LEVEL III

A099374

(1)

7U

R & D STATUS REPORT

Apr. - Nov. 81

13p

AD A110605

ARPA Order No.:	3771
Contractor:	Caltech
Contract No.:	N00014-79-C-0597, ARPA Order 3771 Amount funded - $3,311,894
Effective Date of Contract:	1 March 1979
Expiration Date of Contract:	31 May 1983
Principal Investigators:	Dr. Charles L. Seitz Dr. Carver Mead Dr. Lennart Johnsson
Telephone number:	213/356-6569, 213/356-6839
Short Title of Work:	Submicron Systems Architecture
Reporting Period:	June 30 1981 through September 30 1981

DTIC
ELECTE
S FEB 8 1982 D
H

DTIC FILE COPY

071550

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

California Institute of Technology

Computer Science

SUBMICRON SYSTEMS ARCHITECTURE PROJECT

November 1981

Co-principal investigators: *Charles L. Seitz, Carver A. Mead, Lennart Johnsson*

Other faculty: *Randal Bryant, Jim Kajiya, Alain Martin, Martin Rem*

Ph.D. students: *Chris Carroll, Sheue-Ling C. Lien, Marina Chen, Young-Il Choo, Erik DeBenedictis, Dick Lang, Bob Lewis, Peggy Li, Mike Ullner, Dan Whelan*

M.S. students: *Chao-Lin Chiang, Howard Derby, Erik Holstege, Chris Kingsley, Chris Lutz, Parveen Shukla, Jeffrey Soong, Doug Whiting, George Williams, Winston Wong*

The Submicron Systems Architecture Project is concerned with the architecture, design, and testing of VLSI Systems. The ~~following are our~~ principal activities in ~~the~~ period from April 1981 to November 1981.

The Tree Machine Project

The Tree Machine operating system that was briefly described in the April site report has been documented in an internal document. During the past few months work has focused on improving the efficiency of the loader by having the tree processors performing a larger fraction of the name generation and thereby decrease the amount of code to be supplied through the root. The new scheme is entirely linear in the number of different nodes in the tree. The previous algorithm loaded programs for different nodes once only, i.e., replication of code was made within the tree where needed, but the node names were generated and loaded entirely from outside the tree. Hence, loading time had a term proportional to the size of the tree irrespective of how many nodes were identical. For many problems there are only a few different types of nodes even for very large trees. Additional simulation experience has also been gained.

Documentation: Li, Peggy, "The Tree Machine Operating System", Computer Science, Caltech, internal document (4618), July 1981.

DTIC COPY INSPECTED 2

Accession For
NTIS GRA&I
DTIC TAB
Unannounced
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A

The COPE Machine

Simulation descriptions of Boolean N-Cube, Tree, Chordal Ring and Toroidal Mesh packet switching networks have been written and debugged. Experience with running these simulators with various parameters has resulted in a body of data which bears out the appropriateness of the Boolean N-cube for the COPE architecture. A display file describing the simulation results is forthcoming.

Some new ideas concerning the structure and the garbage collection of the COPE architecture are being considered. Briefly, the tree-like structure underlying the Boolean N-cube might be eliminated reducing cost and complexity. This development requires a rethinking of how the functions of garbage collection and "virtual object space" are managed. Modification of the "rubber banding" heuristics and a different garbage collection algorithm may fill the needs of the modified machine. Investigation of these ideas is underway.

Documentation: Lang, Dick, "A Distributed Class Object Processing Architecture," Computer Science, Caltech, internal document (4199), February 1981.

The Homogeneous Machine

Work is proceeding on the homogeneous machine project. Two boards have been designed, one fabricated and the second is presently being fabricated.

One goal is to have a sufficient portion of the machine working by the end of the year so that software development can begin. Since it is not necessary to have the entire array constructed to develop software, most of the array construction will be deferred. Our plan is to have the dedicated host, the interface processor, and one other processor of the array working by the end of the year.

The second goal is to have the entire machine functional by February. The additional work to meet this goal is to have pc boards layed out and fabricated, to insert chips, and to construct a backplane.

All software work is being deferred until several board level products can be delivered.

Computational Arrays

In the study of linear equation solvers we have so far devised concurrent algorithms for Gaussian elimination with or without partial pivoting, the QR-method based on Householder transformations, and the Discrete and Fast Fourier Transforms. The linear equation solvers are assumed to have a fixed, problem independent number of processors. The focus has been on band matrix problems. Gaussian elimination without partial pivoting in an array with M processors is for a matrix of bandwidth b and size N performed in time O(b*b*N/M). For a sequential machine M = 1, and for a full instantiation in space of the algorithms M = b*b as for the hexagonal and orthogonal arrays. If partial pivoting has to be used and a two-dimensional array is fully pipelined, one dimension is in general "lost". The solution time is on the average O(b*b*N/sqrt(M)). The solution time for the QR-method in a linear array is

O(b*r*N/M) where M equals b for a full instantiation in space.

In studying these algorithms particular attention has been devoted to communication issues, broadcasting, pipelining, and especially the explicit control that is necessary when an algorithm is only partially instantiated in space.

Pipelining often allows the cycle time to be decreased and the overall performance to increase accordingly. When the data flow is not unidirectional due to the existence of loops, an input can not in general be accepted every cycle. Partial results from a preceding set of input data may require a few cycles before they are in a place where they should interact with the new input data. Hence, in a pipelined network with "turbulent" flow data may have to be "spaced out" in time so even though the cycle time can be reduced due to pipelining the total performance may not improve. For example, data can only be accepted every third cycle during Gaussian elimination in a fully pipelined two-dimensional array. For some other operations data can be accepted every second cycle and for yet some others every cycle.

Documentation: L. Johnsson, "A Computational Array for the QR-method", submitted to the MIT Conference on Advanced Research in VLSI, January 25-27, 1982

L. Johnsson, "Computational Arrays for Band Matrix Equations", internal document (4287), Computer Science, Caltech, May 1981.

Johnsson, Lennart and Danny Cohen, "Computational Arrays for the Discrete Fourier Transform," *Proceedings of the Twenty-Second Computer Society International Conference, COMPCON 81*, February 1981.

Switch-Level Model for MOS Logic Design

The simulators MOSSIM (Bryant) and TSIM (Terman) have demonstrated that MOS designs can be simulated as networks of transistor switches rather than of logic gates. These simulators directly model such idiosyncrasies of MOS circuits as bidirectional transistor effects (e.g. sneak paths), dynamic memory, and charge sharing. Furthermore, the network can be derived directly from the mask specifications by a relatively straightforward circuit extraction program (Baker).

Both MOSSIM and TSIM were developed based on the intuitions of the designers and on much experimentation without any formal basis. The algorithms tend to be difficult to describe and to implement and have various quirks with regard to their generality and accuracy (especially in propagating the X state).

In the thesis "A Switch-Level Simulation Model for Integrated Logic Circuits" (report MIT/LCS/TR-259) a formal model of switch-level networks is developed and a set of equations describing the operation of a network in a simple, discrete algebra is derived. Implementing a logic simulator then becomes a process of developing an algorithm for solving these equations, where efficiency is gained by exploiting the sparseness and locality in the network. The resulting algorithm is much cleaner and easier to implement and is provably correct. Its performance should equal or better its predecessors.

In the next few months we plan to implement this new simulator and integrate it into the Caltech design environment. It is also hoped that this more abstract specification of the algorithm will enable others to implement simulators suitable for their own programming languages, machines, and design environments. Furthermore, the switch-level model provides a new methodology for understanding and teaching MOS logic design, and could be incorporated into other design tools.

Documentation: Bryant, Randal E., "A Switch-Level Simulation Model for Integrated Logic Circuits", Laboratory for Computer Science, MIT, Technical Report MIT/LCS/TR-259, March 1981.

Bryant, Randal E., "A Switch-Level Model of MOS Logic Circuits", Proceedings of the First International Conference on VLSI, VLSI 81, Academic Press, 1981, pp. 329-340.

Testing

A report on the test system "FIFI" was distributed at the meeting in April. During the past six months the development has focused on an interactive user interface.

The user interface is based upon a general context-free grammar. Each application for the interface consists of an expanded meta-language that describes the grammar and provides additional information (such as prompts, help text, and error recovery information). There is a program, the parser, that interprets this description and interacts with the user. The output of this program is data structure trees describing the input in a digested form.

The program that assembles parse trees has the ability to prune unnecessary parts of the parse tree, as well as provide data structures with only the appropriate information in the proper places. The parse trees will also have lists, or arbitrary length groups of parsed input.

The following are examples of interface features:

- Prompts: the interface will output a prompt at the beginning of each line that is a mini-help text. For example some prompts may be "FILE NAME>" to indicate a file to type, or "ASSIGNMENT OPERATOR>" to indicate a variation of an assignment operator is to be typed.

- Help text: In addition to providing a description of the lexical objects that are acceptable as input, a semantic description is provided. The semantic description is provided by the designer of the grammar, and implemented with a heuristic in the parser. Such help as "the name of a pin" or "name of a formal parameter to the procedure" are possible. In the event that the input to that point is ambiguous the help will indicate so.

- Recognition: Recognition employs a relatively complex heuristic. Some examples are:

  * Recognition of keywords. This is the most conventional type of recognition. A keyword is recognized from a unique abbreviation of its

name.

* Semantic recognition. If when the language is described a lexical object is referred to a dynamically constructed symbol table then recognition can be done on that table.

* Default fill-ins: Defaults can be provided in the language design for implementing recognition with no input.

Error Recovery

There are two types of error recovery: (1) Error recovery on interactive input and (2) Error recovery when reading a file:

- When accepting input from a terminal it is possible to inform the user that some part of the input is erroneous and dynamically request that it be fixed to conform to the grammar. In the present implementation when a syntax error is detected the current line is output to the terminal with the cursor stopped immediately after the unparsable input, and an error message is output. The user will delete the unparsable input and continue.

- When accepting input from a file it is not possible to have the input corrected and the best action is to try and locate the error and recover. This issue is still under consideration, but it appears possible to detect and recover from any missing or spurious input. The action would be to type to a terminal the expected problem with input and mark the resulting parse tree as invalid.

At present a model of this parsing system is implemented in TOPS-20 SIMULA.

Documentation: DeBenedictis, Erik, "FIFI Test System, Preliminary User's Manual," Computer Science, Caltech, internal document (4270), April 1981.

DeBenedictis, Erik, "A Preliminary Report on the Caltech ARPA tester project," Computer Science, Caltech, Technical Report 4061, April 1980.

Local Network and Designer Workstations

In our project on local computer networks an Ethernet system with several nodes has been built. A large amount of software has been written to enable these nodes to communicate with each other. This includes a debug monitor, an event driven multi-process scheduler, an Internet Protocol package and user processes on top of this. This project has reinforced the effect of software protocols on system performance. Less weight has been put on the physical network protocols than on software protocols. Future development should include the implementation of software protocols as pipelined hardware modules. A noticeable improvement in system performance should be the result.

In the effort to evaluate the feasibility of Apollo Domain computer systems as a basis for designer workstations a PASCAL CIF plotting system has been implemented on our

VAX with plans to port it to the Apollo. This CIF plotting package will be interfaced with our Applicon color plotter. Eventually, the Applicon plotter may be connected to one of the Apollo nodes which could become an Applicon server.

A color frame buffer system has been developed. It should serve at least three purposes. First, it is designed to work on the SUN Workstation and therefore should be an asset to the ARPA community. Furthermore, it should function with the Apollo Domain computers that we have acquired, rounding out that system as a potential VLSI Design Workstation. Also, configurations of SUN Workstations with a color graphics system should fill the needs for stand alone graphics terminals.

Documentation: Whelan, Dan and Ray Eskenazi, "An Inexpensive Multibus Color Frame Buffer", Computer Science, Caltech, internal document (4334), June, 1981.

Whelan, Dan, "A Versatile Ethernet Interface", Computer Science, Caltech, Technical Report 4654, July 1981.

Earl

Earl is an interpreted language for the design of integrated circuits encompassing several new design disciplines. First of all, it is based on the separated hierarchy principles that were described by Jim Rowson. Many design verification problems become simpler if a chip is described out of cells that are either pieces of the final circuit, or are compositions of cells. Earl allows one to design leaf cells of the hierarchy, and compose cells by stretching and abutment. Earl contains an algorithm to join the ports of connecting cells, while satisfying designer constraints on the separations between points.

The layout primitives are based loosely on the circular, "Boston", layout scheme that Carver Mead has been experimenting with. Contact cut structures are circular, and a path of a wire can be specified to miss a point by a given distance. Because it is interpreted, the design iteration time is very short; there is no edit, compile, execute, and check-plot cycle. Earl is currently being used in the VLSI design class at Caltech.

Documentation: Kingsley, Chris, "Earl, an Interpreted Circuit Design Language", Computer Science, Caltech, internal document (4682), November 1981.

Self-timed Systems

A chip set which will be used to link a multiple processor network together via an asynchronous bus is being designed. The bus is similar in spirit to the TRIMOSBUS, though it is implemented quite differently. There can be multiple receivers for a single message. Arbitration (for sendership) is done with a cyclic priority scheme. The chips will have queues for both input and output so that the messages can be buffered. Also, we plan to eventually design as part of the chip set a filter, which can be used to connect two such busses.

Two methods for the verification of speed independent circuits are being studied. One is a method of ternary simulation proposed by Randy Bryant in his thesis. This

method checks a circuit for any critical races on the way to a final input state. The algorithm is quite simple, and is only linear in circuit size, but unfortunately it has some problem with dynamic memory. That is, a node capacitance can be modeled to store the state, but the model then assumes that there is some feedback required to keep this information. But feedback paths can take arbitrarily long using this scheme, which gives incorrect results. If a way could found to select the feedback paths that need to be investigated and which can be assumed 'safe', then this method would work. It would also be possible to simulate C elements, etc.

The other method involves a Petri net expansion using Concurrency Algebra. Each element in the circuit leads to several productions in the algebra, and the initial state is then expanded following the rules of the algebra. The algebra has several nice features, like handling concurrent events as a shuffle, and producing a detailed description of the possible ordering of transitions. Further, it only appears to be quadratic in circuit size, instead of exponential. However, there is a problem with wire delay again. Whenever a wire forks, the two branches are assumed entirely independent, which is not really true for an equipotential region. Thus the method produces several behaviors which would not actually be seen in practice. A proposed solution to this problem is to model all delays in the elements and none in the wires.

Characterization of Deadlock Free Resource Contention

An important class of deadlock problems is caused by conflict over shared resources. Resource sharing without a global scheduler is considered in this research. The only scheduling allowed is the delay of a process requiring a resource already in use. The acquisition of resources is concurrent and asynchronous, and no global information is available. This type of "minimal scheduling" is particularly important in VLSI, where the requirement to have global scheduler would seriously degrade its computing potential.

A theorem has been proved for necessary and sufficient conditions for collections of synchronized processes to be free from deadlock in such concurrent systems. The general theorem which characterizes the danger of deadlock in such collections is proved by applying Philip Hall's Theorem on system of distinct representatives (SDR). It is shown that there exists a polynomial time algorithm for testing the deadlock free condition. Also a special case in which all resources are of different types is discussed.

Documentation: In preparation.

Concurrency Algebra

Rigorous definitions for interleaving of arbitrary length strings over the ordinals have been made. These definitions makes it possible to rigorously define the shuffle in a straightforward manner which seems more intuitive than existing definitions (cf. David Park's "On the Semantics of Fair Parallelism").

Using the semantics, theorems can be derived for the various algebraic rules of the Concurrency Algebra.

One desirable property of a concurrent system is fairness, i.e., that no able process is held waiting forever. Another nice property is compactness, i.e., that infinite behavior is the limit of finite behavior. We have found that Petri Nets with self-loops are fair though not compact. We conjecture that Petri Nets without self-loops are compact.

Dynamic graphs, like Petri Nets, are necessary for describing both the static topological structures as well as the temporal dependance of various events occuring in the structure. Petri Nets are sufficient for most uses requiring dynamic graphs of one sort or another if some redundancy is allowed.

Currently the different methods suggested for composition of smaller Petri Nets to form larger ones are being surveyed. Conceptually they all boil down to one main principle, the "union" operation, that is, the identification of places and transitions with same names.

Documentation: Choo, Young-il, "Concurrency Algebra: Towards an Algebraic Semantics of Petri Nets," Computer Science, Caltech, internal document (4085), December 1980.

Language Design

Unification of expressions is an important function in the execution of logic programming languages such as for instance Prolog. The unification operation is a time consuming operation in current implementations.

As a M.S. thesis project a hardware implementation of the unification operation has been carried out. The chip design is based on a modification of Robinson's algorithm. The UNIF chip is divided functionally into four parts: controller, data-registers, stacks, and an equation table. The controller is implemented as a PLA. The data-registers stores temporary information. The stacks hold multiple arguments of the expressions, replacing the software recursion with hardware. The equation table is a random access memory where expressions to be unified are stored. The equation table is not part of the UNIF chip. The chip size is approximately 3000 x 4500 lambda.

Documentation: Lien, Sheue-Ling, "Towards a Theorem Proving Architecture", Computer Science, Caltech, Technical Report 4653, July 1981.

Logic for Program Verification

λ-logic is a deductive system based on combinatory λ-terms. Its language is concieved by extending the set of λ-terms through the addition of new terms which are logical connectives. The model for λ-logic is Dana Scott's D_∞, which can be represented as a pseudo-Boolean algebra.

A detailed proof has been made that shows that D_∞ can be constructed as a Hoyting algebra, thus being a model for some Heyting intuitionistic logical system. This relationship between algebraic models of computer languages and the algebraic model theory is of great interest in establishing a logical framework for the

verification of programs.

Documentation: Rudin, Leonid, " λ -Logic", Computer Science, Caltech, Technical Report 4521, May 1981.

A Dynamical Model of Computational Systems

By considering systems governed by Hamilton's equations in a general phase space, it is possible to show that certain fundamental physical laws must govern the dynamics of information storage and processing systems. This is accomplished by formulating a definition of storage in dynamical terms; that is, instead of using the usual state machine model where storage is abstractly represented by a symbol in a set of states, storage is now defined in dynamical terms via the postulate that a system stores information when there is a region of its phase space in which it must stay if initialized in that region.

The notions of stability and robustness are incorporated into this model by postulating that the trajectories of the system must be crossing the boundary of the region. This postulate seems reasonable in the light of known properties of many electrical information storage systems and is currently being investigated for applicability to biological storage systems as well. Once such a postulate is adopted as a valid hypothesis, certain conclusions follow about the energetics of the system. For instance, it is a consequence of Liouville's theorem that no such system can conserve energy, i. e. it must be dissipative. Thus the Hamiltonian dynamics mentioned above must be perturbed in a dissipative way to create regions where information can be stored.

In statistical mechanics a systems entropy is defined as the volume it occupies in phase space. Using this definition one can directly correlate the entropy created and destroyed in switching with the energetics of the system. Based on these concepts and bifurcation theory a rigorous definition of reversible transformation in thermodynamical systems can be formulated.

Publications, Technical Reports and Internal Memorandas (ARPA)

Barton Antony F., "A Fault Tolerant Integrated Circuit Memory," Computer Science, Caltech, Technical Report 3761 (Ph.D. Thesis), April 1980.

Browning, Sally A., "The Tree Machine: A highly concurrent computing environment," Computer Science, Caltech, Technical Report 3760 (Ph.D. Thesis), January 1980.

Browning, Sally A., "Generating Padding Processors for Arbitrary Fanout Trees", Computer Science, Caltech, internal document (3827), July 1980.

Browning, Sally A. and Charles L. Seitz, "Communication in a Tree Machine," *Proceedings of the Second Caltech Conference on VLSI*, January 1981.

Carroll, Chris R., "A Smart Memory Array Processor for Two-Layer Path Finding," *Proceedings of the Second Caltech Conference on VLSI*, January 1981.

Carroll, Chris R., "Hardware Path Finders," Computer Science, Caltech, internal document, September 1980.

Chen, Marina and Carver Mead, "HARMOS A Notation for designing concurrent systems," Computer Science, Caltech, internal document (3927), August 1980.

Chen, Marina and Martin Rem, "A Fundamental Theorem on Deadlock in Computer Systems," Computer Science, Caltech, internal document, January 1981.

Chen, Marina, "A Semantics for Systolic Array", Computer Science, Caltech, internal document (4635), August 1981.

Choo, Young-Il, "Concurrency Algebra: Towards an Algebraic Semantics of Petri Nets," Computer Science, Caltech, internal document (4085), December 1980.

DeBenedictis, Erik, "FIFI Test System, Preliminary User's Manual," Computer Science, Itech, internal document (4270), April 1981.

DeBenedictis, Erik, "A Preliminary Report on the Caltech ARPA tester project," Computer Science, Caltech, Technical Report 4061, April 1980.

DeBenedictis, Erik, "Justification for a Tree Machine", Computer Science, Caltech, internal document (3751), May 1980.

Johnsson, Lennart, "Gaussian Elimination on Sparse Matrices and Concurrency," Computer Science, Caltech, internal document (4087), December 1980.

Johnsson, Lennart, "A Note on Householder's Method, Sparse Matrices and Concurrency," Computer Science, Caltech, internal document (4089), December 1980.

Johnsson, Lennart, Uri Weiser, Danny Cohen and Alan L. Davis, "Towards a Formal Treatment of VLSI Arrays," *Proceedings of the Second Caltech Conference on VLSI*, January 1981.

Johnsson, Lennart and Danny Cohen, "Computational Arrays for the Discrete Fourier Transform," *Proceedings of the Twenty-Second Computer Society International Conference, COMPCON 81*, February 1981.

Johnsson, Lennart, "Computational Arrays for Band Matrix Equations", Computer Science, Caltech, internal document (4287), May 1981.

L. Johnsson, "A Computational Array for the QR-method", submitted to the MIT Conference on Advanced Research in VLSI, January 25-27, 1982

Johnsson, Lennart and Danny Cohen, "A Mathematical Approach to Modelling the Flow of Data and Control in Computational Networks", *The CMU Conference on VLSI Systems and Computations*, Pittsburgh, October 19-21, 1981.

Kajiya, J. and Mike Ullner, "Filtering High Quality Text for Display on Raster Scan Devices", SIGGRAPH-81, Computer Science, Caltech, April 1981.

Kingsley, Chris, "Earl, an Interpreted Circuit Design Language", Computer Science, Caltech, internal document (4682), November 1981.

Lang, Dick, "A Distributed Class Object Processing Architecture," Computer Science, Caltech, internal document (4199), February 1981.

Li, Peggy, "The Logarithm Machine," Computer Science, Caltech, Technical Report (MS Thesis), Caltech, November 1980.

Li, Peggy, "The Tree Machine Operating System", Computer Science, Caltech, internal document (4618), July 1981.

Lien, Sheue-Ling, "Towards a Theorem Proving Architecture", Computer Science, Caltech, Technical Report 4653, July 1981.

Locanthi, Bart, "The Homogeneous Machine", Computer Science, Caltech, Technical Report 3759 (PhD Thesis), January 1980.

Mead, Carver and Rem, Martin, "Minimum Propagation Delays in VLSI", *Proceedings of the Second Caltech Conference on VLSI*, January 19-21, 1981, also published as Computer Science Technical Report 4601, 1981.

Rem, Martin, "Communication in a Binary Tree, I and II, Some observations on a tree mapping problem," internal documents, Computer Science, Caltech, June and July 1980.

Rem, Martin and Carver Mead, "A Notation for Designing Restoring Logic Circuitry in CMOS," *Proceedings of the Second Caltech Conference on VLSI*, January 19-21, 1981, also published as Computer Science Technical Report 4600, 1981.

Rudin, Leonid, "λ-Logic", Computer Science, Caltech, Technical Report 4521, May 1981.

Seitz, Charles L, et al, "Proposed instruction set for the tree machine processor," internal document, Computer Science, Caltech, July 1980.

Whelan, Dan and Ray Eskenazi, "An Inexpensive Multibus Color Frame Buffer", Computer Science, Caltech, internal document (4334), June, 1981.

Whelan, Dan, "A Versatile Ethernet Interface", Computer Science, Caltech, Technical Report 4654, July 1981.

Internal documents may be obtained only from the authors. Technical Reports and Technical Memoranda can be obtained from the Computer Science Librarian, Room 256-80, California Institute of Technology, Pasadena, California 91125. Please identify yourself to the librarian as a member of the ARPA community!