MAY84

System Implementation Languages: Experience and Assessment University of Kent September 1984

This paper gives a good discussion on the ideas behind the design of OCCAM and the close relationship between Occam and the TRANSPUTER. Occam offers a new approach for implementing concurrent processing systems. It allows a design to be expressed through concurrent processes that operate independently, most of the time. Occam has a small set of primitives and constructs used for concurrency. This set is sufficient, however, to facilitate clear definition of concurrent components in the design of a program.

A close relationship exists between the Occam programming language and the hardware component, the Transputer, in conjunction with which Occam was designed. The transputer is a programmable VLSI device, with memory, processor and communication links, which allows networks to be constructed from transputers in which each component operates concurrently with links being used for inter-transputer communication. Occam syntax reflects this. Systems in Occam are described as collections of communicating concurrent processes. Each process is associated with a transputer. Communication in Occam is via channels. Channels are the Occam equivalent of transputer links.

Use of Occam and the Transputer allows a greater execution speed as a system's size increases. With conventional systems using a global memory accessed by a system such as a bus, there is a degradation in performance as the system size increases. With the Occam/Transputer combination, large systems may be constructed using localized processing and communication. In addition, since the transputers in a network communicate only through links, the memory is allowed to be dedicated entirely for use by the processor. This allows a gain in performance since no communication to an external device is required for a memory access.

Runtime overhead of concurrency in Occam is not a major worry. It is reduced through a number of language restrictions: dynamic allocation is not allowed; recursive procedures are not allowed. In this way the compiler may determine the number of processors required and space needed for each component to be executed in a concurrent fashion.

Occam allows concurrent processes to be implemented easily. A reason for this is that it gives the ability to successively decompose a process into concurrent component processes. Channels play an important role here. Used to transmit data between two processes, a channel must be specifically named and associated with two processes. Once the association has been made, both processes may execute without any knowledge of the details of the other. In addition, the compiler itself enforces proper usage of channels by checking for the following: for a given channel named between two components, one component may only output to the channel, the other component may only input from the channel.

Overall, Occam is approaching concurrency from a more realistic angle than existing concurrent programming languages. While current languages provide simulated concurrency on a single machine, Occam and the transputer provide real concurrency through the execution of a single task by many communicating computers.

The design of the Transputer and Occam should be viewed as important developments. They allow for easy design of concurrent systems, increase performance in concurrent environments and allow real as opposed to simulated concurrency.