Abstract: A Highlevel Synthesis Compiler that Optimizes Wire Lengths

We consider a fundamental problem in high-level synthesis (HLS) concerning compilation of programs to circuits with minimized time and wire-lengths product (denoted by $T \times W$). The resulting circuits are executed by FPGAs or as part of ASIC chips and are used as accelerators to speedup the execution of a given program. HLS compilers are becoming essential tools in designing such accelerator circuits for various SOCs of embedded systems, high performance computing applications and data analytics. As an example to this trend of using FPGA acceleration of programs is Intel's Xion cores that contains massive FPGA that is memory coupled with the main core. This core+FPGA is programed using Altera's HLS compiler from OpenCL to Verilog. The Vivado system of Xilinx also contains an HLS compiler widely use to accelerate C programs.

Typically, HLS compilers first minimize the execution time T of the resulting circuit (C), and then separate FPGA/ASIC synthesis tools apply place&route algorithms to minimize W the wire-lengths of this C. Beside being a fundamental problem in the research of HLS systems, minimizing wire lengths is practically significant for:

- As VLSI technology moves to the sub 20nm transistor width long wires in ASIC chips incur larger delays and the ability of the proposed system to minimize wire lengths yields faster and more robust chips.
- As was shown long wires are responsible for significant part of the chips power consumption.
- For FPGAs, minimizing wire lengths simplifies the routing complexity of embedding the circuits' topology over the FPGA routing grid. This implies improved execution times and reduced power consumption.

The problem in developing HLS compiler that minimize both time-delays and wire lengths ($T \times W$ product) is that there is an inherent tradeoff between T and W which is hard to resolved. The few works in HLS that do address this issue propose, basically, iterative applications of separate time-minimization and place&route stages. We on the other hand propose an algorithm that, for the first time, simultaneously minimizes $T \times W$ in a single algorithmic framework.

Let a program be represented as a directed acyclic graph G of arithmetic, logical and memory operations. We developed a formal model wherein the optimal product $T \times W$ of G can be resolved. Through this model, we found the optimal strategies to scheduling+place&route two types of graphs: grids and binary trees. The proposed algorithm for general graphs works by recursively decomposing G to either grid-like or tree-like induced sub-graphs. Based on the type of these sub-graphs (grids or trees), the algorithm compose their optimal solutions backwards obtaining a complete solution to G.

This algorithm will be implemented in the LLVM compiler forming an HLS compiler that minimizes both W and T of the resulting circuit. The $T \times W$ complexity of the resulting circuits will be compared to that of current HLS synthesis systems. We hope to prove that the proposed algorithm will lead to significant

improvements not only in the $T \times W$ product but also in the $T \times power$ product since a significant part of the power consumption of a circuit is due to the cost of interconnections.

There will be two stages in the research: A) compiling G to Verilog/VHDL modules with improved $time \times wire/power$ product and 2) integrate the proposed algorithm with an existing ASIC/FPGA synthesis system (Open source) to measure the benefits of the proposed algorithm. The resulting tool will, not only, academically prove that overcoming the $T \times W$ tradeoff is possible but also yield a practical system that can be used to accelerate C/C++ programs over FPGAs and ASIC chips.