



## PyPS a programmable pass manager

Serge Guelton, Mehdi Amini, Ronan Keryell, Béatrice Creusillet

► **To cite this version:**

Serge Guelton, Mehdi Amini, Ronan Keryell, Béatrice Creusillet. PyPS a programmable pass manager. The 24th International Workshop on Languages and Compilers for Parallel Computing, Sep 2011, Fort Collins, United States. hal-01087303

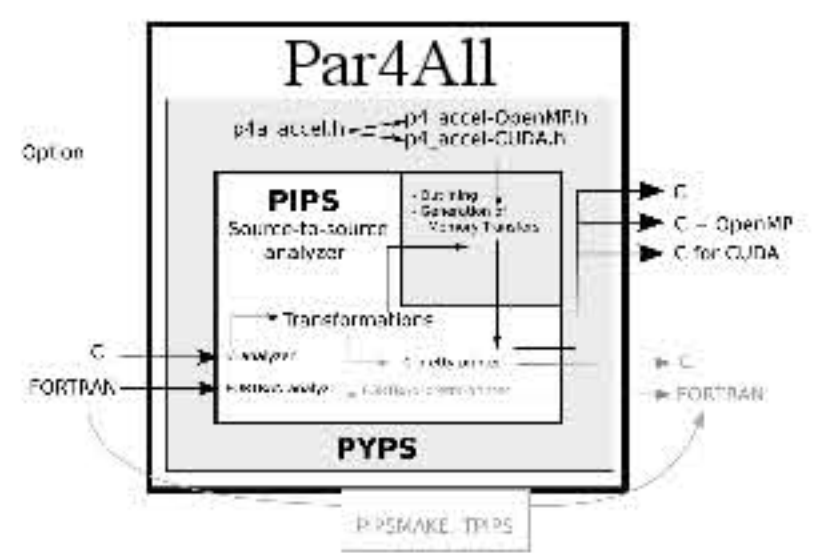
**HAL Id: hal-01087303**

**<https://hal-mines-paristech.archives-ouvertes.fr/hal-01087303>**

Submitted on 25 Nov 2014

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



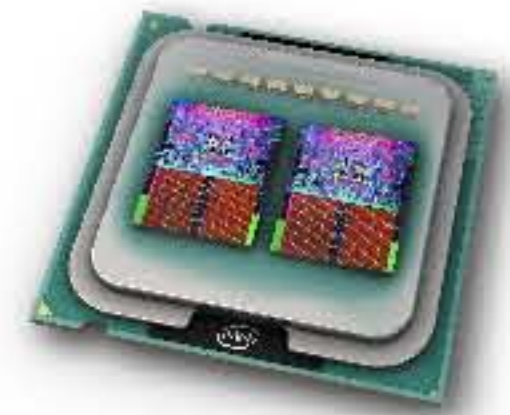
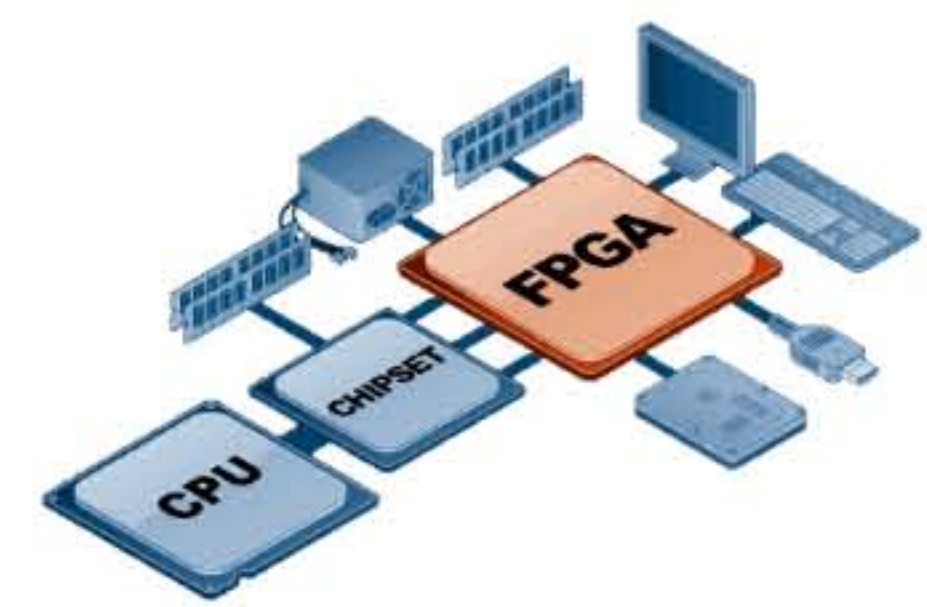
A code transformation is an application  $P \rightarrow P$  that preserves the semantics of the program, that is to say:  
 $\forall p \in \mathcal{P}, \forall v_{in} \in \mathcal{V}_{in}(p), P(p, v_{in}) = P(t(p), v_{in})$



### 1 - Complex Environment



Parallel and heterogenous hardware  
Compilers must be multi-target, collaborate, and be specialized.



### 2 - Source-to-Source

Many source-to-source successful compilers. Flexible transformation systems for heterogeneous computing :  
 → parallelism detection algorithm,  
 → variable privatization,  
 → communication generation,  
 → etc.



### 3 - Model for Code Transformations

Sequencing of code transformations is the compiler core.  
Formal point of view for interactions between passes : several transformation composition rules.

A failsafe operator :  
 $\forall t \in \mathcal{T}, \forall p \in \mathcal{P}, \tilde{t}(p) = \begin{cases} t(p) & \text{if } t(p) \neq \text{error} \\ p & \text{otherwise} \end{cases}$

A failsafe composition:  
 $\forall t_0, t_1 \in \mathcal{T} \times \mathcal{T}, t_1 \circ t_0 = \tilde{t}_1 \circ \tilde{t}_0$


A conditional composition:  
 $\forall t_0, t_1, t_2 \in \mathcal{T} \times \mathcal{T}, \forall p \in \mathcal{P} \quad ((t_1, t_2) \circ t_0)(p) = \begin{cases} (t_1 \circ t_0)(p) & \text{if } t_0(p) \neq \text{error} \\ t_2(p) & \text{otherwise} \end{cases}$

An error propagation operator :  
 $\forall t_0, t_1 \in \mathcal{T} \times \mathcal{T}, t_1 \circ t_0 = (t_1, \text{id}_{\mathcal{T}}) \circ t_0$

# PyPS

## a programmable pass manager

### 4 - Based on a Scripting Language

on the shoulders of giants  `print "Hello, world!"`

No DSL, does not reinvent the wheel, build over a high level language with a rich ecosystem which widens the set of possibilities.

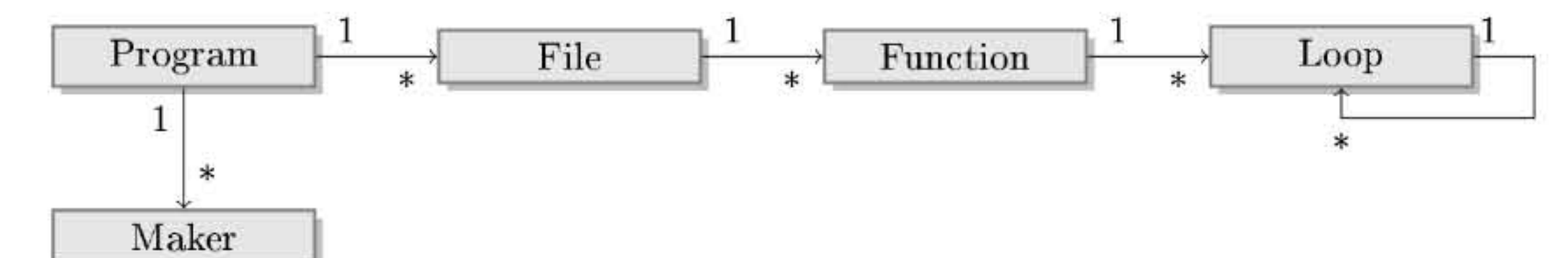
### 7 - Targets

**OpenMP** ; classic scheme but still illustrating basic functionalities.  
**TERAPIX** is a fpga based accelerator for image processing from thales.  
**SAC** vectorizer targets AVX, SSE, or NEON.  
**An Iterative Compiler** ; try different compilation schemes or transformation flavors.  
**Par4All** makes use of other compilers to provide automatic parallelization of applications to multiple hybrid architectures.

### 6 - Control Structures

**Conditionals**: manage switches, choose different compilation schemes.  
**For loops**: iterate over the callgraph, loop nests.  
**Exceptions**: recover from compilation failure, impossible transformations, etc.  
**While loops**: look for fixed point for sequence of passes.

### 5 - Abstractions



**Classes**: high level program representation  
 high level compilation scheme  
**Methods**: compose transformations into more complex ones  
**Inheritance**: compose scheme for heterogeneous targets

### Related Work

Automated Programmable Control and Parameterization of Compiler Optimizations. Yi. CGO 2011.  
 Finding effective optimization phase sequences. Kulkarni et al. LCTES 2003.  
 MILEPOST GCC: machine learning based research compiler. Fursin et al. GCC Developers' Summit, 2008.



Serge Guelton, Mehdi Amini, Ronan Keryell, Béatrice Creusillet

