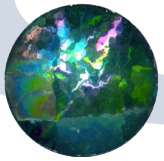
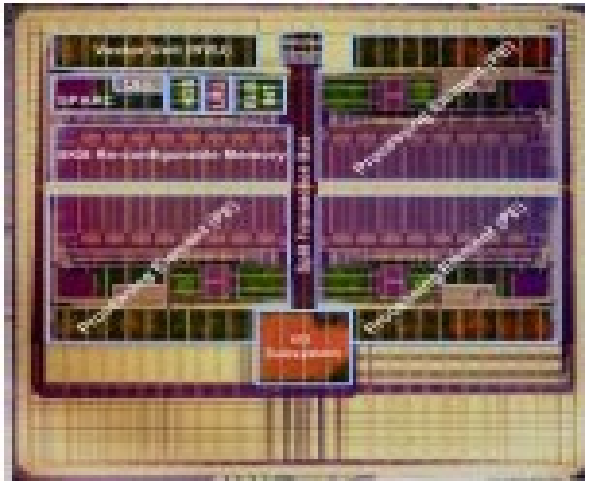


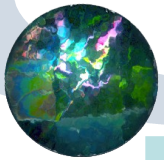
Co-model for co-design de UML à SystemC par transformations de modèles

jean-luc dekeyser



Dart Inria Project :

West High Performance **E**mbedded **S**ystem Design **T**urning point



Vector / SIMD

Automatic parallelization

Monte Carlo
Finite elements

Out Of Order

Languages/compilation

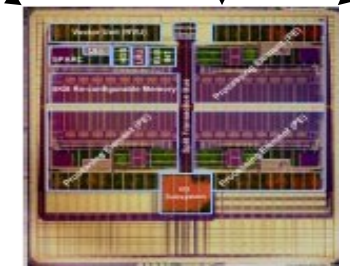
Architectures

Algorithms

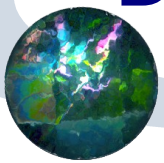
Meta models
UML profile
Mapping / scheduling

Co-design
Simulation
Synthesis
MppSoC

Data/control flow
Signal processing
Finite elements
ES 4 transport



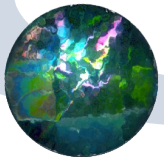
DaRT permanent members today



Boulet Pierre	professor (USTL)
Dekeyser Jean-Luc	professor (USTL)
Dumoulin Cédric	assistant professor (USTL)
Etien Anne	assistant professor (USTL)
Gamatié Abdoulaye	CR (CNRS)
Marquet Philippe	assistant professor (USTL)
Meftali Samy	assistant professor (USTL)
Niar Smaïl	HdR assistant professor (UVHC)

Thématique et Contexte Scientifique

Co-model for Co-design: projet INRIA Dart

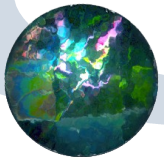


- Parallélisme de données dans des systèmes embarqués
- Une approche modèle : un profil UML de co-design de System on Chip
  - Vérifiable : Modèles formels (ou semi-formels)
  - Simulable : Co-simulation en SystemC
  - Exécutable : Transformations, génération de code
  - Synthétisable : SystemC/VHDL pour FPGA

SoC future is parallel!

- Parallelism exists in applications
  - Multimedia/Telecom/Detection system
- Need for efficiency
  - Power/Speed/Cost,
  - Design Time/Customization,
- Economical Reason
  - Developing cost
  - Manufacturing Cost
- All together implies Re-Usability needs

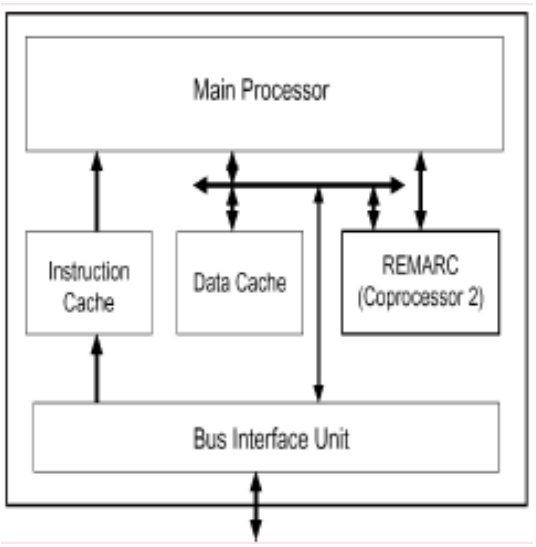
Regular Architecture:Examples

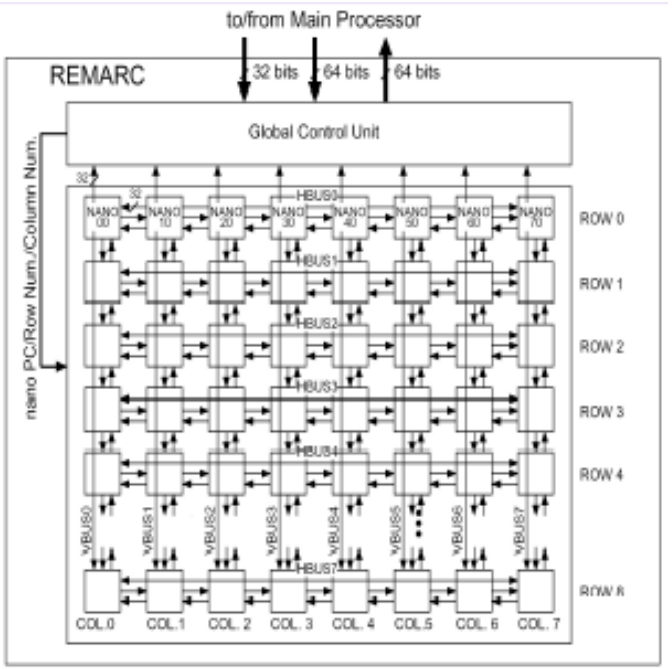


- Processor Array.
  - PicoChip(430 processors)
  - D-Fabrix (Toshiba MeP RISC processor based)
  - PACT XPP(64 ALU-PAEs with 8 × 8 array)
  - BRESCA from Silicon Hive(Philips)(4×4 VLIW PEs)
  - QuickSilver´s
  - ACM architecture (64 Arithmetic(RISC)cluster)
- FPGA
- DP-FPGA
- SIMDVLIW
  - XeTaL (320 PEs)
  - TriMedia (5-way VLIW processor)

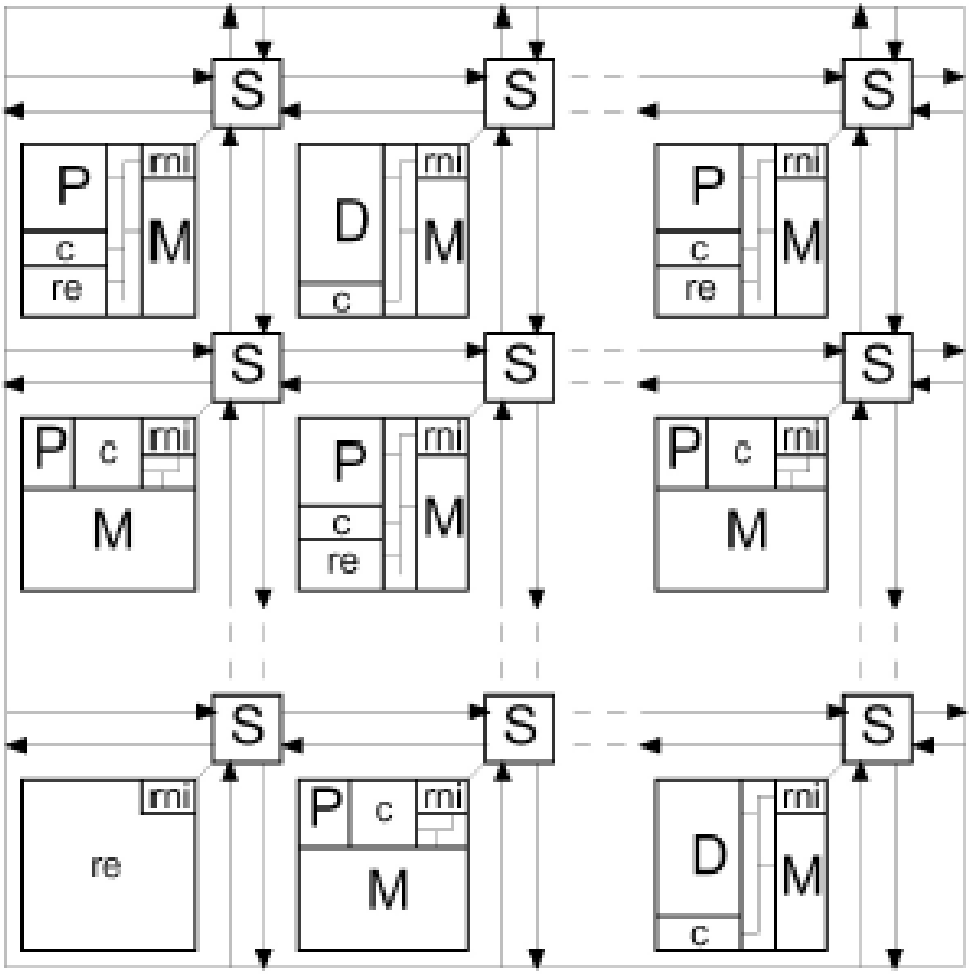
REMARC: Reconfigurable Multimedia Array Co-Processor

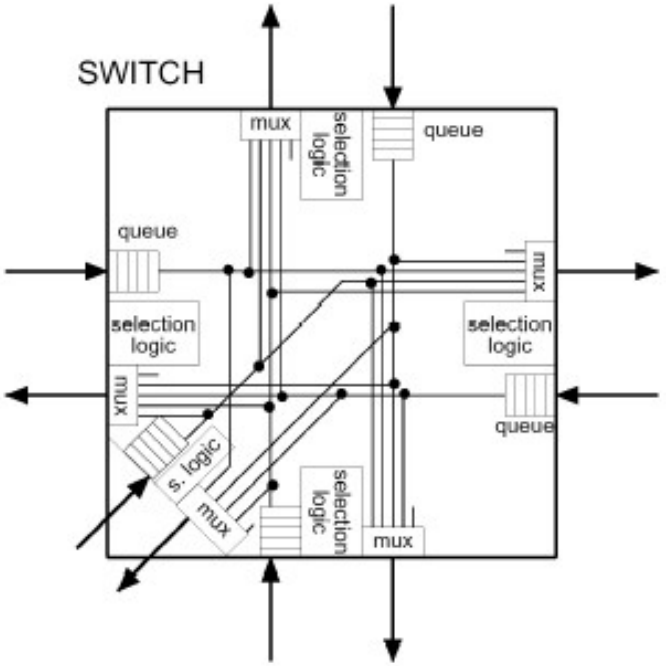
- *(a) Block Diagram of a microprocessor with REMARC*
- *(b) Block Diagram of REMARC(Nano-processor)*



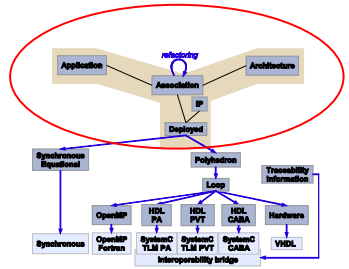


A General NoC Architecture





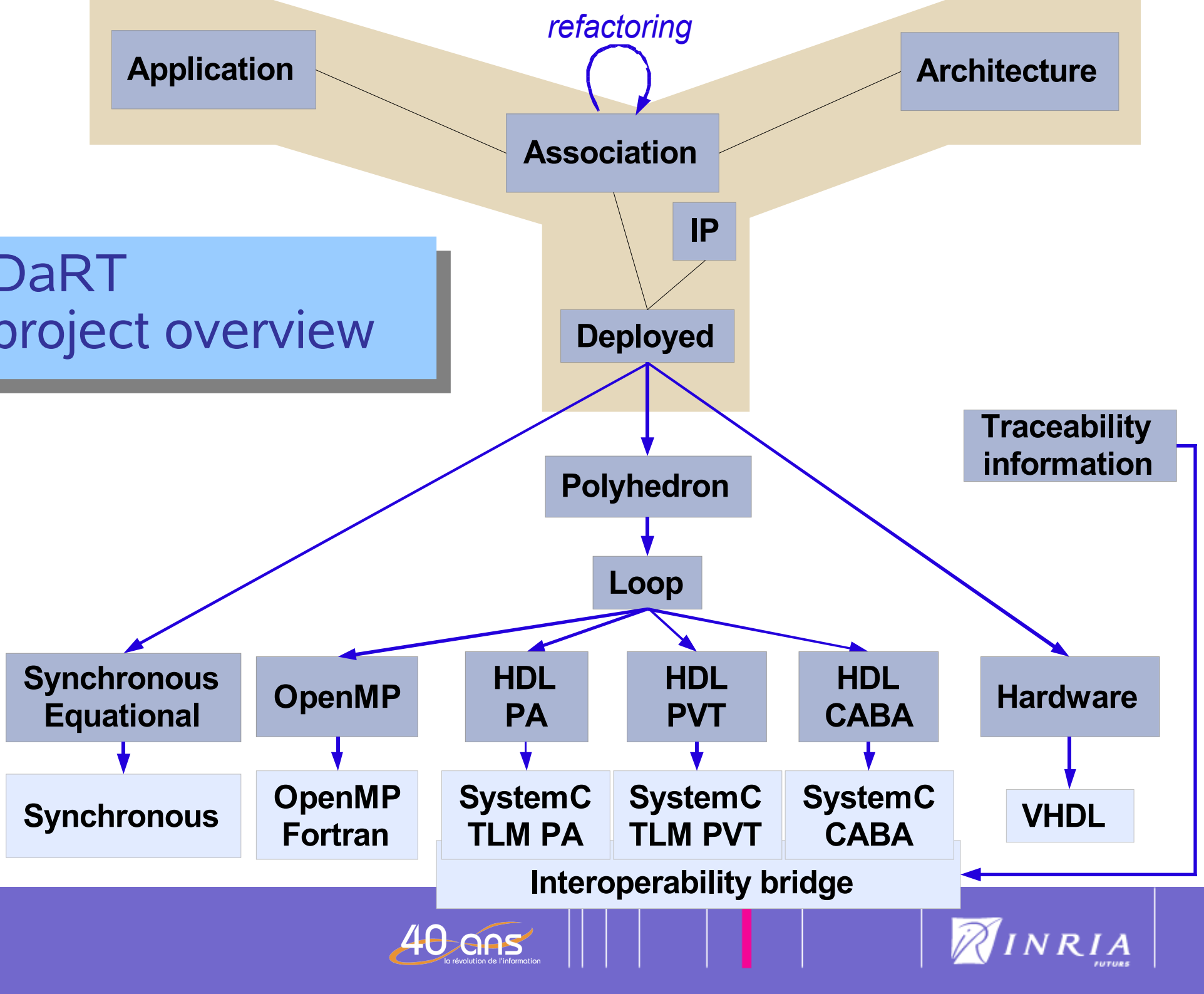
Co-model for co-design



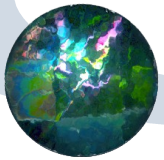
Metamodels for co-design

- High level data parallel constructions
  - Hierarchical
  - Repetitive
  - Application, architecture and association models
- Iterative dependency expression
- Data flow and control flow mixing

DaRT project overview

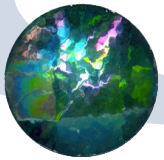


Principes de conception



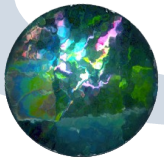
- Une architecture matérielle
  - Blocs standards (CPU, mem)
  - Blocs spécifiques
  - Bus de communication
- Des ressources logicielles
- SoC = cohabitation de ces ressources sur un même chip, prise en compte globale pour la réalisation hard/soft

Notion d'IP (Intellectual Property)



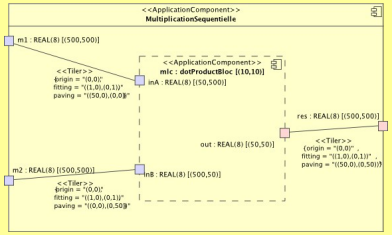
- Blocs fonctionnels complexes réutilisables
  - Hard: déjà implanté, dépendant de la technologies, fortement optimisé
  - Soft: dans un langage de haut niveau (VHDL, Verilog, C++…), paramétrables
- Normalisation des interfaces (OCP)
- Environnement de développement (co-design, co-specif, co-verif)
- Bloc réutilisable
  - connaître les fonctionnalités
  - estimer les performances dans un système
  - être sûr du bon fonctionnement de l'IP
  - intégrer cet IP dans le système
  - valider le système

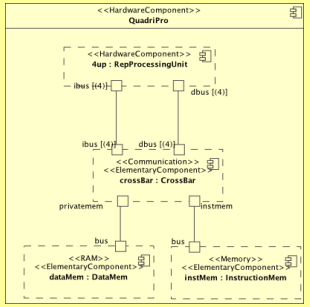
Méthodologies de conception

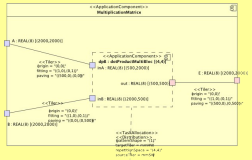


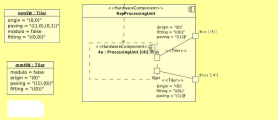
- Procédure pour concevoir un système.
- Comprendre une méthodologie aide à garantir la sécurité de la conception.
- Flot de conception : Σ de compilateurs, outils de développement logiciel, outils de conception assistée par ordinateur (CAD), etc., permettant de:
  - aider à automatiser les étapes de la méthodologie;
  - garder trace de l'application de la méthodologie (gestion de version, rapports, accélération des itérations).

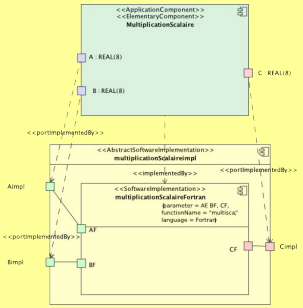
Models

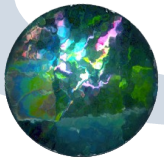

<<ApplicationComponent>>
MultiplicationSequentielle
m1 : REAL(8) [(500,500)]
<<Tiler>>
{origin = "(0,0)"
fitting = "((1,0),(0,1))"
paving = "((50,0),(0,0))"
<<ApplicationComponent>>
mlc : dotProductBloc [(10,10)]
inA : REAL(8) [(50,500)]
res : REAL(8) [(500,500)]
out : REAL(8) [(50,50)]
<<Tiler>>
{origin = "(0,0)" ,
fitting = "((1,0),(0,1))" ,
paving = "((50,0),(0,50))"}
m2 : REAL(8) [(500,500)]
<<Tiler>>
{origin = "(0,0)"
fitting = "((1,0),(0,1))"
paving = "((0,0),(0,50))"
inB : REAL(8) [(500,50)]


<<HardwareComponent>>
QuadriPro
<<HardwareComponent>>
4up : RepProcessingUnit
ibus [(4)]
dbus [(4)]
ibus [(4)]
dbus [(4)]
<<Communication>>
<<ElementaryComponent>>
crossBar : CrossBar
privatemem
instmem
bus
bus
<<RAM>>
<<ElementaryComponent>>
dataMem : DataMem
<<Memory>>
<<ElementaryComponent>>
instMem : InstructionMem



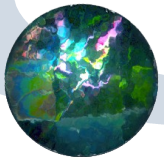



<<ApplicationComponent>>
<<ElementaryComponent>>
MultiplicationScalaire
A : REAL(8)
B : REAL(8)
C : REAL(8)
<<portImplementedBy>>
<<AbstractSoftwareImplementation>>
multiplicationScalaireImpl
<<implementedBy>>
<<portImplementedBy>>
AImpl
<<SoftwareImplementation>>
multiplicationScalaireFortran
{parameter = AF, BF, CF,
functionName = "multiScal"
language = Fortran}
<<portImplementedBy>>
AF
CF
CImpl
BImpl
BF



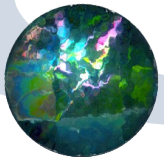
Ingénierie dirigée par les modèles

Définition



- Méthode:
  - technique de résolution de problème caractérisée par un ensemble de règles bien définies qui conduisent à une solution correcte
- Méthodologie:
  - un ensemble structuré et cohérent de modèles, méthodes, guides et outils permettant de déduire la manière de résoudre un problème
- Modèle:
  - une représentation d'un aspect partiel et cohérent du «monde» réel
  - précède toute décision ou formulation d'une opinion
  - est élaboré pour répondre à la question qui conduit au développement d'un système

Rôle d'un modèle pour les systèmes



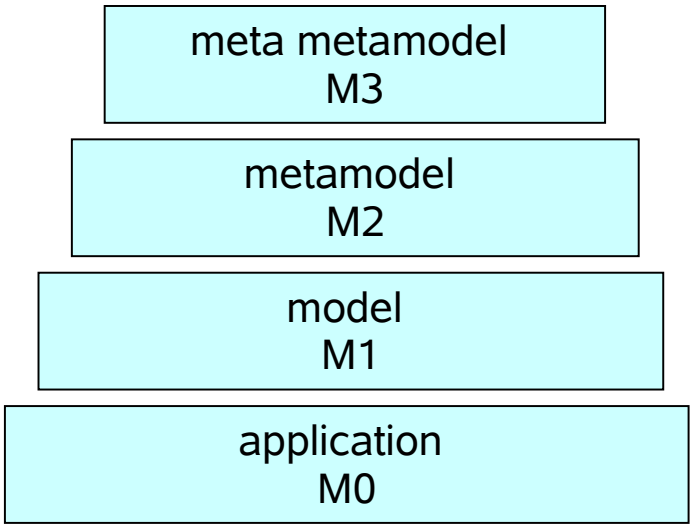
- Abstraction
  - Eliminer des détails, focaliser sur un point de vue du système
  - Travailler à différentes échelles de complexité et de temps
- Analyse
  - Etude des propriétés du modèle (vérification de propriétés)
  - Extrapolation au système réel représenté
- Communication
  - Discussion et échanges avec d'autres personnes
  - Echanges entre outils
- Génération/Production
  - Produire une représentation d'un autre niveau (autre modèle)
  - Produire le système réel
- => Modèle à retenir: fonction de l'objectif visé

MDE/MDA Focus

- Proposes
  - *Increase the reuse* of existing developments
  - *Reduce the time to market*
  - *Increase the lifetime* of current and future developments
  - *Ease the integration* of new technologies with long proven business models
- Means
  - *Clear separation*
    - Of the *fundamental logic* of the specification
    - From the particular *implementation technologies*

Model and Metamodel

- Meta-metamodel
  - Described with the MOF (Meta Object Facility)
  - Provides XMI production rules, JMI specification, …
- **Metamodel**
  - Can be seen as a “language” definition:
    - Available modeling elements
    - Construction rules
- **Model**
  - Follow the rules expressed in the “language”
  - Describe an application
- **Application**
  - Concrete realization of a model
  - Example : generated code



Problematics

- Design application and hardware architecture
  - with the same language
- Have different levels of abstractions
- Reuse/perpetuate developed models
  - with actual and upcoming tools (simulation or synthesis)
- Provide the same modeling environment for the whole co-design process
  - possibly supporting a visual specification.
- Benefit from standard formats for exchange and storage.
- Be able to go from one level to another
  - transformation rules from model to model.

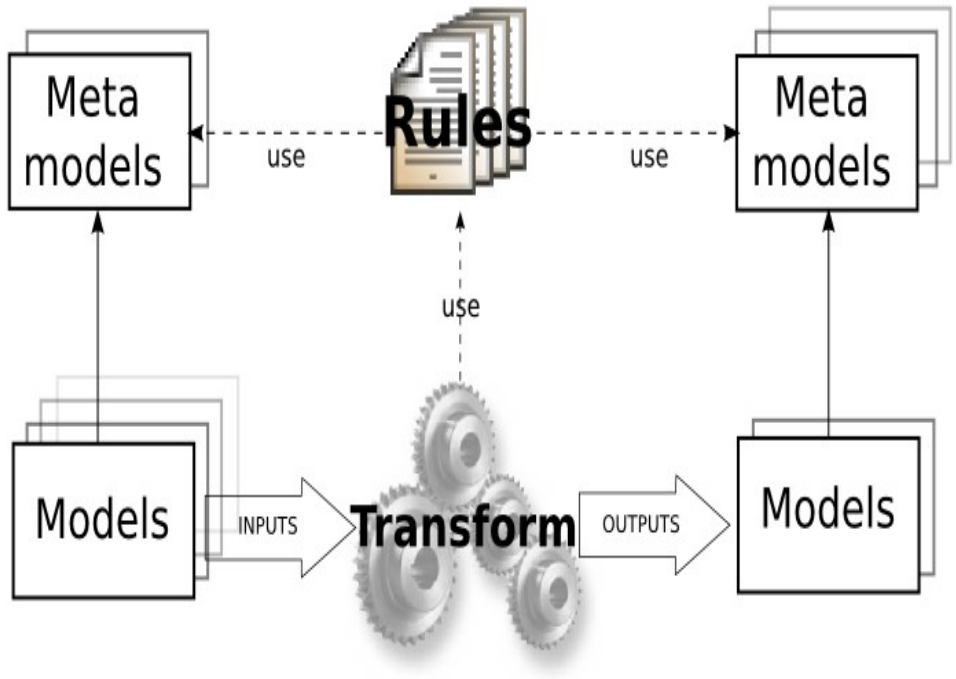
Model Driven Engineering (MDE)

- Model <=> abstract view of reality
- Metamodel <=> language used to describe the model.
- MDE development process:
  - Goes from a high level of abstraction ...
  - ... to low levels / technological levels of abstraction ...
  - ... through intermediate levels of abstraction.
- The high level models contain only domain specific concepts
- Targeted levels / technological levels are for:
  - code generation, simulation, verification, ...
- Smooth introduction of the technological concepts in the intermediate levels.

Model Transformation: principle

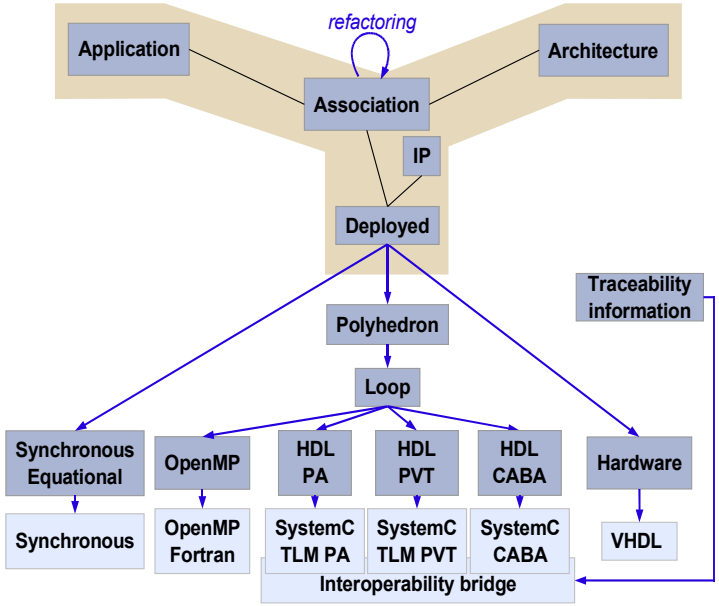
- Transformations is used to :
  - Go from one model at a given abstraction level ...
  - ... to another model at another level

- Definition based on metamodels
- Execution on models
- Decomposition of the transformati into rules
  - Describes what should be transformed into what
  - Divide and Conquer approach



MDE in the DaRT project

- Models are everywhere:
  - application, architecture, association, deployment,
  - technologies,
  - targeted languages
- Different targeted languages for different purposes
- Intermediate models
- Model to model transformations
- Model to code generations



MDE is the Gaspard skeleton

MDE results

- 3 Operational transformations chains:
  - towards VHDL
  - towards Synchonous
  - towards OpenMP

40.000 lines of code

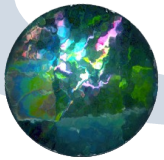
- 5 MDE tools
  - ModTransfL (Model Transformation Language)
  - Gaspard
  - MoMoTE (Model to Model Transformation Engine)
  - MoCodE (Model to Code Engine)
  - TrML (Transformation Modeling Language)
- Organizer of IDM 06 in Lille

Traceability

- Introduction of traceability in model transformations
  - used to know what have been transformed into what
  - a trace model is generated along with model transformations.
- Trace is used to generate an interoperability bridge model
  - Definition of metamodels for traceability and interoperability bridge
  - Code generation for interoperability from the bridge model

Related Works

- QVT specification = OMG standard
- Several transformation engines defined as part of the QVT proposals:
  - ATL, (INRIA ATLAS)
  - Kermeta, (INRIA Triskell)
  - Fujaba (graph community), ...
- Projects
  - OpenEmbedD (ANR project)
  - ModEasy (Interreg III)
- Conferences
  - Models
  - ECMDA
  - IDM (French)



Profils UML pour l'embarqué

UML pour l'embarqué : état des lieux
SysML
UML for SoC
RFP MARTE

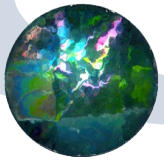
ECSI Workshop on UML…

- **SysML** (Wolfgang Mueller, C-Lab) .
- **MARTE** (Laurent Rioux, Thales)
- **UML for SoC** (Sreeranga P. Rajan, Fujitsu)
- UML for SystemC (Sara Bocchio, Alberto Rosti, ST & Elvinia Riccobene, U Milan, P Scandurra, U Catania)
  - A UML 2.0 profile of the SystemC language for use in UML structural and behavioral diagrams to model the functionality expressed by processes and channels in a SystemC specification.
- MARTE and UML for SoC - Architecture models and platform representations (Pierre Boulet, LIFL, France).
- Models of Computation (Fernando Herrera, UCantrabria, Spain) : the detection of needs and issues to connect UML profiles with heterogenous specification methodologies used for implementation of embedded systems.
- Use of UML2.0 standard extension mechanism for component-based design flow compatible with commercial UML 2.0 tools (Tero Tangas, TUT, Finland)

....continued

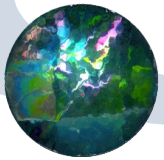
- **Nokia** (Tero Tangas , TUT) : demonstrating the Koski design flow by the implementation of full featured WLAN terminal implementation starting from a behavior entirely specified in UML2.0.
- **Thales** (TBD) : prototyped engineering solutions for distributed and heterogenous embedded systems
- **ST Microelectronics** (Alberto Rosti) : link between UML specification and SystemC TLM
- **Airbus** : the TOPCASED integrated development process from system specification to product architecture.
- **Mentor Graphics** (Thomas Ulber) : Why Systems-on-Chip needs more UML like a Hole in the Head - Executable and Translatable UML (XtUML), a selected subset of UML to support the needs of execution- and translation based development for system partitioning, automatic hardware/software interface generation and system integration.
- **CoFluent** (Vincent Perrier) : Secure prediction of behavior and performance through system-level modeling and simulation from partial hardware and software using model-driven architecture design approaches for system architecting and timed-behavioral modeling.
- **SparxSystems** (Peter Lieber) : Enterprise Architect and the embedded world A generic UML profile mechanism for loading and working with different profiles. UML profiles are specified in XML files allowing to create own profile to describe modelling scenarios peculiar to your development environment
- **Artisan SW** (Olivier Casse) and Telelogic/I-Logix are well-known providers of UML modelling tools for modeling for real-time embedded systems modeling and software engineering, and provide support for SysML.

UML 2.1
http://www.uml.org/



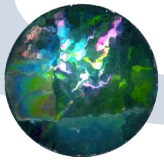
- adopté fin 2005 / 2.1.1 le 3 fevrier 2007
  - superstructure finale (définition des diagrammes)
  - infrastructure (classes de base UML 2.0 et MOF 2.0), OCL, format d'échange de diagrammes en cours de finalisation
- nouveautés par rapport aux versions 1.x
  - plus d'abstraction
    - séparation claire de la structure (statique) et du comportement (dynamique)
    - nouveau diagrammes: structure, activité + langage d'action
  - plus d'automatisation
    - précision de la sémantique
    - format d'échange de diagrammes (XMI 2.0)

Profils dédiés aux systèmes embarqués et au temps réel



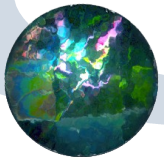
- SPT : Scheduling, Performance and Time
  - profil pour UML 1.x
  - http://www.uml.org/
- SysML : UML for Systems Engineering RFP en cours de définition pour UML 2.0
  - http://www.sysml.org/
- UML for SoC
  - proposition d'un consortium japonais
  - concepts très proches de SystemC
  - RFC en cours d'acceptation
- MARTE : UML Profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE) RFP

SysML



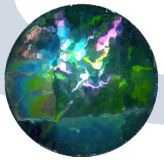
- langage complet étendant UML
  - peu de modification structurelles d'UML
  - profil définissant les concepts nécessaires pouvant être spécialisé pour différents domaines
- étendue
  - specification, analysis, design, verification and validation of a broad range of complex systems
  - that may include hardware, software, information, processes, personnel, and facilities
- unification des différents langages de modélisation utilisés dans l'ingénierie des systèmes
- SysML to SystemC (FDL07....)

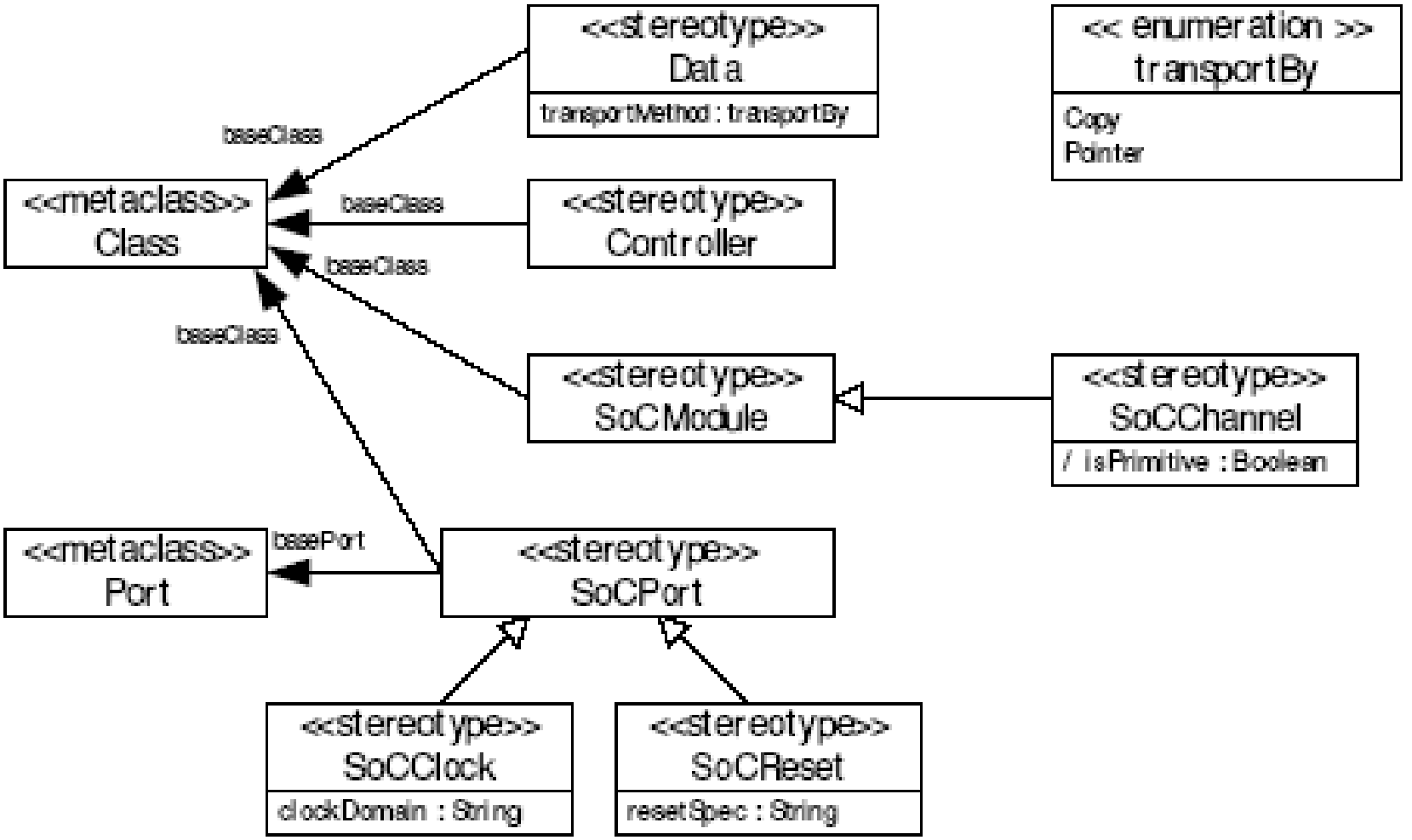
UML4SoC: Hardware Modeling



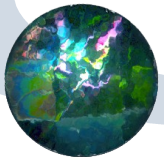
- concepts very close to SystemC
  - allow automatic SystemC code generation
- abstraction level
  - from transactional level modeling (TLM)
  - to register transfer level (RTL)
- mainly uses the structure (or assembly) diagram

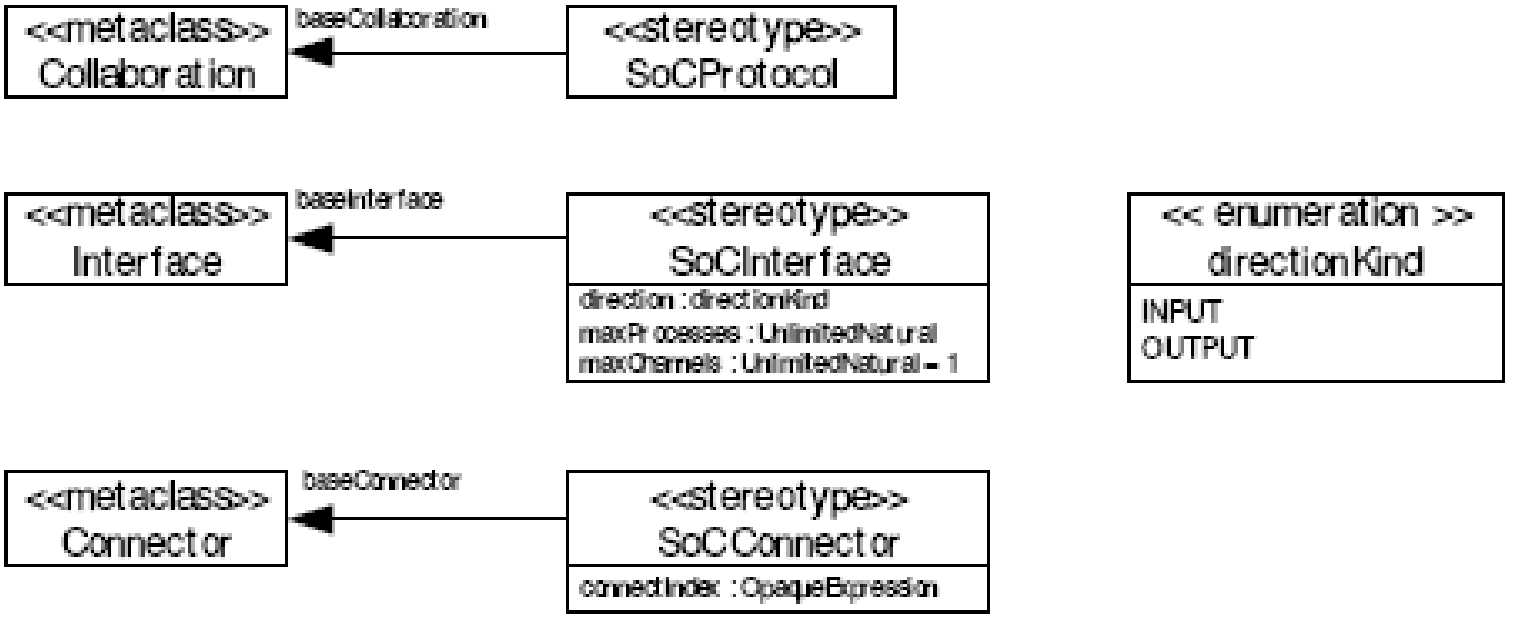
Stereotypes
Structural Modeling





Stereotypes Communication Modeling





Modeling and Analyzis of Real-Time Embedded systems

- Standard UML profile proposal to the OMG
- Status
  - Public draft of revised submission available
  - "Vote-to-vote" in June 07
- Submission team
  - Alcatel, Lockheed Martin, **Thales**
  - Artisan, IBM, Mentor Graphics, Softeam, Telelogic, Tri-Pacific
  - Carleton Univ., **CEA**, ESEO, ENSEITA, **INRIA (Aoste, DaRT, Espresso)**, INSA Lyon, Carnegie Mellon Univ.

Profile Structure (1/2)

- PART I – MARTE foundations
  - Core Elements
  - *Non-Functional Property Modeling*
  - Time Modeling
  - General Resource Modeling
  - *General Component Model*
  - Allocation modeling
- PART II – MARTE design model
  - RTE Model of Computation & Communication
  - Detailed Resource Modeling
    - Software Resource Modeling
    - *Hardware Resource Modeling*
-

Profile Structure (2/2)

- PART III – MARTE analysis model
  - Generic Quantitative Analysis Modeling
  - Schedulability Analysis Modeling
  - Performance Modeling
- PART IV – Annexes
  - *Value specification language*
  - Clock handling facilities
  - MARTE model libraries
  - Repetitive structure modeling

Problematics

- Reuse
  - Component-based design
  - Separation of concepts (application, architecture, association)
- Computation intensive applications
  - Parallelism in applications
    - Data parallelism
    - Task parallelism
  - Parallelism in hardware
    - Massively Parallel System-on-Chip
- Factorization mechanism
  - Compact notation (no unrolling)
  - Regularity in data access patterns
  - Multidimensional arrays

Factorization: Array-OL

- Observations for systematic signal processing
  - Data structures = multidimensional arrays
    - With toroidal structure
  - Regular repetition of regular data access patterns
  - Usually data flow languages
  - Difficulty = variety of data accesses

Principles of Array-OL

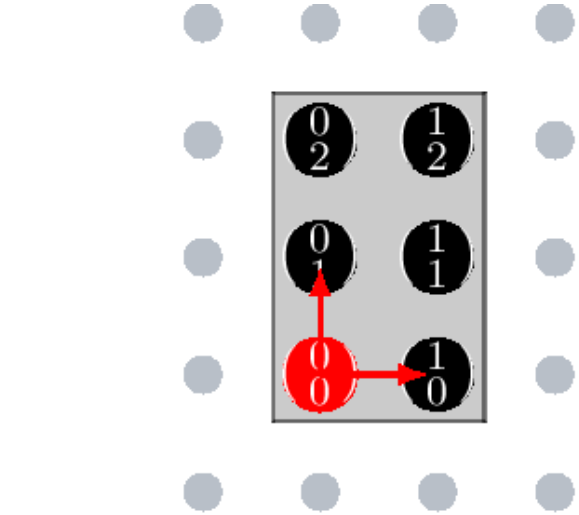
- First order functional language
  - Single assignment
  - Only true data dependences ⇒ full parallelism
- Focus on data accesses
  - Only data structure = multidimensional array
  - Pattern-based data access (via sub-arrays)

Parallelism in Array-OL

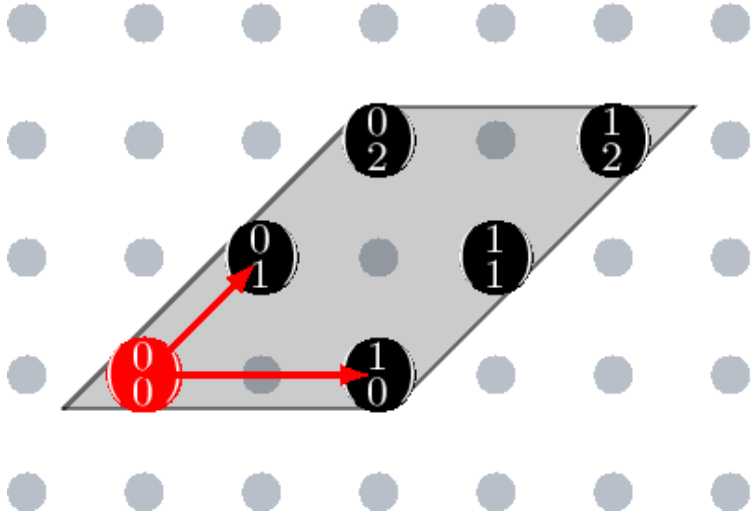
- Elementary components = black box
- Task parallelism: compound component = task graph
- Data parallelism
  - Regular repetition of a component
  - Multidimensional repetition space R (time, space, frequency, etc.)
  - Tiling of each input and output array
    - Access by conform sub-arrays (*tiles*) defined by a *pattern*
    - Regular spacing of the tile elements = *fitting*
    - Regular spacing of tiles = *paving*
    - For repetition $\boldsymbol{r} \in s_{\text{repetition}}$, element of index $\boldsymbol{i} \in s_{\text{pattern}}$ in the pattern has the following coordinates in the array:

$$\boldsymbol{o} + \begin{pmatrix} P & F \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{r} \\ \boldsymbol{i} \end{pmatrix} \bmod s_{array}$$

Fitting Examples

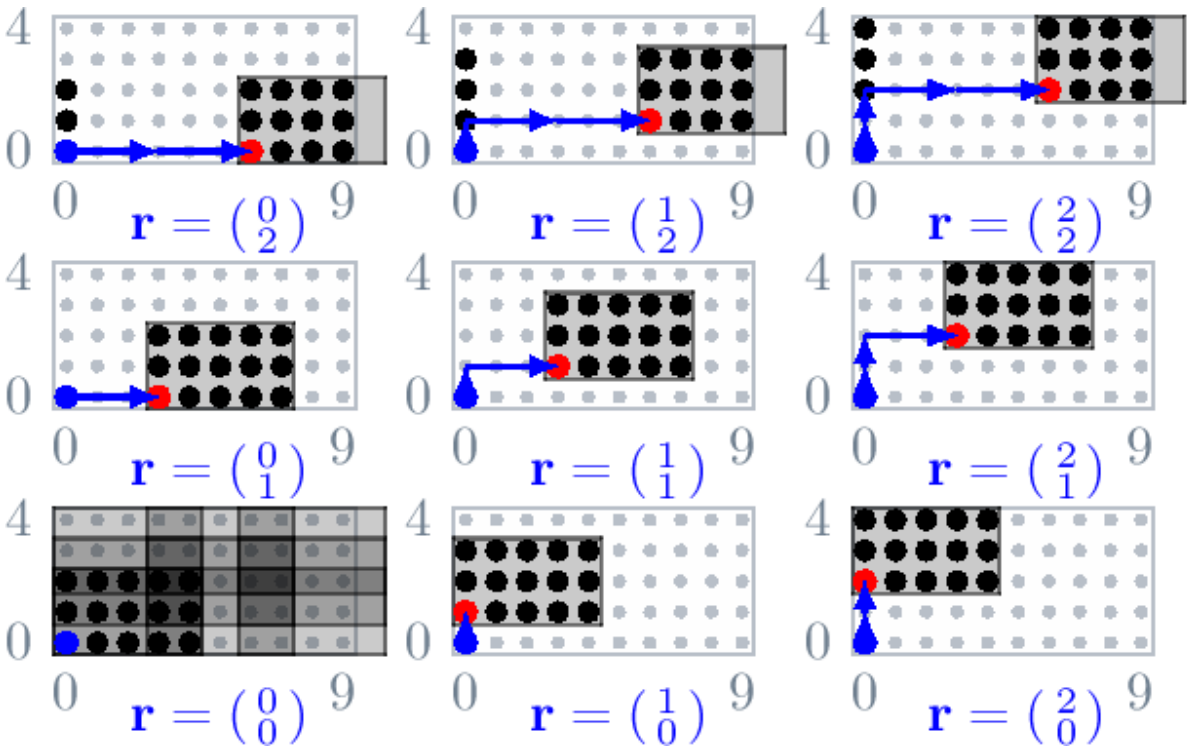


$$F = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \mathbf{s}_{\text{pattern}} = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$



$$F = \begin{pmatrix} 2 & 1 \\ 0 & 1 \end{pmatrix} \quad \mathbf{s}_{\text{pattern}} = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

Paving Example

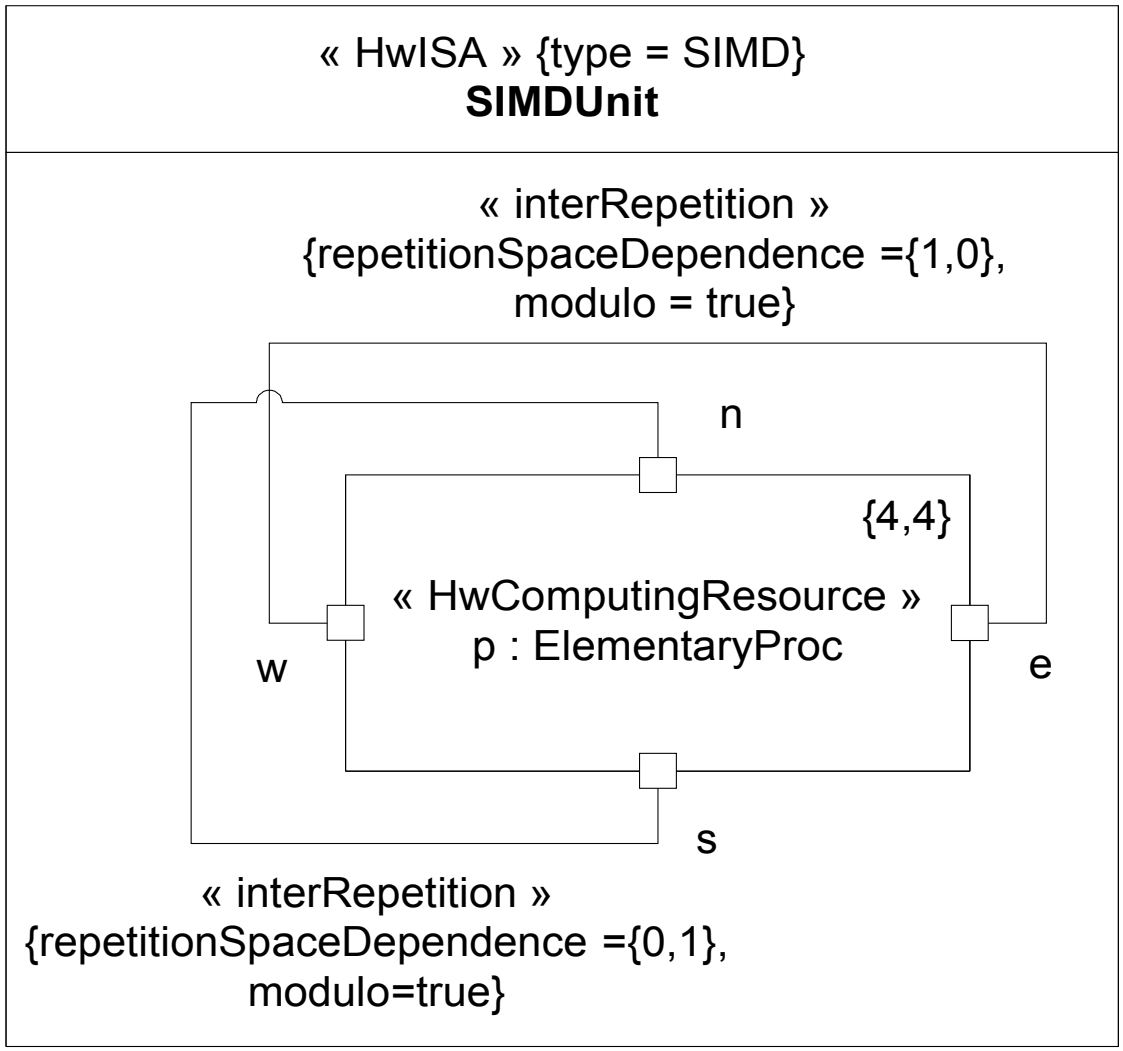


$$F = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \mathbf{s}_{\text{pattern}} = \begin{pmatrix} 5 \\ 3 \end{pmatrix}$$

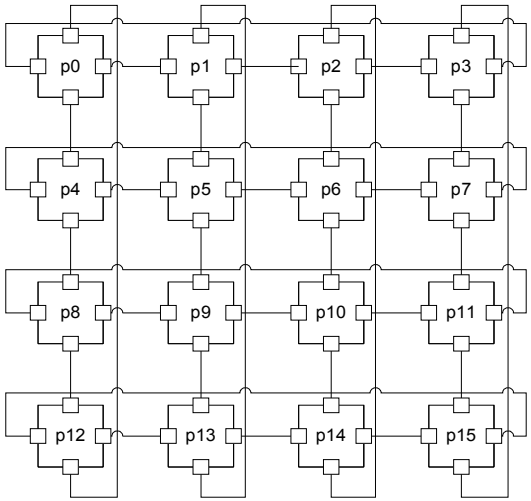
$$\mathbf{o} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \qquad \mathbf{s}_{\text{array}} = \begin{pmatrix} 10 \\ 5 \end{pmatrix}$$

$$P = \begin{pmatrix} 0 & 3 \\ 1 & 0 \end{pmatrix} \qquad \mathbf{s}_{\text{repetition}} = \begin{pmatrix} 3 \\ 3 \end{pmatrix}$$

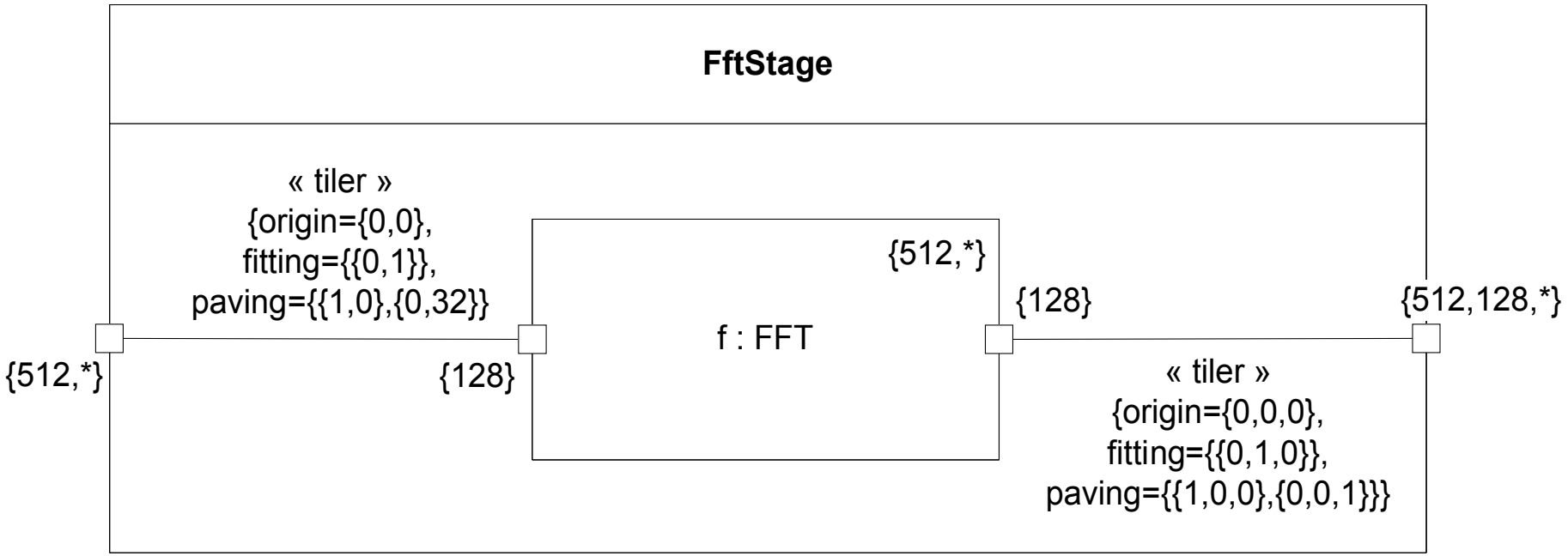
Hardware Platform Example



- SIMD unit
  - 16 processors
- Topology
  - Toroidal 4×4 grid
  - Bidirectional connections
    - North-South
    - East-West

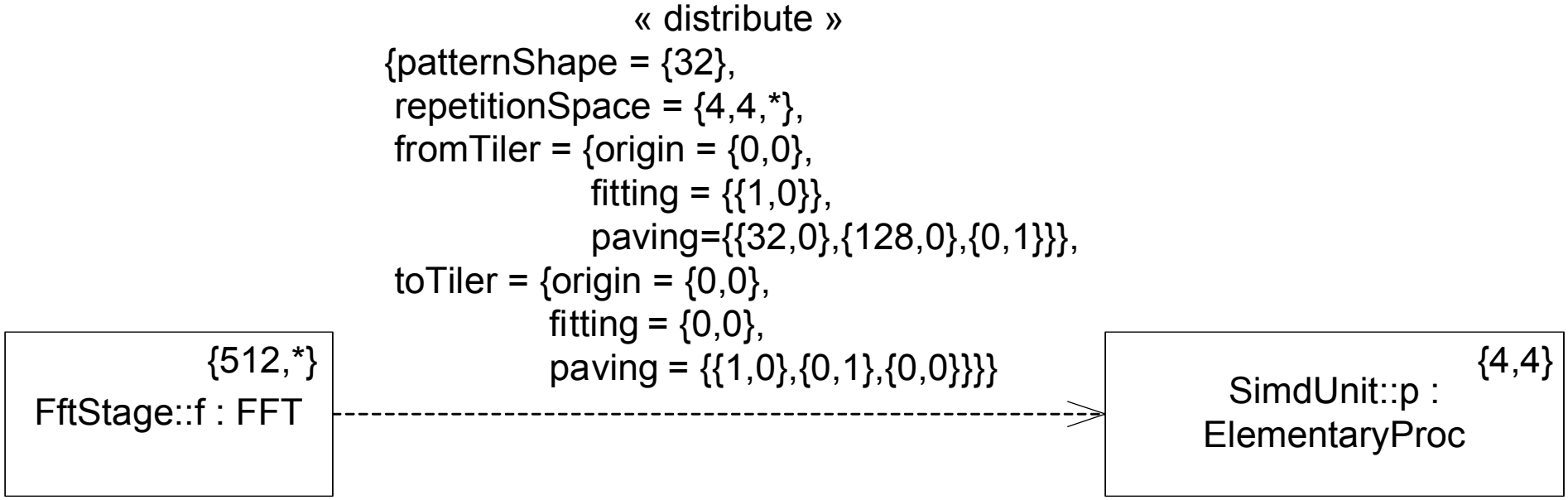


Application Example



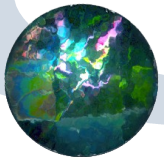
- Samples from 512 hydrophones around a submarine
  - Shape of the input data = 512×∞
- Repetition of FFTs
  - For each hydrophone
  - Sliding window of 128 samples every 32 time steps

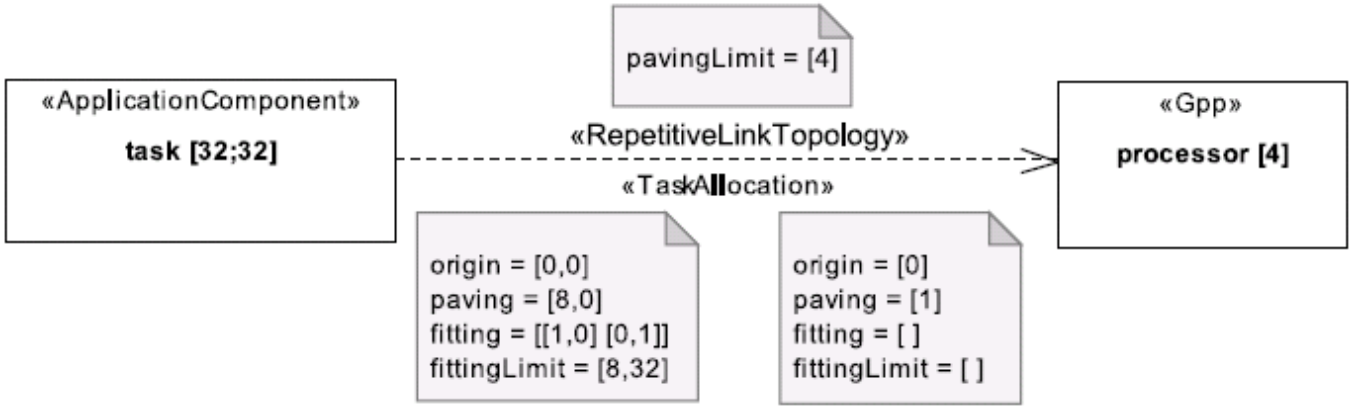
Allocation Example

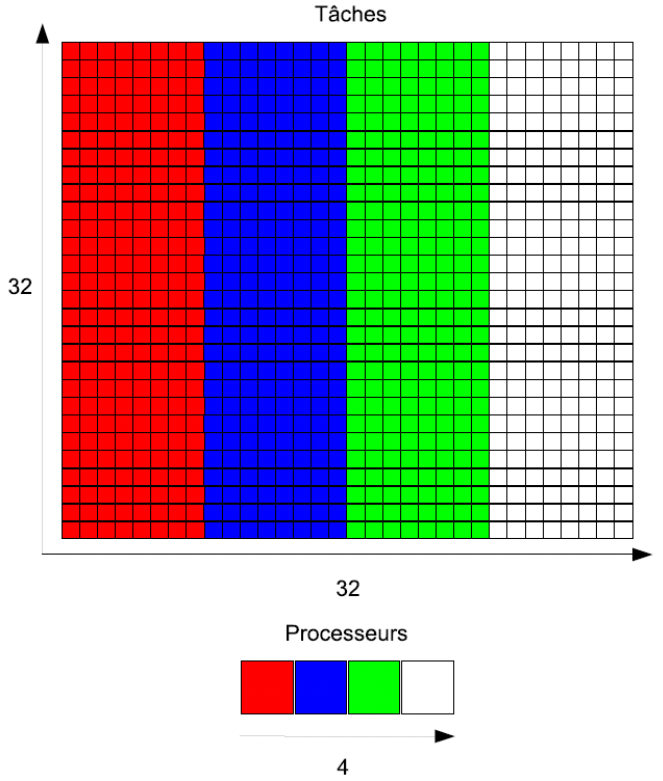


- Distribution of the FFT computations to the SIMD unit
  - No spatial distribution of the infinite dimension (time steps)
  - Bloc distribution of the 512 FFTs for each time step
    - Size of the bloc = 32
    - On the 16 elementary processors
- « distribute » specializes « allocate » for repetitive structures

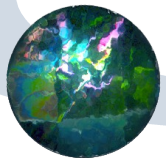
PACKAGE ASSOCIATION

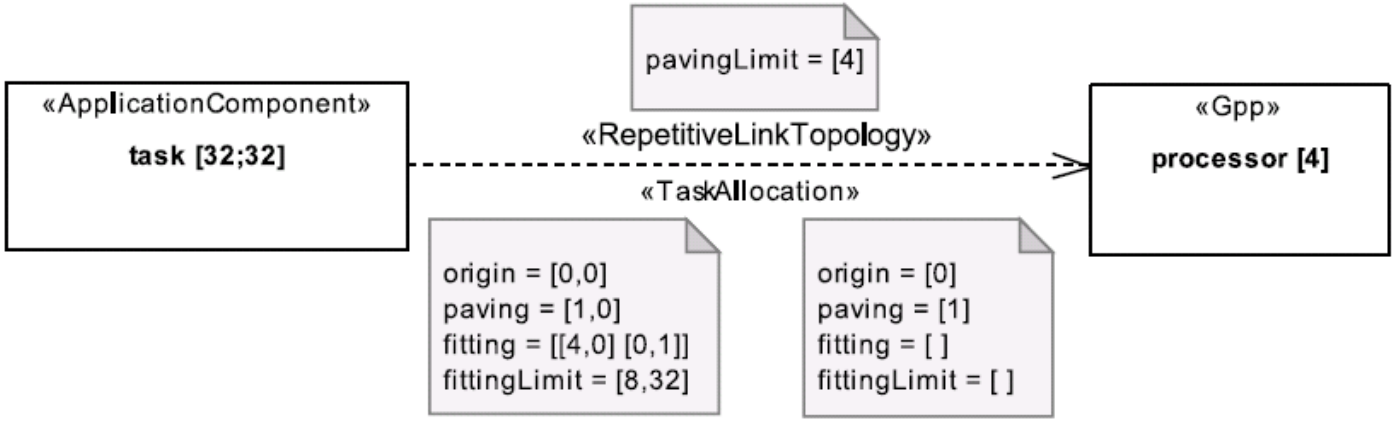


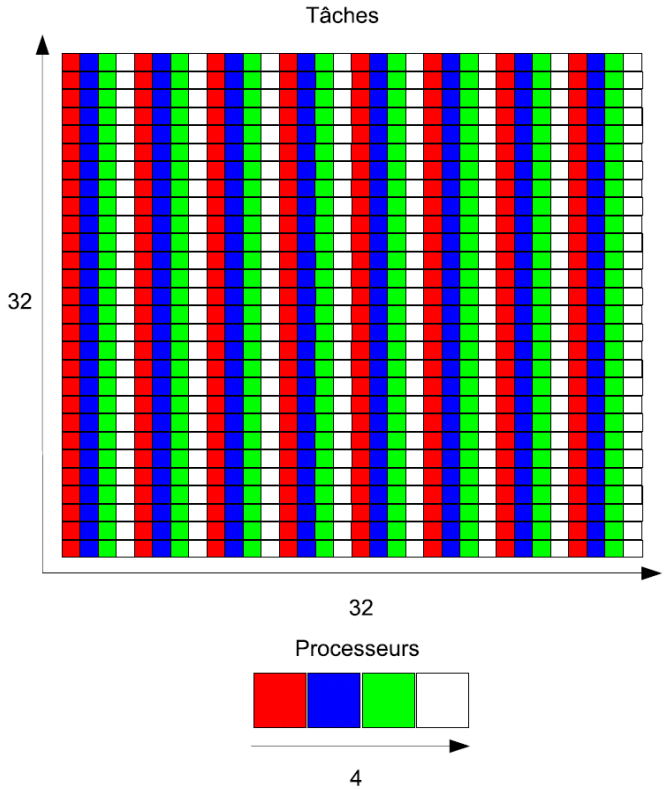




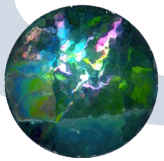
PACKAGE ASSOCIATION

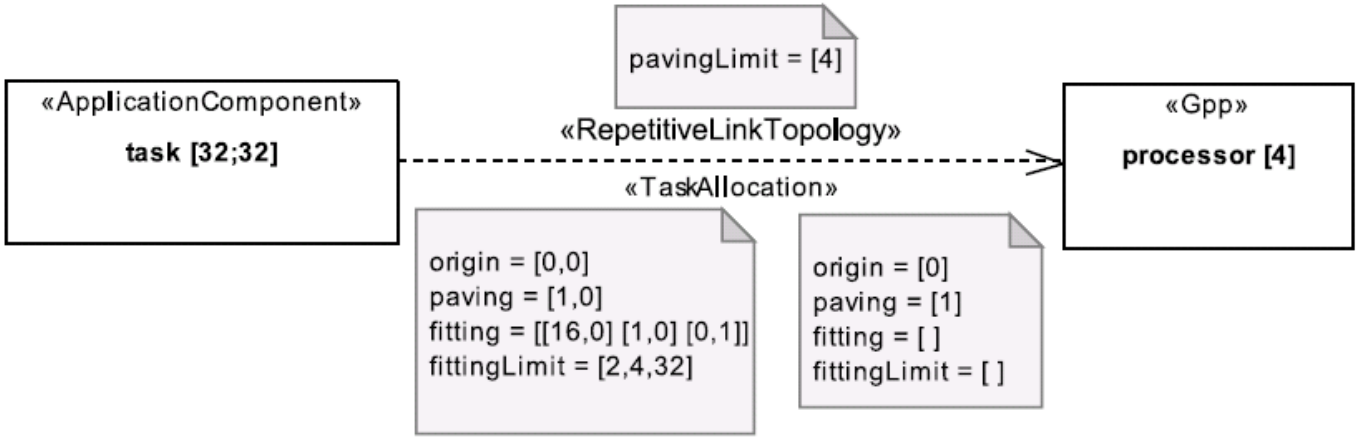


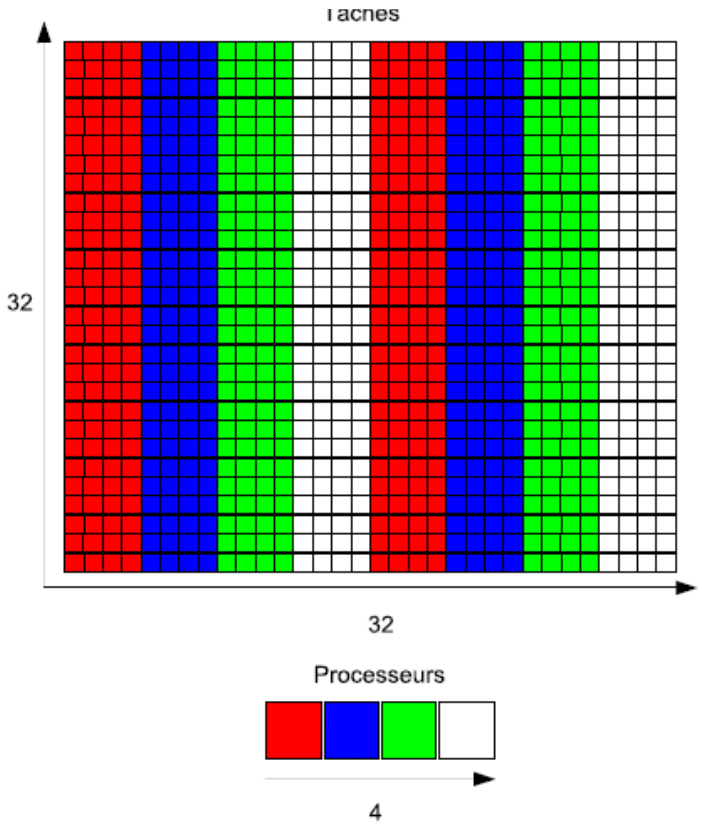




PACKAGE ASSOCIATION



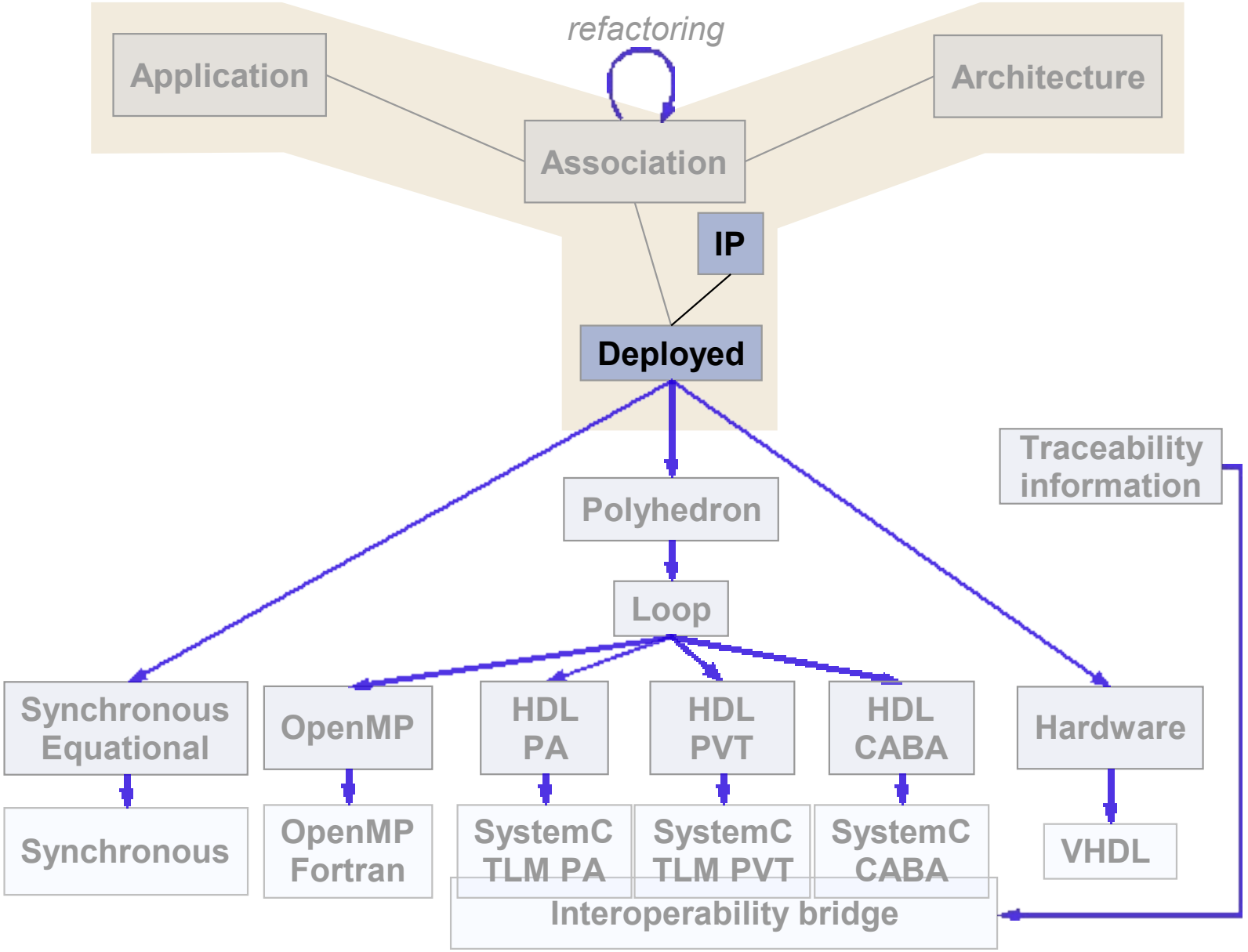




Related Works and Perspectives

- Other profiles for SoC comodeling
  - UML for SoC
    - Much more limited scope
  - Schedulability, Performance and Time
    - MARTE ⊃ SPT v2
  - UML for SystemC
    - Not in the standardization process
    - May become UML for SoC v2
- Still to do for MARTE
  - Finalization task force (1 to 2 years)
  - Documentation, tutorials, etc.
  - Extraction of common usage parts (annexes) to integrate in UML?

Deployment Specification



Problematics

- Elementary components are like black boxes
- Need for code generation
  - possibility to have several targets (eg: SystemC, VHDL, Lustre...)
  - transformation should be entirely automatic
- IP reuse
  - software library (eg: Genial, VSPIL, FFTW, in-house...)
  - hardware library (eg: SoCLib, UniSim, in-house...)
- Interoperability between components
  - a component can be connected to any other component
  - standard data format of the software components
  - multi-level simulation (eg: TLM, CABA...)
  - communication protocol for hardware components (eg: VCI, OCP...)

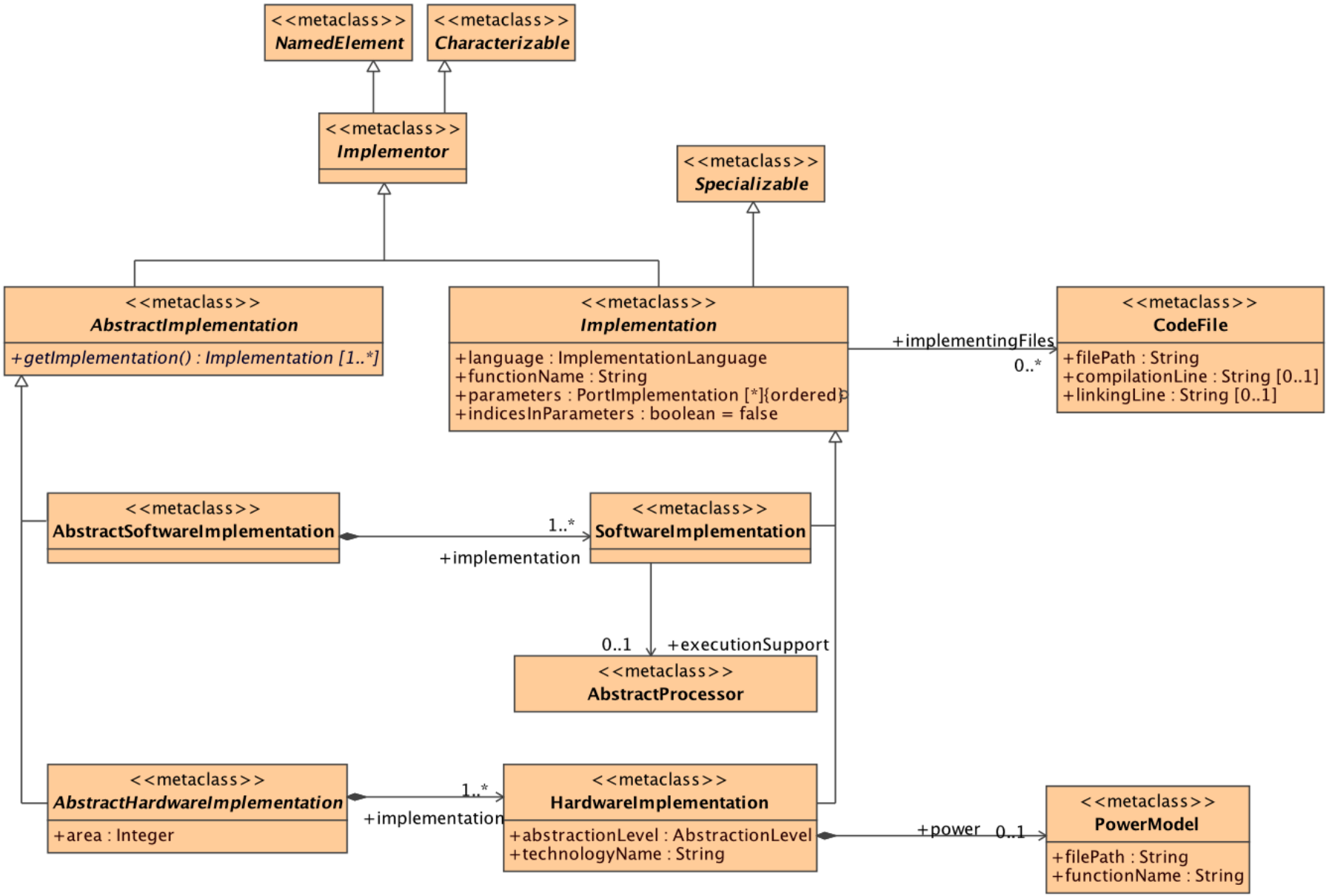
Implementations

- AbstractImplementation
  - represents a functionality
  - platform independent
  - each elementary component linked to *one* AbstractImplementation
- Implementation
  - several by AbstractImplementation
  - platform dependent
  - selected automatically according to the target, or forced by the user
- PortImplementation
  - represents arguments of functions or pins of hardware
  - ordered, to distinguish them

Enriching the Models

- CodeFile
  - filePath, compilationLine, linkingLine
  - file containing the implementation
- PowerModel
  - filePath, functionName
  - power estimation function
- Specialization
  - name, value
  - list of data to parametrize the implementation
- Characteristics
  - name, value
  - list of data to pass details to the transformation

Deployment Specification Meta-model (extract)

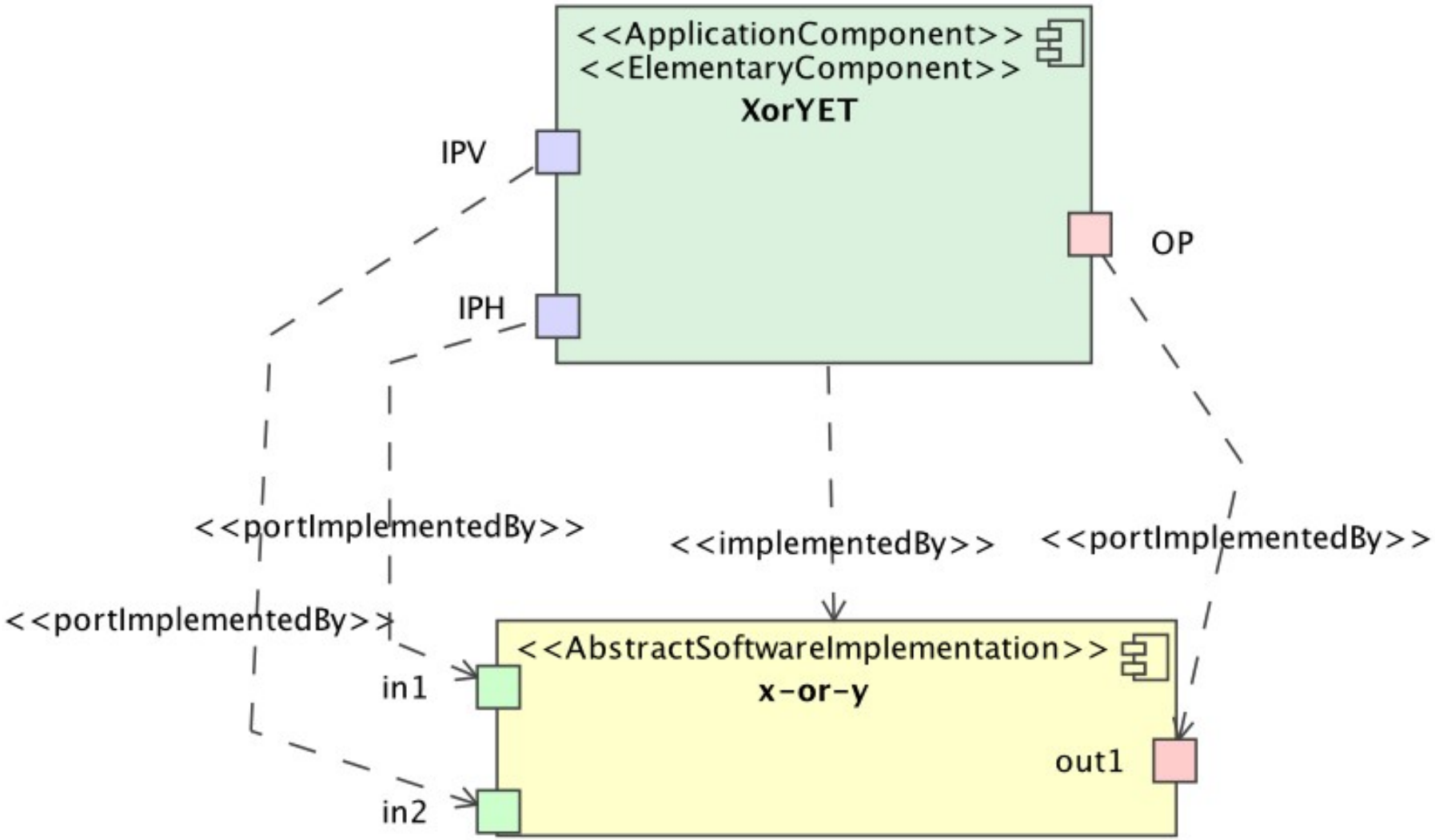


Library of Components

- Function call
  - standard: only one way to pass data to/from a function
  - defined for each language, in the Gaspard specification
  - to import libraries: use of wrappers
- GaspardLib
  - library of hardware and software components
  - several implementations (varies by abstraction level, library...)
  - contains all the code files of the components
  - one deployment model, imported by the designer

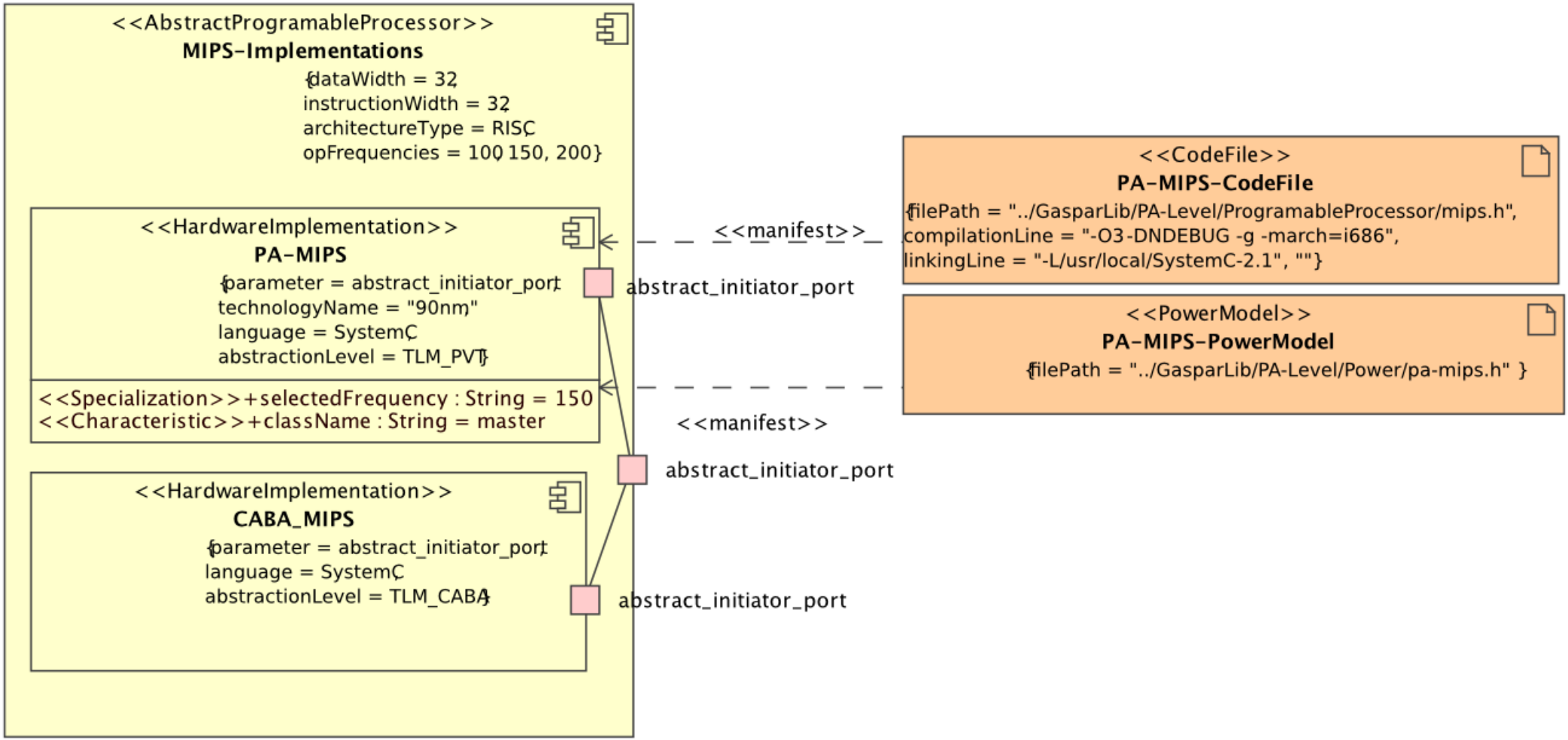
Example

- Software task implemented with a library component (not detailed)



Example

- Hardware library: several implementations of a MIPS processor



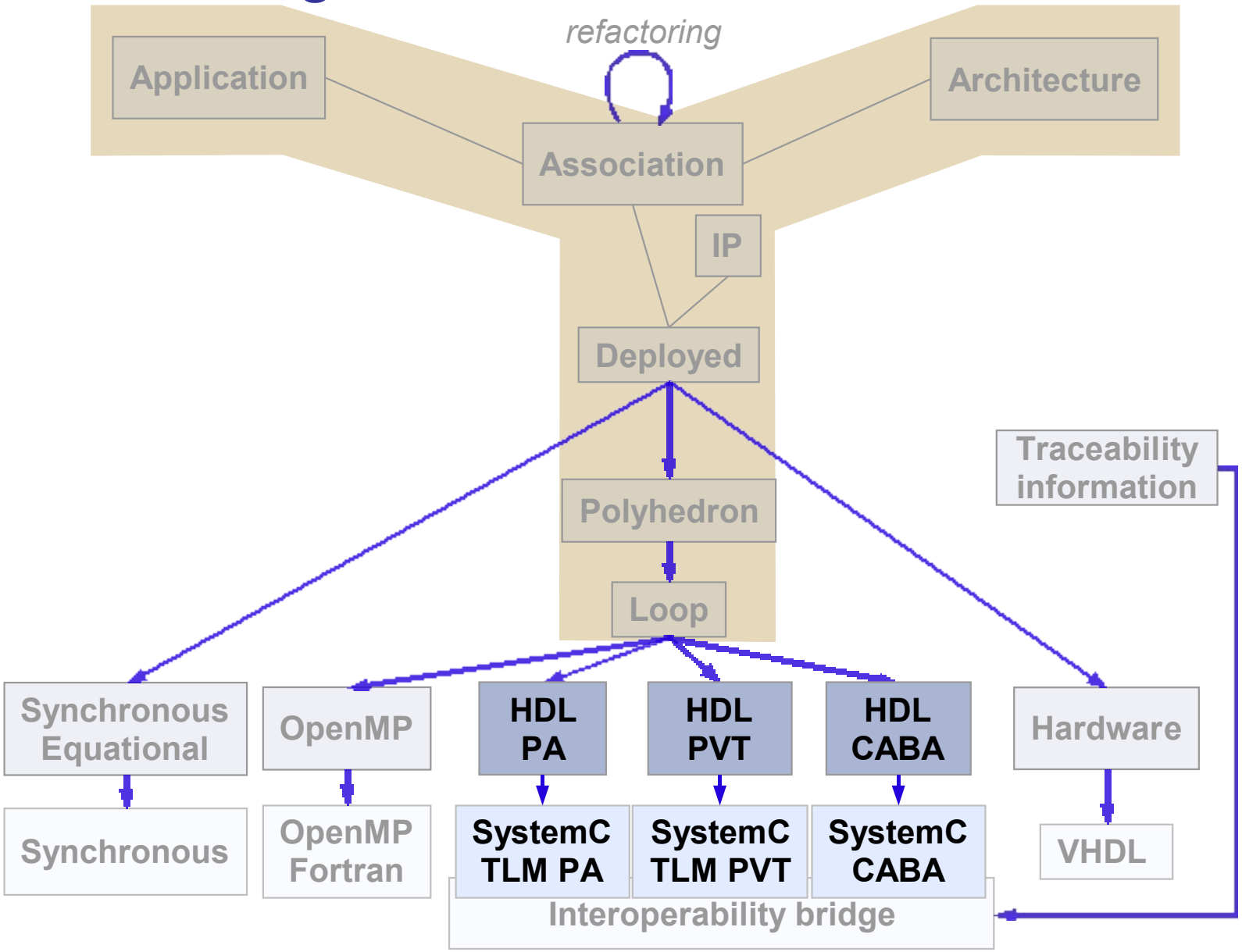
Related Works

- SPIRIT
  - standard to express the communication between IPs (hardware components)
  - XML based
- SynDEx
  - possibility to assign code to elementary component
  - usage of code with special values to reference the component ports
  - no genericity

Perspectives

- Automatic generation of wrappers
  - from the IP interface description
- Language interoperability
  - based on the deployment specification
- Automatic selection of the best implementation
  - given a set of criteria to optimize (consumption, time...)
  - designer just selects the functionality
  - use of characteristics on each implementation

TLM Modeling and Perf. Estimation



Problematic and Objectives

- Problematic
  - Energy consumption and performances are key issues in modern SoCs
  - IP reuse represents 90% of new designs (ITRS)
- Objectives
  - Define standard Transactional Level Modeling abstraction levels
    - Allowing fast simulations
    - Offering good accuracy
    - Integrating energy consumption and performances estimation

Related Work : TLM

- OSCI TLM standard
  - CP
  - CP + T
  - PT
  - PV + T
  - CABA
  - RTL
- IP reuse becomes easier but nothing about performances and power consumption estimations

Abstraction Levels in Garpard

- Timed Transaction Level Modeling (TLM-T)
  - TLM pattern accurate (TLM-PA)
    - Architecture well defined
    - Processors are replaced by tasks
    - Fast system verification
    - Dynamic contention monitoring in the interconnection network
    - Approximative execution time and power consumption estimation
  - TLM event accurate (TLM-EA)
    - Instruction set simulator (ISS) are used
    - Communication protocol is specified
    - Accurate execution time and power consumption estimation

Related Work : Energy Consumption Estimation

- RTL level: SPICE (University of Berkeley), ELDO (Mentor), PETROL (Philips)…
- Architectural level: Wattch (University of Harvard and IBM T. J. Watson Research Center) and Simplepower (University of Pennsylvania) …
- Concern single-processor Systems-on-Chip (SoC)
- MPSoC design requires rapid and accurate tools for estimating performance and energy consumption.
- Functional level: Tiwari et al. have introduced the concept of Instruction Level Power Analysis (ILPA)

Energy consumption estimation

- Component power model :
  - Abstraction level
  - Pertinent activities
  - Power cost for each activity
  - Activity occurrences
- Activity Power cost :
  - Low level measurements (From RTL)
  - Using analytical models
  - Taking into account static and dynamic power dissipation

Performance Estimation at CABA Level

- Micro-architectural details are implemented
- Performance is given by the simulator in number of cycles
- Architectural parameter specifications such as cache size

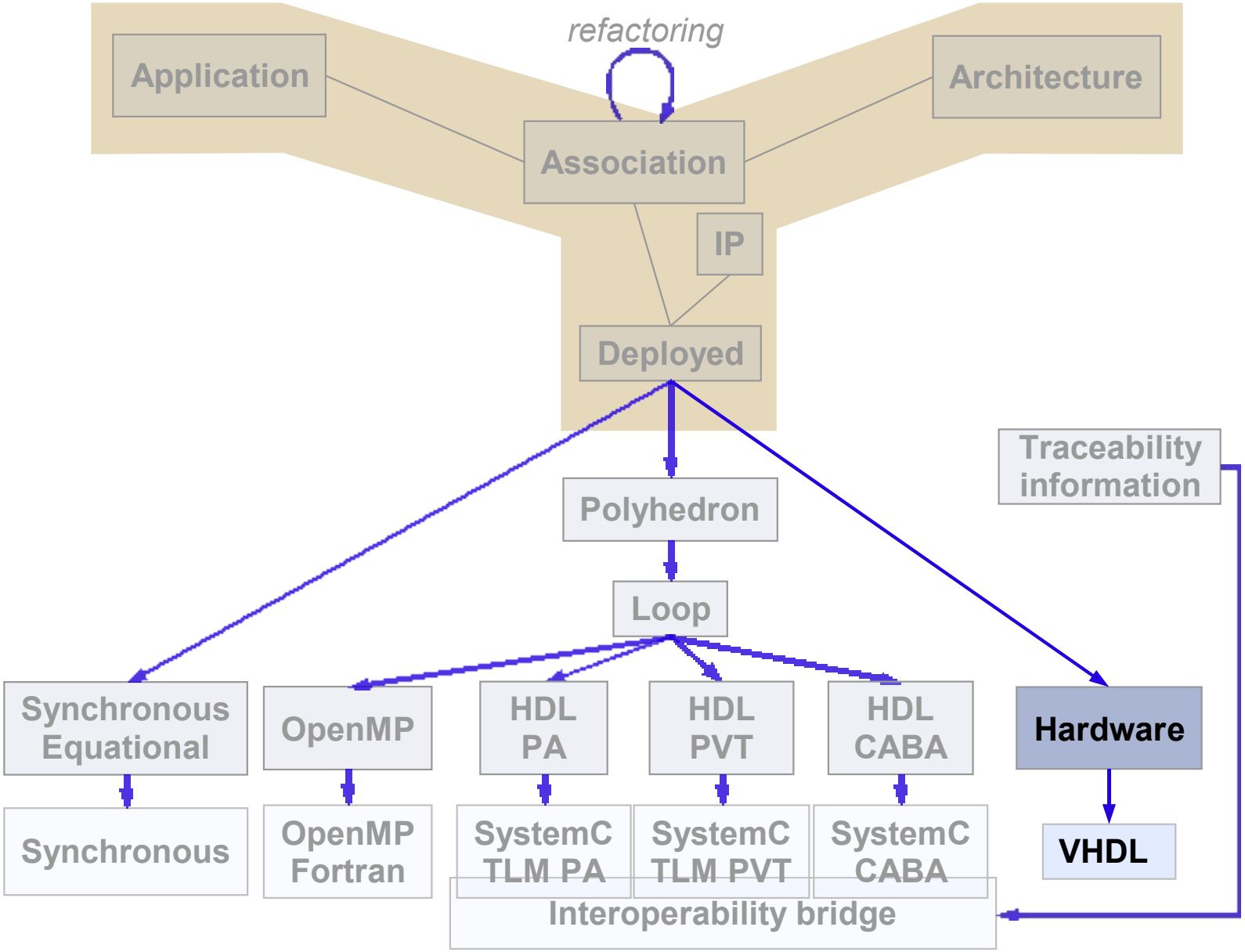
Performance Estimation at PA-TLMT Level

- Details related to micro-architecture and communication are omitted
- Transactions are performed through channels
- Identifying each component's pertinent activities
  - types of executed instructions, hits and misses for the caches, the number of transmitted/received packets, the number of read and write operations for the shared memory modules, etc.
- An execution time is attributed to each activity
- *wait(..)* instructions with arguments
- Activities execution time and platform configurations parameters will be specified at the deployment phase

Results

- Definition of 2 TLM sub levels allowing fast simulation and accurate descriptions
- Physical measures for power consumption in most used components in modern SoCs
- Analytic models
- Significant design platform at TLM level integrating energy consumption models
- Perspectives
  - Power estimation in reconfigurable architectures
  - Estimations integration in application and architectures metamodels

Synthesis


refactoring
Application
Architecture
Association
IP
Deployed
Traceability information
Polyhedron
Loop
Synchronous Equational
OpenMP
HDL PA
HDL PVT
HDL CABA
Hardware
Synchronous
OpenMP Fortran
SystemC TLM PA
SystemC TLM PVT
SystemC CABA
VHDL
Interoperability bridge

Problematics

- Hardware execution of Gaspard applications
- Configurable hardware modelling (several views)
- Transformation towards hardware metamodel
- VHDL code generation
- Simulation and synthesis
- FPGA configuration

Hardware Metamodel

- Close to a hardware architecture (clock, reset, signal, etc.)
- Independent of the target language (VHDL, Verilog HDL, etc.)
- Pipelined datapath adapted to systematic signal processing
- Multiple views of a target FPGA
  - Black box view
  - Quantitative view: number computing / IO / storage resources
  - Physical view: geographical details of resources
  - Virtual views: hierarchy build on the physical view to match the application

Transformation

- Mapping of an application component on a hardware accelerator
- Produce a datapath model realizing the application
- Manage complex data dependencies
  - shift register, temporal multiplex/demultiplex
- Keep the initial application hierarchy

VHDL Code Generation

- Manage multi-dimensional arrays
- Keep the hierarchy
- Keep the repetition factorization

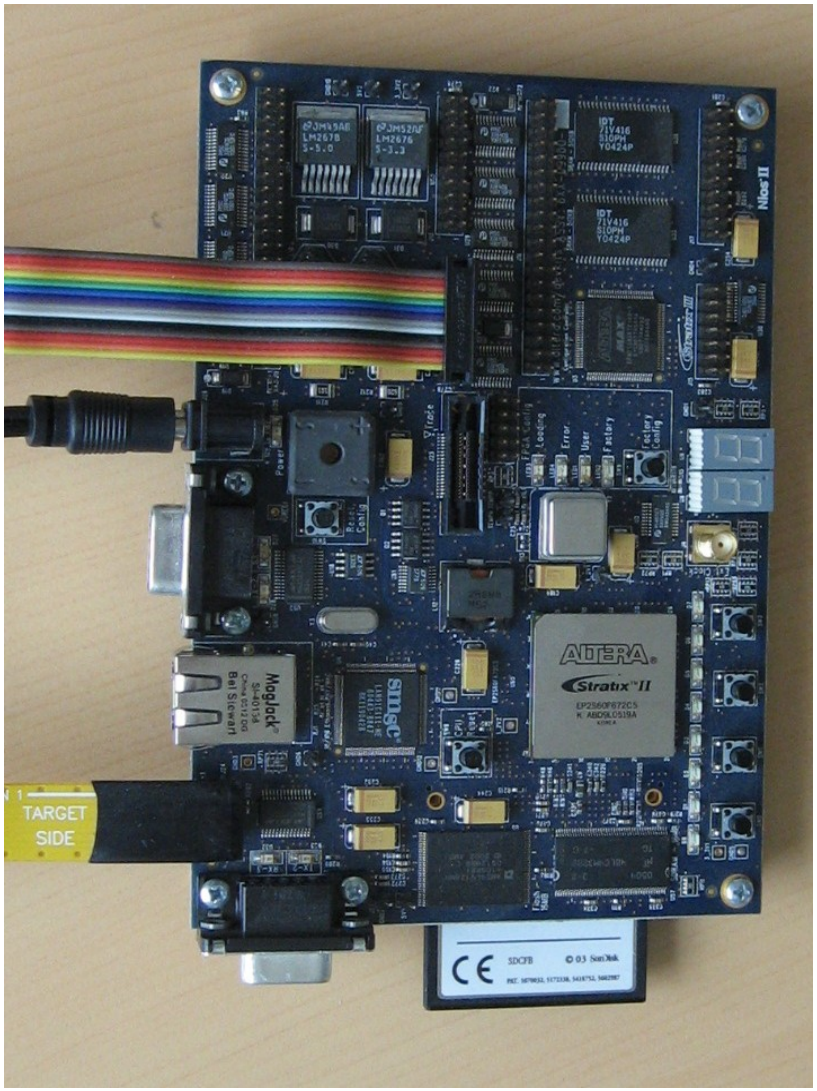
```
genit0 : for it0 in 1 to 4 generate
    genit1 : for it1 in 1 to 4 generate
        instanceOfMyComponent : MyComponent
        port map(
        clk => clkConnectorlignecolonne,
        raz => razConnectorlignecolonne,
        InAMyComponent =>RepetitionConnector_1999b96(it0)(it1),
        OutMyComponent =>RepetitionConnector_14fe736(it0)(it1),
        InBMyComponent =>RepetitionConnector_1d1e713(it0)(it1));
    end generate;
end generate;
```

Design Flow

- Evaluation of the produced hardware
  - area and pins
- Refactoring of application
  - to fit the available resources
- Folding and Unfolding
  - Array-OL transformations
  - increase/decrease required area & IO pins
  - decrease/increase execution time
- Decorrelate IO and processing
  - addition of multiplexors & demultiplexors

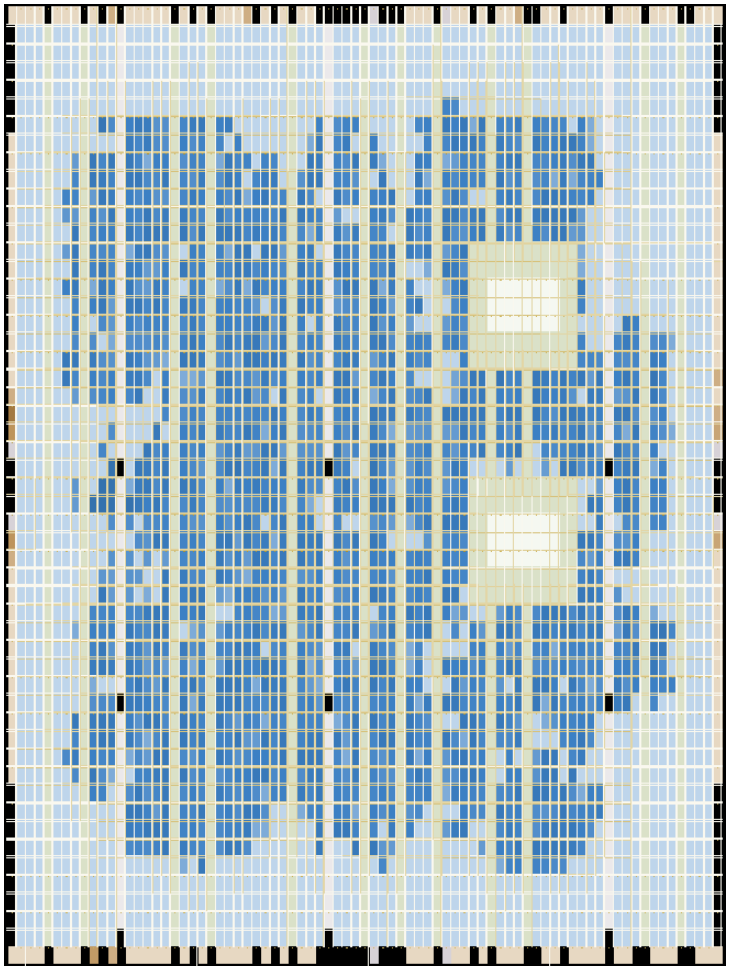
Validation

- Anti-collision radar algorithm
  - generated from UML
  - as powerful as the one tuned by hand
- FIR and IIR filters
- Image processing
- Matrix multiplication

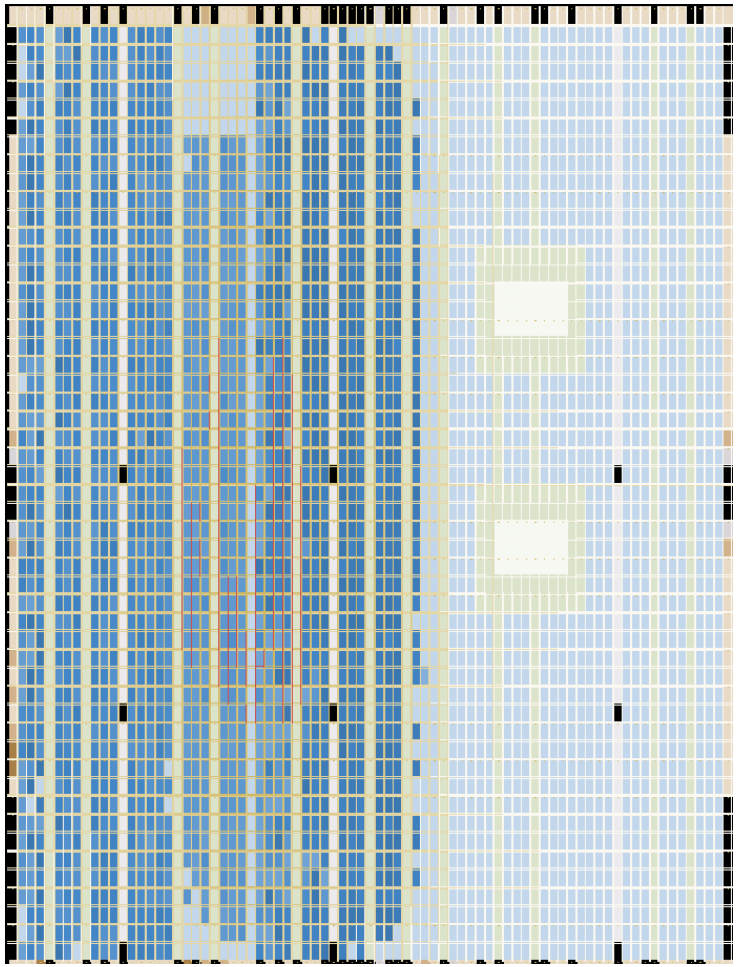


Parallel Mapping Validation

- uncontrolled mapping



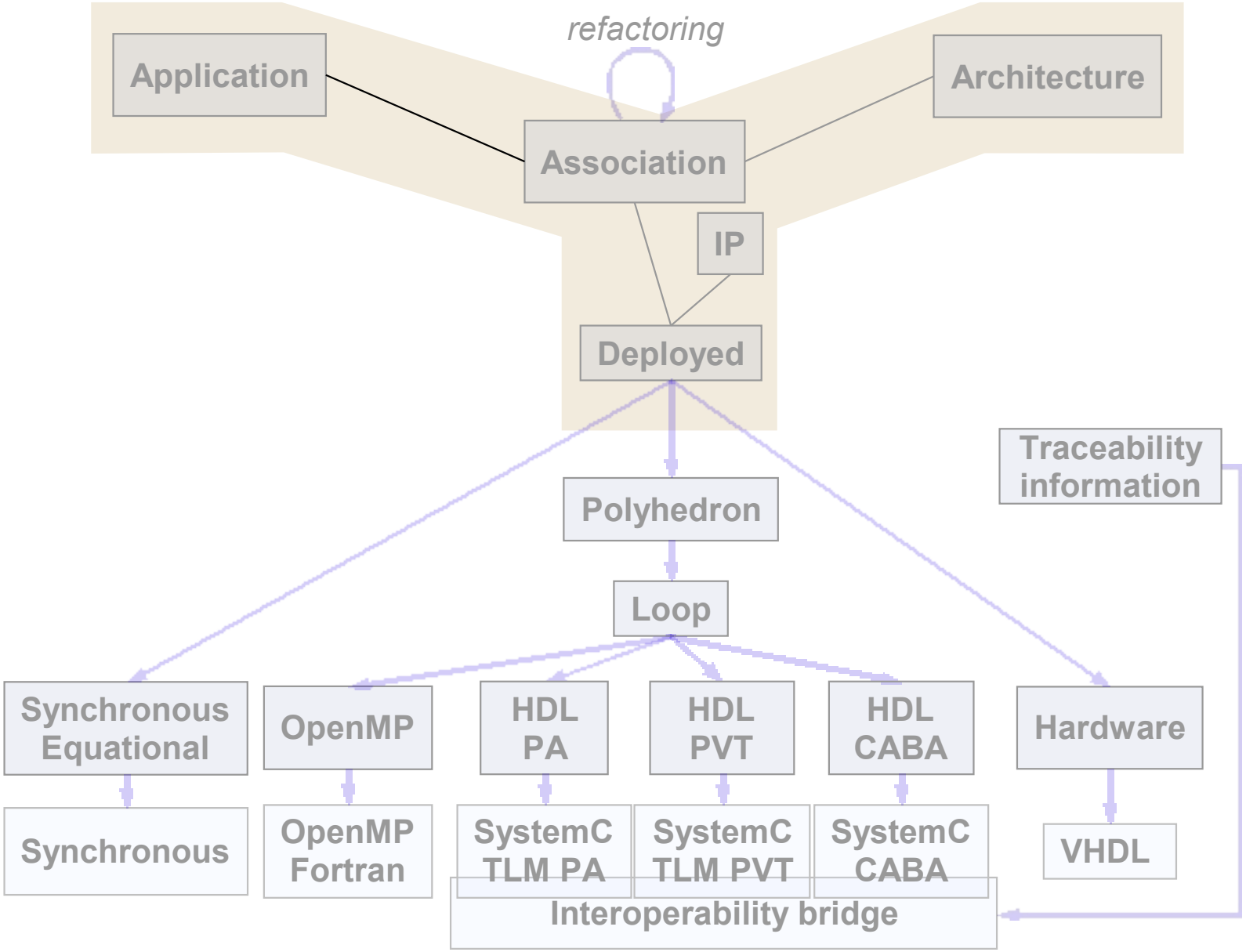
- controlled parallel mapping



Perspective

- CPU with FPGA
  - Hardcore/softcore processor in FPGA
  - FPGA + CPU in a SoC
- Automatically link the FPGA views with the VHDL code generation (parallel mapping)
- Multi-clock domain management
- Dynamic configuration via mixed data and control flow modelling

Application domains



Intensive Signal Processing

Software Radio Receiver

- Front end systematic signal processing including signal digitalizing, channel selection, and application of filters to eliminate interferences.
- Data are decoded in a second and more irregular phase (synchronization, signal demodulation...).
- Thales

Sonar Beam Forming

- First and systematic step provides frequency and location correlations from a continuous flow of data delivered by the hydrophones. It is based on signal elementary transformations: FFT (Fast Fourrier Transformation) and discrete integration.
- The second step analyses a given set of beams and their history to identify temporal correlation and association to signal sources.
- Thales (ex Underwater System)

Image processing

- JPEG-2000 Encoder/Decoder
  - JPEG-2000 works in a two-steps approach.
    - The first part (from preprocessing to wavelet decomposition) is systematic.
    - The second part of the encoder includes irregular processing (quantification, two coding stages).
    - The decoder works the other way around: a first irregular phase is followed by a systematic phase.
  - Thales, Phillips (ITEA Sophocles)

Correlation Algorithm Formulation

General formulation:

$$Ccy(j) = \frac{1}{N} \cdot \sum_{i=0}^{N-1} c(i) \cdot y(i+j)$$

- N: reference code length
- c: reference code
- y: received waive

Our specific study during Modeasy project:

$$Ccy(j) = \sum_{i=0}^{1022} c(i) \cdot y(i+j)$$

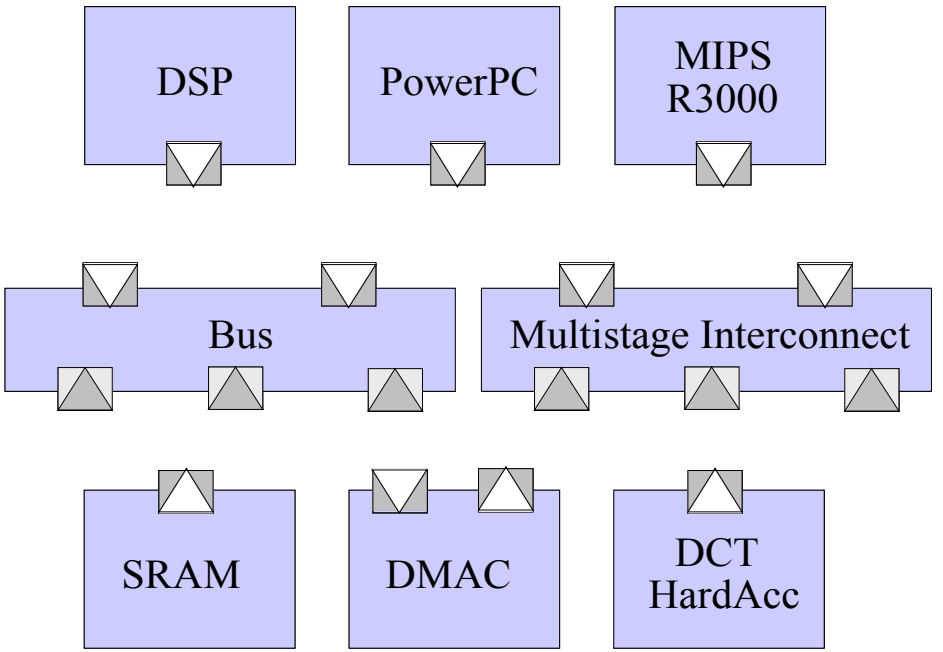
Architecture/Application Adequacy

Design space exploration based on performance and power estimation

- Selected processor (PowerPc, MIPS, ARM...)
- Adequate interconnect (Bus, Multistage...)

Mapping verification

- Mapped tasks on processors
- Mapped data arrays on memory banks



High Performance Computing

Large numerical system to solve => long time of simulation

Parallel numerical simulation

3D electromagnetic simulation

- Applied on a Valeo claw alternator

Non linear equation solver

- Jacobi
- Conjuguate Gradient
- Multigrid

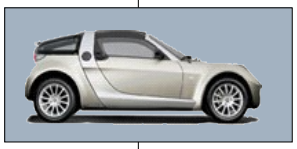
Could be used in others numerical simulations

Auto cruise control application

GPS + RADAR

Deduce the speed limit according to area

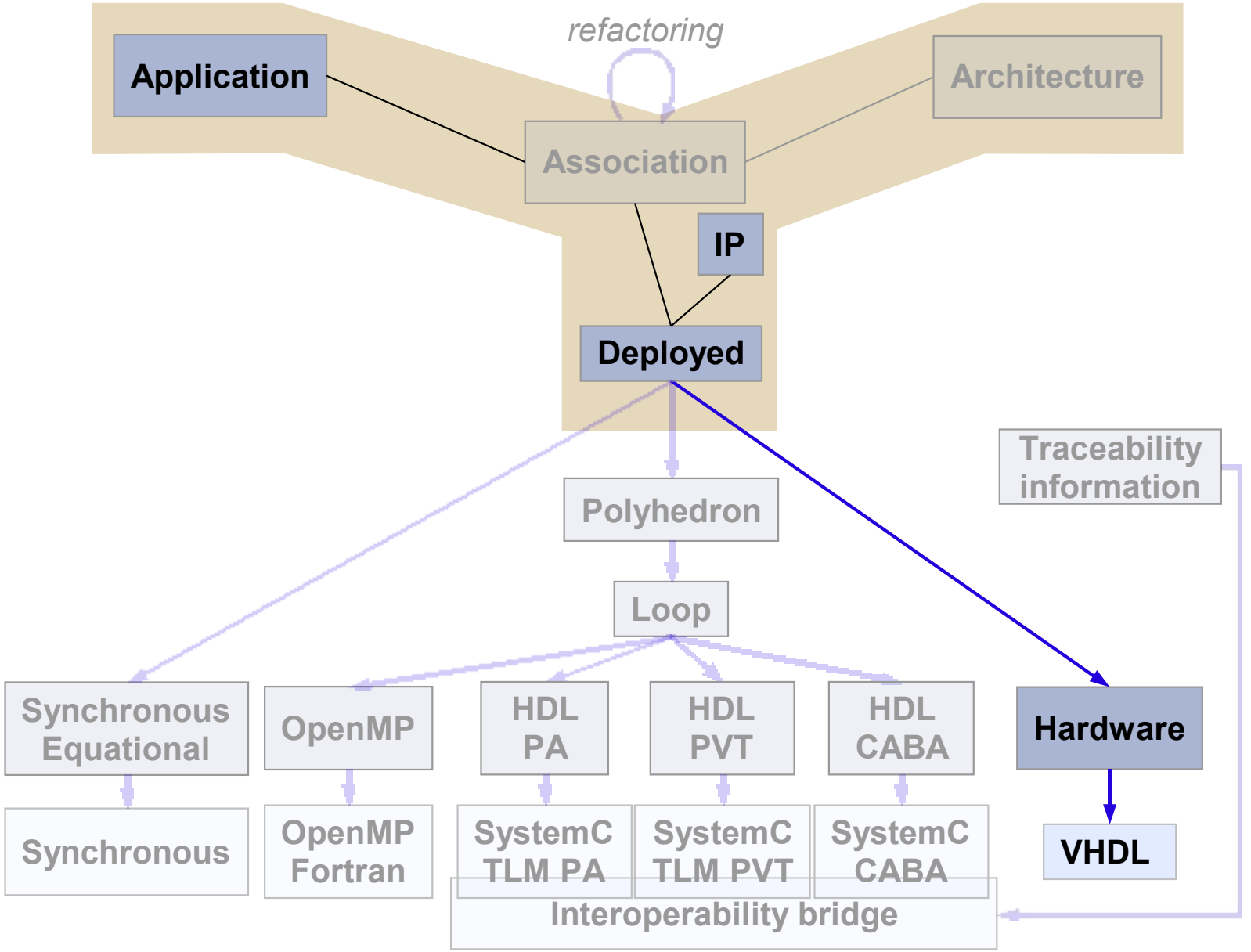
GPS system



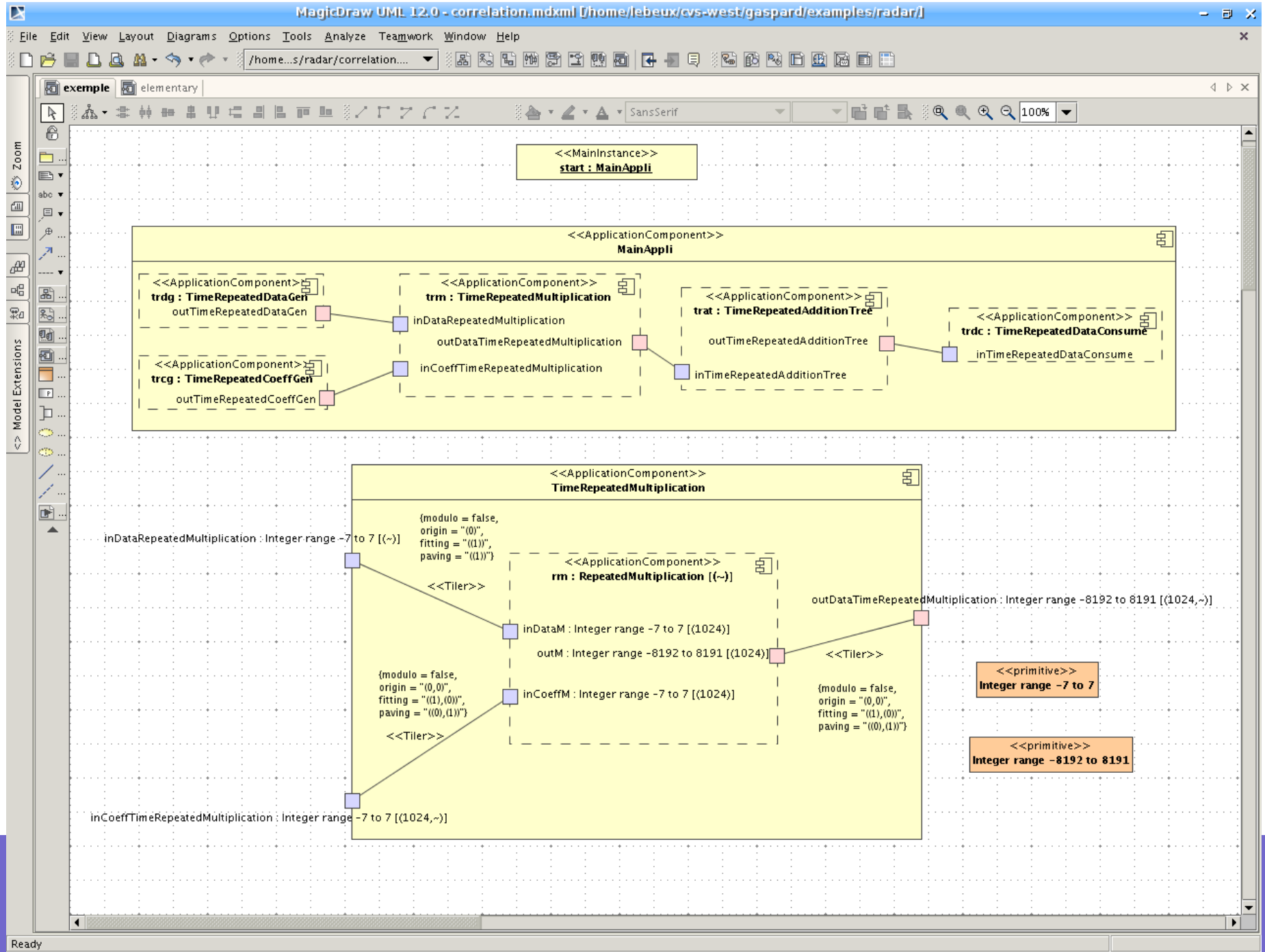
Measure the distance to the closest car and his speed

Radar system

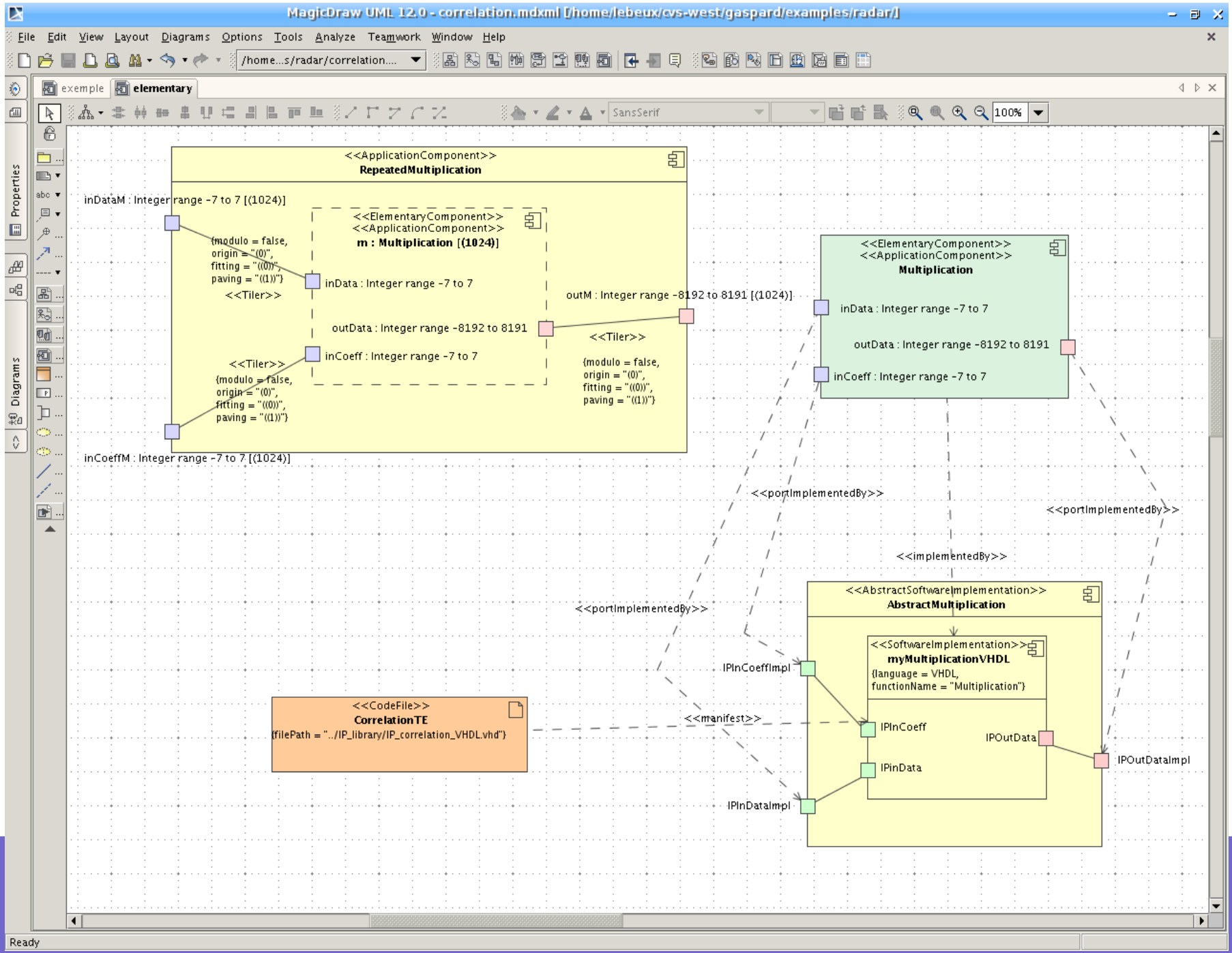
Data flow to VHDL



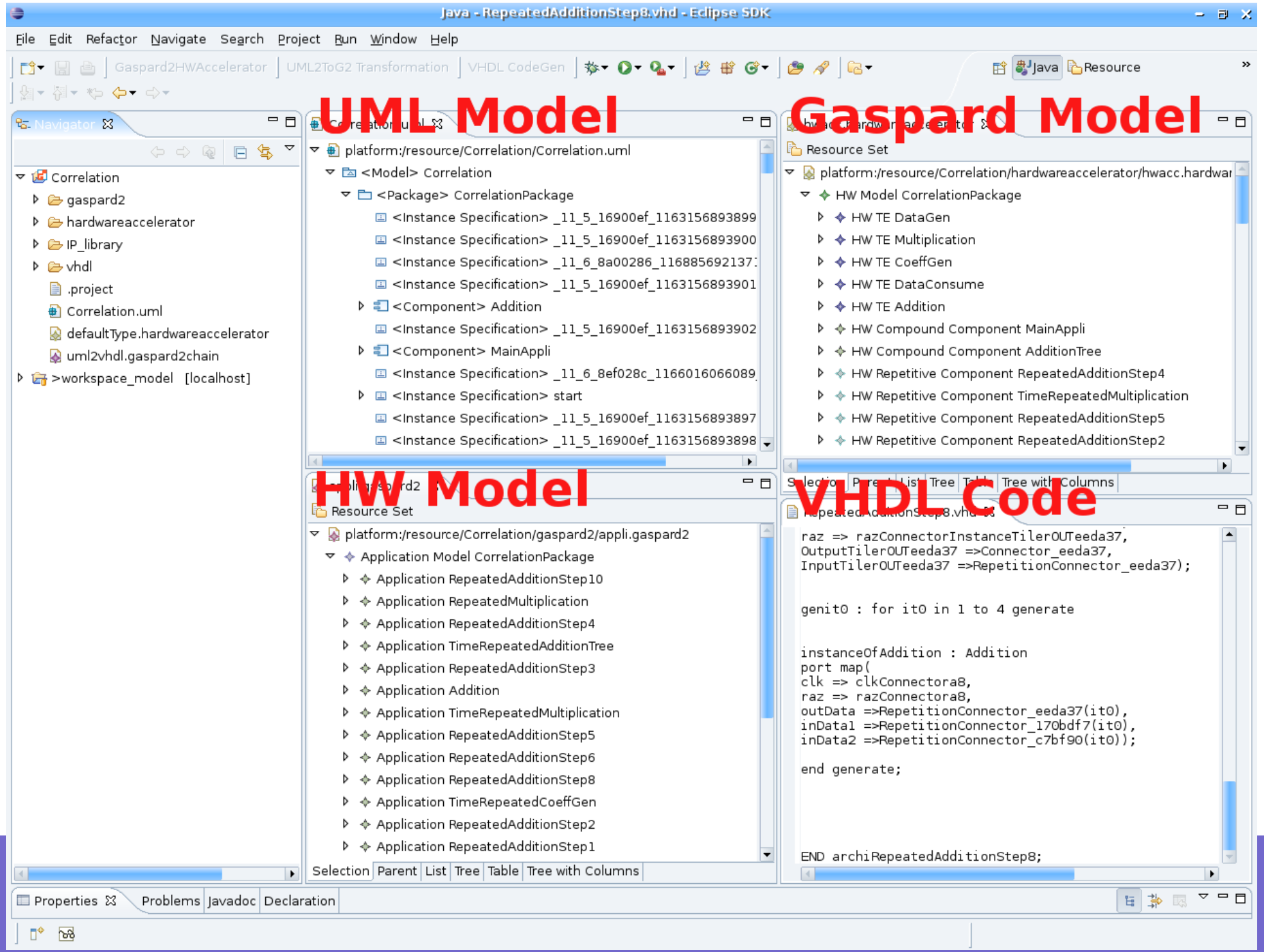
UML Model (1/2)


MagicDraw UML 12.0 - correlation.mdxml [/home/lebeux/cvs-west/gaspard/examples/radar/]
File Edit View Layout Diagrams Options Tools Analyze Teamwork Window Help
exemple
elementary
<<MainInstance>>
start : MainAppli
<<ApplicationComponent>>
MainAppli
<<ApplicationComponent>>
trdg : TimeRepeatedDataGen
outTimeRepeatedDataGen
<<ApplicationComponent>>
trcg : TimeRepeatedCoeffGen
outTimeRepeatedCoeffGen
<<ApplicationComponent>>
trm : TimeRepeatedMultiplication
inDataRepeatedMultiplication
outDataTimeRepeatedMultiplication
inCoeffTimeRepeatedMultiplication
<<ApplicationComponent>>
trat : TimeRepeatedAdditionTree
outTimeRepeatedAdditionTree
inTimeRepeatedAdditionTree
<<ApplicationComponent>>
trdc : TimeRepeatedDataConsume
inTimeRepeatedDataConsume
<<ApplicationComponent>>
TimeRepeatedMultiplication
inDataRepeatedMultiplication : Integer range -7 to 7 [(~)]
{modulo = false, origin = "(0)", fitting = "((1))", paving = "((1))"}
<<Tiler>>
<<ApplicationComponent>>
rm : RepeatedMultiplication [(~)]
inDataM : Integer range -7 to 7 [(1024)]
outM : Integer range -8192 to 8191 [(1024)]
inCoeffM : Integer range -7 to 7 [(1024)]
{modulo = false, origin = "(0,0)", fitting = "((1),(0))", paving = "((0),(1))"}
<<Tiler>>
{modulo = false, origin = "(0,0)", fitting = "((1),(0))", paving = "((0),(1))"}
<<Tiler>>
outDataTimeRepeatedMultiplication : Integer range -8192 to 8191 [(1024,~)]
inCoeffTimeRepeatedMultiplication : Integer range -7 to 7 [(1024,~)]
<<primitive>>
Integer range -7 to 7
<<primitive>>
Integer range -8192 to 8191
Zoom
Model Extensions
Ready

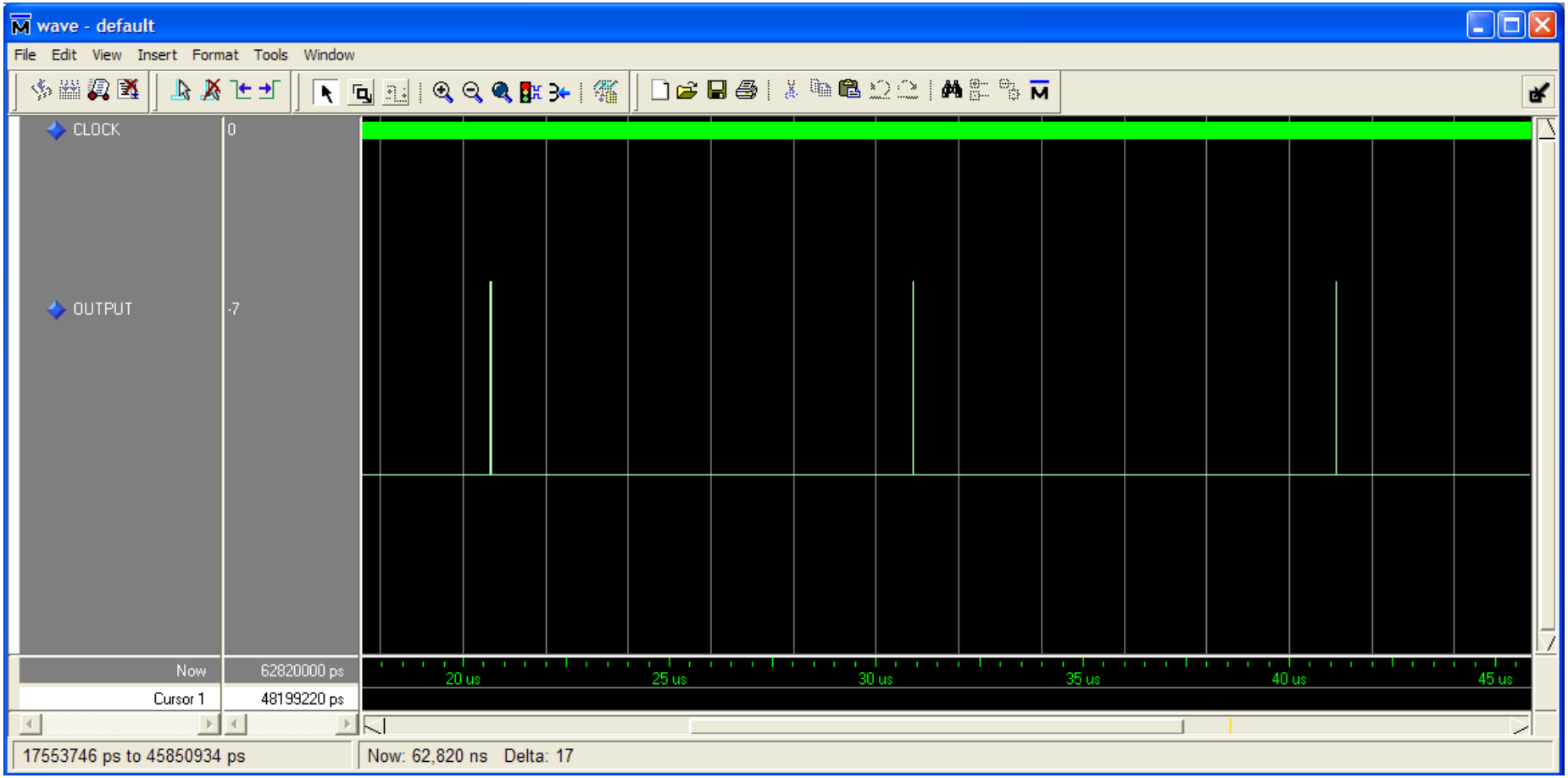
UML Model (2/2)


MagicDraw UML 12.0 - correlation.mdxml [/home/lebeux/cvs-west/gaspard/examples/radar/]
File Edit View Layout Diagrams Options Tools Analyze Teamwork Window Help
exemple
elementary
<<ApplicationComponent>>
RepeatedMultiplication
inDataM : Integer range -7 to 7 [(1024)]
<<ElementaryComponent>>
<<ApplicationComponent>>
m : Multiplication [(1024)]
{modulo = false, origin = "(0)", fitting = "((0))", paving = "((1))"}
<<Tiler>>
inData : Integer range -7 to 7
outData : Integer range -8192 to 8191
inCoeff : Integer range -7 to 7
outM : Integer range -8192 to 8191 [(1024)]
inCoeffM : Integer range -7 to 7 [(1024)]
<<ElementaryComponent>>
<<ApplicationComponent>>
Multiplication
<<portImplementedBy>>
<<implementedBy>>
<<AbstractSoftwareImplementation>>
AbstractMultiplication
<<SoftwareImplementation>>
myMultiplicationVHDL
{language = VHDL, functionName = "Multiplication"}
IPInCoeffImpl
IPInDataImpl
IPInCoeff
IPInData
IPOutData
IPOutDataImpl
<<CodeFile>>
CorrelationTE
{filePath = "../IP_library/IP_correlation_VHDL.vhd"}
<<manifest>>
Ready

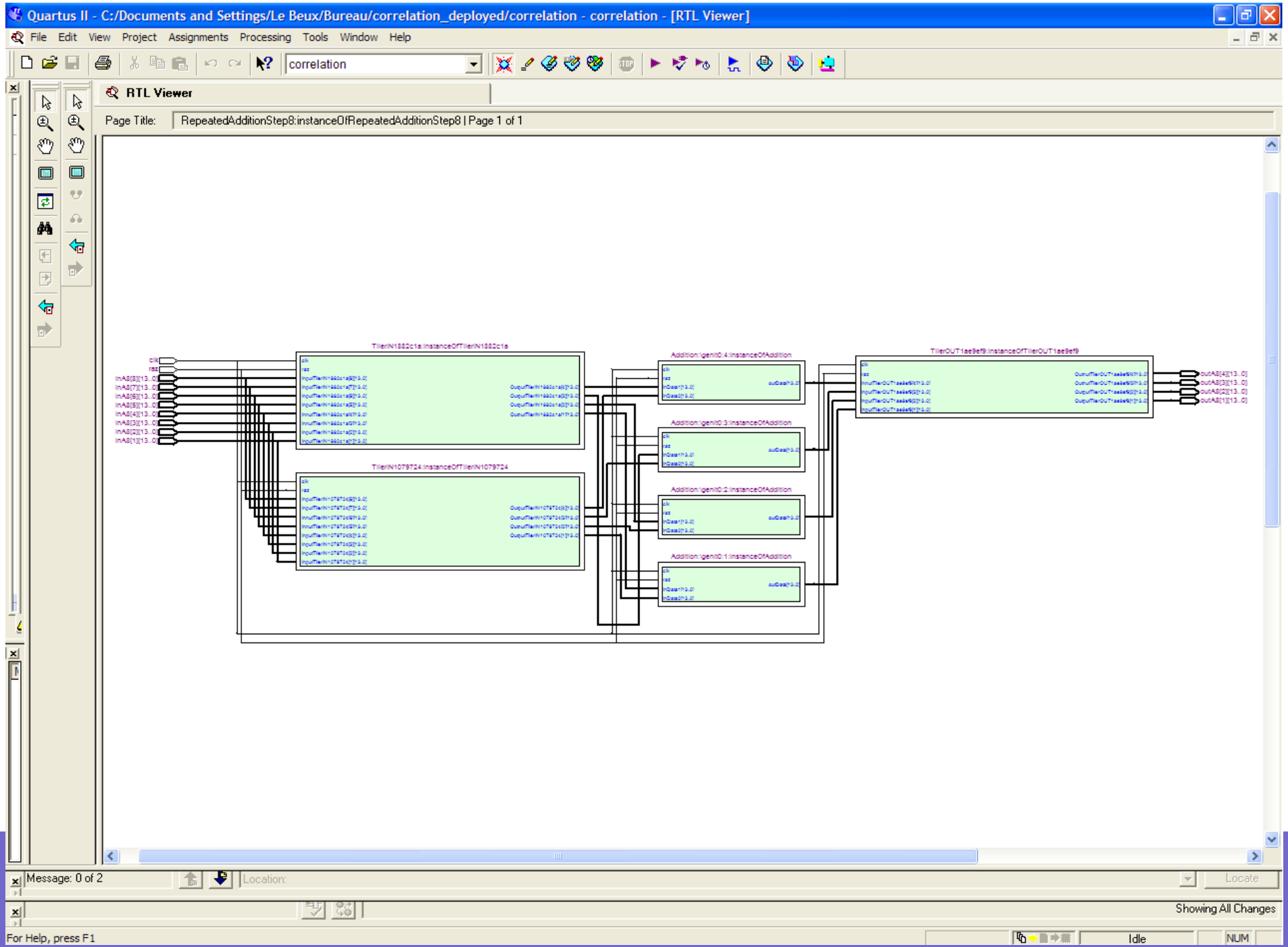
Transformation in Eclipse



VHDL simulation



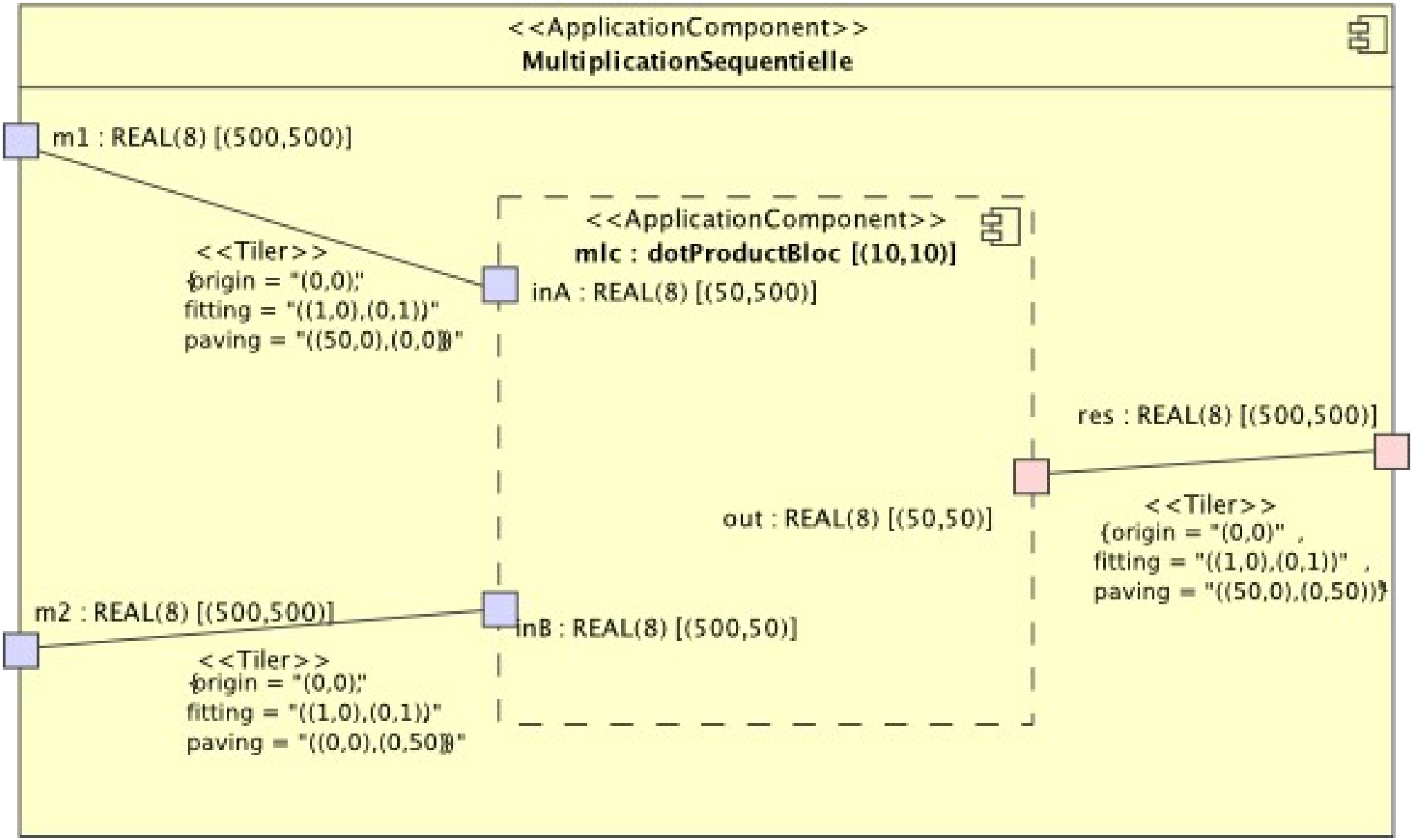
VHDL synthesis

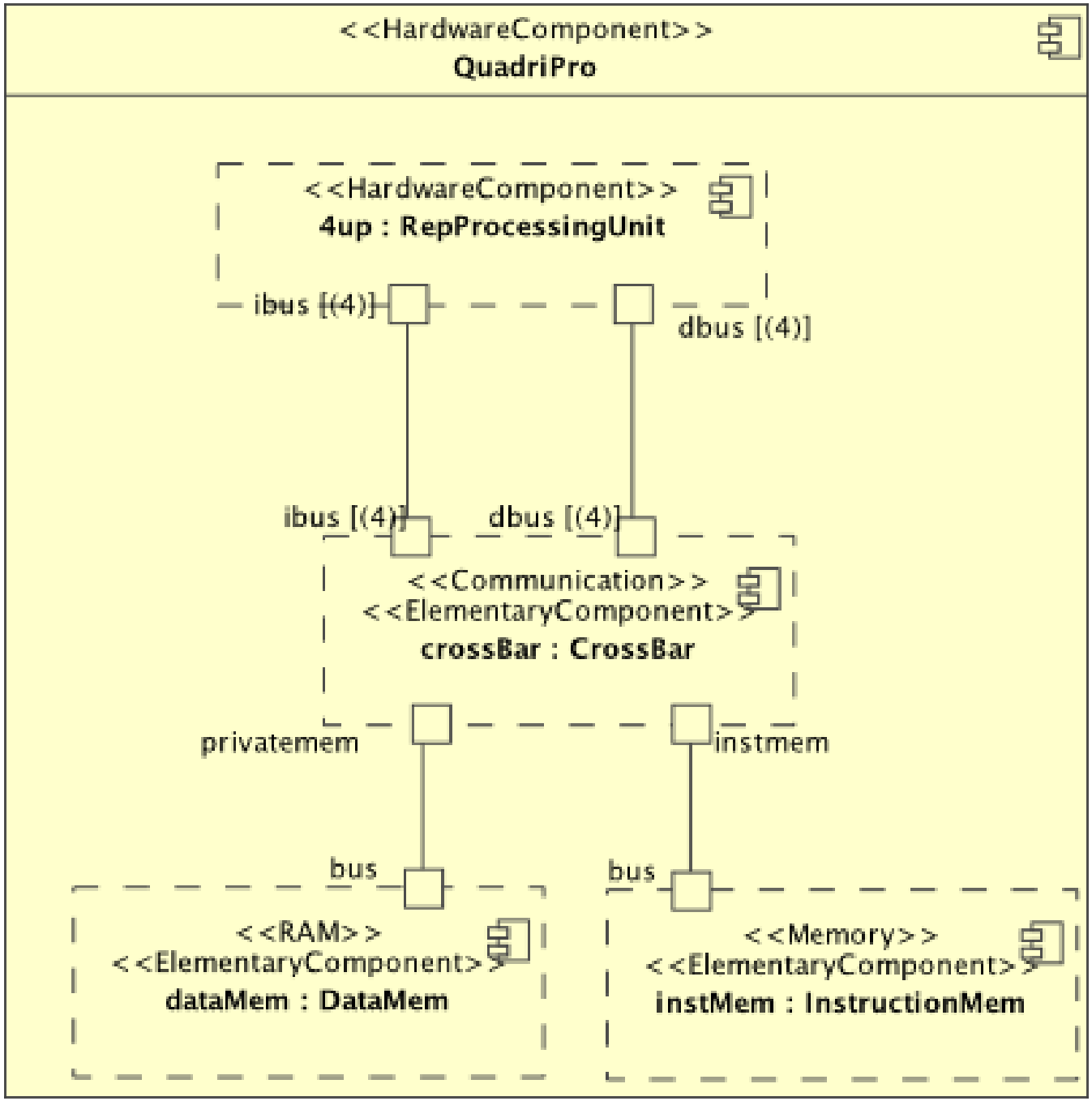


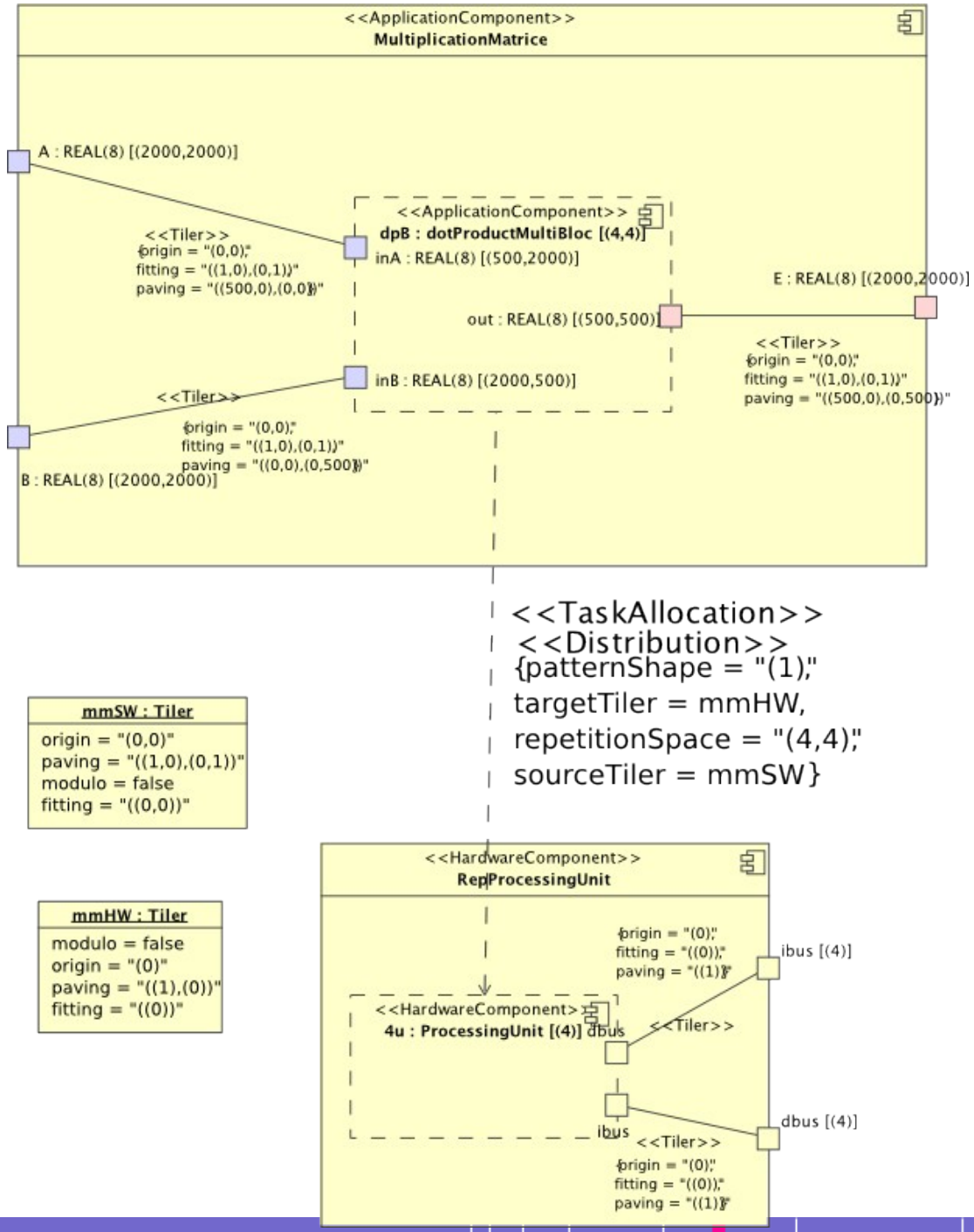

Prototype
Load MAP
Connect
Disconnect
MAP
GPS Information
ATTENTION !
Zone non couverte
Current area
X1 3.133238
Y1 50,61605
X2 3.14514
Y2 50.614004
Speed 50
Current position
Latitude
50,6161316666667
Longitude
3,14264333333333
Speed
0,555 Km/h
Max authoriz
Km/h
Next area
Cruise control
ON
OFF
14 Km/h
Resume
Set
Accelerate
Coast
Clutch pedal
Brake pedal
Accel pedal
Démarrer
MAP
Presario 2100

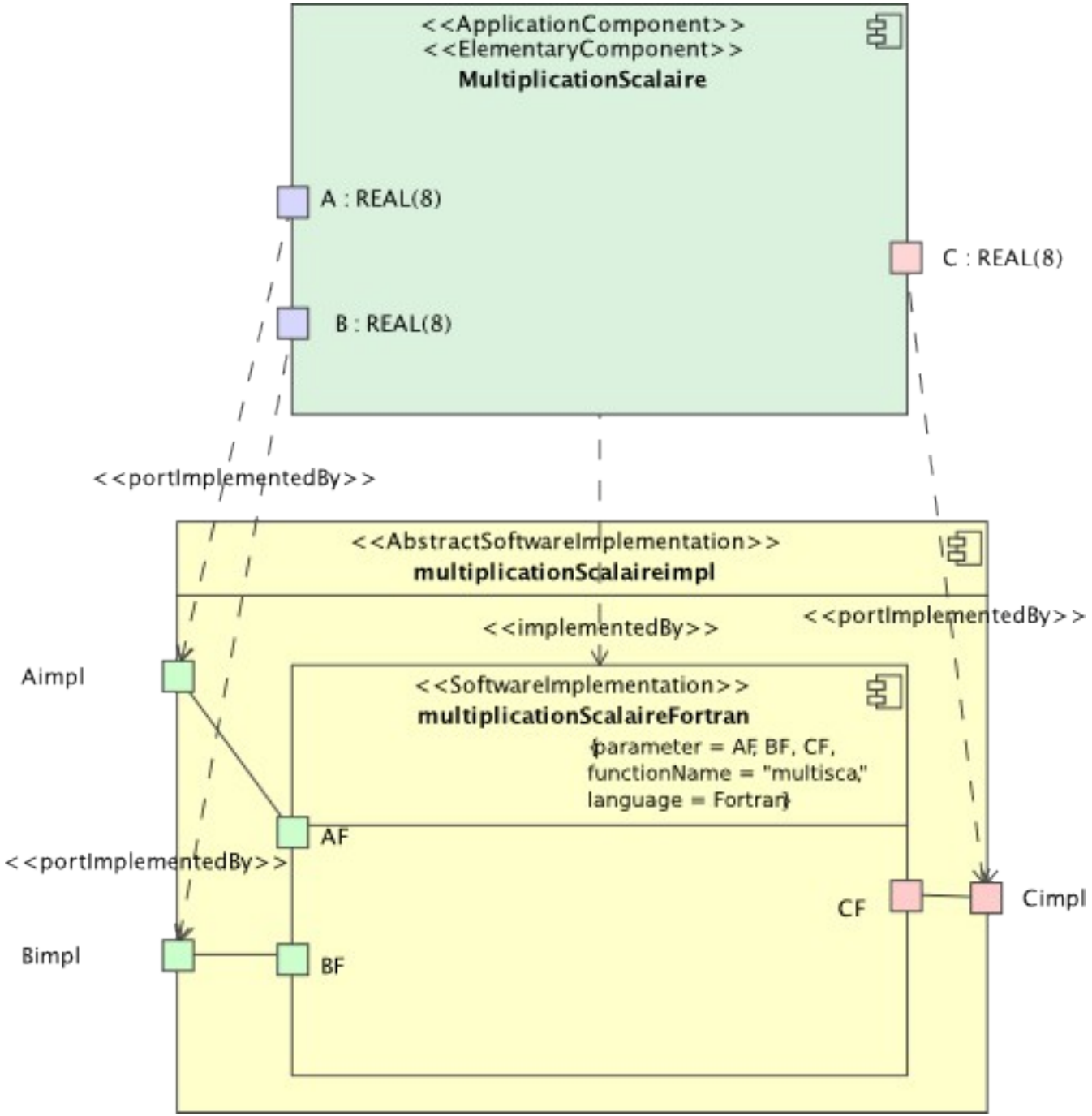
Research continues….

- More MARTE in Gaspard2
- Improve MDE technologies to support metamodel composition
- Validation techniques: Synchronous techniques, test-based techniques
- Automatic architecture exploration
- MppSoC on FPGA as experimentation SoC platform for Gaspard2
- Technology transfer of Gaspard2 (Thales TRT)
- Application also focused on rail transportation (Pôle iTrans)
- Adequacy of Gaspard2 to HPC (Pôle MEDEE)


<<ApplicationComponent>>
MultiplicationSequentielle
m1 : REAL(8) [(500,500)]
<<Tiler>>
{origin = "(0,0)"
fitting = "((1,0),(0,1))"
paving = "((50,0),(0,0))"
<<ApplicationComponent>>
mlc : dotProductBloc [(10,10)]
inA : REAL(8) [(50,500)]
res : REAL(8) [(500,500)]
out : REAL(8) [(50,50)]
<<Tiler>>
{origin = "(0,0)" ,
fitting = "((1,0),(0,1))" ,
paving = "((50,0),(0,50))"}
m2 : REAL(8) [(500,500)]
inB : REAL(8) [(500,50)]
<<Tiler>>
{origin = "(0,0)"
fitting = "((1,0),(0,1))"
paving = "((0,0),(0,50))"


<<HardwareComponent>>
QuadriPro
<<HardwareComponent>>
4up : RepProcessingUnit
ibus [(4)]
dbus [(4)]
ibus [(4)]
dbus [(4)]
<<Communication>>
<<ElementaryComponent>>
crossBar : CrossBar
privatemem
instmem
bus
bus
<<RAM>>
<<ElementaryComponent>>
dataMem : DataMem
<<Memory>>
<<ElementaryComponent>>
instMem : InstructionMem


<<ApplicationComponent>>
MultiplicationMatrice
A : REAL(8) [(2000,2000)]
<<ApplicationComponent>>
dpB : dotProductMultiBloc [(4,4)]
inA : REAL(8) [(500,2000)]
<<Tiler>>
{origin = "(0,0)",
fitting = "((1,0),(0,1))"
paving = "((500,0),(0,0))"}
E : REAL(8) [(2000,2000)]
out : REAL(8) [(500,500)]
<<Tiler>>
{origin = "(0,0)",
fitting = "((1,0),(0,1))"
paving = "((500,0),(0,500))"}
inB : REAL(8) [(2000,500)]
<<Tiler>>
{origin = "(0,0)",
fitting = "((1,0),(0,1))"
paving = "((0,0),(0,500))"}
B : REAL(8) [(2000,2000)]
<<TaskAllocation>>
<<Distribution>>
{patternShape = "(1)",
targetTiler = mmHW,
repetitionSpace = "(4,4)",
sourceTiler = mmSW}
mmSW : Tiler
origin = "(0,0)"
paving = "((1,0),(0,1))"
modulo = false
fitting = "((0,0))"
mmHW : Tiler
modulo = false
origin = "(0)"
paving = "((1),(0))"
fitting = "((0))"
<<HardwareComponent>>
RepProcessingUnit
{origin = "(0)",
fitting = "((0))",
paving = "((1))"}
ibus [(4)]
<<HardwareComponent>>
4u : ProcessingUnit [(4)] dbus
<<Tiler>>
ibus
dbus [(4)]
<<Tiler>>
{origin = "(0)",
fitting = "((0))",
paving = "((1))"}


<<ApplicationComponent>>
<<ElementaryComponent>>
MultiplicationScalaire
A : REAL(8)
B : REAL(8)
C : REAL(8)
<<portImplementedBy>>
<<AbstractSoftwareImplementation>>
multiplicationScalaireimpl
<<implementedBy>>
<<portImplementedBy>>
Aimpl
<<SoftwareImplementation>>
multiplicationScalaireFortran
{parameter = AF, BF, CF,
functionName = "multisca",
language = Fortran}
AF
<<portImplementedBy>>
CF
Cimpl
Bimpl
BF