Analyses, Hardware/Software Compilation, Code Optimization for Complex Dataflow HPC Applications
CASH team proposal
(Compilation and Analyses for Software and Hardware)

Matthieu Moy and Christophe Alias and Laure Gonnord

University of Lyon 1 / Inria (LIP Laboratory)

November 22, 2017
Who

- Christophe Alias:
  - CR Inria, LIP (temporarily ROMA team)
  - HLS (hardware generation), ...
- Laure Gonnord:
  - MCF Lyon 1, LIP (temporarily ROMA team)
  - Static Analysis, ...
- Matthieu Moy:
  - 2005: Ph.D: formal verification of SoC models (ST/Verimag)
  - 2006: Post-doc: security of storage (Bangalore, Inde)
  - 2006: Assistant professor, Verimag / Ensimag
    Work on SoC models & abstract interpretation
  - 2014: HDR: High-Level models for Embedded Systems
    Shift towards critical, real-time systems on many-core
  - 2015: Synchrone team leader, Verimag
  - 2017: Assistant professor, LIP / UCBL
Scientific Context: Growing HPC Challenges

- Power-efficiency
  \(\leadsto\) New kind of accelerators (CPU \(\rightarrow\) GPU \(\rightarrow\) FPGA)

- Data movement = bottleneck (memory wall)
  \(\leadsto\) Optimize communication and computation

- Programming model: efficient SW and HW implementations
  \(\leadsto\) Express or extract efficient parallelism

\(\leadsto\) Optimized (software/hardware) compilation for HPC software with data-intensive computations
Power-efficiency and FPGA

Best power-efficiency without FPGA
≈ 9.46 GFlops/W
(Cluster of Tesla P100 GPU)

- ≈ 2006: end of Dennard scaling
  ⇒ no more free lunch with energy efficiency!
- 2015: Microsoft achieves 40 GFlops/W with 500,000 FPGA
- 2015: Intel acquires Altera
- 2016: Intel begins shipping Xeon Phi with integrated FPGA

⇝ How to program FPGA?
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 langages and interfaces. Still not adopted by circuit designers.
- 2015: FPGA become a credible building block for HPC. Industry is now pushing HLS technologies!

\[
\text{FPGA} + \text{HLS} = \text{best of software and hardware?}
\]
CASH’s Vision

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

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

\[\mapsto \text{Dataflow} = \text{transverse and fundamental topic of CASH.}\]
Building Blocks of CASH (1/2)

Dataflow models:
- as source language (SigmaC, Lustre, ...)
- as intermediate representation within compilers (e.g. Dataflow Process Network within HLS compiler)
- **Added value**: combination of diverse formal reasoning on programs. Collaboration with Kalray (Many-Core).

Compiler algorithms:
- Heavyweight analysis (polyhedral model and future extensions for irregular applications)
- Low-cost program-wide analysis (abstract interpretation)
- Memory management (minimize data movement)
- **Added value**: experience on design and implementation of scalable analyses
Building Blocks of CASH (2/2)

Hardware compilation (HLS) for FPGA:
- Parallelism extraction from sequential programs
- Scheduling for I/O optimization and latency hiding
- **Added value**: 4 years of case-study-driven research (Xtremlogic startup, co-founded by C. Alias)

Simulation of Systems on a Chip (SoC):
- Fast simulation of large SoCs
- Parallelization of simulations
- Heterogeneous simulations (functional + physics)
- Application to HLS
- **Added value**: 15 years of collaboration w/ STMicroelectronics
Overview of the Team

Compilation and Analysis for Software and Hardware

C. Alias, L. Gonnord

Polyhedral Model

L. Gonnord, M. Moy

Dataflow semantics

Program

Analyses

HPC data-intensive application

Abstract Interpretation

High-Level Synthesis

Code generation

Simulation

FPGA

General-purpose platforms

M. Moy, C. Alias, L. Gonnord

L. Gonnord, M. Moy

C. Alias

L. Gonnord, M. Moy, C. Alias

C. Alias
Application domain

- HPC (Solvers, Stencils) & Big Data (Deep Learning, Convolution Neural Networks)
- Typical applications heavily use linear algebra kernels (matrix operations, decompositions, ...)
- Examples applications using FPGA
  - HPC: Oil & Gas prospection (ex: Chevron, system running on FPGA)
  - Big Data: Torch scientific computing framework (ex: Facebook, already has an FPGA backend)
Parallel & Heterogeneous SoC Simulation (1/2)

Other simulator

Physical Environment (real or model)

Other System

Not yet implemented

Power/Temperature Model

In parallel!
Parallel & Heterogeneous SoC Simulation (2/2)

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

**Short/Medium-term:**
- Work with CEA-LIST and LIP6 on convergence of approaches
- Deal with loose information (intervals instead of individual values for physics)

**Long-term:**
- Framework for parallel and heterogeneous simulation: simulation backbone and adapters
Dataflow Compiling & Scheduling 1/2

Dataflow program

Parallel Machine

Formal Verification

Parametrization Dev. Interaction

Matthieu Moy and Christophe Alias and Laure Gonnord
Locks:
▶ Different levels of granularities that do not coexist well.
▶ What’s the frontier between static and dynamic?
▶ Many syntax-based optimisations.

Medium-term:
▶ Unify all kinds of parallelism in a same formal semantic framework.
▶ Express compilation/analysis activities for this model.
▶ Implement a proof of concept, validate on literature examples (video algorithms, neuron networks).

Long-term:
▶ Find suitable (intermediate) representations to compile from and to (and a language)
▶ Implement a mature compiler infrastructure/toolbox.
Scalable static analyses for general programs 1/2

Static analyses for optimising compilers: improve accuracy (abstract interpretation) but remain cheap (linear runtime): **sparse analyses**.

```c
void saxpy(float a, float * x, float * y, int n)
{
    for(int i = 0; i < n; ++i)
    {
        x[i] = a * x[i] + y[i];
    }
}
```

Loop can be parallelized only if arrays x and y do not overlap.
Scalable static analyses for general program 2/2

**Locks:**
- Classic abstract interpretation is too costly
- How to design optim-based analyses.
- Many syntax-based optimisations inside compilers.

**Medium-term:**
- Rephrase/revisit syntax-based optimisations in the AI framework.
- Revisit the polyhedral model optimisations.
- Design new low cost analyses.

**Long-term:**
- Find a theoretical framework (SSA-based?) to design scalable analyses.
- Better interfaces for analyses and their clients (optims).
High-Level Synthesis for Reconfigurable Circuits

Dataflow representation

FPGA configuration

Input Program → C-to-dataflow → Dataflow Compilation

Custom parallelism and I/O
Dynamic control/data

Dataflow Compilation

Dataflow Optimization
Control/channel factorization
Fifoization

Synthesis

Cost Model
Fast resource estimation
Roofline model for FPGA
Roadmap

Locks:
- Memory wall: huge computing resources, low memory bandwidth
- Exact dataflow analysis required: dynamic control/data?
- Fine-grain parallelization does not scale well

Short/Medium term:
- Models and algorithms for tuning operational intensity
- Dataflow compilation: channels/control factorization
- Algorithms and hardware mechanisms for static/dynamic parallelization

Long term:
- Scalability: abstractions and parametric parallelization.
- Rephrase polyhedral analysis with dataflow semantics
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
Recruitment strategy

- Junior research and teaching positions $\Rightarrow$ team visibility (GDR, Compilation group, . . . )
- Transfer from other places (several persons potentially interested)
- Non-permanents: Ph.D students, . . . $\Rightarrow$ implication in local courses, internships, . . .
Summary

- **Recent but strong** interest for reconfigurable circuits (FPGA) and high-level synthesis (HLS) in HPC. Ever-growing level of **parallelism** for software implementations.

- **Synergies**:
  - Compilation ↔ abstract interpretation
  - Compilation ↔ Hardware (FPGA)
  - Theory ↔ Practice

- **Industrial partnerships**: STMicroelectronics (simulation), Kalray (many-core), Xtremlogic (HLS)

- **Fertile context**: LIP + Inria + “Fédération Informatique de Lyon”: HPC and theory (AriC/Avalon/Plume/Roma)
Overview of the Team

Compilation and Analysis for Software and Hardware

C. Alias, L. Gonnord

Polyhedral Model

L. Gonnord, M. Moy

Dataflow semantics

Program

Analyses

Abstract Interpretation

High-Level Synthesis

HPC data-intensive application

Code generation

Simulation

C. Alias

M. Moy

FPGA

General-purpose platforms

C. Alias, M. Moy

M. Moy, C. Alias, L. Gonnord