MONARCH AND MONARCH-RELATED PROJECT RESULTS
(12/28/87)

The Monarch parallel processor design is aimed at producing a multiple instruction stream, shared memory computer capable of 6 MIPS and 2 scalar-megaflops per processor in configurations having approximately 10^2, 10^3, 10^4, and 5×10^4 processors. The machine will use 64-bit arithmetic operands and in the above configurations should be capable of 600, 6,000, 60,000 or 300,000 MIPs and one-third as many scalar-megaflops. This is an aggressive machine design for which the implementation is in progress but not yet complete.

In the course of our parallel processing work on Monarch and a potential follow-on to Monarch, we have produced many valuable interim results. These include new and novel ideas, techniques, and artifacts. Some are valuable new electrical engineering, some are new forms of algorithms, and some are new approaches to machine design. We have listed some of them here with a brief description of each.

Monarch System Design - The Monarch illustrates how a parallel computing system can be designed around the interconnection mechanism rather than around a processor or a memory. It also illustrates the value of coordinating chip and system design. The Monarch design uses four custom VLSI chips and about an equal number of off-the-shelf chips. With so few chips used many times, it is possible to have a very low-cost system.

Most computers are complex in their architecture because computer architects tend to include special facilities for particular aspects of a problem, (e.g. auto incrementing addressing modes). The Monarch has an exceptionally straightforward structure which may be an example for other designs.

The Monarch is easy to describe — thousands of identical processors access a single multiported memory through a switching network. We expect that such a uniform architecture will let the programmer attack the parallelism in his program more effectively than other designs, removing machine imposed barriers such as declaring shared variables, local memory limits, and the coordination of message traffic. If a programmer can see parallelism in his application, he should be able to put it on this machine.

Algorithms for Large-Scale Parallelism - Our experience with coding algorithms for Butterfly have led to several new ways to approach problems on a parallel machine. A few of these techniques are listed below.

> **Recursion Breaking** - An algorithm which requires a number of steps, where each step depends on the previous result, is referred to as a recursion. We have developed several techniques that split the string of results into pieces so that the parallelism is enhanced.
>
> **Large Memory Techniques** - In addition to a large number of processors, the Monarch has a large amount of memory. We have found many ways to precompute tables that aid in problem solution. Surprisingly, this often helps improve the parallelism in an application. These techniques may be useful for large memory uniprocessors (like the Princeton Massive Memory Machine).

Data Synchronization - The **steal** primitive provides efficient synchronization between parallel tasks with much finer grain than has previously been attempted in a shared memory machine.

Algorithms - In addition, we have examined numerical algorithms showing 10,000 way parallelism for dynamic programming, sorting, searching, pattern matching, bit operations and many others. We have now begun to benchmark these algorithms on a simulator (16-processors). The first benchmark simulated gives results better than the theoretical estimate. A thousand processor version of the simulator should be running soon.

Electrical Engineering Results - The Monarch is designed around a high-performance interconnection network. We have developed techniques that allow operation at high data rates using normal CMOS technology. Some of the particular results are listed below:

Dynamic Delay Adjustment - In a normal system design, signalling between components of the machine is either synchronous or asynchronous. In a synchronous design, a central clock establishes when signals should be sent and when they should be received. Normally, the difference in delay between signals and the skew in clock phase as seen by different components limits the speed at which a machine can operate. This forces very special design techniques in the Cray and other machines where circuits are tuned to control skew and drives Cray to physically smaller machines.

The alternative is asynchronous design where the timing of signals is passed with the signals. Unfortunately, this requires arbitration when two events occur at nearly the same time. The uncertainty in which came first gives rise to the "classic synchronizer problem" and introduces additional delay.

With the help of Professor Lance Glasser, we have developed a third approach which is relevant to systems where custom or semicustom LSI is being used. We distribute a central clock to establish an accurate frequency of operation, but introduce an artificial delay into every signal in the machine. This delay is automatically adjusted to correct for any signal skew. We have just recently received a patent on this technique.

This approach has greatly eased the design of the Monarch and has enabled us to consider machines with several tens of thousands of processors. The technique will be of great value to many other system design projects.

High-Performance I/O Pads - The signalling rate of the Monarch will be at least 100 megabits per second, and theoretical analysis, timing simulations, and fragmentary empirical results from test chips indicate the signalling rate may be as much as 400 Mbps. This is unusual in an age where most CMOS is running at 16 to 33 MHz. We have developed special high performance pads in our Monarch design frame to drive transmission lines at these high speeds.

On-Chip Termination Resistors - We have integrated the line termination resistors into the Monarch chips so that we can turn them off when they are not needed. This provides better terminations, significant space savings, and this could dramatically reduce power consumption and thus requirements for heat dissipation. This work is similar to the recent work by Tom Knight (MIT and Symbolics).

High Speed CMOS design - We have been using MOSIS design techniques to make very high-performance CMOS designs for register files, switching networks, and other circuit components. These designs or techniques are at least 10 times faster than most other CMOS designs. The idea that CMOS can't run at these speeds is simply wrong, and our demonstration of high speed CMOS design will add to the list of successful high speed CMOS designs in the DARPA community.

We have theoretical results for each of the above at present and expect to have empirical results after the switch chip comes back from MOSIS (expected in February).

Control Network - In the Monarch design, every custom chip is connected to a monitoring and control network. (This network is in no way associated with the primary inter-processor connection network of the machine.) Initially, this was to allow us to parameterize a few chips and monitor a few statistics. We have been surprised at how powerful this notion has become. For example, using the control network, it is possible to test electrical continuity throughout the machine. Similarly, it is possible to run diagnostics on all of the chips in the machine from the control net directly. This is a level of diagnostic and maintenance support that may be revolutionary, and makes it possible to build reliable massively parallel machines.

The control network also allows control over the configuration of the machine. Switch outputs can be enabled or disabled, error detections recorded, and switch load measured directly. Even minor pieces of the machine can be controlled. For example, each switch has its own random number generator. The control network sets all of these generators to different values at initialization.

Monarch Design Frame - The custom IC's in the Monarch are designed using a common collection of logic around the perimeter of the chip. This logic, which is known as a "design frame" by VLSI designers, deals with the high-speed I/O for the chip, the control network, power, and clocking. Using this design frame approach, a designer can concentrate on the design of the "core" circuitry that is placed inside the design frame. In the Monarch, the actual switching matrix is an example of "core" circuitry.

Once the Monarch design frame is proven, other groups might use it to build components (like a specialized processor) that work with the Monarch. Alternatively, the design frame could be used in other DARPA computer design projects.

Switching Network Design - The Monarch switching network uses switch nodes that have more outputs than inputs (e.g. 6 inputs routed to 8 destinations) to reduce the load on the outputs and reduce contention. The switches have two paths in parallel to each destination so that the probability of conflict is substantially reduced. This changes the network statistics so that it can run at close to full capacity without over congestion rather than having to run at quarter capacity to avoid congestion.

Since we have switches with more outputs than inputs, there are a lot of wires between columns of switch nodes. In order to reduce this, we provide a statistical concentrator which reduces the number of wires between boards and reduces the number of switch nodes required in the second and third columns.

The Monarch design also provides for redundancy throughout. The system can be reconfigured around the failure of any processor, switch, concentrator, or memory subsystem in the machine.

This switching network design is well integrated into the Monarch computer but could be used in other systems as well. Thought of as a switch, it supports 8 Megabytes per second of memory reference on each port and can be built with from 4- to 64000-ports for a total bandwidth of from 32 Million to 0.5 Trillion bytes per second. Imagine using the Monarch switch as a very high performance very local area network, e.g., to connect the LISP machines in the MIT dataflow simulator.

We have designs, theoretical and simulation studies now, and we should have empirical studies by summer.

Processor Design - Our goal has not been to innovate in the design of the Monarch processor, but none-the-less, several innovations have occurred. The processor performs 64 bit operations. However, it is implemented in a serial fashion with 2 bit wide data paths. This has allowed us to hide the latency of the switching network.

The Monarch processor design is mostly compatible with the MIPS core instruction set. However, in its implementation, there are aspects of a very large-word processor. Instructions are matched to the pattern of periodic communication with memory. The processor may execute a memory reference instruction, then execute several register-register instructions or control instructions.

This is a nice enhancement of the reduced instruction set principles, and should be usable to increase several fold the speed of such instruction execution, on our own and other machines.

Memory System - The memory system of a computer is often an underrated component. In the Monarch, the memory system has been designed with as much care as the processor. The memory consists of a large number (half the number of processors) of independent memory modules. Each module is optimized for memory bandwidth through clever use of serial access modes.

The design also includes time multiplexing for better utilization, two-port memory system design, and tagged memory at little extra cost.

Packaging - Our board interconnection ideas have received some informal publicity and deserve a more formal description. Our ideas for controlled-impedance connectors should also be generally interesting.

Tools - We have developed a simulator for the Monarch. We have developed several VLSI tools to aid our design effort and a library of high-speed logic cells.

Services - We have built a small (seven people now) very strong staff of high speed MOSIS CMOS designers. We have an extraordinarily creative team of parallel processing system architects and very extensive experience deploying parallel processors (50 Pluribus systems in mid-70's, 85 Butterflies in mid-80's, 228 parallel processor based Computer Image Generation systems in the last two years) and programming them for numerous applications (several communications systems, sonar signal processing, graphics processing, speech recognition, vehicle dynamics simulation, fluid mechanics, finite elements, AI, etc.).