LEVEL II

CALIFORNIA INSTITUTE OF TECHNOLOGY

PASADENA, CALIFORNIA 91125

COMPUTER SCIENCE 256-80

071 550

5 May 1981

AD A099374

Col. Duane Adams
Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, Virginia 22209

Dear Sir:

Enclosed please find the interim technical report for the period of January 1, 1981 through March 31, 1981 for Contract No. N-00014-79-C-0597, "Submicron Systems Architecture", ARPA Order No. 3771.

Sincerely,

Chas L. Seitz

Charles L. Seitz
Associate Professor of Computer Science

CLS:nb:4278

Enclosure
(as stated)

cc: Director, NRL, Washington
Defense Documentation Center
Mr. Clint Werner, ONR, Pasadena
Mr. A. J. Lindstrom, CIT
Dr. Clifford Lau, ONR, Pasadena

R & D STATUS REPORT

ARPA ORDER NO.:	3771
CONTRACTOR:	Caltech
CONTRACT NO.:	N00014-79-C-0597 New Amount Funded - $1,424,692
EFFECTIVE DATE OF CONTRACT:	1 March 1979
EXPIRATION DATE OF CONTRACT:	31 May 1981
PRINCIPAL INVESTIGATORS:	Dr. Charles L. Seitz Dr. Carver Mead
TELEPHONE NO.:	(213) 356-6569
SHORT TITLE OF WORK:	Submicron Systems Architecture
REPORTING PERIOD:	January 1, 1981 through March 31, 1981

DESCRIPTION OF PROGRESS:

See attached report

CHANGE IN KEY PERSONNEL:

None

SUMMARY OF SUBSTANTIVE INFORMATION DERIVED FROM SPECIAL EVENTS:

None

PROBLEMS ENCOUNTERED AND/OR ANTICIPATED:

None

ACTION REQUIRED BY GOVERNMENT:

None

FISCAL STATUS:

1.	Amount currently provided by contract:	$1,424,692
2.	Expenditures and commitments as of March 31, 1981:	1,028,675
3.	Funds required to complete work:	$ 396,017

(15) N00014-79-C-0597, ARPA Order-3771

(10) Charles L. Seitz
Carver Mead

California Institute of Technology

Computer Science

(11) 5 May 81

(6) SUBMICRON SYSTEMS ARCHITECTURE

(12) 7

(9) Progress report, ~~January 1 - March 31 1981.~~
1 Jan - 31 Mar 81

The Tree Machine Project

An assembly language has been defined for the tree machine and its instruction set. The language supports both the notation proposed and used by Sally Browning and the notation proposed and used by Marina Chen in her work on HARMOS. The assembly language contains pseudo instructions for the definition of the interconnections in the logical tree and the logical name of the ports. Macro definition and expansion features are included. These features are used in the generation of padding macros for mapping the logical tree onto the physical tree.

The assembler works in three passes. It generates the object code for each processor, does the conversion from the logical tree to the physical tree, and produces information required by the loader. The loader loads a precoded bootstrap loader into each node of the tree before it loads the machine code with proper headers into the tree. Each processor in the tree runs its bootstrap loader and stores its own code into its own memory. The host computer initiates the execution on completion of the loading.

Work on the processor layout is underway again, now that Earl (q.v.) is starting to work.

The COPE Machine

The COPE Machine (Class Object Programming Engine) attempts to exploit the concurrency available in programming languages having constructs similar to the class concept in Simula. The COPE machine is an abstract machine in which objects can execute concurrently. The physical structure and organization of the hardware is transparent to the user.

An object is identified by a unique "handle", which also defines the class of the object. Objects contains code and data. The code is shared with all other objects in the same class, while the data is unique for the object. Objects communicate and synchronize via messages. An object can send messages to all other objects to which it has handles. Messages are queued by the receiving object.

A simulator is currently being developed for the evaluation of the COPE Machine concepts, physical structures and various implementation issues.

071550

The Logarithm Machine

Chips for the Logarithm Machine have recently been returned. The chips are currently being tested.

Mapping of one communication structure onto another

The mapping of an algorithm of a certain communication structure onto an array of a different structure can be divided into several subclasses of problems. One such class is the mapping necessary to use an array designed for a particular algorithm and problem size for the same algorithm but a problem size larger than the array can accommodate directly. The control of the data flow which typically is implicit in the interconnection scheme for the basic arrays has to be made explicit. We are currently studying suitable control and data organizations for orthogonal arrays for the solution of band matrix equations. For the array and problem studied a simple modification of the cells and the control of the array makes it readily useful for "oversize" problems. There is however a need for a fairly extensive data management outside the array. The external data management can be simplified at the expense of more complex cells in the array.

In the computational arrays for the Discrete Fourier Transform described next, the mapping is made by exploiting symmetry and other special properties of the data.

The mapping of large trees onto ensembles of a limited number of processors interconnected in regular patterns is also being studied.

Formal description of Computational Networks

A formal description of computational networks is being developed in collaboration with Danny Cohen at USC/Information Sciences Institute. The goal is to be able to proceed in an entirely formal way from a mathematical expression defining a function to be computed to a description of a computational network computing the desired function. In the initial joint efforts the notation proposed earlier by Cohen has been used to formally derive computational arrays for the Discrete Fourier Transform. Arrays implementing the FFT are of particular interest since the data flow is not laminar.

In the notation used the control can be modeled explicitly. Explicit modeling of the control is necessary when the resources are shared over time.

Wire Routing

The building of a small machine (16 - 32 Pathfinder chips) is near completion. The principal of analog control of digital computations, as used in the Pathfinder chip, is being evaluated. The project is expected to be completed within the next few months.

Testing

Our development of a tester and a test language have now progressed to a stage where both our medium tester and the proposed test language are being used on an experimental basis, A preliminary user's manual is available and will be handed out during the testing session.

Local Network

Presently, we are exploring network connection schemes. We have had a version of the Internet Protocol running on our nodes for some time. Since the systems software on the nodes provides event scheduled multi-processing, our connection management protocols provide inter-network inter-process communications. To complete our effort to provide a network that acts as a terminal concentrator we plan to make our connection management software more robust, for instance by modifying the process scheduling software to handle situations involving wait for multiple conditions to become true.

Earl

A new design system Earl, an interpretive "silicon assembler," is now sufficiently complete that it is starting to be used by a small group of designers. Bugs are being removed and new features added. Earl will be available for distribution this summer. If you are interested in getting copies of our VAX software package, including CLAP (a C language based LAP), CIF2OP (a CIF 2.0 plotter), and Earl, send mail to Chuck@CIT-20.

Earl is written in C to run on a VAX under Berkeley Unix. It allows the definition of cell geometry with constraints, and maintains information about the ports of a cell. Its composition operators do constraint solving to fit cells or compositions of cells together by abutment, stretching the geometry as required. Earl also includes an interesting geometrical primitive that allows one to specify a wire path between points while "missing" one or more other points clockwise or counterclockwise by a specified radius. Earl thus allows not only "funny angles" but even circular arcs.

Notations for design of Concurrent Systems

A theorem on deadlock in concurrent computing environments has been proved. The theorem states that a system is free of deadlock if the number of available resources in contention plus the number of processes competing for these resources is greater than the total number of resources requested.

Self-timed Systems

Quite a few new designs for self-timed elements in nMOS and CMOS technology were developed in this period, and many are currently queued up for fabrication. These designs include self-timed precharge PLAs that are denser than conventional PLAs even though they generate completion signals, a CMOS arbiter, and an interesting FIFO/LIFO structure (designed by an ARPA visitor from Linkabit, Dr. Klein Gilhausen).

On the theoretical side, some progress was made (in collaboration with Susan Owicki at Stanford) in verifying the sequencing properties of compositions. The sequencing rules for the elements or systems being composed are represented in a sequence-net (s-net) notation -- a form of Petri net --, and the reasoning about composing the parts is based on the reduction of a merged s-net.

Concurrency Algebra

Current research focus on the problem posed by Petri Nets of infinite behavior. All finite behaviors can be proved by ordinary induction. To show two infinite strings to be identical requires transfinite induction, which requires the behavior of Petri Nets to have some "compactness" property.

Logic for Program Verification

Logic for lambda calculus programs is being studied. Lattice models as suggested by Dana Scott are used. The models do not define a Boolean algebra, but a pseudo-Boolean or Heyting algebra. Using a new logic, LAMBDA-LOGIC, a formalization of program verification can be made.

Language Design

Our effort to design a unification chip is now completed. The chip represents a hardware implementation of the unification program by J. A. Robinson. The UNIF-CHIP unifies a pair of expressions and computes the substitution set for this pair of expressions. The result of a successful unification is stored off chip in a RAM. The chip consists of three parts: controller, data register, and a stack memory. The size of the chip is approximately 3000 x 4500 lambda.