Abstract

Some Studies on Designing an Efficient Data flow Computer

Four major sources of inefficiency in data flow computers have been addressed. These are respectively the problems of unbounded parallelism, tag-size overhead, data-structure related overhead and communication cost. The first three have been resolved by the incorporation of demand driven mode of evaluation on data flow computer.

An architecture, based on an extension of the Manchester data flow machine, has been presented. The main feature of this architecture is that it provides a demand driven evaluation system on a data flow machine for source programs written in the Functional Programming System, FPS. It has been shown that the first three sources of inefficiency of a data flow computer is considerably reduced (or even eliminated).

The performance of the proposed architecture has been studied with simulation experiments. Stochastic Petri Net models for the Manchester as well as the proposed architecture have been given. Experimental results giving performance figures have been obtained.

The demand driven mode of evaluation provides a mechanism for avoiding recomputation of a function. Convinced about its role in reducing inefficiency an attempt to implement this feature on the existing list-processing-oriented data flow architecture has been made. Studies have been made on the impact of this technique in reducing inefficiency.

For the last problem, a problem common in almost all kinds of

parallel processing systems, a simple method for packet arbitration is first given. A new data flow program graph partitioning scheme has been presented.

Keywords and C.R. classifications:

B.1.2 Parallel processors
B.1.3 Data flow architectures
B.4 Modeling Techniques
C.3.2 Applicative languages, Data flow languages
C.3.3.3 Concurrent programming structures
C.4.4 Message sending, network communication
C.4.8 Modelling and prediction, simulation
D.2 Composite structures
E.2.2 Sequencing and scheduling
E.3.1 Lambda calculus and related systems
E.3.4 Functional constructs, programs and recursion scheme
F.2.2 Graph algorithms