GRIP: A parallel graph reduction machine

Simon L. Peyton-Jones, Chris Clack and Jon Salkild
University College, London

Abstract

GRIP – Graph Reduction In Parallel – is a Fifth Generation machine designed to execute functional languages in parallel, using graph reduction. Its design and construction is a project funded by the Alvey Directorate as a collaborative project between University College London, ICL and High Level Hardware Ltd. It is expected that a working prototype will be completed during 1987. The paper gives a brief outline of the principles of graph reduction and of GRIP's architecture.

1 Functional languages and graph reduction

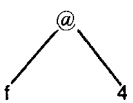
Functional languages are Fifth Generation programming languages which address two of the major current challenges to computer science, those of correctness and parallelism.

The challenge of correctness is the difficulty we experience in writing large, correct programs, a problem which Hoare eloquently outlines in his Turing award lecture[4]. Darlington gives an introduction to functional programming languages[2], showing how their use can alleviate some of these problems.

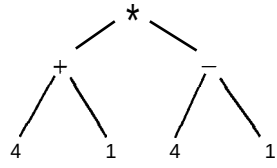
The organisation of a number of independent processors to co-operate in the execution of a single program is the challenge of parallelism. Functional languages contain inherent parallelism and so are a suitable medium in which to express parallel programs. To understand where the parallelism comes from we will look at a functional program:

```
let f x = (x + 1) * (x - 1)
in f 4
```

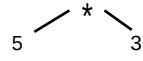
The "let" defines a function "f" of a single argument "x", which computes "$(x + 1) * (x - 1)$". The program executes by evaluating "f 4", that is, the function "f" applied to 4. We can think of the program like this:



where the "@" stands for the function application. Applying f to 4 gives



We may now execute the addition and the subtraction simultaneously, giving



Finally we can execute the multiplication, to get the result

15

From this simple example we can see that

(i) Executing a functional program consists of evaluating expressions
(ii) A functional program has a natural representation as a tree, or more generally a graph.
(iii) Evaluation proceeds by means of a sequence of simple steps, called ***reductions.*** Each reduction performs a local transformation of the graph, hence the term ***graph reduction.***
(iv) Reductions may safely take place simultaneously since they cannot interfere with each other.
(v) Evaluation is complete when there are no further reducible expressions.

Graph reduction is described in detail by Peyton-Jones[5].

GRIP – Graph Reduction In Parallel – is designed to execute functional programs by performing parallel graph reduction. Despite the opportunities for parallelism offered by functional languages, only the ALICE project at Imperial College has so far attempted a parallel implementation in custom hardware[1,3]; the main features of ALICE are described briefly in the paper by Townsend in this issue[6]. GRIP is intended to provide state-of-the art performance at moderate cost by extracting the maximum performance from a fast bus. This means that within its performance range GRIP should provide more power for unit cost than more extensible designs, such as ALICE. Our performance target for a fully populated GRIP is one million reductions per second.

2 The GRIP architecture

Most proposals for parallel graph reduction machines look like Fig. 1. The Processing Elements (PE) traverse the graph held in the Intelligent Memory Units (IMU), discovering reducible expressions and reducing them. The

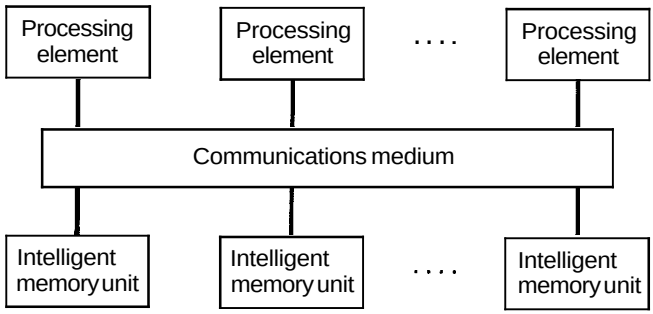


Fig. 1 Physical structure of a parallel graph reduction machine

principle variation between different designs lies firstly in the choice of communications network and secondly in the intelligence in the IMUs.

2.1 The bus

In the case of GRIP we have chosen to use a fast bus, the IEEE Futurebus, for the communications network. A bus offers an extremely cost-effective switch, but at the cost that only one transaction can take place between a PE and an IMU at once, thus limiting concurrency. This places a fundamental limit on the parallelism achievable but gives an extremely cost-effective solution up to this limit. In the case of GRIP we expect to be able to integrate up to 80 or so PEs, on 20 boards, before running out of bus bandwidth and physical space.

The use of a bus allows us to address one research issue, that of parallel reduction, at a time, rather than try to solve several difficult problems at once. By using a bus therefore we expect to exploit a cost/performance/concurrency "window" and to build a working prototype in the relatively short time of 2–3 years.

2.2 The Intelligent Memory Units

GRIP's IMUs will each consist of 5 megabytes of RAM arranged in 40-bit words, with a simple bit-slice microprogrammable processor on the front. Instead of supporting just READ and WRITE commands as normal memories do the IMUs will support an instruction set of high level operations, chosen to support parallel graph reduction. These operations are the unit of indivisibility supported by GRIP (all concurrent machines must provide some indivisible operations to ensure correct synchronisation of parallel activities). In addition, the use of high level operations reduces the requirement for bus bandwidth for communication with the IMUs.

2.3 The Processing Elements

The PEs are autonomous units responsible for performing reductions on the graph held in the IMUs. They will be of straightforward design, based

around a microprocessor, the MC68020, and will include their own private memory which is inaccessible to the rest of the system. The processor within a PE executes a program held in local memory.

2.4 Physical arrangements

Although the PEs and IMUs are logically separate we shall integrate several PEs, and one IMU on each board plugged into the bus. This maximises the use of the bus slots, which are our scarcest resource, and also the number of concurrent activities in the system by putting several on each board.

Most transactions between a PE and an IMU will be carried out on a split cycle basis, to make the best use of scarce bus bandwidth. The PE will write a transaction request into a fast transaction buffer held in the bus interface section. The bus interface will then acquire the bus (which may take some time), send all pending requests to the corresponding buffer on the destination board and relinquish the bus. The request will be processed by the recipient IMU which will write a reply transaction into the transaction buffer, and this reply will then get transferred back to the requesting PE by the same mechanism.

Achieving an implementation of this protocol without imposing substantial latency on transactions is one of the major hardware challenges of the project.

3 Project status

The GRIP project is funded by the Alvey Directorate as a collaborative project between University College London, ICL and High Level Hardware Ltd. Three full-time Research Assistants form the main team based at UCL and led by Simon Peyton-Jones. Work began in the late autumn of 1985; construction of the machine is now under way and the expectation is that a prototype will be working during 1987.

The GRIP architecture is described in more detail in Peyton-Jones[7] and Peyton-Jones et al[8].

References

1 DARLINGTON, J. and REEVE, M.: 'ALICE – a multiprocessor reduction machine for the parallel evaluation of applicative languages'. Proc. ACM conference on functional programming languages and computer architecture, New Hampshire, Oct. 1981, 65–75.

2 DARLINGTON, J.: 'Functional programming'. In: Distributed Computing, Duce, D.A. (editor), Peter Peregrinus, 1984.

3 DARLINGTON, J.: 'Software development using functional programming languages'. ICL Technical Journal Vol.5, No. 3, 1987, 492–508.

4 HOARE, C.A.R.: 'The Emperor's old clothes'. CACM, Vol. 24, No. 2, 1981, 75–83.

5 PEYTON-JONES, S.L.: 'Implementation of functional programming languages'. Prentice-Hall (to be published March 1987).

6 TOWNSEND, P.: 'Flagship hardware and implementation'. ICL Tech. J., Vol. 5, No. 3, 1987, 575–594.
7 PEYTON-JONES, S.L.: 'Using Futurebus in a fifth-generation computer'. Microprocessors and Microsystems Vol. 10 No. 2, March 1986
8 PEYTON-JONES, S.L., CLACK, C.D., SALKILD, J. and HARDIE, M.: 'GRIP–a high-performance architecture for parallel graph reduction', Internal Note 2079, University College London (submitted to IFIP Conference on Functional Programming and Computer Architecture, Portland, 1987).