Coarse Grain Highlevel Synthesis a technique to Reducing MUX-complexity

Y. Ben Asher

Abstract

We consider the problem of reducing the number of MUX-gates of circuits synthesized by highlevel synthesis compilers. Our goal is to devise a fast HLS compiler from C to Verilog/VHDL that can be used to accelerate sequential C programs on Intel's Xeon+FPGA machines. Typically an HLS compiler first transforms C-code to a graph of operations G, schedules the nodes of G into a $\overset{rows}{clock\_cycles} \times \overset{resource}{hardware\_units}$ table T and emits a circuit that executes the rows of T at consecutive clock cycles. This technique yields increase use of MUX-gates. This is because the resource hardware units must be reconfigured (through MUX-gates) to access different arguments in each clock_cycle/row of T. Increase use of MUX-gates yields increased routing complexity and a slowdown of the execution. For example, a simple 10lines of C code in Vivado-HLS compiled into a 700 lines of Verilog code, 242 registers and 118 MUX gates. We propose to first partition G into coarser sub-graphs that will be schedule to coarser hardware units containing several operations each. Consequently the number of rows in T will decrease and so is number of the hardware units that are used. We believe that this will reduce the number of MUX-gates and routing complexity of the resulting circuits.

In this proposal we plan to build the following system:

- An LLVM module that compile each suitable loop/functions of a given $C/C++$ program to a graph of operations G.
- An algorithm that can find a good partition of G's nodes/edges to sub-graphs whose HLS scheduling will minimize the use of MUX-gate. Though this is a complex problem we plan that the algorithm will be linear in $|G|$ so that the resulting HLS compiler will be fast.
- A synthesis pass containing the scheduling of the partitioned G to T and the generation of the final Verilog code.