Configuration Compression for the Xilinx XC6200 FPGA

Scott Hauck, Zhiyuan Li, Eric Schwabe

Year of publication: *1998*
Area: *Tools*

	00	01	10	11
00	1	1	2	DT
01	1	1	2	DT
10	1	3	2	3
11	3	3	2	DT

Figure 3. Example for demonstrating the potential for configuration compression.

This paper shows how to automate bitstream compression by exploiting the wildcard addressing scheme in the XC6200. It is an excellent example of identifying a new optimization problem that arises with novel uses of FPGAs, formulating the problem carefully, and providing a sophisticated solution based on a non-trivial reduction of the problem to powerful, known optimization techniques.

Bitstream load time was a small concern for load-once applications. However, as engineers started to consider run-time reconfiguration applications that exploited the fact that FPGA configurations could be changed dynamically during operation, it became a bigger concern---one that could severely limit the cases where run-time reconfiguration was a benefit. Since bitstream load is typically I/O bandwidth limited, one way to reduce bitstream load time is to compress the bitstream; this can be particularly effective for regular applications where many resources are identically configurable. The XC6200 was a short-lived FPGA family aimed at regular computing applications that had an open bitstream format that supported random access into the bitstream. In addition to random access, it allowed wildcard row and column addressing so that the same configuration could be simultaneously written into a set of configurable cells. While it was somewhat clear how to use this for carefully hand-designed logic and layout, it came with no CAD tools to automate the use of the wildcard addressing or automatically identify how to use the wildcards for arbitrary designs.

This paper showed that the problem of determining the minimum set of wildcard writes to configure the FPGA could be reduced to the same set-covering problem that is at the core of two-level logic minimization. Furthermore, it shows how to formulate the ability of the FPGA to overwrite cells as don't-cares in the two-level optimization problem and how to leverage the espresso two-level optimization tool to perform the core covering operation. The result is an elegant optimization formulation and solution for this new challenge that provides a practical tool that can compress regular, hand-crafted bitstreams to about one-fifth their size and CAD mapped designs to about half their size.

The formulation and development of this optimization technique also tells us about the value of the wildcard addressing architecture. Without the algorithm, it was not possible to quantify the actual savings this architectural feature offered. The paper quantifies the benefit on a set of applications as well as introducing the tool needed to perform further evaluations.

André DeHon

DOI: http://dx.doi.org/10.1109/FPGA.1998.707891