



Vectorising a non-strict data-parallel functional language

Hill, Jonathan; Clarke, Keith; Bornat, Richard

For additional information about this publication click this link.

<http://qmro.qmul.ac.uk/jspui/handle/123456789/4631>

Information about this research object was correct at the time of download; we occasionally make corrections to records, please therefore check the published record when citing. For more information contact scholarlycommunications@qmul.ac.uk

Vectorising a non-strict data-parallel functional language

Jonathan M.D.Hill, Keith M. Clarke, and Richard Bornat

Department of Computer Science
Queen Mary and Westfield College
University of London

The role of a vectorising compiler for an imperative language is to transform the for-loops of a program into the vector instructions of a data-parallel machine. In a functional language, constant complexity map is the essence of data-parallelism, where a function is applied to every element of a data-structure all at the same time. As map can be considered to be an abstraction of an imperative for-loop, the goal of vectorising a functional language is to transform map expressions into vector operations. This paper presents a vectorisation process in terms of transformations to programs expressed in an extended lambda-calculus. Of particular interest is the way in which algebraic data-types are transformed into a form that is susceptible to the synchronous parallel evaluation on a data-parallel machine. Results are presented for a vectorising Haskell compiler that generates code for the CPP DAP, a massively parallel SIMD machine.

[Download Postscript](#)