High-Quality, Deterministic Parallel Placement for FPGAs on Commodity Hardware

Adrian Ludwin, Vaughn Betz, Ketan Padalia

Year of publication: *2008*
Area: *CAD*

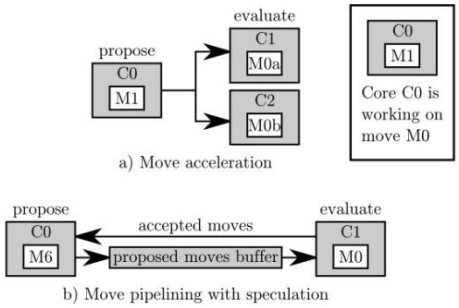

Figure 3: Accelerated vs. pipelined moves.

One of the key issues facing the FPGA industry today is the long compile times for processing designs, brought about by the stagnation in the improvement of processor performance and the increasing size of FPGAs with each new generation of process technology. One of the weapons to combat this trend will be to employ the multiple processor cores that are now ubiquitous on the desktops and servers typically charged with running the compile.

This paper addresses the parallelization of one of the slowest parts of the FPGA CAD flow, placement, and does it in several highly unique and important ways: this is the first and only attempt to parallelize actual industrial placement software, resulting in released, working and successful commercial software. While there have been numerous prior attempts at parallel placement that have been published, they have all been on academic versions of algorithms which don't have to deal with massive device databases, complex timing analysis and the many detailed issues that show up in commercial software.

Second, the paper makes an important statement about the need for *determinism* – that each run of the algorithm gives exactly the same results regardless of how many processors it runs on. Although controversial among academics, determinism is essential in a commercial environment where it is necessary to replicate results both for consistency and debugging. The paper describes the careful work that needs to be done to ensure determinism and shows that the cost in performance loss is small.

Third, the paper performs an insightful analysis of the effect of memory architecture on the performance of the parallel algorithm, and interestingly, shows fairly significant effects that occur from different memory organizations.

The overall performance improvement achieved is impressive and useful: on the placement phase, the speedup is 2.2 times using four cores. On a large design this could save time approaching hours. Since the full compile consists of several more timing-consuming phases, there is more work to be done; this paper is an exemplar for how to do that.

Jonathan Rose

DOI: http://doi.acm.org/10.1145/1344671.1344676