Explicitly Isolating Data and Computation in High Level Synthesis: the Role of Polyhedral Framework

Riccardo Cattaneo, Gabriele Pallotta, and Marco D. Santambrogio
Politecnico di Milano
Dipartimento di Elettronica, Informazione e Bioingegneria

7 -9 Dec 2015,
Mayan Riviera, Mexico

SIGDA
University Booth

Rationale

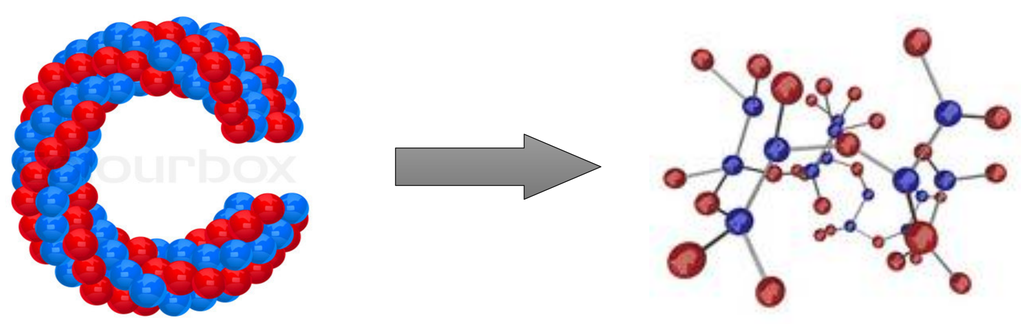
We focus on the exploitation of Field Programmable Gate Arrays (FPGAs) as generic fabric for implementing custom hardware accelerators. This solution exploits the power efficiency of hardware-based computation without losing in flexibility thanks to reconfigurability. The goal is to design a methodology that allows users to specify their workloads into a widely known programming language (C) and to automatically and power efficiently implement those functions on an FPGA-based SC, without the need to directly manipulate Hardware Description Languages (HDLs).

C-to-Parallel C

We analyze serial, statically defined C code, and derive a parallel representation of it using the **polyhedral analysis** framework.

This allows us to automatically generate independent parallel hardware cores. Thus, we are able to compute non-conflicting instructions simultaneously/in parallel.

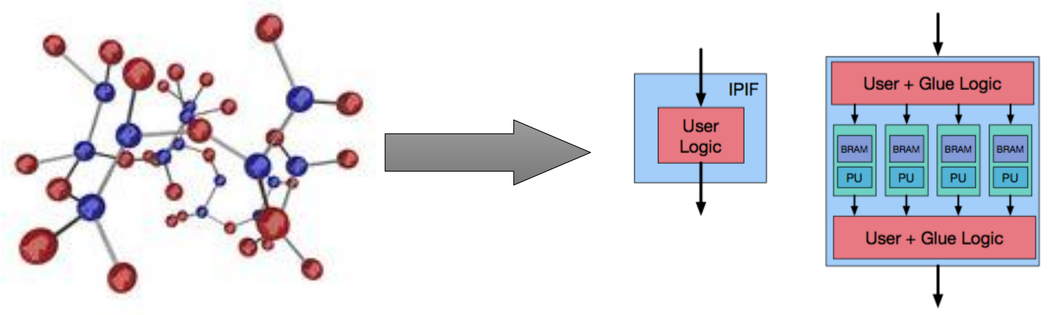
The kernels are also shaped to contain the least amount of memory needed.



Parallel-C-to-HDL

For each such parallel section, we generate the HDLs through adequate **High Level Synthesis** (HLS) steps. Here we can still manipulate the original data types such that we can trade arithmetic accuracy versus logic complexity, or vice versa.

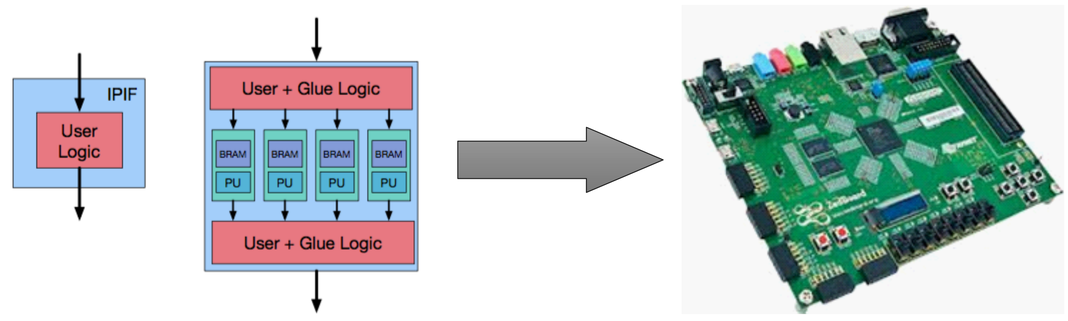
At this step we are able to apply different HLS directives further improving performance.



HDL-to-.bit/.elf

We connect each previously generated module with the other components, respecting the prescribed topology. This allows us to **automatically** derive the **.bit/.elf** files required to run on the Zynq processor by means of synthesis and compilation.

At this step each conflict on the data side is resolved in the main memory of the system, although it is possible to create architectures capable of exchanging data between cores.



Preliminary Results and Future Work

We design a methodology to generate the .bit + .elf file corresponding to a sequential-to-parallel C input program.

To test the overall solution we focused on Polybench workloads, which are highly data-parallel and are HLS-friendly. We managed to easily port most linear algebra and machine learning kernels. All the solutions worked out-of-the-box. With this methodology it is possible to dramatically increase the performance compared to simple kernels created relying only on traditional HLS directives. Not only this solution is orthogonal to all the HLS directives to improve performance but can create a scalable design that can be put on to multiple FPGA.