X-TREE: A TREE STRUCTURED LSI MODULAR COMPUTER SYSTEM

by

A.M. Despain, D.A. Patterson, and C.H. Séquin

Memorandum No. UCB/ERL M78/65

11 September 1978

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

X-TREE: A Tree Structured LSI Modular Computer System

Alvin M. Despain, David A. Patterson and C. H. Séquin
Computer Science Division
University of California
Berkeley, CA 94720

ABSTRACT

The design of a tree-structured general purpose computing system based on multiple monolithic LSI processor chips is being investigated at the University of California, Berkeley. This research has been started to determine how future complex computers can be constructed in a modular way from only a few types of large scale integrated circuit (LSI) chips, how this constraint affects computer architecture, and in return, how architectural issues shape the design of the LSI chips.

Research sponsored by the Joint Services Electronics Program Grant F44620-76-C-0100.

INTRODUCTION

For more than a decade the functional computing power that can be implemented on a single chip of silicon has doubled every year, and this trend is likely to continue.[1] Commercially available microcomputers now include the processor, program and data storage and several I/O ports on a single chip. And in the next decade the capability of such single-chip computers will exceed that of present-day minicomputers with sizeable main memories (1 Megabit). On the other hand, packaging technology limits the number of leads that interface to such a chip to less than about 100; a limit that might expand, at best, linearly with time. A challenge thus exists to design a computer architecture that takes these possibilities and constraints into account to provide the most cost effective computing power.

CONCEPT AND GOALS

We have started to study the design of modular computer systems constructed from a single type of very large scale integrated (VLSI) circuit chip such as can be expected around 1985. Taking the Zilog Z8 single chip processor as a benchmark for 1978, demonstrating the possibility to put a processor, 1K bit of registers, 1K bit of RAM, 16K bit of ROM and 32 I/O lines on a single chip, one can expect single chip computers with 1 Megabit of RAM and 100K bit of ROM will be feasible by 1985.

Such a VLSI chip is taken as the basic building block for a modular computer system with incremental expandability. In our anticipated design, each processor is coupled to several other processors through a fast communication link, typically a parallel bus. Although each node of this system is a self-contained computer in its own right, the function of a particular processor chip depends on its location in this computing system and will dynamically change to best serve the need of the system. Processors near I/O devices or storage devices will contain programs that act as communication controllers, memory managers, or disk or tape controllers. Processors in the next higher levels of the system's hierarchy (see following section) will mainly be used for data transmission and to perform the required computational tasks. Finally, processors near the top of the system's hierarchy are supervising the distribution of tasks and monitor overall system performance. Neighboring processors or "nodes" may interact to share the load for a particular computation - so the exact number of "individual computers" represented by the system will be fuzzy and time variant.

An important requirement is that the system should be modular and incrementally expandable with no limits on size or in the number of input/output devices. The system should be operational for even a small number of processors so that the initial capital investment can be matched to the means available. It should further be upward compatible with new VLSI processors emerging after 1985, so that its overall computational capabilities and performance can be upgraded without the necessity to replace all the already existing nodes with newer processors.

From a functional point of view this system should act as a general purpose mulitprocessing machine. Architecture and hardware should efficiently support high-level languages and advanced concepts such as concurrent processes, modules and synchronization signals. Overall the system's architecture should provide some fault tolerance with respect to the failure of individual nodes or branches (processors or communication links).

INTERCONNECTION TOPOLOGY

Many different multiprocessor connection schemes have been evaluated within the framework of these constraints. Bus and Ring connected structures were rejected since the bus(es) would represent a serious communications bottleneck for a large number of processors, and a crossbar arrangement would become unreasonably expensive. Star-type networks lack individuality and are limited in growth by the control power of the central processor. Fully connected networks or n-dimensional cube arrangements, on the other hand, require more and more connections per processor as the system grows, conflicting with the requirement for easy expandability as well as with the pin limitations of the processor chips. Regular arrays or lattices (othogonal or hexagonal) maintain a constant number of ports per processor as the system grows, and they can be easily expanded by adding processors at the periphery. Their main disadvantage is that the mean distance between processors grows proportional to $\sqrt[n]{N}$ for systems with N nodes in n dimensions. Among the various configurations then, tree structures or cluster bus organizations have appeared to be most suitable to implement the modular, expandable computer system outlined above. In a tree structure the mean distance between nodes grows no more than logarithmically with the number of nodes. A tree can readily be expanded at its leaves. Furthermore the topology of the interconnection graph implies a hierarchical control organization that can be mapped directly onto the physical structure.

While some details of the tree structure are still being investigated and open to discussion - therefore the project name X-TREE - other design parameters have become firmed up: A binary tree seems most advantageous. It needs the least number of ports per node to form the basic tree structure. It has the further advantage that node addresses (node numbers) can be associated in a very simple and straightforward way with the location of a node in the tree. This simplifies the routing algorithms for node to node communication, which can consist of only a few very simple local checks. Input/output devices, except for some operating system control inputs at the root, will be restricted to the leaves of the tree in order to preserve full modularity and recursive formulation of hierarchical control programs. There are never more than two I/O devices connected to the same tree, and these addresses of I/O devices are also given by their position in the tree. The number of bits in the address, starting to count with the leading one, readily indicates the level in the tree to which the particular node or I/O device belongs. Dropping bits from the right readily gives the addresses of the chain of antecedents all the way to the root of the tree, which itself carries address "1".

Additional communication links are included in the binary tree to provide redundant paths between nodes that make the system fault tolerant with respect to the removal of individual nodes or branches. In addition those extra links yield a more uniform message traffic destibution throughout the tree. The optimum set of these redundant branches has yet to be determined. Half-ring and full-ring interconnection schemes are reasonably good contenders. They compare favorably to other interconnection schemes in static evaluations of the mean distances between arbitrary nodes as a function of tree-size.[2] They are now being subjected to dynamic message traffic evaluations in a simulator that randomly generates messages of various lengths, and then routes them to different nodes according to certain user-defined statistics. More complicated interconnection schemes for the redundant branches (such as threading, shuffles or multiple interleaved trees) sometimes perform somewhat better in these simulations but this advantage is offset by the more complicated routing algorithms. These trade-offs are currently being studied.

COMMUNICATION BETWEEN NODES

As long as X-TREE is of reasonable size so that it can be housed in a single room or small building, the communications links between nodes are parallel busses (8 bit data or address, plus control lines for asynchronons handshake operation). X-NODE, the VLSI processor located at each node of X-TREE, requires 4 or 5 complete Bus-Ports for the half-ring or full-ring interconnection scheme, respectively. Limitations in the width and number of the busses connecting to X-NODE are primarily given by the limited number of pins available on a single IC package.

Routing algorithms will be based on a local evaluation of the destination address of the message. Dependent on the outcome of this evaluation the message will be routed to one of the nearest neighbors. Should the communication link or the neighbor itself be defective, a second and third choice for a routing direction are contained in a look up table that implements the local routing algorithm, and the message is thus sent on a detour. In the presence of a large number of missing links or nodes, a sequence of such detours can lead to closed loop paths and the message will never arrive at its destination. A detour counter field is therefore included in the message header, which permits to determine when more than a tolerable number of detours have been encountered. Special action can then be taken, such as purging the message or returning it to the originator.

To minimize the delay associated with the communication between distant nodes and reduce interaction between different traffic streams, multiple streams of messages must be able to flow simultaneously through one node without interference and, for most cases, without local processor interaction. Message destination address decoding, routing, and buffering control should then be implemented in firmware, possibly using a separate controller on the same chip. Complicated trade-offs exist between "path switching" and "message packet routing". To have the best of both worlds, five different types of message headers are introduced. The first type is suitable for short messages and will cause a routing of the corresponding message packets without leaving a permanent trace. Another type sets up a permanent communication path along which subsequent data will flow without the need for repeated address decoding. After a whole block of information has been transmitted, the set up path is torn down by yet another special message header. A special header is also introduced for broadcasting messages to all nodes in the whole tree. This format is useful for start-up and for overall systems control.

SOFTWARE STRUCTURE

A common, very large virtual address space (48-64 bits) is shared by the whole system. The memory on the VLSI chip that constitutes X-NODE functions mainly as a large cache (~ 1 Megabit) within this virtual memory. The actual storage devices are attached to the leaves of X-TREE. The address of an individual word or item is thus composed of the leaf-address to which the storage device is attached and an address defining the location inside the storage device indicated. The operating system is also primarily resident at the leaves, except for the small part that is necessary to start up the system, which is permanently stored in firmware at each node. Parts of the operating system, programs, and data will be paged into the local memory of a processor when needed at that location, and processes will run independently of the particular location where they are executed. In general, the system will try to minimize the transmission of bits between nodes. Rather than repeatedly moving a rather bulky block of data to a distant processor, the program, which in many applications is considerably smaller than the data, will be moved towards the data. Therefore, in data-base applications the actual data manipulations will be performed in the nodes close to the leaves to minimize the number of bits transmitted between nodes.

On the other hand, complicated iterative algorithms on a piece of data small enough to fit inside one processor can be performed almost anywhere in X-TREE. If the large data or program memory is too small to hold all information pertinent to such a calculation, a cluster of N neighboring processors will share the load, thus multiplying available memory space by a factor of N. Of the N available data processing units in such a cluster of N nodes the processing will be done in the ALU and in the process registers on the same chip with the memory that holds the program part that is currently being executed. Again this is done to minimize inter-node communication and to maximize execution speed. In such applications X-NODE functionally looks like an "intelligent memory" with on-chip -- and therefore fast -- processing capability.

The nodes at the top of the hierarchy (near the root) will mainly serve in supervisory functions. They monitor the operation of the whole system, and depending on the state of affairs (message traffic density, number of defective nodes or links) they may broadcast guidelines to all nodes concerning routing strategies, the presetting of the detour counters, and algorithms for task sharing among neighboring nodes.

To alleviate some of the software problems typical for multiprocessor systems we intend to design the hardware of X-NODE to provide maximum support for advanced software concepts such as proceeses and data abstraction. Rather than inventing a new language, we intend to adopt MODULA[3] or concurrent PASCAL[4] which have these features built in.

IMPLICATIONS ON HARDWARE

The organization of this tree structured computer system and the novel approach of dealing with movable operating systems and program parts will dramatically affect the design of "X-NODE", the processor that forms the modular building block of this system.

A considerable amount of memory is required at each node so that each processor can execute sizable tasks completely within a single LSI chip, thus enhancing computation speed and reducing power consumption. In the envisioned VLSI implementation of X-NODE, all of the local memory will be implemented in silicon technology on the same chip as the processor, and therefore will be inherently fast. Nevertheless speed differences too big to be neglected exist between high density, dynamic single-transistor-cell memories and static memories using more power and more area. There will thus be a hierarchy of memories even on the same silicon chip, consisting of a dynamic program memory, a fast, static writeable control store and a fast ROM microprogram store for the basic and often used operations.

A writeable control store is important for the described dynamic computer system. To serve efficiently in the various different functions imposed by the location in the hierarchy of X-TREE and by the dynamically changing systems needs, X-NODE must be able to change its"personality". Depending on the task to be performed, the most suitable instruction set is loaded from main memory into the control memory. Since the relative needs for control memory change as a function of time, the possibility is explored to combine several of these storage functions in one and the same physical structure with multiple-access capability.

In a single-chip VLSI processor memory space will always be at a premium, and the controller may occupy a smaller and smaller fraction of the total chip area. Thus other means to increase the effective amount of storage on a chip have to be found. Is the storage density difference between serial memories (CCDs) and RAMs sufficient to warrant yet another level in the memory hierarchy on one and the same chip? Is it worthwhile to use automatic on-chip data compression schemes not only for data but also for programs and even microcode? At the same time, suitable data compression would also reduce the number of bits that have to be transmitted between processors. These ideas are currently being investigated.

CONCLUSION

X-TREE is at an early stage of research. Many basic issues concerning the interconnection topology, the communications protocol and the operating system have to be resolved before we can think about starting an actual hardware design. Obviously, the concept would first be tested with a processor built from commercially available MSI and LSI parts. However, the real pay-off from such a modular LSI approach to a general purpose computer can only be consummated when indeed X-NODE is realized as a single-chip LSI processor.

So far, X-TREE has proven to be an exciting project that attracts the interest of faculty members with quite different backgrounds, and to-date has stirred up many fundamental questions in graph theory, computer architecture, communications, integrated circuit design, operating systems and high level languages.

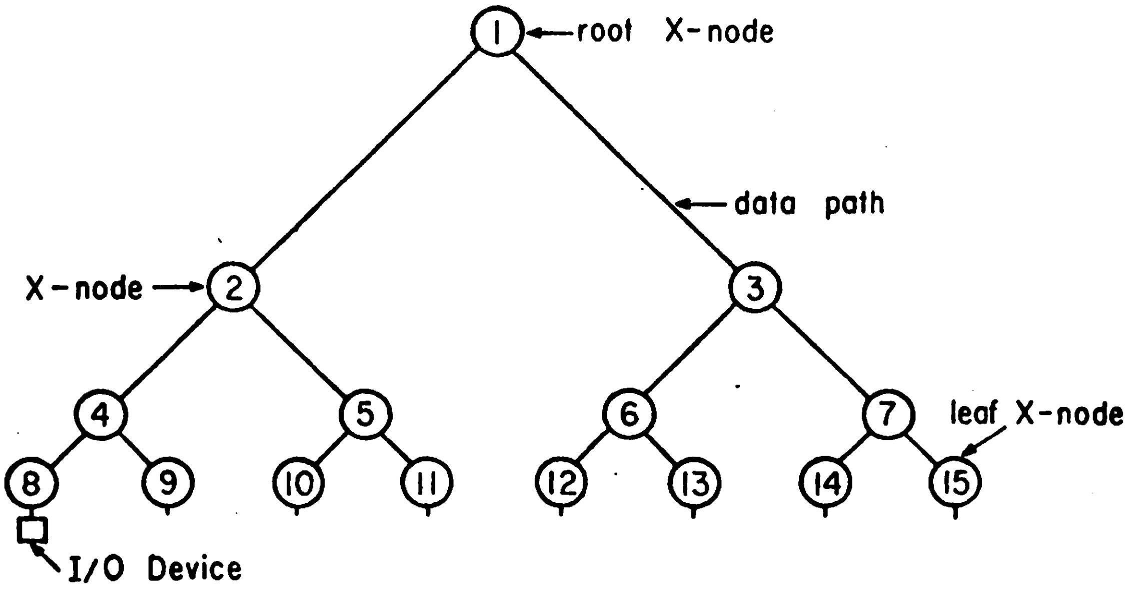
REFERENCES

1) R.N. Noyce, "Microelectronics", Scientific American, Vol.237, No.3, pp.62-69, Sept. 1977.

2) A.M. Despain and D.A. Patterson, "X-TREE: A Tree Strucured Multi-Processor Computer Architecture". 5th Symposium on Computer Architecture, Palo Alto, April 5, 1978.

3) N. Wirth, "Modula: A Language for Modular Multiprogramming", Software - Practice and Experience, Vol. 7, pp.3-35, 1977.

4) P.B. Hansen, "Concurrent Programming Concepts", Computing Surveys, Vol. 5, No. 4, pp.223-245, Dec. 1973.



root X-node
data path
X-node
leaf X-node
I/O Device
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15