Monarch Parallel Processor: Assessment of Programmability

September 8, 1988

This report presents my assessment of the programmability of BBN's Monarch parallel processor. While learning to use this machine, I have taken special care to track the intellectual obstacles I have encountered and to reflect upon how difficult they have been to overcome. I have chosen a programming project which demands a variety of techniques from parallel algorithms and constitutes a rigorous test of Monarch programmability.

Having no prior experience with parallel programming, I have spent approximately forty hours learning Monarch C and developing software which can be compiled and run on BBN's simulator. Initially, with help from the Monarch engineers, I concentrated on developing simple programs for summing large lists of integers and performing matrix multiplications. These projects occupied about half of my time and allowed me to become familiar with the handful of techniques required for successful parallel programming. I have devoted the remaining time to my primary project: implementing a parallel graph algorithm for computing minimum spanning trees (MST). Toward this end, I have developed nearly 300 lines of Monarch C code. In the course of generating this code, I have found it possible to take full advantage of the Monarch's MIMD architecture and achieve the asymptotically optimal speedup for parallel MST algorithms on shared memory machines.

Computing minimum spanning trees has been an active area of research in parallel computation and many algorithms for the problem have been published. I found that I was able to implement the fastest of these algorithms in Monarch C using just the basic techniques I learned from BBN's engineers. The steal operation proved to be a powerful tool for synchronizing processors and implementing the algorithm's crucial logarithmic combining phases. Similarly, parallel do loops proved easy to develop and allowed me to allocate processors to disparate tasks simultaneously. In this manner, I was able to bring the full force of Monarch's powerful MIMD architecture to bear on the inherent parallelism in the spanning tree problem.

In assessing the Monarch's programmability, it is important to bear in mind that parallel programming techniques are richer and more powerful than their sequential counterparts. There are new abstract concepts which need to be learned. However, my experience with the Monarch indicates that parallel programming concepts are no more difficult to master than the basic sequential concepts taught in introductory programming courses. Newcomers to Monarch programming may initially stumble over logarithmic combining and the steal operation. But after some practice, they will find these concepts no more difficult to master than pointers, loops, recursion, or other basic ideas of computer science.

In conclusion, I have been very favorably impressed with the programmability of the Monarch. While successful parallel processing does require mastery of some new programming concepts, these are not difficult to learn on BBN's Monarch. I am confident that competent programmers will experience little difficulty learning to program this machine.