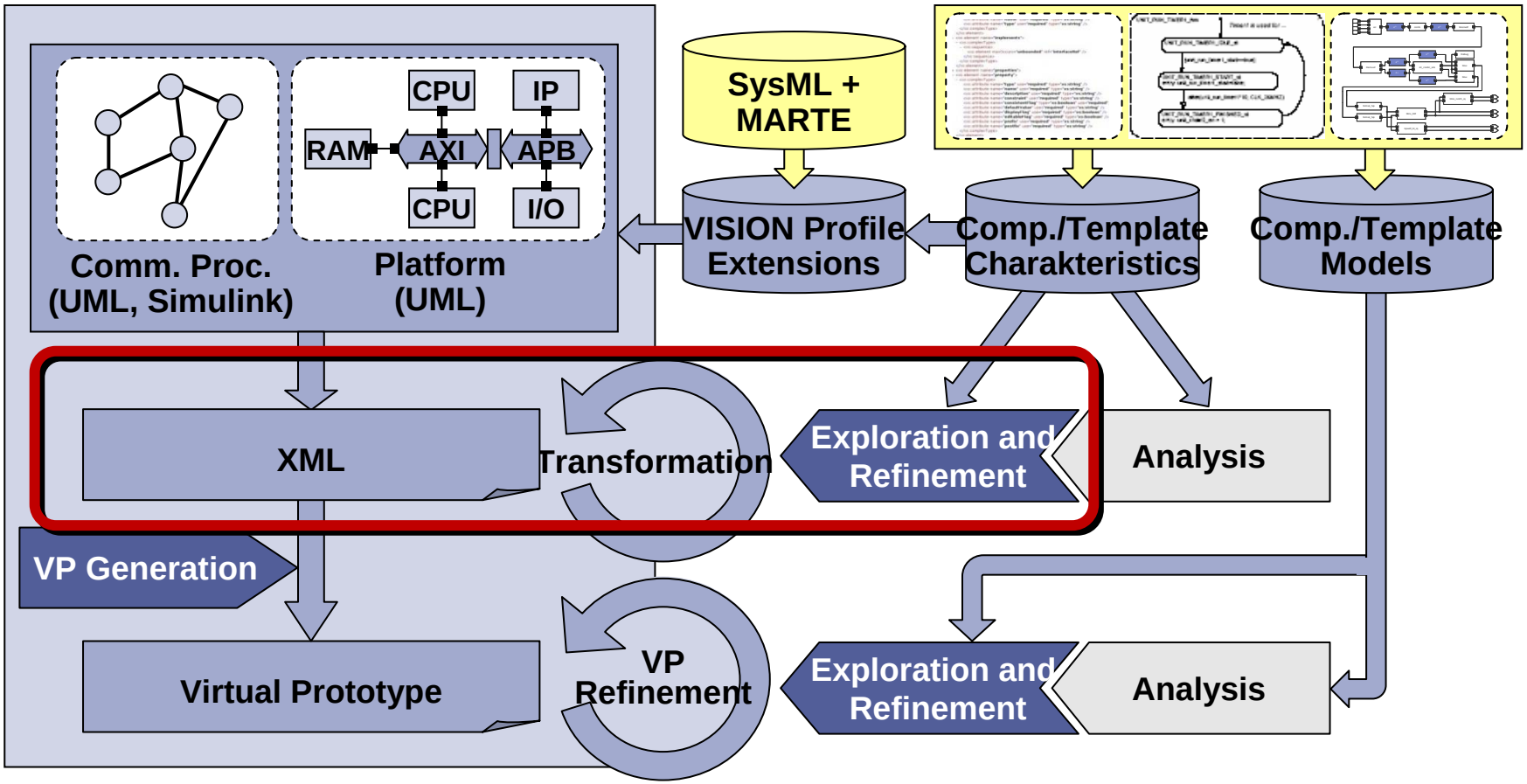


Mapping Software onto HW Platform Components



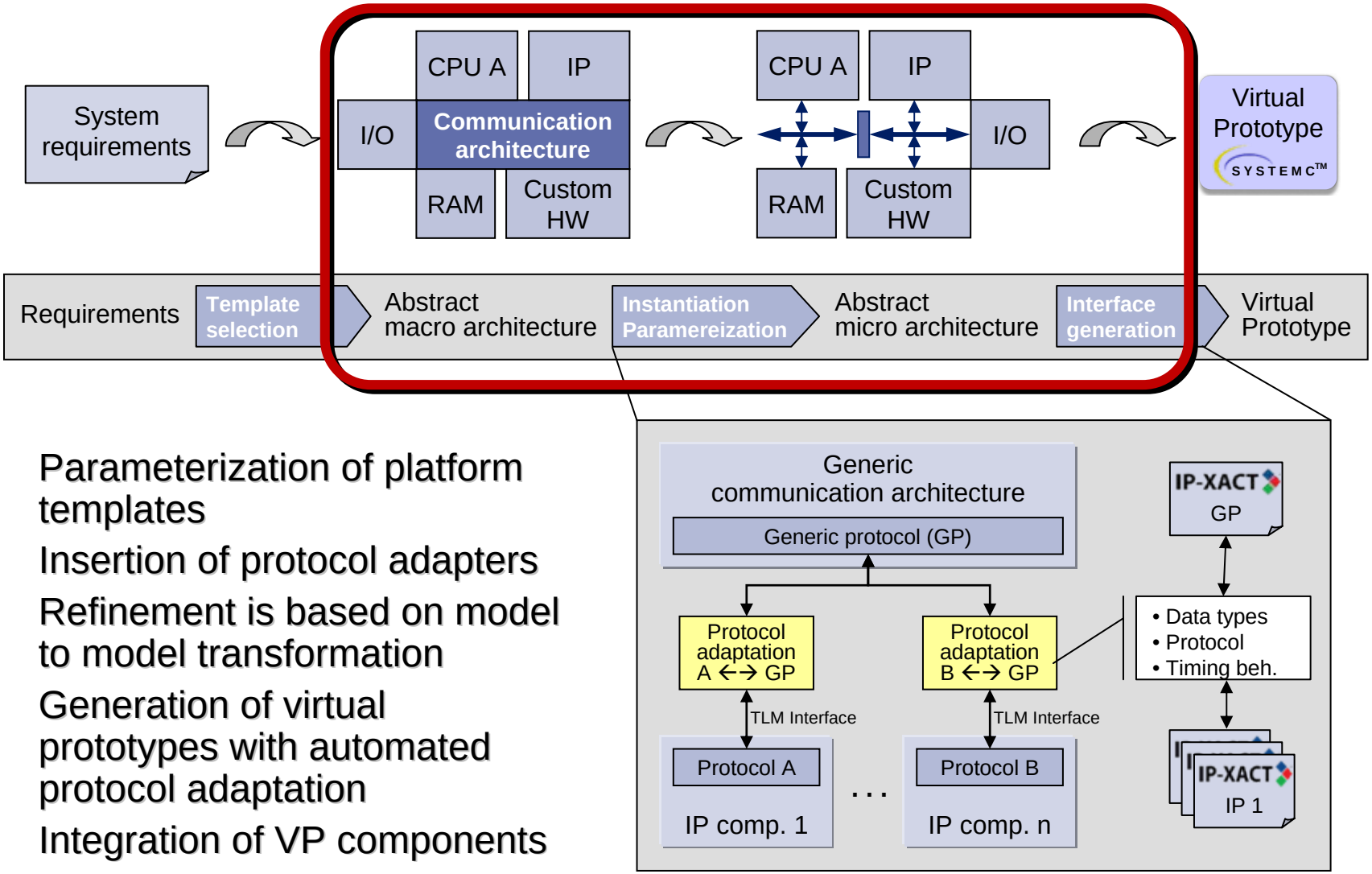
- Software mapping (deployment) is modeled using artifacts
  - Mapping of C/C++ source code
  - Mapping of UML behavior diagrams

Platform Composition and Refinement

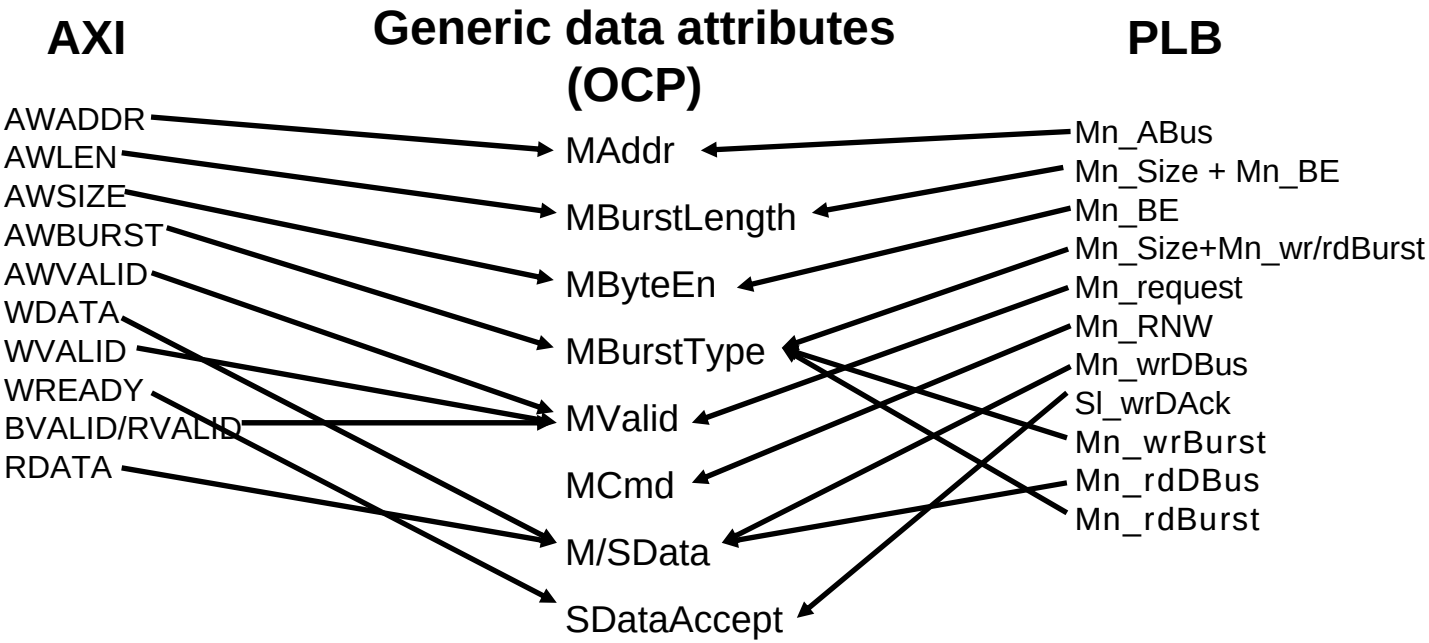


- Modeling techniques providing a holistic system
- Derivation of an optimized network architecture
- Generation of abstract executable models (virtual prototypes)

Platform Refinement

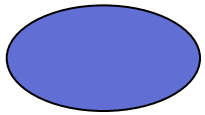


- Parameterization of platform templates
- Insertion of protocol adapters
- Refinement is based on model to model transformation
- Generation of virtual prototypes with automated protocol adaptation
- Integration of VP components



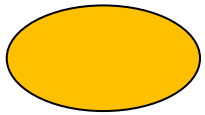
- Mapping of component specific protocol parameters onto the generic attributes of the interconnect
- Automatic type conversion or stub generation for manual conversion of complex data types
- In some cases few manual steps are needed for complete mapping (no specific attribute) or attribute generation (no generic attribute)

Mapping Specific to Generic Protocols

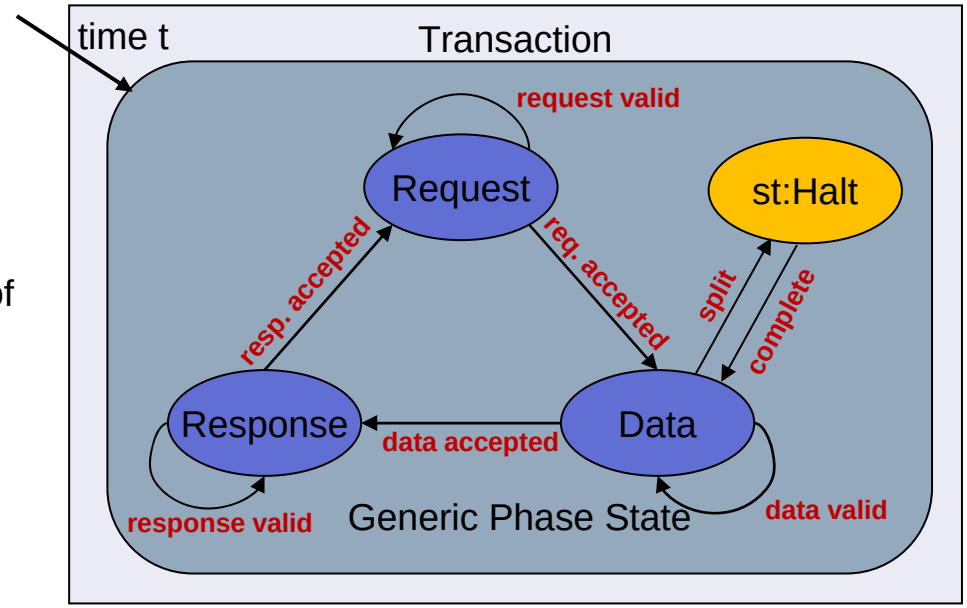


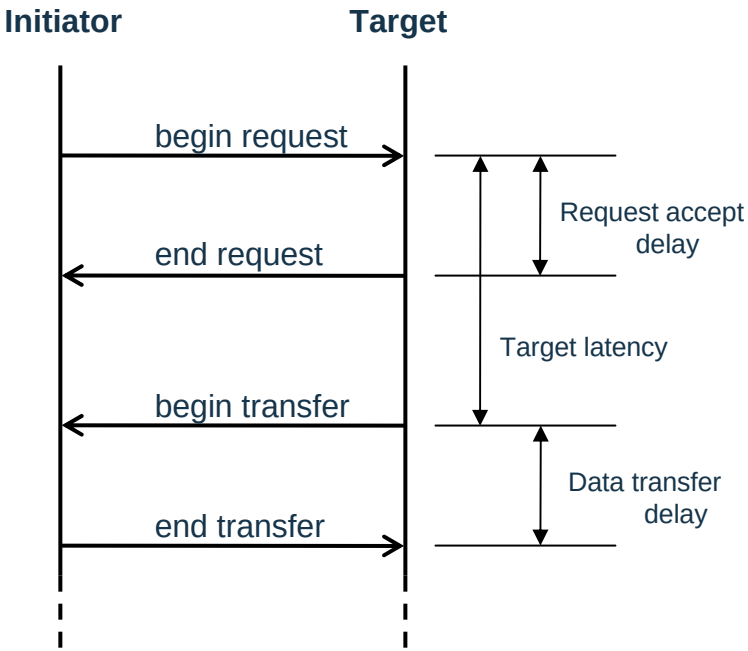
Phase states of the generic protocol

Phase transition by specific protocol



Additional phase state of specific protocol



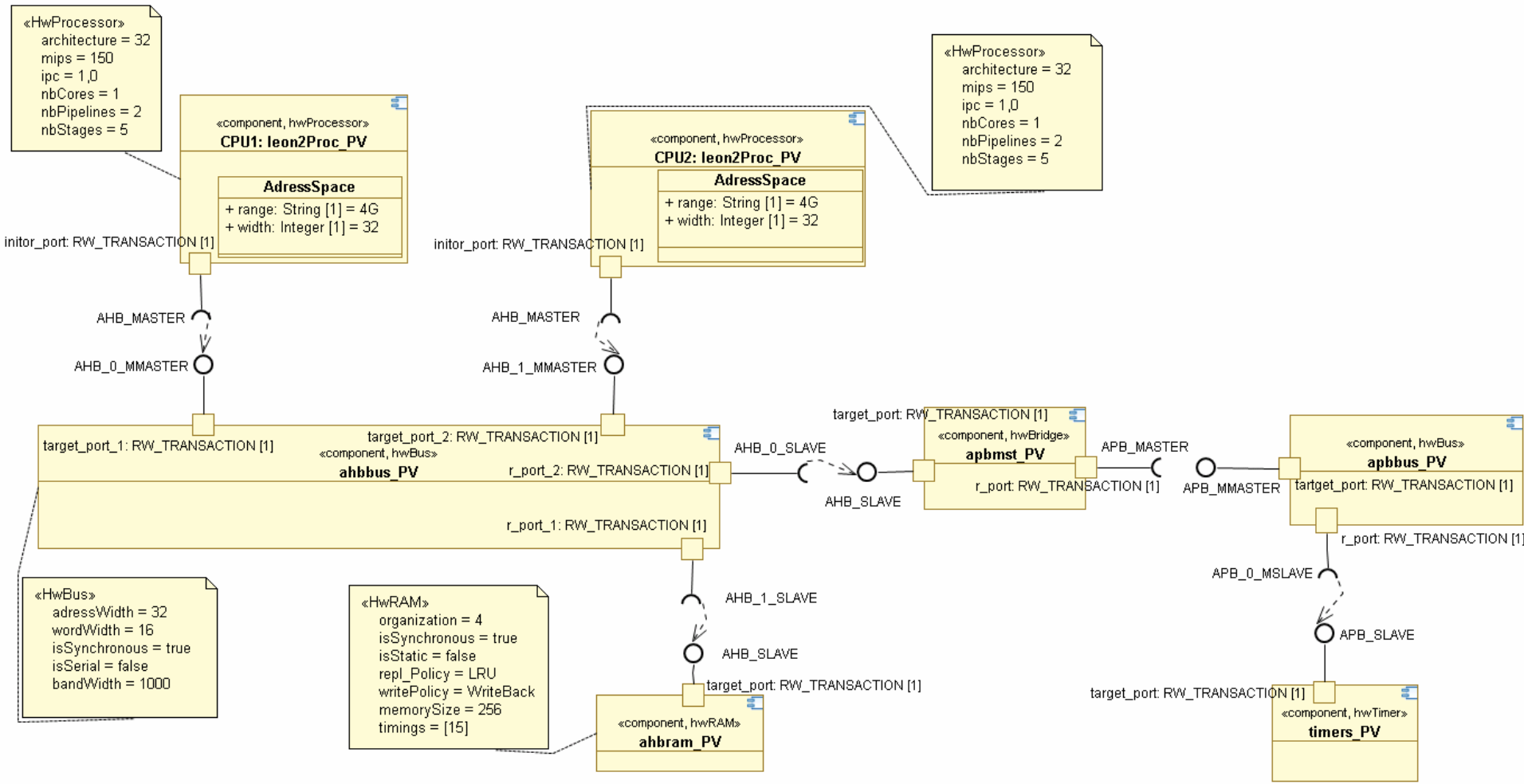


Assertion-like protocol behavior specification

property Transaction
router r = {cycleTime ct, pipelineDepth pd, …}, globalTime t,
communicationDelay d, requestTimeOut rto, …
Request(r,t,rto) |-> Data(r,t,d) |-> …
endproperty

sequence Request (router r, time t, timeOut m)
#1 (initiator.request_valid) (r.enqueue(Transaction))
#1 (target.request_accepted@[t+1;t+m];timeout(t+m+1))
(r.dequeue(Transaction));
endsequence

Results: Refined Typed Platform



Results: Traffic Sign Recognition

Results: Traffic Sign Recognition

Conclusion

- Holistic modeling of interconnected microelectronic systems needs a combination of different specification models
- Application specific XML editor to describe the component characteristics (address spaces, registers, ports, …)
- Transformation of IP-XACT component descriptions into UML components
- Platform composition by use of component diagrams
- Performing platform refinement by automated protocol adaptation *“from un-typed platform models towards typed platform models”*
- Automated generation of virtual prototypes using SystemC

Thank you very much for your attention!

Questions?

Oliver Bringmann
FZI Forschungszentrum Informatik
Microelectronic System Design
E-Mail: bringmann@fzi.de
Web: www.edacentrum.de/vision