Compilation and Analyses, Software and Hardware

CASH Joint Inria Team

Christophe Alias, Laure Gonnord, Ludovic Henrio, Matthieu Moy, Gabriel Radanne, Yannick Zakowski

University Claude Bernard Lyon 1 / CNRS / ENS Lyon / Inria (LIP Laboratory)

March 2021

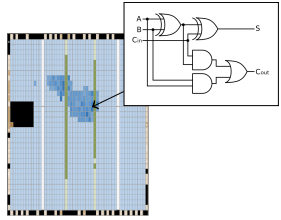
Outline

Outline

CASH Context: High-Performance Computing

- Power-efficiency
  $\rightsquigarrow$ New accelerators (CPU $\rightarrow$ GPU $\rightarrow$ FPGA)

GPU

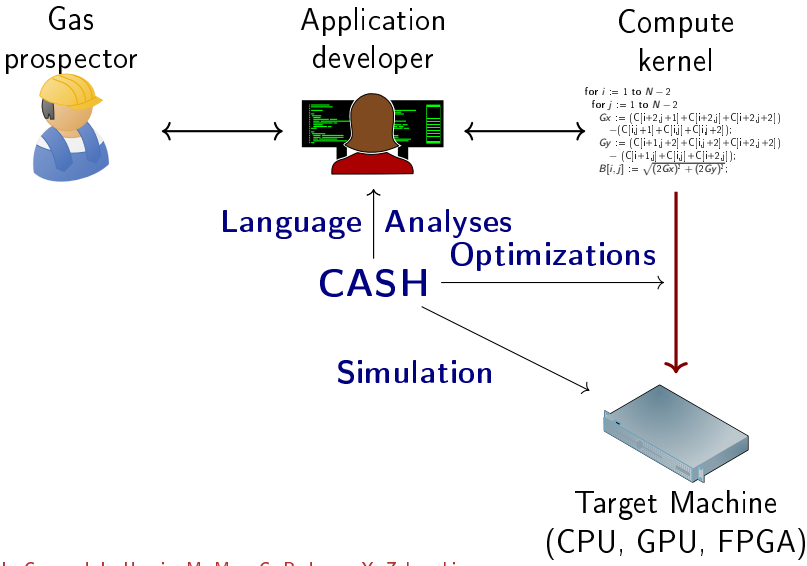


FPGA

- Parallelism is a pain, hardware design even worse $\rightsquigarrow$
  - High-level programming models
  - Express or extract efficient parallelism, including High-Level Synthesis (HLS, C $\rightarrow$ Hardware)
  - Need for code analysis, transformation & simulation

Goal: Efficient parallelism for HPC applications
Target: software & hardware

Our "end-users"



CASH Timeline

2015 C. Alias and L. Gonnord form the "compilation" subgroup of ROMA

2017 Arrival of M. Moy

2018 Creation of the CASH LIP team

Arrival of L. Henrio

2019 CASH is an Inria joint team

2020

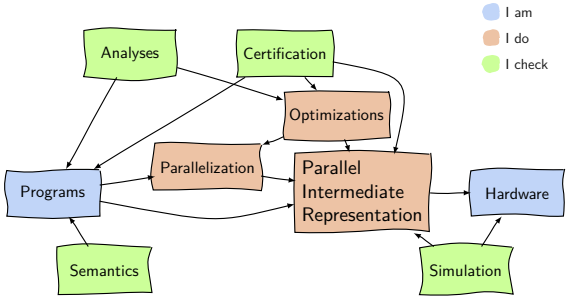
2021 Arrival of G. Radanne & Y. Zakowski

Who?

- Permanent members:
  - Christophe Alias (CR Inria)
  - Laure Gonnord (MCF Lyon 1)
  - Ludovic Henrio (CR CNRS)
  - Matthieu Moy (MCF Lyon 1)
  - Gabriel Radanne (ISFP Inria)
  - Yannick Zakowski (CR Inria)
- Ph.D students:
  - Paul Iannetta
  - Julien Braine
  - Amaury Maillé
  - Julien Emmanuel

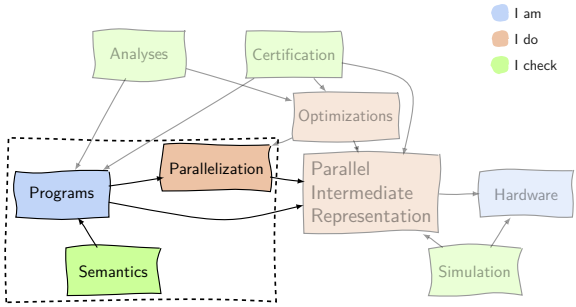
Outline

CASH: Compilation and Analysis, Software and Hardware



1. Parallel and dataflow programming models
2. Expressive, Scalable and Certified Static Analyses
3. Optimizing Program Transformations
4. Simulation and Hardware

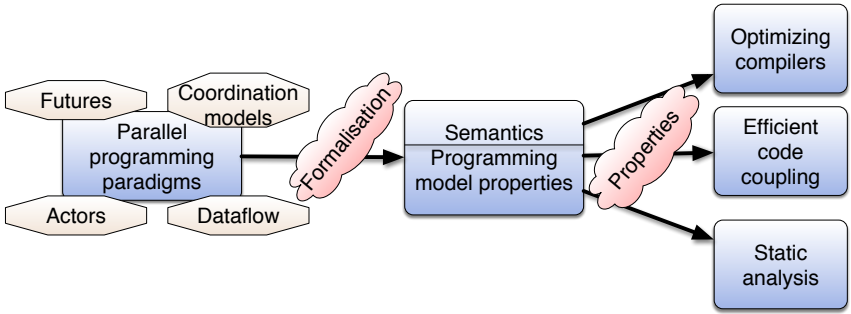
CASH: Compilation and Analysis, Software and Hardware



Direction 1

1. **Parallel and dataflow programming models**
2. Expressive, Scalable and Certified Static Analyses
3. Optimizing Program Transformations
4. Simulation and Hardware

Parallel and Dataflow Programming Models



Challenges:

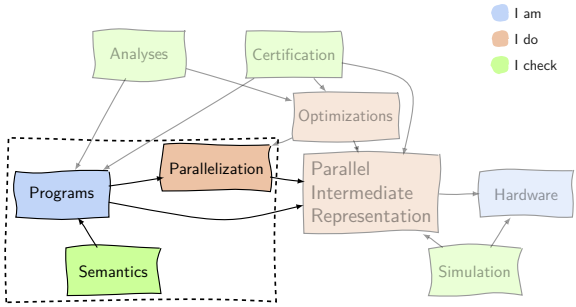
- Different levels of granularity
- Different paradigms (shared/distributed memory, message passing)

Parallel and Dataflow Programming Models

```
{ Act a,b; Fut«Int» y,w; Int z;
 a=new Act(); b=new Act();
 y = a.fact(3,1);
 w = b.fact(y,1);
 z = get y }
```

- Scientific Focus:
  Dataflow Explicit Futures
  [ECOOP'2019] implemented in the Encore compiler.
  - Future: programming construct for safe parallelism
  - Dataflow Explicit Future: new construct, minimizes synchronization points
  - First implementation in a language with parametric types
  - Proof of correctness of an optimization (using bisimulation)

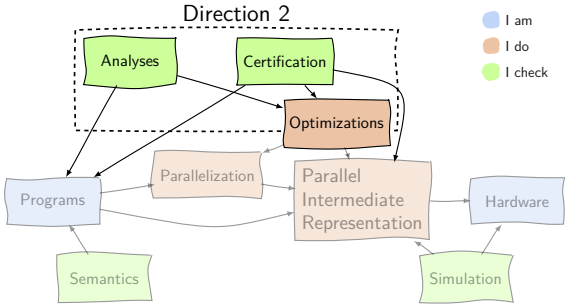
CASH: Compilation and Analysis, Software and Hardware



Direction 1

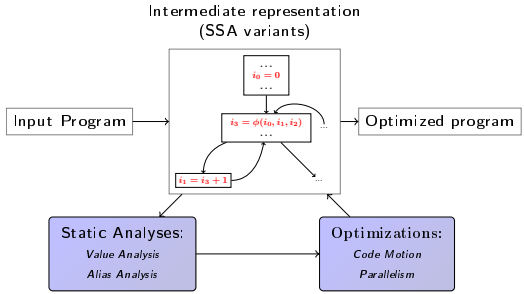
1. **Parallel and dataflow programming models**
2. Expressive, Scalable and Certified Static Analyses
3. Optimizing Program Transformations
4. Simulation and Hardware

CASH: Compilation and Analysis, Software and Hardware



1. Parallel and dataflow programming models
2. **Expressive, Scalable and Certified Static Analyses**
3. Optimizing Program Transformations
4. Simulation and Hardware

Expressive, Scalable and Certified Static Analyses

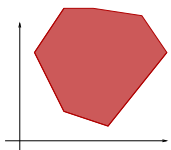


Challenges:

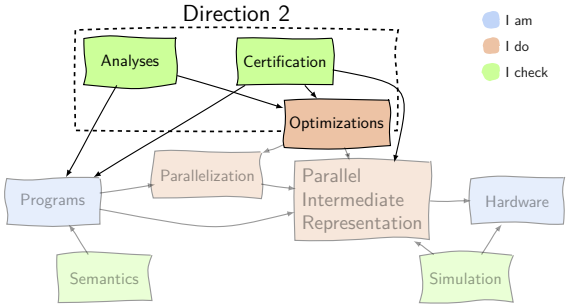
- Scale better than classic abstract interpretation
- Relationship between analyses and optimizations
- Semantics-based vs syntax-based optimizations

Expressive, Scalable and Certified Static Analyses

- Scientific Focus: Polyhedral Invariants for binary code [VMCAI'2019]
  - Variables/Memory: addresses.
  - Abstract interpretation with polyhedral values.
  - Improve precision of existing WCET estimation tools.

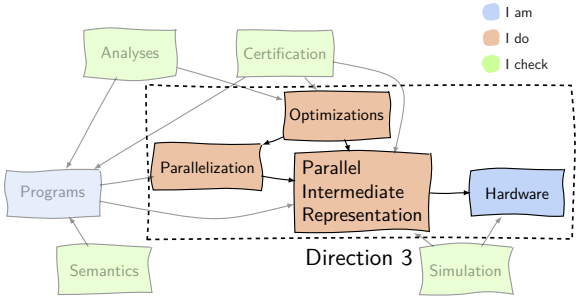


CASH: Compilation and Analysis, Software and Hardware



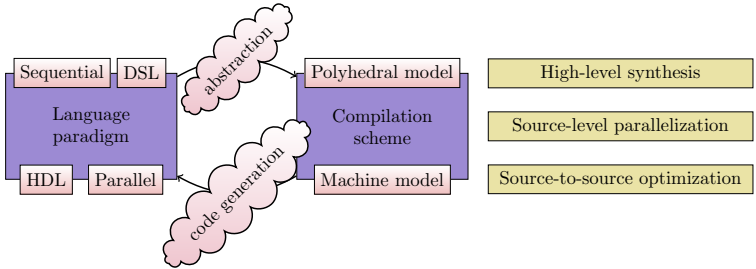
1. Parallel and dataflow programming models
2. **Expressive, Scalable and Certified Static Analyses**
3. Optimizing Program Transformations
4. Simulation and Hardware

CASH: Compilation and Analysis, Software and Hardware



1. Parallel and dataflow programming models
2. Expressive, Scalable and Certified Static Analyses
3. **Optimizing Program Transformations**
4. Simulation and Hardware

Optimizing Program Transformations

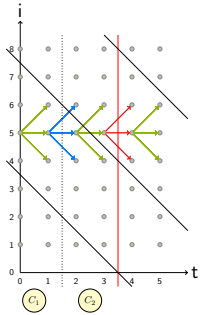


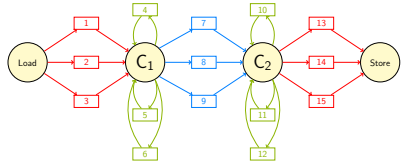
Challenges:

- Automatic parallelization precise and scalable!
- Composition of optimizations
- HLS: No parallel runtime, no cache $\Rightarrow$ complete chain to rebuild.

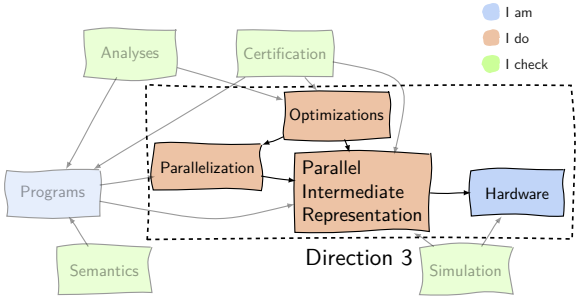
Optimizing Program Transformations

- Scientific Focus: HLS with Data-Aware Process Networks [CC'21]
  - FPGA requires parallelism + data spilling
  - DPN = dataflow model + iteration space partitioning
  - Front-end (C to DPN) and backend (DPN to FPGA)
  - **Industrial transfer:** XtremLogic



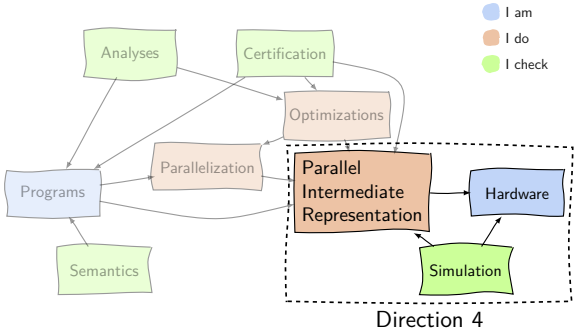


CASH: Compilation and Analysis, Software and Hardware



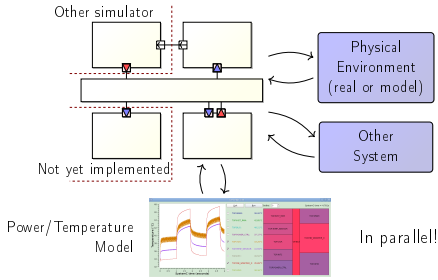
1. Parallel and dataflow programming models
2. Expressive, Scalable and Certified Static Analyses
3. **Optimizing Program Transformations**
4. Simulation and Hardware

CASH: Compilation and Analysis, Software and Hardware



1. Parallel and dataflow programming models
2. Expressive, Scalable and Certified Static Analyses
3. Optimizing Program Transformations
4. **Simulation and Hardware**

Simulation and Hardware

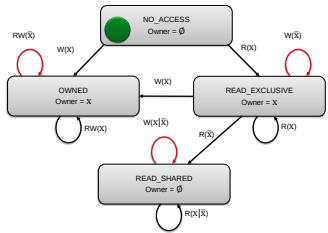


Challenges:

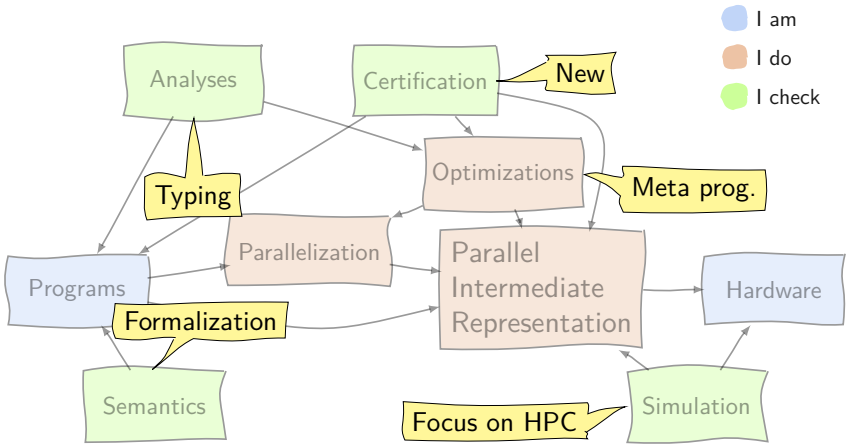
- Heterogeneous simulation (functional, physics, ...)
- Scale up (parallelism)

Simulation and Hardware

- Scientific Focus: standard-compliant parallel SystemC simulation [ASP-DAC'2020]
  - Parallel execution equivalent to a sequential execution
  - Instrumentation of memory accesses
  - Very good performance (x19 on 32 cores)

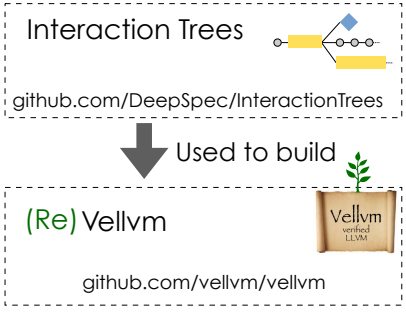


Overall Evolution of Directions



Focus on new Activity: Certified Compilation

Formal guarantees for realistic compilation infrastructures



A *toolkit* to define **and** reason about the semantics of interactive systems

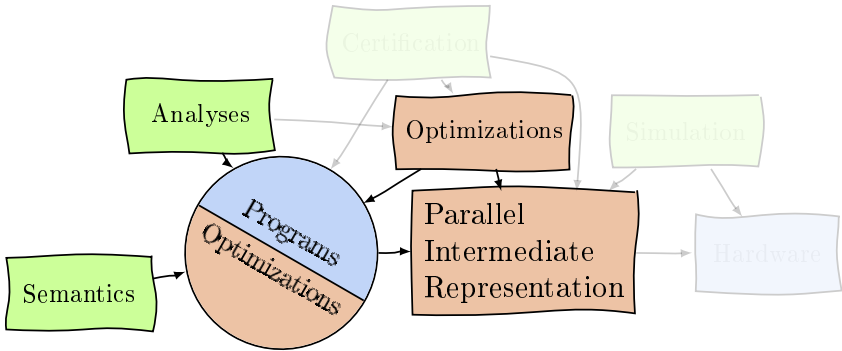
Compositional, Modular, Executable

A formal semantics for LLVM IR

Large case study:
a verified front end for **Helix**

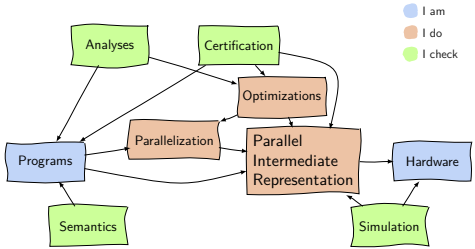
Focus on new Activity: Domain Specific Language for HPC

- **Proposal**: Provide a Domain Specific Language where optimisations are **first class objects**, like functions:
  - Easy to define (no need to change compiler!), compose and abstract
  - Can be customized and adapted to numerous HPC domains



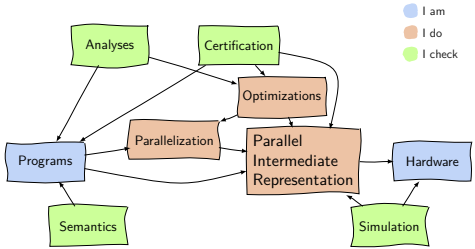
Outline

Conclusion



- Wide spectrum $\rightsquigarrow$ Involvement in several communities & cross-fertilization:
  - Programming languages $\leftrightarrow$ analyses
  - Abstract interpretation $\leftrightarrow$ compilation
  - Compilation $\leftrightarrow$ hardware (FPGA)
  - Compilation $\leftrightarrow$ formal proofs

Conclusion



- Wide spectrum $\rightsquigarrow$ Involvement in several communities & cross-fertilization:
  - Programming languages $\leftrightarrow$ analyses
  - Abstract interpretation $\leftrightarrow$ compilation
  - Compilation $\leftrightarrow$ hardware (FPGA)
  - Compilation $\leftrightarrow$ formal proofs

Thanks!

Outline

Related teams in Lyon

- Within LIP :
  - **Avalon**: same application domain (HPC). Avalon targets application-level programming models, we target compute kernels.
  - **AriC**: arithmetic operators, float to fix point transformation: could be integrated into an HLS flow.
  - **Plume**: dataflow semantics, abstract interpretation, parallel languages semantics and verification
  - **Roma**: scheduling and resource allocation for I/O, throughput and energy, I/O models for FPGA
- CITI:
  - **SOCRATE**: programming models for software defined radio, simulation of SoCs
- LIRIS:
  - **Beagle** (modeling, simulations):
    potential case-studies

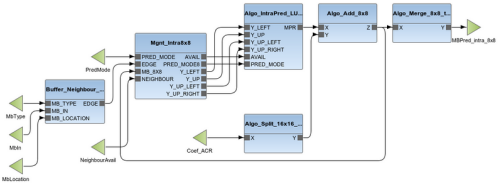
Inria teams in Grenoble

- **CORSE**: Static vs Dynamic compilation
- **CTRL-A** & **SPADES**: formal methods, components.
- **DATAMOVE**: data management for HPC.
- **CONVECS**: languages for concurrent systems.

Other Inria teams

- Compilation, scheduling, HLS:
  - **CAIRN**: HLS for FPGA & polyhedral model
  - **CAMUS**: Compilation, parallelism, polyhedral model (static + dynamic)
  - **PACAP**: Dynamic compilation and scheduling, embedded systems
  - **PARKAS**: Compilation of dataflow programs for embedded systems, deterministic parallelism
- Abstract Interpretation:
  - **ANTIQUE**: Abstract interpretation, data-structures, verification.
  - **CELTIQUE**: Abstract interpretation, decision procedures and interactive proofs

Dataflow and Parallelism



Credo: **dataflow** is a good model to handle complex HPC applications:

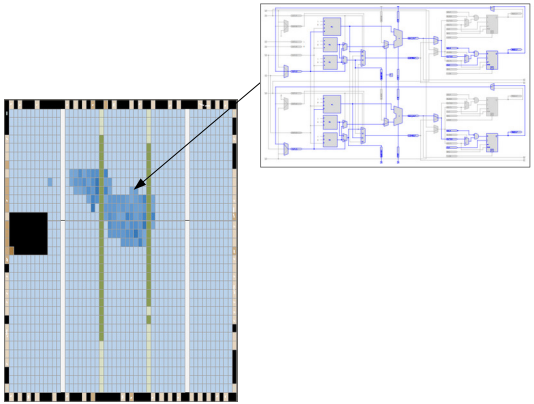
- All the available parallelism is expressed
- Natural intermediate langage for an HPC compiler (compile to/from dataflow program representations)
- Suitable for static analysis of parallel systems (correctness, throughput, etc.)

$\rightsquigarrow$ Dataflow = transverse and fundamental topic of CASH.

Power-efficiency and FPGA

FPGA = reconfigurable circuit

- ⇝ hope: better than GPUs (Microsoft CNN/FPGA: 53% more energy efficiency than GPU implementation).



High-level synthesis (HLS) required!

- ⇝ need for robust, static automatic parallelization

High-Level Synthesis (HLS)

- 1990's — VHDL/Verilog are the only way to produce hardware
- 2000's — Early steps of High-Level Synthesis (HLS):
  - Focus on computation, not communication
  - Marginal raise of abstraction level, semantics unclear
- 2010 — Better input languages and interfaces. Still not adopted by circuit designers.
- 2015 — FPGA become a credible building block for HPC. Industry is now pushing HLS technologies!

FPGA + HLS = best of software and hardware?

Application domain

- HPC (solvers, stencils) & big data (deep learning)
- Typical applications heavily use linear algebra kernels (matrix & tensor operations)
- Examples applications using FPGA
  - HPC: oil & gas prospecting (ex: Chevron, system running on FPGA)
  - Big Data: **Torch** scientific computing framework (Facebook, already has an FPGA backend)

Outline

Main Collaborations

CEA/CITI (Lionel Morel) Compilation and scheduling, polyhedral model (coadvising P. Iannetta)

CEA-LIST (Tanguy Sassolas) Simulation of System-on-a-Chip

Colorado State University (Sanjay Rajopadhye) Automatic parallelization, polyhedral model

Oslo, Uppsala, Darmstadt Semantics & typing of concurrent languages

Linköping Cache coherence mechanisms, verification

Verimag/PACSS (David Monniaux) Proving correction of programs with arrays (coadvising J. Braine)

STMicroelectronics Simulation of hardware

Xtremlogic startup High-level synthesis

Positioning

- CASH = Only compilation-centered team in Lyon
- France: compilation (CORSE, ...), analysis (ANTIQUE, ...), HLS (CAIRN, ...). Particularities of CASH:
  - Emphasis on static aspects
  - Static analysis for compilation
- International:
  - HPC: High-level languages (PELAB, Linköping; programming languages, Uppsala; ...)
  - HLS: VAST, California; System group, London; ...
  - Static analysis: Automatic verification, Oxford; ...
  - Dataflow: Compaan, Netherland; ...
  - Simulation: Rolf Drechsler, Bremen; ...

(Details on positioning in the long document)

Outline

Optimizing Program Transformations

Short/Medium term:

- **Dataflow-to-HLS code generator**

  start with the DPN model (used by XtremLogic)

- Factor channels and control
- Dataflow optimization for throughput

  solved for a single process [MICPRO 2012]

Long term:

- **Models and algorithms for data movement minimization**

  [PhD Plesco 2010]

- Parametrization for scaling parallelization

  Parametric tiling [PhD Iooss 2016]

- Hardware synthesis for dynamic control/data

Simulation and Hardware

Short/Medium-term:

- **Work with CEA-LIST and LIP6 on convergence of approaches**

  ANR Project submitted

- Deal with loose information (intervals instead of individual values for physics)

Long-term:

- **Framework for parallel and heterogeneous simulation: simulation backbone and adapters**

  [PhD Becker 2017]

Compiling & Scheduling Dataflow Programs

Medium-term:

- **Express compilation/analysis activities for the dataflow model.**
- Understand the impact of local parallelism optimization on global performance

Experiments with SigmaC

Long-term:

- **Unify several kinds of parallelism in a same formal semantic framework.**

  Experience on concurrent programming languages, dataflow synchronization, semantics.