ABSTRACT

Over the years, it has become increasingly evident that future solutions for high-speed computing can not depend on technology alone. The advances in hardware and software has to be combined and speed in computation has to be harnessed. The work reported in this thesis is a step towards this. An important class of programming languages, Functional Languages, have been implemented on advanced hardware, a parallel Architecture.

The map function, essence of data-parallelism, applies function to every element of a parallel data structure all at the same time. In a non-strict functional language the function applications only occur to those elements required by sub sequent computation. The thesis deals with a non-strict data-parallel evaluation mechanism based upon “aims” that combines the ‘only evaluate what is required’ philosophy of non-strict evaluation with the ‘evaluate every thing synchronously and in parallel’ mechanism of a data parallel paradigm. The technique is novel as non-strictness and data-parallelism don’t fit together. Incorporation of data parallel extensions into lazy functional language is also proposed.

Program transformations inspired by the ‘map’ distributivity law are used to vectorise functional programs. Twelve higher-order parallel algorithms are developed. These algorithms form the building blocks for many parallel applications and express the use of non-strictness in data parallel algorithms. The resulting vectorized programs are compiled into machine code that mimics an abstract machine.

As part of design and development, an unbounded parallel data structure, Parallel Objects with arbitrary Dimensions(PODS), POD comprehension, a frame work within which commands and parallel operations on PODS can be expressed and higher order parallel functions have been developed.