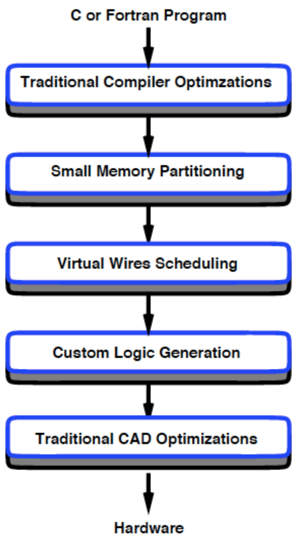
Parallelizing Applications into Silicon

Jonathan Babb, Martin Rinard, Csaba Andras Moritz, Walter Lee, Matthew Frank, Rajeev Barua, Saman Amarasinghe


C or Fortran Program
Traditional Compiler Optimzations
Small Memory Partitioning
Virtual Wires Scheduling
Custom Logic Generation
Traditional CAD Optimizations
Hardware

Year of publication: *1999*
Area: *Tools*

In 1999 FPGA architecture was at the inflection point where the performance of the core computing logic in FPGAs had improved to the point where the communication was fast becoming the blocking factor for performance improvement. A high-level synthesis approach that assumed that wires were for free was bound to fail, and the insights in this paper represent a quick response to the changes in the economics of the underlying architecture. Various prior efforts had proposed compilation from sequential programming languages, but none had really tackled issues of large-scale distributed memories and control.

This paper shows how sequential programs written in C and FORTRAN can be mapped to digital logic while taking care to model the memory organization of an application to try and improve the degree of parallelism that can be automatically extracted. Specifically, the authors attempt to transform one large slow memory into many smaller fast memories by analyzing the memory access patterns in each application. This transformation process enables fine-grained, highly parallel computation with short communication paths and enables a higher aggregate memory bandwidth than is possible with a monolithic memory model. The authors highlight an important realization about the economics of high performance computing systems: the cost of communication is the bottleneck rather than the cost of computation.

The compilation process exploits basic pointer analysis methods to help decompose the heap into regions that can be accessed independently, yielding impressive performance. Experiments in the paper reported a 6x speedup of an MPEG kernel compared to the same algorithm executed on a MIPS R2000 processor. The authors also reported comparisons with the Raw machine using 16 processors which produced an 8x speedup (a hand wired version ran at 32x).

Today engineers expect modern synthesis systems convert C or behavioral HDL code into efficient circuits with judicious use of memory blocks and the ability to process nested loop computations over arrays into parallel fine-grained computations. However, this 1999 paper was well ahead of its time, identifying and solving several issues that lead to a promising synthesis flow.

Satnam Singh

DOI: http://dx.doi.org/10.1109/FPGA.1999.803669