

The DARPA Data Transposition Benchmark on a Reconfigurable Computer

S. Akella, D. A. Buell, L. E. Cordova
Department Computer Science and Engineering,
University of South Carolina,
Columbia, South Carolina, 29208.
{akella | buell | cordoval}@cse.sc.edu

J. Hammes
SRC Computers, Inc.
4240 North Nevada Avenue
Colorado Springs, Colorado 80907
hammes@srccomp.com

Abstract

The Defense Advanced Research Projects Agency has recently released a set of six discrete mathematics benchmarks that could be used to measure the performance of high productivity computing systems. Benchmark 3 requires transposition, bit by bit in blocks, of a long bit-stream. The problem definition for this benchmark is given below.

- Let $\{A_i\}$ be a stream of n-bit integers of length L. Consider each successive block of n integers as an $n \times n$ matrix of bits. For each such matrix, transpose the bits such that bit $b_{i,j}$ is interchanged with bit $b_{j,i}$.
- Output the resulting stream $\{A'_i\}$
- Parameters for the three sets of sub-benchmarks of the data transposition benchmark are:

Length of Stream	Bit-width (n)	No. of Iterations
10^7	32	400
10^7	64	230
10^7	1024	12

In this paper we:

- Discuss the design and implementation of the data transposition benchmark on a high performance reconfigurable computer – the SRC Computers SRC-6.
- Look at various design implementations and exploit fine grained parallelism to develop parallel constructions of the benchmark on the reconfigurable platform to obtain better performance.
- Evaluate the performance of this machine by benchmarking our implementations against a standard C based implementation of the same algorithm on a single processor Pentium PC.
- Discuss and present the obtained results. Our best implementation provided a speedup of 75 times over a 2.8 GHz Pentium in a standard server.
- We note that our methodology follows that of traditional programming. The initial implementation, entirely in a high level language, was functionally correct but provided insufficient performance improvement. Just as programmers in an earlier era would have then rewritten critical functions in assembly language, we created macros using hardware description languages. For this application, with its high level of bit manipulation inside word boundaries, the HDL macros were critical in providing an additional factor of about three to four times the performance over the high level language on the reconfigurable hardware.