Distributed Computing (1986) 1:185–186

DISTRIBUTED COMPUTING

Editorial

Distributed computing issues in hardware design

Edmund M. Clarke
Carnegie Mellon University Pittsburgh

The selection of papers in the first few issues of this journal will probably determine to a large extent what areas of study are included under the heading of *Distributed Computing*. It is important for our readers to realize that many interesting problems properly fall under this heading, but deal neither with communications networks nor distributed databases. Likewise, if our subject is to remain a fertile area for research, it is important to encourage prospective authors who may wonder if our journal is the appropriate place to publish a paper on a somewhat offbeat but relevant topic. With these goals in mind I suggested to Mohamed Gouda that we should organize a special issue on hardware design. The six papers that we have included in this issue represent some of the most active areas in that discipline; four of the papers deal with asynchronous circuits, one with data flow computing, and one with systolic arrays. The papers span quite a spectrum as far level of theory and potential applications are concerned. Some deal with new and practical hardware structures, some address fundamental semantical issues, and some propose new methods for implementing algorithms in hardware. However, in my opinion, all expose exciting problems of common interest to researchers both areas.

The first four papers deal with asynchronous circuits. Here the relationship with distributed computing is clear and immediate. In order to ensure that an asynchronous circuit works properly, it is necessary to consider very carefully all assumptions regarding the delays of signals exchanged between various circuit components; ideally the circuit should function correctly regardless of the delays associated with wires or individual circuit components. The advantages of using asynchronous circuits instead of synchronous circuits can be just as impressive as the use of a distributed program instead of a sequential program. Nevertheless, there are enough differences between asynchronous circuits and computer networks so that the theories that have been developed for dealing with the complexity of distributed systems cannot be directly adapted and must be thought through afresh.

The first paper, "The Torus Routing Chip" by Dally and Seitz makes a persuasive argument for the use of self-timed designs. The paper describes a self-timed routing chip for multiprocessor systems based on the k-ary n-cube. A prototype of this chip has achieved an order of magnitude higher throughput than the communications network used by the Cosmic Cube. Deadlock in the network is avoided by a clever algorithm based on *virtual channels*. The basic idea is to duplicate those physical channels that lie on cycles in the channel connection graph. Edges are then deleted from the resulting graph until an acyclic graph is obtained that has the same connectivity as the original graph but cannot deadlock.

Although Dally and Seitz state that the design and testing of their chip was no more difficult than for a synchronous circuit of comparable complexity, this does not appear to be the case in general. Most researchers acknowledge that asynchronous design can be quite tricky. In his paper Jan Tijmen Udding tries to lay a solid foundation for the study of asynchronous circuits by carefully defining what it means for such a circuit to be *delay-insensitive*. The behavior of a circuit and its environment is modelled as a collection of traces where each trace is a finite sequence over an appropriate alphabet of input and output symbols. Axioms for delay-insensitivity are given in terms of traces and illustrated by examples. The axioms lead directly to

For photograph and biography see Distributed Computing (1986) 1:150–166

the controversial conclusion that it is *impossible* to have a delay insensitive *fair* arbiter.

The paper by David Black also addresses the question of whether it is possible to nave a delay-insensitive fair arbiter, but reaches exactly the opposite conclusion from Udding. According to Black, the problem with Udding's negative result is that his theory only permits finite traces and is therefore not expressive enough to deal properly with most interesting notions of fairness. By extending Udding's Work to handle infinite traces, he shows that it is indeed possible to have a delay-insensitive fair arbiter. Black also introduces new composition operators that simplify some of the constructions in previous papers on trace theory.

In the last paper on asynchronous circuits Alain Martin describes a methodology for compiling delay-insensitive circuits from high-level, programming language specifications. An algorithm is initially specified as a network of communicating processes in a language that is similar to Hoare's CSP. The algorithm is then translated into a circuit by a series of transformations which guarantee that if the original algorithm is correct then resulting circuit will be both correct and delay-insensitive.

The paper by Keller and Panangaden presents a trace semantics for data flow networks that contain indeterminate modules. A data flow module is *determinate* if each output stream is a mathematical function of the input streams. A module is *indeterminate* if such a functional relationship does not exist. When determinate modules are composed, the result is always a determinate module; consequently, it is fairly easy to give a good semantics for networks that are composed entirely of determinate modules. Networks containing indeterminate modules can exhibit surprising behaviors and are much less well understood in spite of half a dozen years of research. Since indeterminate modules can occur in realistic data flow networks (e.g. imagine a load balancing module in a data flow network with four possible multiply units), research on this problem is important.

The final paper, "Global Operations on the CMU WARP Machine" by H.T. Kung and Jon Webb, discusses algorithm implementation on a programmable systolic array machine being developed at Carnegie Mellon University (the "WARP"). Like other designs for systolic array machines WARP can efficiently perform local operations in which each output depends on a small number of contiguous inputs. However, some important algorithms like the Fast Fourier Transform (FFT) and various image warping algorithms used in computer vision require global operations in which each output can depend on any of the inputs. Kung and Webb show that such operations can also be efficiently implemented on WARP, thus extending its usefulness to a considerably larger class of algorithms.

Edmund M. Clarke
Carnegie Mellon University
Pittsburg, PA 15213, USA